

Non-parametric Probabilistic Machine Learning Methodologies suited to Real-world Data



Gargi Roy

Supervisor: Dr. Dalia Chakrabarty

Statistics and Data Science Research Group,
Department of Mathematics,
Collage of Engineering, Design and Physical Sciences,
Brunel University London,
London, United Kingdom

A Thesis Submitted in Fulfilment of the Requirements for the Award of the
Degree of Doctor of Philosophy (Ph.D.)

March 2025

I would like to dedicate this thesis to Sri Shibendu Lahiri and my loving parents.

Declaration

I declare that all the material contained in this thesis is my work.

I declare that no material contained in the thesis has been used in any other submission for an academic award.

I declare that while registered as a candidate for the University's research degree, I have not been a registered candidate or an enrolled student for another award of the University or other academic or professional institution.

I confirm that I have undertaken the programme of related studies in connection with the programme of research following the requirements of my research degree registration.

Gargi Roy
March 2025

Acknowledgements

I would like to convey my deepest gratitude to Dr. Dalia Chakrabarty for her extraordinary support, excellent supervision and encouragement throughout my Ph.D. journey. A debt of gratitude is owed to my beloved supervisor Dalia for all that I have learnt from her, without her teaching, this endeavour would not be completed.

I also convey my sincere gratitude to my panel members for evaluating my progress in the reviews and providing valuable feedbacks. I express my warm thanks to all the faculty members of my department along with the head of the department for their support. I am grateful to EPSRC for fully funding my Ph.D. being an international doctoral researcher and the “Prachi Dwivedi award” for providing some financial support towards the travel during my doctoral studies.

I would like to thank all the academic staffs and technical staffs for providing timely support, especially when we all struggled to survive a pandemic caused by COVID-19. I am grateful to be a part of the stimulating environment provided by the fellow doctoral researchers at the department of Mathematics, Brunel University London. I thank my fellow doctoral researchers Kane and Joe for meet-ups and technical discussions.

I also thank my housemates and friends Unmesh Wagh, Anirban Mitra, Swati Wagh, Sayali, Swapna for providing me the required work environment during the work-from-home set-up and being patient with a doctoral researcher! Their friendship and emotional support helped me accomplish this tough Ph.D. journey while I was away from my family in the United Kingdom.

Last but not the least, I am deeply thankful to my husband, my mother and father, without their love, understanding, patience and support, this thesis would not get a real shape. Finally, I would like to convey my deep gratitude to my spiritual father Sri Shibendu Lahiri for his teachings of Kriya Yoga which helped me maintaining a good mental health, a good balance and stress management when I passed through the phase of failures before succeeding in the innovation, especially in the pandemic phase.

Abstract

Recent years have seen a massive increase in the use of supervised learning for prediction tasks in various real-world applications. In supervised learning, the relationship between two variables - an input and an output - is sought. In spite of several existing learning models, the learning of this sought relation poses several challenges given real-world data due to different characteristics of this data, such as noise of the observations; non-stationarity in the data; inhomogeneities present in the correlations between different output pairs that are realised at inputs located differently in the space of the input variable. Moreover, real-world data can be high-dimensional, where both the input and the output can be tensor-valued in general. These challenges induce the following desiderata in the prediction that is performed after such learning is undertaken: fast prediction that is accurate, uncertainty-included and reliable, as well as low in computational complexity, for easy implementation. Additionally, the prediction exercise - foreshadowed by the learning - needs to be scalable to high-dimensions.

Although, some of the existing learning models address a subset of the challenges given non-stationary data, (towards reliable, uncertainty-included predictions), they typically require learning of a large number of hyperparameters, making these learning techniques computationally intensive. Also, they do not come with easy-to-implement algorithms, which makes these models infeasible for applications using medium sized real-world data. Furthermore, some of the existing models are designed for dedicated applications, using domain-specific model assumptions, generalisability of which outside the domains can be questioned.

This thesis addresses challenges of real-world data, and presents applications of a generic, completely non-parametric learning model that is reliable, accurate, parsimonious, and works given non-stationary data that can be high-dimensional in general. Equipped with an easy-to-implement algorithm, such a learning technique overcomes the limitations of existing models. More precisely, this thesis attempts demonstration of reliable learning of a function (that represents the relation between a pair of random variables), by modelling this function as a sample function of a Gaussian Process. Such learning will be followed by fast prediction of the output that is realised at test inputs, where said prediction offers closed-form mean and variance of this output.

In fact, in this approach, the predictions that follow the learning of the inter-variable relation, follows from the identification of the posterior predictive distribution of outputs realised at test inputs.

The illustration of this approach has been performed for applications in various domains such as finance, energy consumption and astrophysics, where the data is inhomogeneously correlated and have diverse dimensions. From finance sector, real-world time-series data has been considered where both the input and output is scalar-variate. In the real-world energy consumption data, the input is vector-variate and the output is a scalar. The astrophysics application uses an astronomical simulation data where the input is a vector and the output is a matrix, yielding the sought function to be high-dimensional.

Inference is undertaken throughout my doctoral work using Markov chain Monte Carlo (MCMC) sampling techniques. This thesis also highlights the sensitivity of predictions achieved with Deep Neural Networks (DNN), to the architecture of the DNNs.

The chapter-wise distribution is as follows. The first chapter introduces the topic. The second chapter discusses various MCMC techniques and illustrations of these inference techniques to perform parametric learning with a small real-world data. The third chapter introduces the background of Gaussian Process (GP) based learning, application of GP-based supervised learning for efficient learning of uncertainty with an under-constraint MCMC for prediction. A probabilistic, non-parametric, non-stationary, parsimonious learning strategy is presented in the fourth chapter along with results on applying the model, and on comparison against existing models. This application is relevant to the case of both input and output is scalar-variate and the data is inhomogeneously-correlated. The fifth chapter includes the application of the learning strategy with a multivariate (with vector input and scalar output), inhomogeneously-correlated real-world data. This chapter also discusses some ideas about inhomogeneities in the correlation structure of the training data and the DNN exposition in both univariate and multivariate setups. An application of the learning of a high-dimensional function is discussed in the sixth chapter. In this application, the input is a vector and the output is a matrix. Prediction of the output at a test input vector is then presented. Finally the thesis has been concluded in chapter seven.

The first appendix includes the application of the presented non-parametric learning strategy towards forecasting, along which a new strategy for designing of priors for performing forecasting with real-world inhomogeneously-correlated data. The second appendix includes some of the results of inference performed with various MCMC techniques included in chapter two.

Table of contents

List of figures	xv
------------------------	-----------

List of tables	xxxiii
-----------------------	---------------

1	Introduction	1
1.1	Introduction	1
1.2	Objectives of this thesis	5
1.3	Achieving the objectives of this thesis	6
1.3.1	MCMC-based inference	8
1.3.2	Why I preferred to avoid INLA	8
1.4	Novelty of this thesis work	9
1.5	Publications out of this thesis	10
1.6	Thesis outline	11
2	Bayesian Inference using Markov Chain Monte Carlo (MCMC)	13
2.1	Introduction	13
2.2	MCMC literature	13
2.3	Motivation	15
2.4	MCMC introduction	22
2.4.1	Markov Chain in the context of MCMC	22
2.4.2	Markov Chain Monte Carlo (MCMC)	23
2.4.3	Why Metropolis-Hastings (MH) algorithm works	24
2.5	Various MCMC algorithms	27
2.6	Application with real-world data	28
2.6.1	Data	28
2.6.2	Model for the O-ring data	28
2.6.3	Results	31
2.7	Conclusion	34

3	Efficient Uncertainty Quantification for Under-constraint Supervised Learning using Gaussian Processes	37
3.1	Introduction	37
3.2	Gaussian Process Regression	39
3.2.1	A toy example	41
3.2.2	Constrained prediction and Gaussian Process	41
3.3	Data and model	43
3.3.1	Learning parameters of the logistic regression model	45
3.4	Learning with training data sets generated at parameter summaries with DNN exposition	47
3.4.1	Learning ℓ under the constraint given D_{mean}	47
3.4.2	Learning with DNNs	49
3.4.3	Learning ℓ under constraint, with MCMC, using D_{right}	50
3.5	Efficient uncertainty percolation using pipe-lined architecture	53
3.6	Conclusion	57
4	A Fully Non-parametric, Non-stationary, Parsimonious, Kernel for Supervised Learning	59
4.1	Introduction	59
4.2	Need for non-stationary, non-parametric kernels for real-world data	62
4.3	State-of-the-art kernels	64
4.3.1	Discussion on hierarchical GP models	66
4.3.2	Discussion on Multiple Kernel Learning	69
4.3.3	Discussion on vector-valued kernels	70
4.4	Methods, non-parametric model and inference	71
4.4.1	Model	72
4.4.2	Example of non-parametric kernels	79
4.4.3	Inference	80
4.4.4	Single GP layer based models	85
4.5	Non-stationary extension of the non-parametric kernel	87
4.6	Blended model: a non-stationary non-parametric kernel	91
4.7	Results	93
4.7.1	Data	94
4.7.2	Results from SQE-looking non-parametric kernel:	95
4.7.3	Results of Matérn-looking non-parametric kernel:	96
4.7.4	Results of learning with a non-parametric model, given noisy data	97
4.7.5	Results of non-stationary model extending from non-parametric model	101

4.7.6	Results of Blended model	102
4.7.7	Single GP layer based model using stationary SQE kernel	104
4.7.8	Single GP layer based model using stationary Matérn kernel	104
4.7.9	Non-stationary kernel from Remes et al. (2017)	106
4.7.10	Non-stationary kernel from Paciorek and Schervish (2003)	107
4.7.11	Comparative performance for predictions made with all models	107
4.7.12	Learning with DNNs	110
4.8	Conclusion	111
5	Non-parametric Learning with Multivariate Data and Deep Neural Network	
	Exposition	113
5.1	Introduction	113
5.2	Inhomogeneity in correlation structure of data	115
5.3	Application of non-parametric Learning with Multivariate Data	117
5.3.1	Application with data	118
5.3.2	Results of multivariate learning	120
5.4	Learning with Deep Neural Network models	127
5.4.1	DNN model	128
5.4.2	Single layer GP based learning model for univariate data	131
5.4.3	Datasets	132
5.4.4	Results	134
5.5	Conclusion	142
6	Non-parametric Learning in High Dimensions	145
6.1	Introduction	145
6.2	Non-parametric learning with high-dimensional data	147
6.2.1	Prediction of matrix-variate variable given vector input	149
6.2.2	Data	150
6.2.3	Learning of covariance matrices	153
6.2.4	Inference	157
6.2.5	Computational issues	158
6.3	Single layer GP based learning with high-dimensional data	159
6.4	Results	160
6.4.1	High-dimensional prediction with comparison and analysis	166
6.5	Conclusion	175

7 Conclusion	177
7.1 Limitations of current work and future work	178
Appendix A Forecasting with Non-parametric Model	181
A.1 Forecasting with Gaussian Process	181
A.1.1 One-step ahead strategy for forecasting	181
A.1.2 A novel strategy for forecasting with One-step ahead approach . . .	183
A.2 Information redundancy modelling for time series data	187
A.3 Results on forecasting with new strategy	188
Appendix B Figures for Table 2.2	197
B.1 Monte Carlo Simulation:	197
B.2 Figures for Table 2.2	198
References	231

List of figures

3.1	Toy example of GP modeling a one-dimensional function with $\ell = 0.3$ with training data and GP predictive mean.	42
3.2	Ten samples from GP prior and GP posterior from the GP modeling a one-dimensional function with $\ell = 0.3$ on the toy example.	42
3.3	Example of GP modeling a one-dimensional function with different ℓ on the toy example.	43
3.4	Results of learning α, β using test flight data D . The top row depicts the trace (variation with iteration index) of α, β and logarithm of the joint posterior of these 2 parameters, given data D , as I move from the left to the right panel. In the lower rows, at the extreme left are the histogram representations of the marginal posterior of α and β given data D , and that of the log of the joint posterior. The middle column of panes in the lower row represent the histograms of α (upper) and β (lower panel) values sampled during the MCMC chain, over three distinct, equally-wide (covering 20000 consecutive iterations), non-overlapping iteration windows. the three histograms are coloured in blue, red and black, and overplotted. The right most panel in the lower row depicts the $\Pr(Y = 1 X = x_i)$ computed at 16 different temperatures using the mean of the α and β learnt in this chain, using data D . This MCMC chain was run using a Metropolis-within-Gibs algorithm. The total number of iteration in this chain is 200000; the variances of the truncated Normal proposal densities for α and β are 0.074^2 , and 0.008^2 respectively. Normal priors were used on the parameters.	46

3.5	Results of learning ℓ using the generated training data D_{mean} in which, at design temperature x_i , the output ($w_i \equiv \Pr(Y = 1 X = x_i)$) is computed using the mean of the learnt α and β . Here the 2-stepped prior is imposed on ℓ . The top left and second-to-left panels respectively depict traces of learnt values of ℓ and logarithm of posterior of ℓ given training data D_{mean} . The 2 further panels to the right present the histogram representations of the marginal of ℓ given D_{mean} and the histogram of the posterior values. The bottom row depicts the 3-window convergence test, (see caption of Figure 3.4) to confirm convergence of the MCMC chain.	49
3.6	DNN predictions with varying architectures. “HL”: Hidden layer, “N”: Neurons, “64N/HL”: 64 neurons in each hidden layer.	50
3.7	As in Figure 3.5, except these depict results obtained using training data D_{right}	51
3.8	GP predictions with 2.5 standard deviation of uncertainties with different ℓ s	52
3.9	Two approaches of dealing with uncertainty	54
3.10	Traces and plots for learning α, β, ℓ in pipe-lined architecture; above, “Conv.” substitutes for “convergence”	56
3.11	In the MCMC chain that updates the α and β parameters, while learning ℓ , predicted mean output $W \equiv \Pr(Y = 1 X = x_{test})$, at each test input x_{test} , is depicted in filled circles. Said prediction follows learning of the input-output functional relationship by modelling this sought function with a GP, the covariance structure of which is parameterised by ℓ . 2.5 times the predicted standard deviation of the output value is depicted on each side of the predicted mean, within the grey shaded region.	56
3.12	Variation of predicted mean of the output $W \equiv \Pr(Y = 1 X = x_{test})$ with iteration index, at few values of x_{test}	57
4.1	Flow of the dual-layered GP-based learning using fully non-parametric kernel.	83
4.2	The methodical flow of non-stationary extension of the SQE-looking non-parametric kernel Cartoon graphical representations. The cartoon graphs represents inputs along X axis and outputs along Y axis, four neighbouring points of two training inputs x_i and x_j are depicted in short vertical lines along with its corresponding predictive means in stars, triangle shows the training output realised at x_i and x_j	90
4.3	The methodical flow of three stages along with Blended model using SQE-looking non-parametric kernel. The dotted box presents the whole flow of non-stationary prediction presented in Figure 4.2.	94

4.4	Brent crude oil price training data. Along x axis is dates and along y is the oil price in U.S Dollars.	95
4.5	Results of learning hyperparameters with dual staged model using SQE-looking non-parametric kernel using Brent crude oil price data. The panels from left, in the top row depict the trace of ℓ that is updated in the first block, δ learnt in the second block, predicted ℓ using the inner GP at the end of second block, the logarithm of posterior in the outer GP, given the training data. The panels from left, in the bottom row depict the trace of logarithm of the posterior of δ in the inner GP, given the dynamically-varying lookback data and the convergence diagnostics pertaining to the ℓ , predicted ℓ and δ respectively.	96
4.6	Traces of prediction of the test points with SQE-looking non-parametric kernel. The x-axis and y-axis of each plot denotes the iteration and the oil prices respectively.	97
4.7	Results of learning ℓ and ν using the non-parametric model with Matérn-looking kernel, given Brent crude oil price data. <i>Top row, from left:</i> trace of ℓ updated in the first block of the conducted Metropolis-within-Gibbs chain; trace of ℓ as predicted from the $\mathcal{GP}$ in the inner-layer, at the ℓ and ν that are updated in the first block; trace of δ ; trace of ν updated in the first block of the chain; trace of ν value predicted using the $\mathcal{GP}$ in the inner-layer. <i>Middle row, from left:</i> trace of δ_ν ; trace of logarithm of the joint posterior of ℓ and ν given training data; traces of logarithm of posterior of δ_ℓ and δ_ν respectively, given relevant (dynamically-varying) lookback data; convergence diagnostic of ℓ . <i>Bottom row, from left:</i> convergence diagnostics for predicted ℓ , δ_ℓ , ν , predicted ν , δ_ν	98

- 4.8 Traces of prediction of the test points for nested learning with Matérn-looking kernel. The x-axis of each plot denotes the iterations and the y-axis shows the oil prices. The left most panel of the top row shows the histogram for test point zero, the second left panel shows the histogram for test point one, and similarly incrementally the right most panel shows the histogram for test point four; in the second row from top, the left most panel shows the histogram for test point five, the second left panel shows the histogram for test point six, and Similarly incrementally the right most panel shows the histogram for test point eight; Similarly with this pattern of increment the bottom most row presents the histogram for test point 17, the second left panel shows the histogram for test point 18 and right most panel shows the histogram for test point 20. 99
- 4.9 Results of learning with the non-parametric model that uses the SQE-looking kernel, given noisy Brent crude oil price data. The panels from left in the top row depict traces of ℓ ; of ℓ predicted in second block; of δ ; of the error variance σ_ε^2 (with the true value of σ_ε^2 depicted in white line); and the logarithm of the joint posterior of ℓ and σ_ε^2 , given the data noisy data. The panels from left in the bottom row display convergence diagnostics pertaining to the learning of ℓ , predicted ℓ , δ and σ_ε^2 100
- 4.10 Figure displaying histograms of output values predicted at five test inputs, obtained from the fully non-parametric model - with an SQE-looking kernel that parametrises $\mathcal{GP}_{outer}$ - given noisy data (in black dashed lines), and noise-free data (in blue solid line). The histogram displaying results given the noise-included data, is wider than that depicting prediction results obtained with the noise-free data. 101
- 4.11 Correlation matrices of the non-stationary equivalent of the non-parametric model that invokes an SQE-looking kernel. The left most plot presents the correlation matrix computed between outputs realised at pairs of design points in the training dataset $\mathbf{D}_{train}$, while the middle plot shows the correlation matrix computed between outputs realised at input pairs that each comprise test inputs in D_{test} . Compared to these panels, the right-most panel depicts the correlation matrix between an output realised at a design point, and another that is realised at a test input. 101
- 4.12 Similar to panels in Figure 4.11, except here, contour plots of the computed length scales - of the non-stationary equivalent of the non-parametric model - are depicted. 102

- 4.13 Figure displaying histograms of output values predicted at five test inputs (every fifth), obtained from the non-stationary equivalent of the non-parametric model. 102
- 4.14 The left panel depicts the trace of the length scale hyperparameter ℓ_B of the Blended model. The middle panel depicts logarithm of the posterior *pdf* of ℓ_B given data $\mathbf{D}_{train}$. The right panel presents the convergence diagnostic of the learning of ℓ_B 102
- 4.15 Figure displaying length scale hyperparameters relevant to the implementation of the Blended model. The top row depicts lengths scales computed at input pairs, each of which is a design point. These inputs relevant to the presented plots are (from the left) at the 10-th and 5-th design inputs; the 15-th and the 2nd; the 15-th and the 50-th design points; the 30-th and the 10-th; the 60-th and the 65-th design points; the 180-th and the 10-th. The middle row depicts traces of length scales that are computed at input pairs, each of which is a test input. From the left, panels depict traces of length scale parameters that are relevant to the pair that comprises the 1st and 3rd of the chosen 20 test inputs; the 3rd and 4-th; 2nd and the 14-th test points; the 5-th and the 15-th; the 10-th and the 18-th; and the pair comprising the 19-th and 18-th test points. Bottom row depicts traces of length scales that are computed at input pairs, one of which is a design point, and the other a test input. From the left, the presented traces are pertinent to the pair of the 1st test and the 10-th design inputs; the 1st test and the 25-th design point; the 2nd test and the 100-th design inputs; the 5-th test and the 50-th design; the 9-th test and 25-th design; and the 19-th test and the 180-th design points. 103
- 4.16 Traces of mean outputs predicted at the test points (every fifth test input), where the predictions are obtained using the Blended model. 103
- 4.17 The left-most plot presents a heat map of the correlation matrix, any element of which is the inter-output correlation in the Blended model, where each correlation element is between outputs realised at a pair of design inputs. The heat map in the middle plot represents the correlation computed at inputs pairs, where each of the pair is a test input. The right-most plot depicts the heat map of the correlation matrix, an element of which is the correlation between an output realised that is predicted at a test input, and another that is realised at a design time. 104

4.18	Similar to Figure 4.17, except this plots depicts the contour plots of length scales at which an inter-output correlation is kernel-parametrised by the kernel of the Blended model.	104
4.19	Traces of learnt length scale parameter ℓ (left panel), logarithm of posterior <i>pdf</i> of ℓ given the training data (middle panel) and the convergence diagnostics on the learning of ℓ (right panel), using a stationary SQE kernel.	105
4.20	Traces of mean outputs predicted at the test points (every fifth test input), where the predictions are obtained using the stationary SQE kernel.	105
4.21	From left: Traces of hyperparameters ℓ and ν learnt with a stationary Matérn kernel, (given the training data on Brent crude oil price data); the trace of the joint log posterior of the hyperparameters given this data; convergence diagnostics on the learning of ℓ and ν respectively.	105
4.22	Traces of mean outputs predicted at the test points (every fifth test input), where the predictions are obtained using the stationary Matérn kernel.	105
4.23	Results of the learning performed with the non-stationary spectral kernel developed by Remes et al. (2017). <i>Top row, from left:</i> Traces of the hyperparameters w , ℓ and μ and the logarithm of joint posterior respectively. <i>Bottom row, from left:</i> convergence diagnostics of the learning pertaining to the w , ℓ and μ	106
4.24	Traces of mean outputs predicted at the test points (every fifth test input), where the predictions are obtained using the non-stationary kernel developed by Remes et al. (2017).	106
4.25	Results of learning performed with the non-stationary kernel developed by Paciorek and Schervish (2003), given the Brent crude oil price data. Trace of the length scale hyperparameter ℓ of the Matérn GP prior - that is assigned a fixed $\nu = 5.3979$ - is displayed in the left panel, while logarithm of the posterior density of this hyperparameter is on the middle and the convergence diagnostics of the learnt ℓ is shown in the right panel.	107
4.26	Traces of mean outputs predicted at the test points (every fifth test input), where the predictions are obtained using the non-stationary kernel developed by Paciorek and Schervish (2003).	107

- 4.27 Figure displaying prediction of the oil price output at each of 20 identified test inputs (in red), undertaken following the learning of the temporal variation of the price made using the Brent crude oil price data. 2.5 times the predicted standard deviation of the output is added to either side of the predicted mean, as an error bar (in red). Means of the predicted outputs are presented in red stars; true values of the output are in blue circles; and training points are in green triangles. In the left panel of the top row, predictions made using the fully non-parametric model with the SQE-looking kernel are depicted. The middle panel in this row depicts predictions with the fully non-parametric model using Matérn-looking kernel and the top right panel presents predictions using the non-parametric model with an SQE-looking kernel, given noisy oil price data. In the middle row, the left panel shows the predictions made using the non-stationary version of the fully non-parametric model, (where the non-stationary equivalence is constructed using the non-parametric model with an SQE-looking kernel). The middle panel in this row presents predictions made using the Blended model, while the right panel of the middle row shows predictions made using the single GP layer based model with a stationary SQE kernel. In the bottom row, the left panel shows predictions made using a single GP layer based model with stationary Matérn kernel, while the middle panel depicts predictions made using the non-stationary kernel from Remes et al. (2017) and the right panel shows predictions made using the non-stationary kernel of Paciorek and Schervish (2003). 108
- 4.28 Columns in the left, middle and right depict results obtained using the NN-1, NN-2 and NN-3 models respectively. The top row shows predictions at test inputs; the bottom row presents results from the last 10,000 epochs to illustrate the convergence of loss. 111
- 5.1 The left most panel shows the heat map and the right panel shows the 3D bar plot of the design training input of the multivariate Tetouan city data. 119
- 5.2 Output training data plotted against index of design training points of the multivariate Tetouan city data. 120

- 5.3 Results of learning $\ell_T, \ell_H, \ell_W, \ell_D$ using the non-parametric model with SQE-looking kernel, given multivariate data. *Top row, from the left:* traces of $\ell_T, \ell_H, \ell_W, \ell_D$ updated in the first block of the Metropolis-within-Gibbs chain undertaken for inference. *Middle row, from the left:* traces of $\ell_T, \ell_H, \ell_W, \ell_D$ as predicted from the $\mathcal{GP}_T, \mathcal{GP}_H, \mathcal{GP}_W, \mathcal{GP}_D$ respectively in the inner layer at the end of second block. *Bottom row, from the left:* traces of $\delta_T, \delta_H, \delta_W, \delta_D$ updated in the second block. 122
- 5.4 *From left:* trace of logarithm of the joint posterior of $\ell_T, \ell_H, \ell_W, \ell_D$ given training data with the non-parametric model using SQE-looking kernel; traces of logarithm of posterior of $\delta_T, \delta_H, \delta_W, \delta_D$ respectively, given relevant (dynamically-varying) lookback data. 122
- 5.5 Convergence diagnostic of the learning of the hyperparameters using the non-parametric model with SQE-looking kernel given the multivariate training data where the values of the hyperparameter are sampled over three distinct, equally-wide (covering 10000 consecutive iterations starting at the 60000-th; 70000-th; 80000-th iterations for the three windows respectively), windows during the MCMC chain. Histograms of samples taken in the three windows are coloured in blue, red and black. From extreme left in the top row, the panels pertain to hyperparameters: $\ell_T, \ell_H, \ell_W, \ell_D$. From extreme left in the bottom row, the panels pertain to hyperparameters: $\delta_T, \delta_H, \delta_W, \delta_D$ 123
- 5.6 Correlation matrices of the non-parametric model that invokes an SQE-looking kernel. The left most plot presents the correlation matrix computed between outputs realised at pairs of design points in the training dataset, while the middle plot shows the correlation matrix computed between outputs realised at input pairs that each comprise test inputs. The right-most panel depicts the correlation matrix between an output realised at a design point, and another that is realised at a test input. 123
- 5.7 Four test sets - *Top row, from left:* test sets “Test-1”, “Test-2”, “Test-3”, “Test-4” where true values of the test points are plotted against the temperature in blue circle and the training data points against the temperature is shown in green dots. *Bottom row, from left:* test sets “Test-1”, “Test-2”, “Test-3”, “Test-4”, similar to top row, however, in addition it shows the trivial function (a piecewise-linear-function fitted to the output training data) in green solid line. 124

- 5.8 Prediction of integrated power consumption obtained at test set “Test-1” using the non-parametric multivariate model with SQE-looking kernel, given the multivariate training data. Predictions are plotted against the index of the test data points. The true values are shown in blue circles while the predictive means are shown in red stars with the computed uncertainty in terms of the variance of the prediction are plotted in both side of the predicted mean in red straight lines. 125
- 5.9 Prediction of integrated power consumption obtained at test sets (from left to right), “Test-2”, “Test-3”, “Test-4”, using the non-parametric multivariate model with SQE-looking kernel, given the multivariate training data. Predictions are plotted against the index of the test data points. The true values are shown in blue circles while the predictive means are shown in red stars with the computed uncertainty given by the variance of the prediction is plotted on both sides of the predicted mean in red straight lines. 125
- 5.10 Predictions obtained at the test sets “Test-1”, “Test-2”, “Test-3”, “Test-4” plotted against all the four input dimensions - temperature, humidity, wind power and diffuse flows. *Rows from top* present: predictive results at the test set “Test-1”, “Test-2”, “Test-3” and “Test-4” respectively. *Columns from the left* display predictions against: temperature, humidity, wind power and diffuse flows respectively. The true values are shown in blue circles while the predictive means are shown in red stars with the uncertainty of predictions, computed as the variance of the predicted output variable, is plotted on both sides of the predicted mean in red straight lines. Outputs at design points are plotted in green dots. 126
- 5.11 Histograms of predictions made at a few test points, from test set “Test-1” with SQE-looking non-parametric kernel given the Tetouan city multivariate data. 126
- 5.12 Histograms of predictions made at a few of the test points from test set “Test-2” with SQE-looking non-parametric kernel given the Tetouan city multivariate data. 127
- 5.13 Similar to Figure 5.12 except here, few of the test points from test set “Test-3” are used. 127
- 5.14 Similar to Figure 5.12 except here, few of the test points from test set “Test-4” are used. 127

- 5.15 The left most panel presents the training data D_1 along with the minimally-rough sample (trivial) function (with black solid line) where the output oil prices are plotted against days in running total. The second panel from the left plots the computed L-values (in black dots) along with their interval (with black vertical line) from data D_1 against their index; the L-values along with their corresponding interval are marked in red which have no overlap with other L-values. Similarly, the third panel from left depicts plots for dataset D_2 where output integrated power is plotted against index of the training points, while showing the trivial function in black solid line. The right most panel displays the computed L-values (in black dots) along with their interval (with black vertical line) from data D_2 against their index; the L-values along with their corresponding interval are marked in red which have no overlap with other L-values. 133
- 5.16 Columns in the left, middle and right depict results pertaining to dataset D_1 with $NN3_1$ (with ReLU), $NN2_1$ (with sigmoid) and $NN3_0$ (with sigmoid) respectively. The top row shows predictions at test inputs “Test-1”; in the middle row the loss function is plotted; and the bottom row presents results from the last 2000 epochs to illustrate the convergence of loss. True values are displayed with blue hollow circle, while the predictions are depicted with red stars. 136
- 5.17 Columns in the left, middle and right depict results pertaining to dataset D_2 using test set “Test-1” with $NN3_1$ (with ReLU), $NN2_2$ (with ReLU) and $NN2_0$ (with ReLU) respectively. The top row shows predictions; in the middle row the loss function is plotted; and the bottom row presents results from the last 2000 epochs to illustrate the convergence of loss. True values are displayed with blue hollow circle, while the predictions are depicted with red stars. 138
- 5.18 The left most, second from left, third from left and the right most panel presents the predictions with the data D_2 using DNN model $NN1_0$, $NN1_1$, $NN2_1$ and $NN2_2$ respectively. True values are displayed with blue hollow circles, while the predictions are depicted with red stars. 138
- 5.19 The prediction obtained using DNN model $NN1_2$ with Sigmoid activation function with the dataset D_2 is depicted in the left most panel, loss from long windows of epochs, and the last 2000 epochs, are presented in the middle and right panel respectively. True values are displayed with blue hollow circle, while the predictions are depicted with red stars. 139

- 5.20 Figure displaying traces of learning made with the GP based model. The left most and second from left panel depict the traces of the length-scale and the logarithm of the posterior with the training data D_1 using stationary Matérn kernel with $\nu = 3/2$. The third from left panel and the right most panel show traces of learnt length-scale and the logarithm of the posterior given data D_2 , where the learning is done using stationary Matérn kernel having $\nu = 1/2$. . 140
- 5.21 Predictions made on the four test datasets (from left: “Test-1”, “Test-2”, “Test-3”, “Test-4”), given training data D_1 , with stationary Matérn kernel having $\nu = 3/2$ 140
- 5.22 Predictions made on the four test datasets (from left: “Test-1”, “Test-2”, “Test-3”, “Test-4”), given training data D_2 , with stationary Matérn kernel having $\nu = 1/2$ 140
- 5.23 The prediction obtained at test set “Test-1” using DNN model $NN6_0$ with ReLU activation function with the dataset D_3 is depicted in the left most panel. The loss function and the loss for the last 2000 epochs are presented in the middle and right panels respectively. True values are displayed with blue solid circle, while the predictions are depicted with red stars. 141
- 5.24 The prediction obtained at test sets “Test-2”, “Test-3”, “Test-4”, given the data D_3 using DNN model $NN6_0$ with ReLU activation function is depicted the left panel onward respectively. True values are displayed with blue solid circle, while the predictions are depicted with red stars. 141
- 6.1 High-dimensional astronomical data obtained from Chakrabarty (2007). In the left panel, input design points $\mathbf{X}$ are depicted in red dots within polar coordinates with the radial component X_1 and the angular component X_2 . The output matrices $\mathbf{Y}$ are depicted in the right panel which are collated to a 3-rd order tensor $\mathbf{D}_Y \in \mathbb{R}^{50 \times 2 \times 173}$ in the training set I use, since my used training set comprises sets of velocity vectors of 50 stars, collected at 173 identified locations (in the galactic disk). The colour coding presents the different values of this tensor. The velocity components are labelled as “0” - for the first velocity component and “1” - for the second velocity component. 151
- 6.2 Train-test split (80-20) in the high-dimensional astronomical data obtained from Chakrabarty (2007) that consists of 216 number of data points. In the left panel, the black dots represents the index of 173 training points and the red diamonds show the index of 43 test points. In the right panel input test points $\mathbf{X}$ are depicted in red dots within polar coordinates with the radial component X_1 and the angular component X_2 152

- 6.3 Inhomogeneity parameter η , computed from the output values generated at the design inputs, in the high-dimensional astronomical data. The left most panel shows the heat map of the η values computed for each star for each of the velocity components. The middle and the right panels depict visualisation in 3D-bar plots of the training data, from two different angles. The velocity components are labelled as “0” - for the first velocity component and “1” - for the second velocity component. 153
- 6.4 An example 3-dimensional data shown in the left most panel with its depth, row and column depicted. The second panel from left shows the row $\times$ column-dimensional matrix as slices obtained by chopping the 3-dimensional data depth-wise. The third panel from left shows the depth $\times$ column-dimensional matrix as slices obtained by chopping the 3-dimensional data row-wise. The right most panel displays the row $\times$ depth-dimensional matrix as slices obtained by chopping the 3-dimensional data column-wise. 155
- 6.5 Results of learning all the unknown hyperparameters using the non-parametric model with SQE-looking kernel, given high-dimensional astronomical data. The traces from extreme left in the top row pertain to, $\sigma_{11}^{(1)}, \sigma_{22}^{(1)}, \sigma_{10}^{(1)}, \ell_1$ updated in the first block, ℓ_2 updated in the first block given training data. The traces from extreme left in the bottom row pertain to, δ_1 and δ_2 , given relevant (dynamically-varying) lookback data, ℓ_1 value predicted from the $\mathcal{GP}_1$ in the inner-layer, ℓ_2 value predicted from the $\mathcal{GP}_2$ in the inner-layer, A_0 given training data. 162
- 6.6 The left panel shows the logarithm of the joint posterior probability density given the high-dimensional astronomical training data considered in this application using the non-parametric model. The middle and the right panel present the logarithm of the marginal posterior probability density of δ_1 and δ_2 of the inner GPs respectively given their relevant lookback data. 162
- 6.7 Convergence diagnostic of the learning of the hyperparameters using the non-parametric model with SQE-looking kernel given the high-dimensional data where the values of the hyperparameter are sampled over three distinct, equally-wide non-overlapping iteration windows (covering 30000 consecutive iterations starting at the 200000 - th, 230000 - th, 260000 - th iterations), during the MCMC chain. The three histograms are coloured in blue, red and black, and overplotted. From extreme left in the top row, the panels pertain to hyperparameters: $\sigma_{11}^{(1)}, \sigma_{22}^{(1)}, \sigma_{10}^{(1)}, \ell_1$. From extreme left in the bottom row, the panels pertain to hyperparameters: $\ell_2, \delta_1, \delta_2, A_0$ 163

- 6.8 Traces of the length scales ℓ_1, ℓ_2 with various lookback size N_{LB} . The top and bottom rows depict traces of ℓ_1 and ℓ_2 respectively. Traces shown in the left most, middle and right columns pertain to chains with $N_{LB} = 300, N_{LB} = 400$ and $N_{LB} = 500$ respectively. These chains are run using the non-parametric model with SQE-looking kernel for 300000 iterations. 164
- 6.9 Results of learning all parameters using the single GP layer based model with SQE kernel, given high-dimensional astronomical data. The traces from extreme left in the top row pertain to, $\sigma_{11}^{(1)}, \sigma_{22}^{(1)}, \sigma_{10}^{(1)}, \ell_1$ given training data. The traces from extreme left in the bottom row pertain to, ℓ_2, A_0 and the logarithm of joint posterior given training data. 165
- 6.10 Convergence quality of the learning of the hyperparameters using the single GP layer based model with SQE kernel where the values of the hyperparameter are sampled over three distinct, equally-wide, non-overlapping iteration windows (of width 16000 iterations, starting at the $75000 - th, 91000 - th, 107000 - th$ iterations), during the MCMC chain. The three histograms are coloured in blue, red and black, and overplotted. From extreme left, the panels pertain to hyperparameters: $\sigma_{11}^{(1)}, \sigma_{22}^{(1)}, \sigma_{10}^{(1)}, \ell_1, \ell_2, A_0$ 166
- 6.11 Heat maps of the compliance measure pertaining to the 50×2 -dimensional output matrix predicted at each of the 43 test input using the non-parametric model with SQE-looking kernel, given the training data. Compliance value *true* is depicted in black which means truth lies within the predicted interval and *false* value is shown in white (when truth lies outside of the predicted interval). 167
- 6.12 Heat maps of the compliance measure pertaining to the 50×2 -dimensional output matrix predicted at each of the 43 test input using the single layer GP based model with SQE kernel, given the training data. Compliance value *true* is depicted in black which means truth lies within the predicted interval and *false* value is shown in white (when truth lies outside of the predicted interval). 168
- 6.13 Heat maps that represent, from extreme left: estimated mean tensor $\hat{\mathbf{T}}_\mu$ computed using Remark 6.2.1 to be used within the likelihood-computation of the inference; the MSE values depicted in the Table 6.2 pertaining to the non-parametric model; the MSE values presented in the Table 6.2 pertaining to single layer GP based model; the compliance score shown in the Table 6.3 pertaining to the non-parametric model (shown in 0-th column) and single layer GP based model (shown in 1-th column). 170

- 6.14 Predictive results of the velocity component U pertaining to the subset of stars S , where the predictions are made with the non-parametric model using SQE-looking kernel, given the training data. True values are shown in blue circles while the predictive means are shown in red stars with the uncertainty plotted as red line along the both side of the predicted mean. 170
- 6.15 Predictive results of the velocity component U pertaining to the subset of stars S , where the predictions are made with the single layer GP based model using SQE kernel, given the training data. True values are shown in blue circles while the predictive means are shown in red stars with the uncertainty plotted as red line along the both side of the predicted mean. 171
- 6.16 Predictive results of the velocity component V of the subset S of stars, where the predictions are made with the non-parametric model using SQE-looking kernel, given the training data. True values are shown in blue circles while the predictive means are shown in red stars with the uncertainty plotted as red line along the both side of the predicted mean. 171
- 6.17 Predictive results of the velocity component V pertaining to the subset S of stars, where the predictions are made with the single layer GP based model using SQE kernel, given the training data. True values are shown in blue circles while the predictive means are shown in red stars with the uncertainty plotted as red line along the both side of the predicted mean. 172
- 6.18 Train-test split (~ 80 -20) in the high-dimensional astronomical data obtained from Chakrabarty (2007) that consists of 216 number of data points. The black dots represents the index of 173 training points and the red diamonds shows the index of 43 test points. 172
- 6.19 Each column presents the prediction of U components (in top row), trivial function (in middle row) and the L values (in bottom row) pertaining to four stars having index 19, 24, 42 and 49; where the predictions are made with the non-parametric model using the SQE-looking kernel, given the training data. The true values are shown in blue circles while the predictive means are shown in red stars with the uncertainty plotted as red line along the both side of the predicted mean. 174
- 6.20 Prediction of U components pertaining to four stars having index 19, 24, 42 and 49 where the predictions are made with the single layer GP based model using SQE kernel, given the training data. The true values are shown in blue circles while the predictive means are shown in red stars with the uncertainty plotted as red line along the both side of the predicted mean. 174

6.21	Trivial function and L values pertaining to the U component of the subset S of ten stars having index 2, 4, 9, 13, 21, 26, 30, 33, 34, 38; whose predictive results with the presented two models given the training data are presented in Figure 6.14 and Figure 6.15.	175
6.22	Trivial function and L values of V component of the subset S of ten stars (indexed as 2, 4, 9, 13, 21, 26, 30, 33, 34, 38), whose predictive results with the presented two models given the training data are presented in Figure 6.16 and Figure 6.17.	176
A.1	Results of forecast at the test points given the updated training data with the non-parametric model having SQE-looking kernel and the single layer GP based model having SQE kernel using one-step ahead strategy with the same model presented in the Chapter 4. <i>Top row, left panel:</i> Forecast at the test points, given updated training data with non-parametric model having SQE-looking kernel. <i>Top row, right panel:</i> Forecast at the test points, given updated training data with non-parametric model having SQE-looking kernel along with the prediction results at the previous five test points with the same model. <i>Bottom row, left panel:</i> Forecast at the test points, given updated training data with single layer GP based model having SQE kernel. <i>Bottom row, right panel:</i> Forecast at the test points, given updated training data with single layer GP based model having SQE kernel along with the prediction results at the previous five test points with the same model. True values are displayed with blue solid circle, forecasts are presented with red diamonds, while the predictions are depicted with red stars.	184
A.2	Results of re-learning hyperparameters with non-parametric model using SQE-looking kernel to perform forecast at test point T_0 given the updated Brent crude oil price data. The panels from left: the trace of ℓ that is updated in the first block, predicted ℓ using the inner GP at the end of second block, δ learnt in the second block, convergence diagnostics pertaining to the ℓ . The trace of the forecast at T_0	189
A.3	Forecasting at T_0 . From left: the prediction and forecasting at the test points plotted in a single plot showing the training data as well; the forecasting generated at T_0 ; forecasting made at T_0 along with predictions made at the previous five test points. True values are displayed with blue solid circle, training points are displayed in green triangle, forecasts are presented with red plus, while the predictions are depicted with red stars.	190

- A.4 Results of re-learning hyperparameters with non-parametric model using SQE-looking kernel to perform forecast at test point T_1 given the updated Brent crude oil price data. The panels from left: the trace of ℓ that is updated in the first block, predicted ℓ using the inner GP at the end of second block, δ learnt in the second block, convergence diagnostics pertaining to the ℓ . The trace of the forecast at T_1 190
- A.5 Forecasting at T_1 . From left: the prediction and forecasting at the test points plotted in a single plot showing the training data as well; the forecasting generated at T_0 and T_1 ; forecasting made at T_0 and T_1 along with predictions made at the last five test points. True values are displayed with blue solid circle, training points are displayed in green triangle, forecasts are presented with red plus, while the predictions are depicted with red stars. 191
- A.6 Results of re-learning hyperparameters with non-parametric model using SQE-looking kernel to perform forecast at test point T_2 given the updated Brent crude oil price data. The panels from left: the trace of ℓ that is updated in the first block, predicted ℓ using the inner GP at the end of second block, δ learnt in the second block, convergence diagnostics pertaining to the ℓ . The trace of the forecast at T_2 191
- A.7 Forecasting at T_2 . From left: the prediction and forecasting at the test points plotted in a single plot showing the training data as well; the forecasting generated at T_0 , T_1 and T_2 ; forecasting made at T_0 , T_1 and T_2 along with predictions made at the last five test points. True values are displayed with blue solid circle, training points are displayed in green triangle, forecasts are presented with red plus, while the predictions are depicted with red stars. . . 192
- A.8 Results of re-learning hyperparameters with non-parametric model using SQE-looking kernel to perform forecast at test point T_3 given the updated Brent crude oil price data. The panels from left: the trace of ℓ that is updated in the first block, predicted ℓ using the inner GP at the end of second block, δ learnt in the second block, convergence diagnostics pertaining to the ℓ . The trace of the forecast at T_3 193
- A.9 Forecasting at T_3 . From left: the prediction and forecasting at the test points plotted in a single plot showing the training data as well; the forecasting generated at T_0 , T_1 , T_2 and T_3 ; forecasting made at T_0 , T_1 , T_2 and T_3 along with predictions made at the last five test points. True values are displayed with blue solid circle, training points are displayed in green triangle, forecasts are presented with red plus, while the predictions are depicted with red stars. 193

- A.10 Results of re-learning hyperparameters with non-parametric model using SQE-looking kernel to perform forecast at test point T_4 given the updated Brent crude oil price data. The panels from left: the trace of ℓ that is updated in the first block, predicted ℓ using the inner GP at the end of second block, δ learnt in the second block, convergence diagnostics pertaining to the ℓ . The trace of the forecast at T_4 194
- A.11 Forecasting at T_4 . From left: the prediction and forecasting at the test points plotted in a single plot showing the training data as well; the forecasting generated at T_0, T_1, T_2, T_3 and T_4 ; forecasting made at T_0, T_1, T_2, T_3 and T_4 along with predictions made at the previous five test points. True values are displayed with blue solid circle, training points are displayed in green triangle, forecasts are presented with red plus, while the predictions are depicted with red stars. 194
- A.12 Results of re-learning hyperparameters with non-parametric model using SQE-looking kernel to perform forecast at test point T_5 given the updated Brent crude oil price data. The panels from left: the trace of ℓ that is updated in the first block, predicted ℓ using the inner GP at the end of second block, δ learnt in the second block, convergence diagnostics pertaining to the ℓ . The trace of the forecast at T_5 195
- A.13 Forecasting at T_5 . From left: the prediction and forecasting at the test points plotted in a single plot showing the training data as well; the forecasting generated at T_0, T_1, T_2, T_3, T_4 and T_5 ; forecasting made at T_0, T_1, T_2, T_3, T_4 and T_5 along with predictions made at the previous five test points. True values are displayed with blue solid circle, training points are displayed in green triangle, forecasts are presented with red plus, while the predictions are depicted with red stars. 195

List of tables

2.1	O-ring data	30
2.2	Different MCMC methods, configurations and observations (Correct value of $\alpha=15$, $\beta=-0.235$). α_0 is seed, <i>prop</i> denotes proposal, μ is mean, σ is std. deviation, <i>pri</i> denotes prior, <i>n</i> is iteration, AR is acceptance rate, C-A denotes Chain-A, NLR: need longer run, JS: Jump Scale, RW: Random Walk MH, IS: Independence Sampler, RWG: Random Walk MH within Gibbs, VMH: Vanilla MH. A block between double horizontal line denotes a chunk of analysis starting from one configuration. Figures for these results are presented in Appendix B.2	35
3.1	Prediction of output <i>W</i> at test inputs - with DNNs of varying architectures, and following GP-based learning of the length scale hyperparameter (ℓ) of the covariance kernel that parametrises the GP covariance structure. Below, “HL” stands for number of Hidden Layers in the DNN, and “N” for number of Neurons.	51
4.1	Table displaying comparative predictive performance and computational times relevant to the different models that I have predicted with. The first column depicts the names of the various models. The second column presents the number of hyperparameter(s) learnt using that model, (typically for each dimension of the input vector). The third column depicts the mean squared error (MSE) computed at the predicted means; the fourth column shows the MSE computed using the predicted 95% HPDs; the fifth and right-most column presents the overall time required to compute the predictions. This includes the time to convergence (achieved by the MCMC chain) and predictions made post convergence.	109

5.1	Learnt values of the unknown hyperparameters. For the hyperparameters ℓ_T , ℓ_H , ℓ_W , ℓ_D , “95% HPD” denotes the 95% HPD credible regions learnt in the first block of the chain. For the hyperparameters δ_T , δ_H , δ_W , δ_D , the 95% HPD credible regions are learnt in the second block. “Mean” denotes the mean of the 95% HPD credible regions of the learnt hyperparameters and “Predicted” depicts the predicted mean value of hyperparameters ℓ_T , ℓ_H , ℓ_W , ℓ_D at the end of second block.	121
5.2	Values of the parameters of the MCMC chain run given the multivariate Tetouan city data. Here σ_{prop} is the variance of the Normal proposal densities for the hyperparameters, μ_{prior} and σ_{prior} are the mean and variance of the Normal priors invoked for the hyperparameters respectively.	121
5.3	Predictive performance in terms of RMSE where the predictions are obtained using the non-parametric multivariate model with SQE-looking kernel, given the multivariate training data on the four test sets “Test-1”, “Test-2”, “Test-3”, “Test-4”. Top row depicts the root mean square error (RMSE) computed at the mean predicted value, while the bottom row shows the RMSE computed using the predicted 95% HPDs.	124
5.4	Performance of experiments with various DNN architectures to predict oil prices at test set “Test-1”, given dataset D_1 with prediction performance in terms of MSE, ‘CI’ denotes the 95% confidence interval computed from the post burn in trace of loss, μ_c is the mean of the 95% CI, μ is the mean of the entire trace of loss.	135
5.5	Performance of experiments with various DNN architectures to predict integrated power consumption at test set “Test-1”, given dataset D_2 , with prediction performance presented in terms of MSE, ‘CI’ denotes the 95% CI computed from the post burn in trace of loss, μ_c is the mean of the 95% CI, μ is the mean of the entire trace of loss, ‘LR’ denotes the learning rate used and tried as well (for some cases).	137
5.6	Predictive performance using data D_3 with various DNN architectures that are utilised to predict integrated power consumption with prediction performance in terms of RMSE. ‘CI’ denotes the 95% CI computed from the post burn in trace of loss, μ_c is the mean of the 95% CI, μ is the mean of the entire trace of loss.	141

5.7	Predictive performances with the three best DNN models and the single layer GP based model, where I have used data D_1 and D_2 with four test sets. For GP based model, RMSE denotes RMSE computed at the predicted means; the RMSE interval shows the RMSE computed using the predicted 95% HPDs.	142
6.1	95% HPD credible region on the learnt hyperparameters using non-parametric model with SQE-looking kernel having various lookback size.	164
6.2	Comparative predictive performance using MSE (over all the predicted values at the test points) for each star with the two models.	169
6.3	Comparative predictive performance using compliance score (in %) for each test data with the two models.	169
A.1	Values of the parameters of the MCMC chain run given the oil price data for forecasting. Where σ_{prop} is the variance of the Normal proposal densities for the hyperparameters, μ_{prior} and σ_{prior} are the mean and variance of the Normal priors invoked to the hyperparameters respectively.	196
A.2	tab forecast all test	196

Chapter 1

Introduction

1.1 Introduction

The human brain possesses the innate ability to acquire facts; skills; knowledge; concepts at several hierarchy; amongst others. Its learning capacity has intrigued and inspired us enough, to attempt replicating this learning in computers, giving rise to Artificial Intelligence, and Mechanistic Learning (Carbonell et al., 1983). The learning techniques that fall within the broader umbrage of machine learning can be of several types including: (i) supervised learning (Muhammad and Yan, 2015) - where the interest is in learning the functional relationship between two variables using supervision or guidance from data (observations, measurements, etc.) on the pair of associated variables, in order to perform tasks such as prediction, forecast etc; (ii) unsupervised learning - when there is no supervision from extant data or past data (Murphy, 2012); (iii) reinforcement learning - when the learning takes suitable action to maximize reward in a particular situation (Kaelbling et al., 1996; Sutton and Barto, 2018).

Prediction task within the supervised learning setup are widely undertaken these days with the advancement of Machine Learning (ML) techniques, in various domains such as finance, health, material science, manufacturing, transport etc. In such supervised learning setup, the functional relationship between the response (output) and predictor (input) variables is sought. For accurate prediction, the sought function needs to be learnt reliably.

Current ML literature includes several supervised learning models, such as Deep Neural Networks (DNN) which are fast, can handle large datasets, produce good predictive performance when a DNN model is chosen with trial-and-error from the repertoire of infinitely many possible models and hence quite popular; yet they are less-reliable, lack explainability, yield results that are point-estimates and sometimes completely wrong Cremer (2021); Waldrop (2019). On the other hand, Probabilistic Machine Learning (PML) Ghahramani

(2015); Murphy (2012) models, which have theoretical solid foundations, are reliable, better-explainable, yield uncertainty-included results, yet computationally intensive.

This thesis aims to achieve reliable and fast learning for prediction (especially regression task) using techniques pertaining to PML and Computational Statistics, given small-to-medium sized data ($\sim 10^4$) in various applications in the domain of finance, energy consumption and astrophysics. The learning task has been performed in these applications using outputs of type scalar-variate and matrix-variate while having typically vector-variate inputs.

The learning of the inter-variable functional relation can be done either through parametric modelling or non-parametric modelling. In parametric modelling Ghahramani (2015), a chosen parametric form is assigned to the sought function, where this form is parametrised by a finite number of parameters. Some examples of such parametric forms of the function include linear, quadratic, logistic, etc. Values of the parameters of the chosen parametric model of the sought function, are then learnt or estimated or optimised, using various techniques given the data, where information from the training data is absorbed by these model parameters. The learnt values of these parameters are then used to obtain predictive output (of the sought function) at new "test" values of the input variable (to this function). Thus the model parameter learning is followed by prediction/forecast - which is the basic aim behind the learning of the sought function that represents the relation between two variables.

Some examples of parametric supervised learning include linear and logistic regression of a response variable, against a finite number of covariates; neural networks with a fixed number of hidden units, etc. However, the problem with this approach is that a chosen model can be misspecified, as far the training data is concerned. Model misspecification is an accepted aspect in domains like econometrics Hansen and Sargent (2001) with the misspecification meant in the sense that something like q covariates are needed for a good fit but p have been used, or a linear model has been wrongly invoked for the data with a non-linear model seemingly better as per some error parametrisation.

So, parametric learning of functions is very limited in scope and accuracy. Irrespective of improvements in quantity and/or quality of training data, if the sought functional relation is assigned a parametric form - parameters of which we may learn from the data - fitting does not improve if the choice of that parametric form was misplaced given the originally-available training set. For example, if motivated by no other reason but computational simplicity, one chooses a linear form for the function parameters of which are to be learnt, then no matter how superior the training set may update to, the quality of the fit will remain poor, if originally poor. Thus, this not learning of the function. This is a gratuitous example of the pitfalls of parametric modelling of a sought function and it reflects the fundamental inability of a parametric form to carry information about the correlation structure in the data onto the

output variable. It may be argued that (1) the parametric form can be chosen well to begin with, and (2) that piecewise fitting of parametric functions can resolve any worries. The argument against the first point is that it is hard to identify an “optimal” parametric form, especially in high-dimensions, and in real-world situations. The argument against the 2nd point is that again this is hard to undertake in high-dimensions, and importantly, renders forecasting using such models a defeated endeavour, while prediction could as well be badly affected.

On the other hand, non-parametric modelling Xuan et al. (2019) does not assume any functional form for the sought function. The model is defined with infinite dimensional parameters - or rather as a function - with the "parameters" referred to before, being the functional values at chosen values of the domain variable(s). The amount of information absorbed by the parameters grows as amount of data grows. Hence, the smoothness of the function is updated as the training data expands. Non-parametric models require access to all the training data when making predictions. A set of characteristics that often considered while performing non-parametric learning, is as follows.

1. Similar points in input space mean similar points in output space.
2. Training data points very close to a test point will contribute to the prediction.

However, as modelling strategies evolve to cleverer ones, one or multiple of the above is/are beginning to be challenged. Non-parametric models, which employ distance measure to carry correlation information, are flexible; provide good models for data from real-world contexts; and give better prediction performance. Some tools for such learning include Gaussian Process, k-nearest neighbour classifier, Support Vector Machines.

However, classical machine learning always gives a single value for the prediction, forecast or any learning task performed, without associating any uncertainties, which often fails to address some important aspects as follows.

1. Multiple models can be consistent with the data (Der Kiureghian and Ditlevsen, 2009) and the data itself can include uncertainties due to measurement errors. There could be further complications owing to hidden variables that cannot be addressed just by gathering more data under the same data collection set up.
2. The prediction of new outputs using the learnt functional relationship, is also uncertain, where such uncertainties can be interpreted to reflect on limitations of the model; misrepresentative training set; noise in the test inputs; and possibility of the input-output relationship to be non-bijective. Hence, the uncertainty quantification has enormous significance, and reliable learning of the uncertainties in the learning of the

sought function itself, is crucial, especially in sensitive domains (Begoli et al., 2019) and when the available training data is small and/or expensive to procure. Examples of sensitive domains include medicine (Begoli et al., 2019); transport systems (Zhu and Laptev, 2017); information security (Smith et al., 2011), etc.

The probabilistic framework on the other hand, is quite capable of objectively learning and representing uncertainty (Jaynes, 2003) about models and predictions. So, probabilistic machine learning Ghahramani (2015) provides the framework for undertaking inference on models that explain the data under consideration, while also quantifying uncertainties objectively.

The probabilistic approach to modelling within Bayesian paradigm, consists of representing the sought function as a random function and all the uncertain unobserved quantities in a model - along with their relationship with the noisy (or noise-free data) data - expressed as random and therefore assigned a probability distribution. Bayesian approach provides a way to incorporate prior information on unknowns to obtain the joint posterior of the unknowns, given the data, and then inferring upon the unknowns using the joint posterior would yield the marginal posteriors of the unknowns with a comprehensive quantification of uncertainties. Thus Bayesian modelling, allows for reliable and uncertainty-included learning while rendering scope for invoking prior knowledge on the unknowns, given the observed data using probability theory.

In Bayesian paradigm, a random function takes a form with a certain probability given by the probability distribution of the random function. It is to be noted that probability distribution on the space of function is nothing but a stochastic process. Therefore, modelling the sought function as random function, actually means that the sought function is a random realisation from a stochastic process. In this thesis, I have chosen this stochastic process to be Gaussian Process (GP) due to the ease of computation in high dimensions and the ability to generalise across the multiple dimensions.

One of the crucial aspects of real-world data is that the correlation structure of the outputs realised at inputs that lie within a subspace of the input space could be different from another subspace of the input space. This inhomogeneity in the correlation structure poses significant challenge towards the reliability of prediction because, if the correlation structure present in the data is not reflected truly by the covariance function of the underlying GP, (which is used to model the sought function), then reliability of the prediction will be a concern. To achieve such reliability in prediction, the covariance function of the underlying GP needs to be constructed such that it is truly non-stationary. While the covariance function of the GP is non-stationary, it also needs to be model-free and non-parametric so that the construction is not only limited to a specific data that has emerged from a specific domain of application.

The literature on spatial statistics includes a repertoire of hierarchical models to achieve regression (Banerjee et al., 2014; Gelfand, 2012; Gelfand and Banerjee, 2017). These hierarchical models typically make assumptions about the distribution of the output variable in the first stage, where the parameters of the first stage of the model are modelled using stationary Gaussian Processes (GPs) in the second stage, where hyperpriors are specified on the parameters of the second stage by considering them as constant unknowns. The assumption of stationary GP in the second stage is not justified and it is also unclear why one should not add more layers of GPs to continue to model the hyperparameters of the previous layer of GPs. Also, the learning of the covariance structure of the GP in the second stage remains unclear, such as in regard to the specifics of the used training data as well as justification in using the empirical training data at this stage as well. Furthermore, we might question the arbitrary fixing of the distributions of the hyperpriors in the later stage of the modelling, in addition to the sensitivity of the results to such choices. We are not entirely sure how easily these choices generalise to other datasets. The hierarchical modelling not only involves GPs, but also other processes such as in the Poisson-Gamma model (Chapter-6, Banerjee et al. (2014)).

Although the literature of spatial statistics includes multiple (hierarchical) models, neither is parsimony of the models clear, nor is such parsimony of one model compared to that of another. Also, the literature does not mention how these spatial models can be used for non-spatial domains, nor the models are readily extendable to the learning of high-dimensional functions.

1.2 Objectives of this thesis

Objectives of this thesis are delineated as follows.

1. **Generic reliable learning given real-world data, across dimensions:** The primary aim of this thesis is to illustrate a universal probabilistic supervised learning method, for regression tasks. This demonstrated learning is generic and applicable to diverse applications given real-world data; uses two nested layers of Gaussian Processes; learns inter-variable relationships that are inhomogeneously correlated, and are identified without imposing any parametric form; learning the covariance structure of the stochastic process that generates such relationships, without imposing a parametric form to it. Additionally, the illustrated learning is such that it demands fewer parameters to be learnt, compared to existing attempts. This thesis also provides an easy-to-implement algorithm of these methodologies so that anyone from an application domain can easily

implement and use these learning models. Thus, the thesis illustrates learning techniques that have been identified recently (Noack et al., 2024a) as relevant to supervised learning techniques for regression.

2. **Comparative studies:** The illustrated learning methods are showcased in comparison to existing probabilistic machine learning methods - bringing out the comparatively higher reliability of the predictions by the demonstrated methods, is another important undertaking of this thesis. The aim is also to compare predictive performance with two-staged fully non-parametric model (developed in Chapter 4) against single layer GP based model. The comparative studies highlight the strength, weaknesses and sensitivity of the learning demonstrated in this thesis, over existing models, when learning is faced with the challenges of real-world data.
3. **Fast and reliable high-dimensional learning in presence of inhomogeneous correlation structure:** Of the challenges offered by the data, the chief difficulty is data that bears inhomogeneous correlation structure, i.e. data that compels the sought inter-variable function to bear an inhomogeneous correlation structure. The significant objective of this thesis is to demonstrate the fast and reliable learning of such a non-stationary high-dimensional function, given such inhomogeneously correlated data, using Markov Chain Monte Carlo (MCMC) based inference.
4. **Robust Bayesian inference across dimensions:** An objective of this thesis is to perform robust MCMC-based inference on the unknown model parameters given (generally high-dimensional) data, in order to reliably learn values of these unknown parameters with uncertainties.
5. **Efficient uncertainty percolation with constrained inference:** I have also studied the effect of various types of uncertainty propagation in a multi-staged learning with constrained MCMC to indicate robust uncertainty propagation. As, a poorly chosen representation of the uncertainty can incur erroneous learning of the sought function and thus yield incorrect predictions.

1.3 Achieving the objectives of this thesis

The aforesaid objectives are achieved through various chapters as follows.

To achieve the 1st objective stated above in Section 1.2, I undertake the following. Learning the non-stationary sought function in a fully non-parametric manner which is suitable for generic applications requiring regression task, has been achieved by developing

two-staged probabilistic learning methodologies that are reliable. The genericness of this universal learning method is achieved by constructing model-free covariance structure of the underlying GP which mimics the covariance structure present in the data. This developed method is fast due to the employment of closed-form prediction, and learning minimal number of hyperparameters (i.e. parsimonious) in contrast to the existing methods in the literature which require huge number of learning of the hyperparameters for data with high dimensions. The parsimony of the method is achieved via invoking system ergodicity (Cohen, 1996; Peebles Jr, 2001; Peters and Gell-Mann, 2014, amongst others). Chapter 4 presents these methodologies in details along with detailed algorithm and flow chart for easy depiction.

To achieve the 2nd objective stated above in Section 1.2, I undertake the following. I have implemented non-stationary kernels developed by Remes et al. (2017) and Paciorek and Schervish (2003) along with other existing stationary models in order to compare their predictive performance against the proposed parsimonious, fully non-parametric kernel presented in Chapter 4. Chapter 4 presents comparative results of predictions from eight models given univariate real-world data.

Chapter 3, 4, 5 and 6 presents comparative study with the proposed models against the relevant existing models and demonstrate the superiority of the proposed models. Chapter 5 includes the comparative learning and predictive results to depict the fact that more the inhomogeneous correlation structure of the empirical data, harder the task of prediction becomes. Apart from comparing relevant models, this thesis also compares the predictive performance of hierarchical GP-based model with a single layer GP based models in the learning of univariate and high-dimensional sought functions in chapter 4 and 6.

Chapter 5 also includes comparative studies amongst several DNN based models to demonstrate its sensitivity towards its architecture and model parameters. This chapter also attempts to select DNN models based on the loss values obtained from the optimizer beyond trial-and-error strategy.

To achieve the 3rd objective stated above in Section 1.2, I undertake the following. The comprehensiveness of the proposed universal method (in chapter 4) allows learning across dimensions with data that is differently sourced (multiple subsets of the training data comes from different sources) and being inhomogeneously correlated. This method, also avoids the choosing of particular covariance structure from the collection of available covariance functions in the literature, to accomplish learning of the sought function - which is a key element of reliable learning in high-dimensions. Because, any imposed parametric covariance structure would incur erroneous learning, especially in high-dimensions and thereby yielding incorrect high-dimensional predictions. Chapter 6 presents an application of high-dimensional learning in fully non-parametric and parsimonious manner where a matrix

variate is predicted given a vector input. The MCMC inference is achieved on 3-dimensional tensor variate complex posterior. Computational challenges have been addressed using stable and efficient computations, parallel processing and ensuring maximal usage of processing units.

To achieve the 4th objective stated above in Section 1.2, I undertake inference with MCMC methods, i.e. all inference is performed by generating posterior-samples using MCMC, given the relevant data. MCMC inference techniques have been used in this thesis to learn the uncertainty-included value of unknown parameters. MCMC-based inference is discussed in Section 1.3.1. Why I have chosen MCMC over other techniques, such as INLA, has been discussed in Section 1.3.2. Chapter 2 presents an introductory study of various types of MCMC along with empirical illustrations.

1.3.1 MCMC-based inference

one of the popular category of inferencing techniques used within the Bayesian learning approach which provides reliable learning (Geyer, 2011; Qian et al., 2003; Simpson et al., 2021) while the original idea is developed by Metropolis et al. (1953). MCMC is capable of learning the marginal posterior density function of the the unknown hyperparameters even when the hyperparameters are correlated in high dimensions (van de Schoot et al., 2021).

1.3.2 Why I preferred to avoid INLA

I preferred rather not to use more recent technique such as Integrated nested Laplace approximations (INLA) which is originally developed by (Rue et al., 2009). As depicted in van de Schoot et al. (2021), the INLA is “An approximate approach developed for the large class of latent Gaussian models, which includes generalized additive spline models, Gaussian Markov processes and random fields”. Moreover, the very use of Laplace approximations, that too in multilevel manner, invokes few worries associated with Laplace’s approximation, as mentioned by Christian P. Robert (Universite Paris Dauphine and Centre de Recherche en Economie et Statistique, Malakoff) in the discussion of the original paper of INLA by (Rue et al., 2009). The worries include: Laplace’s approximation not being parametrization invariant, requiring some black box numerical differentiation and most importantly it is devoid of any clear evaluation of the associated error. Also “Using and comparing different approximations that are all based on the same principle (Laplace’s!) does not provide a clear indicator of the performances of those approximations”. Another literature (Banerjee et al., 2014) criticises INLA as the “overall inference is limited and application to more challenging multi-level models may be difficult” in Chapter-12. So, it is unclear to me whether

INLA would yield correct estimates of the hyperparameters in learning of high-dimensional functions. As my model is generic for any domain (including precision-sensitive and safety-critical applications) and suitable for univariate, multivariate and high-dimensional learning, hence I could not rely on the approximations induced learning and thus I used traditional, reliable MCMC based inference.

To achieve the 5th objective stated above in Section 1.2, I undertake the following. The efficiency in propagating uncertainty has been established by undertaking a comparative analysis to illustrate the relative superiority of the predictions. Chapter 3 depicts this comparative analysis and shows how the comprehensive percolation of uncertainties could yield better and correct predictions than when the uncertainties are inefficiently propagated.

1.4 Novelty of this thesis work

Novelty of the contributions of this thesis are multifold. The contributions of this thesis are delineated as below.

- Existing learning models use two layers of GPs, where the outer GP is non-stationary, with the hyperparameters of its covariance function modelled using assumed-to-be-stationary GPs in the inner layer (thus incurring only two layers of GPs instead of adding more layers). However, the fully non-parametric learning model demonstrated in this thesis, includes the proven sufficiency of only two layers of GPs. While the existing models use two layers without any justification, based on an unjustified assumption, sufficiency of the number of layers of the invoked GPs needs to be addressed. This is addressed in this thesis (see Sections 4.1 to 4.4 in Chapter 4).
- The correlation function of each GP in the inner layer is not learnt in the existing relevant models. Instead, it is either specified by hand or hyperpriors are assigned through fixed distributions - these ad hoc approaches remain questionable. However, the models illustrated in this thesis learn the correlation function of each inner GP by generating datasets relevant for each such GP (see Section 4.4.1 in Chapter 4).
- Another novelty is the parsimony of the illustrated model, achieved by using the system ergodicity. This parsimonious model requires minimal learning of hyperparameters in comparison with the existing non-stationary models or spatial models - thus making it fast and suitable for diverse applications including safely-critical, precision-sensitive applications (see computational complexity of applications presented in Chapter 4, Chapter 5 and Chapter 6).

- The model depicted in this thesis learns non-stationary functions marked by inhomogeneous correlation structure, given real-world data (see Chapter 4, Chapter 5 and Chapter 6).
- The proposed model is generalisable across dimensions and this universal learning methodologies are illustrated in learning univariate, multivariate and high-dimensional functions along with mitigating computational challenges (see applications in Chapter 4, Chapter 5 and Chapter 6).
- A preliminary attempt has been made to quantify the inhomogeneous correlation structure and using which explainability of predictions are attempted in high-dimensional learning (see Section 5.2 in Chapter 5 and Section 6.4.1 in Chapter 6).
- Several comparative studies have been made with existing probabilistic machine learning models and machine learning models such as Deep Neural Networks (DNNs). The experiments depict that both types of predictive models suffer from the inhomogeneous correlation structure of the given data. The DNN exposition also shows that it worked well for medium-sized data when the inhomogeneity present in the correlation structure of the data is not significantly high; whereas they are very sensitive to the architecture and some model parameters. This thesis also attempts for DNN model selection beyond trial-and-error strategy (see Section 4.7.11 in Chapter 4, Section 5.4 in Chapter 5, Section 6.4.1 in Chapter 6).
- Showing the importance and significance of efficient propagation of uncertainties for producing correct results is another novelty of this thesis (see Chapter 3).

1.5 Publications out of this thesis

This thesis has already yielded two peer-reviewed international conference publications, along with one journal paper. Additionally, I have submitted a manuscript to a reputed journal within machine learning domain, and two further journal manuscripts are currently being written towards submission. The detailed publications are as follows.

Conference publications:

1. Gargi Roy, Kane Warrior, and Dalia Chakrabarty. “Covariance Kernel Learning Schemes for Gaussian Process Based Prediction Using Markov Chain Monte Carlo.” EPIA Conference on Artificial Intelligence. Cham: Springer Nature Switzerland, 2023.

2. Gargi Roy, and Dalia Chakrabarty. “Efficient Uncertainty Quantification for Under-Constraint Prediction Following Learning Using MCMC.” International Conference on Neural Information Processing. Singapore: Springer Nature Singapore, 2022. This work is included in Chapter 3.

Publications under review:

1. A New Reliable & Parsimonious Learning Strategy Comprising Two Layers of Gaussian Processes, to Address Inhomogeneous Empirical Correlation Structures - Gargi Roy, Dalia Chakrabarty. Chapter 4 consists of this contribution.

Publications under preparation:

1. A manuscript titled “Inhomogeneously Correlated Data - its Impact on Quality and Reliability of Predictions Obtained from Deep Neural Networks and Gaussian Process” is under preparation and will be submitted to a peer-reviewed journal, tentatively by the end of March 2025. Contents from Chapter 5 contributes to this manuscript.
2. A manuscript on the non-parametric high-dimensional, parsimonious learning is under preparation which will be submitted to a peer-reviewed journal, tentatively by the end of April 2025. Chapter 6 includes this contribution.

1.6 Thesis outline

The outline of the thesis is as follows. The related state-of-the art literature pertaining to each chapter is included in the chapter itself. While the first chapter is introductory to this thesis, the whole second chapter (Chapter 2) and the first two sections of the third chapter (Chapter 3) consist of background materials; the rest of the third chapter, fourth (Chapter 4), fifth (Chapter 5) and sixth (Chapter 6) chapters include the contributions of this thesis.

Chapter 2 presents discussions on various MCMC methods along with an empirical illustration of parametric learning on a small real-world dataset. I undertook this work early in my Ph.D to hone my skills with Bayesian inference using MCMC.

Chapter 3 presents preliminaries of Gaussian Process based learning along with an application of GP-based supervised learning, in various situations, and it includes an efficient uncertainty quantification. This is relevant since it has been shown that many aspects of learning and artificial intelligence implementation pertaining to learning of the sought function, depend crucially on the careful uncertainty quantification.

Chapter 4 presents the universal method for a model-free, non-parametric, non-stationary learning to mimic the empirical correlation structure present in the empirical training data along with the required proofs, protocols for inference, detailed algorithms for implementation, an illustration on univariate real-word data that bears inhomogeneity in its correlation structure along with comparative results obtained using several relevant models from literature. This chapter also presents results from using Deep Neural Networks (DNNs) on the same data that has been used for illustration within the chapter using all the model.

In **Chapter 5**, the following are presented.

1. The notion of inhomogeneity in the correlation structure of a data is presented along with an inhomogeneity parameter that pre-exists in the literature, to compare multiple inhomogeneously correlated data.
2. An illustration of the universal method for a model-free, non-parametric, non-stationary learning with multivariate data which is highly inhomogeneously correlated.
3. An extensive set of experiments involving DNNs and GP based learning is presented in order to depict that the predictions are hard in the inhomogeneously-correlated data in comparison to another data that bears less inhomogeneous correlation. The DNN exposition also depicts the reliability issue and the sensitivity of the predictive performance of DNNs on its architecture. An attempt has been made in this thesis for model selection based on the convergence of the optimisation within the DNN, in addition to the predictive performance.

Chapter 6 presents an illustration of the universal method for a model-free, non-parametric, non-stationary learning with high-dimensional data where the input to the sought function is a vector and the output is a matrix-valued variable. This high-dimensional data is highly inhomogeneous and an attempt has been made to explain the predictions using the notion of inhomogeneity presented in Chapter 5. This illustration is also challenging from the computational perspective to implement the model-free, non-parametric model and inference for learning the high-dimensional sought function. This chapter includes a discussion on the computational aspects of the high-dimensional learning to achieve reliable, fast, accurate, with-uncertainty predictions.

Finally this thesis is concluded in **Chapter 7** with limitations and future work.

Chapter 2

Bayesian Inference using Markov Chain Monte Carlo (MCMC)

2.1 Introduction

This chapter introduces the fundamentals of Markov Chain Monte Carlo (MCMC) inferencing techniques that are used for learning purpose in Bayesian models, (i.e. in probabilistic machine learning). The chapter includes motivation, some basic related algorithms, along with an illustrative application of such algorithms on a small yet real-world data set. This empirical illustration discusses learning of model parameters, within a parametric learning setup. The contribution of this chapter is in putting forward the motivation behind MCMC techniques; gather several MCMC methods from the literature; and present the corresponding computational algorithms, along with the practical illustration on the real-world data set mentioned above.

The chapter is organised as follows. Section 2.2 presents the literature on MCMC. The core motivation behind using MCMC is presented in Section 2.3. Then Section 2.4 includes some fundamentals relevant to the MCMC techniques. Various flavours of MCMC techniques along with the detailed algorithm for implementation are depicted in Section 2.5. Application of these MCMC techniques to learn the parameters in a real-world set up with a univariate data, is presented in Section 2.6. Finally the chapter is concluded in Section 2.7.

2.2 MCMC literature

There exists a vast literature on MCMC methods including many surveys (Changye and Robert, 2020a; Luengo et al., 2020; Roberts and Rosenthal, 1998) depicting several MCMC

methods with its associated theories. Markov chain Monte Carlo methods produce a sequence of simulations $(x_t)_t$ whose limiting distribution is the target distribution $\pi(\cdot)$. Some advantages include, 1) this method needs less information on $\pi(\cdot)$, it is enough to be able to compute the value of $\pi(x)$ at a value x ; 2) high dimensionality can be handled by decomposing into conditional of smaller dimension (as in Gibbs sampling). Some disadvantages include, 1) the samples are correlated, hence they are less informative than i.i.d. samples; 2) the convergence may be slow to achieve. Apart from the Metropolis-Hastings algorithm which is fundamental MCMC method, Random Walk Metropolis-Hastings (Metropolis et al., 1953) is another extended method when the proposal distribution is symmetric. In independent sampler (Tierney, 1994), the proposal distribution has a fixed mean so that the proposed sample is independent of the current sample. In Langevin algorithm the proposal distribution is motivated by the approximating continuous-time Langevin diffusion with the target distribution as stationary distribution (Grenander and Miller, 1994; Neal, 1993; Rossky et al., 1978), for example, when the Normal distribution, which is used as a proposal distribution, does not use the current sample as its mean, then it becomes the Langevin version (Robert and Changye, 2020). Vanilla Metropolis-Hastings algorithm uses a truncated Normal as proposal (Robert and Casella, 2010).

Gibbs sampler (Gelfand and Smith, 1990; Geman and Geman, 1984) is another kind of popular MCMC algorithms. The Gibbs sampler was introduced by Stuart Geman and Donald Geman in 1984 in order to sample from the Markov Random Field (MRF) induced by the Gibbs distribution (Geman and Geman, 1984). This method was designed to break the curse of dimensionality by producing a sequence of low dimension simulations that still converge to the right target distribution, however, it may take longer time to converge. Although, Gelman (1993) showed that the Gibbs sampler is a particular case of the Metropolis-Hastings algorithm.

The Gibbs sampler requires sampling from the full univariate conditional PDFs. Unfortunately, although this should be a much easier task than sampling from the multi-variate posterior PDF, in many real-world applications, these conditional PDFs have non-standard forms and we cannot sample directly from them. To address the problem, another kind of MCMC technique has been introduced, which uses Metropolis-Hastings algorithms within Gibbs sampler (Geweke and Tanizaki, 1999, 2001) resulting in a hybrid sampler (Robert and Casella, 2005).

Another type of MCMC method is a gradient-based MCMC technique which uses the gradient of the log-posterior to enhance the efficiency of the sampling method. The proposal distribution is formed by using the gradient, due to which longer jumps are possible without pushing the acceptance ratio of the method to a low value. Following are the two kinds of

gradient-based MCMC methods. The Metropolis adjusted Langevin algorithm (MALA) is an MH algorithm (Girolami and Calderhead, 2011; Roberts and Stramer, 2002) which uses a stochastic differential equation (SDE) to form the proposal distribution. In the Hamiltonian Monte Carlo (HMC) or the hybrid Monte Carlo (HMC) method (Brooks et al., 2011; Duane et al., 1987), a statistical physical simulation of a physical system is used to form the proposal distribution for the MH algorithm.

Metropolis-Hastings algorithms use arbitrary proposal density, however, in practical run of the algorithm, the choice of proposal density should be wise enough so that we obtain independent-enough samples which have good acceptance rate along with the good approximation of the target distribution in a reasonable time. As the manual tuning of the proposal distribution can be a time intensive work, adaptive MCMC methods enhance the algorithm's performance automatically, by adapting the proposal distribution based on the already seen samples. Giordani and Kohn (2010); Haario et al. (2006, 2001); Luengo and Martino (2013); Mbalawata et al. (2015) presents a wide range of adaptive MCMC techniques.

2.3 Motivation

Over the past few decades, there has been a surge of high volume of data being made available. Thankfully, this rise has been backed by technological advancement in collection and storage of such data, along with the ability to analyse the data using increasingly more powerful and faster machines. Many of the scientific analyses involve making inference on the model of the system under study using the observed data. In general, when I learn the model of a function, $\mathbf{Y} = f(\mathbf{X})$, where this function might be expressing relationship between variables $\mathbf{X}, \mathbf{Y}$, there could be parameters in that sought function, and I would like to identify these parameters. Such parameters are denoted by the notation $\boldsymbol{\theta}$ where $\boldsymbol{\theta} \in \mathbb{R}^d$.

Within a probabilistic approach, $\boldsymbol{\theta}$ is learnt, given the data $data = \{(x_i, y_i)\}_{i=1}^n$. The identification of the parameters $\boldsymbol{\theta} = \{\theta_1, \theta_2, \dots, \theta_p\}$ can be attempted by formulating an objective function, $L(\{\theta_1, \theta_2, \dots, \theta_p\} | \{(x_i, y_i)\}_{i=1}^n)$. One can maximise this function in principle (or maximise $-L(\cdot | \cdot)$ if minimum of the objective is required). The likelihood function is an example of such an objective function. It yields the probability of the data given the unknown parameters i.e. $Pr(Data | \{\theta_1, \theta_2, \dots, \theta_p\})$, which is effectively $Pr(Data | Model)$. Now this likelihood function can be maximised by scanning over all possible values of the unknown parameters $\{\theta_1, \theta_2, \dots, \theta_p\}$. Such an optimisation task indicates a fundamental problem of conditioning on the unknown parameters rather than the data at hand which is simpler than optimising $L(\{\theta_1, \theta_2, \dots, \theta_p\} | Data)$. This is simple because when I condition on something,

I assume that I know the "entity" which I am conditioning upon i.e. an "event" has already occurred and based on that I try to find the probability of another "event". However, when I already have the data and try to find the model, so one should condition upon the data, because that is already at hand - analogous to the "event" that has already occurred, such that the model is the "event", whose probability I am trying to find.

So, the idea is that once I have a model - for example a model of the relationship between an input and an output variable - I can churn out values of the output at given inputs. But the task of learning the model of the relationship between the two variables, given a set of values of the output variable, at chosen values of the input, is much harder. In fact, this is the point of supervised learning, which is highly non-trivial. Indeed, in general it is true that learning the model given data, is many times harder than churning out data given a model. so computing the likelihood of the model parameters given the data - which as is known from the definition of the likelihood, is probability of data given the model (parameters) - is easy, or rather, easier, than the computing the probability of the model (parameters) given the data.

This sought probability of the model - given the data - can be achieved in Bayesian statistics, since Bayes' rule allows us to compute the sought probability of the model (parameters) given the data, using the simpler-to-compute likelihood i.e. the probability of observables given the model. The sought probability of the model given the data demands prior probabilities on the unknown parameters in addition.

Bayes' rule: I now present Bayes' rule which can be directly derived using the definition of conditional probability and be used to estimate the model I seek. When A, B are two events, denote by $Pr(A), Pr(B)$ probability of event A, B respectively (marginal or prior probability), $Pr(A|B)$ probability of condition of occurrence of event A while event B has already occurred (conditional probability), $Pr(B|A)$ is vice versa and $Pr(A \cap B)$ probability of event A and event B (joint probability). Conditional probability gives the following.

$$Pr(A|B) = \frac{Pr(A \cap B)}{Pr(B)}, \quad Pr(B|A) = \frac{Pr(B \cap A)}{Pr(A)}, \quad Pr(B) \neq 0, Pr(A) \neq 0 \quad (2.3.1)$$

Now, as $Pr(A \cap B) = Pr(B \cap A)$, I get the following.

$$Pr(A|B)Pr(B) = Pr(B|A)Pr(A) \implies Pr(A|B) = \frac{Pr(B|A)Pr(A)}{Pr(B)} \quad (2.3.2)$$

In our context, the equation 2.3.2 becomes the following.

$$Pr(Model|Data) = \frac{Pr(Data|Model) \cdot Pr(Model)}{Pr(Data)} \quad (2.3.3)$$

In high-dimensional cases, the $Pr(data)$ is difficult to compute. Using the law of total probability, I get the following.

$$Pr(Data) = Pr(Data|Model_1) \cdot Pr(Model_1) + Pr(Data|Model_2) \cdot Pr(Model_2) + \dots + Pr(Data|Model_n) \cdot Pr(Model_n) \quad (2.3.4)$$

Applying equation 2.3.4 in equation 2.3.3, I get,

$$Pr(Model|Data) = \frac{Pr(Data|Model) \cdot Pr(Model)}{Pr(Data|Model) \cdot Pr(Model) + Pr(Data|Model^c) \cdot Pr(Model^c)} \quad (2.3.5)$$

Due to the lack of model dependence of the denominator in the equation 2.3.5, I have the following proportionality (Robert and Casella, 2010).

$$Pr(Model|Data) \propto Pr(Data|Model) \cdot Pr(Model) \quad (2.3.6)$$

Bayesian inference: Using Bayes' rule, I first formulate the posterior probability density function (pdf) i.e. $\pi(\boldsymbol{\theta}|data)$, and then I want to infer values of these parameters given $data = \{(x_i, y_i)\}_{i=1}^n$. I, not only want to infer just a particular value of the parameter $\theta_j \in \boldsymbol{\theta}$, where $\boldsymbol{\theta} = (\theta_1, \dots, \theta_p)^T$, and $j = 1, \dots, p$ but prefer to learn the uncertainties in the values of the θ_j that are found compatible with the data, $\forall j = 1, \dots, p$. This is done by sampling values of θ_j given the data from the marginal posterior pdf of θ_j given the data. Sampling implies generating the set of values $\{\theta_j\}_{j=1}^p$ from the posterior pdf of θ_j , given the data at hand, such that the frequency distribution of the sample is directly proportional to the posterior pdf of this variable. So, I will first have to compute the marginal posterior pdf of θ_j from the joint posterior pdf of all the parameters (components of vector $\boldsymbol{\theta}$) given the data, i.e. $\pi(\theta_1, \dots, \theta_p|Data)$ and then I will need to formulate a solution to the key issue which is how do I generate this sample, because choosing values randomly will not serve the purpose - after all - the marginal posterior probability distribution of θ_j is not uniform in general. Hence, the sampling method is designed in a way which makes it more likely that I choose the value θ of θ_j that occurs at a higher probability than a value that occurs at lower posterior value, given the marginal posterior of θ_j . that way, the frequency distribution of the generated sample will emulate the marginal posterior pdf of θ_j given the data - up to a scale. In essence, the sampling needs to be done in a way so that the frequency distribution of the sampled values reflects the shape of the posterior pdf, which essentially means that for a parameter θ_j , if the sample is $\{\theta_j^1, \theta_j^2, \dots, \theta_j^n\}$ then, the following relation holds:

$$\frac{\text{frequency}(\theta_j^k)}{\text{frequency}(\theta_j^l)} = \frac{P(\theta_j = \theta_j^k)|data)}{P(\theta_j = \theta_j^l)|data)} \quad (2.3.7)$$

Hence, I need algorithms which sample while satisfying equation 2.3.7.

Rejection Sampling: Rejection sampling, which is depicted in the Algorithm 1, is one way to do such sampling.

The algorithm runs as follows. I choose a random value of the unknown parameter denoted by Θ from a density function $g(\theta)$, where for some real c , $cg(\theta) > f(\theta), \forall \theta \in \mathbb{R}$. It is to be noted that $g(\theta)$ is any distribution which is easy to sample from and injective, whereas $f(\theta)$ is the target density, which when contextualised to our discussion above, will be interpretable as the marginal posterior density function of θ_j . Once a value θ_i is sampled from $g(\theta)$, the algorithm accepts this sampled value if the following criteria holds.

$$u \leq \frac{f(\theta_i)}{cg(\theta_i)}, u \sim U[0, 1] \quad (2.3.8)$$

Section 2.3 depicts the proof that rejection sampling algorithm indeed gives the sampled values with the target posterior density function i.e. $f(\theta)$.

Why Rejection sampling works

Algorithm 1: Rejection Sampling

- 1 $i = 1, c \geq 1$, given $f(\theta) \leq cg(\theta) \quad \forall \theta \in \mathbb{R}$
 - 2 Repeat until $i = N$
 - 3 Sample : $\theta_i \sim g(\theta)$ and $u \sim U[0, 1]$
 - 4 If $u \leq \frac{f(\theta_i)}{cg(\theta_i)}$ then accept θ_i and $i = i + 1$,
 - 5 Otherwise reject θ_i
-

Now, I shall be discussing why the algorithm of rejection sampling yields a sample that bears a frequency distribution with the same shape as the marginal of a given random variable. In order to do so, first, I will recall the algorithm first, secondly I shall prove some other results useful for the sought proof, finally the sought proof will be presented.

Derivation of Cumulative distribution function (CDF i.e. $F_{\Theta}(\theta) = P(\Theta \leq \theta)$) of Uniform distribution $U(0, 1)$.

The probability density function (PDF) for $U(a, b)$ is defined as follows.

$$f_{\Theta}(\theta) = \begin{cases} \frac{1}{b-a} & , a < \theta \leq b \\ 0 & , otherwise \end{cases} \quad (2.3.9)$$

The CDF is as follows.

$$\begin{aligned} F_{\Theta}(\theta) &= P(\Theta \leq \theta) \\ &= \int_{-\infty}^{\theta} f_{\Theta}(t) dt \\ &= \int_{-\infty}^a f_{\Theta}(t) dt + \int_a^{\theta} f_{\Theta}(t) dt \\ &= 0 + \int_a^{\theta} \frac{1}{b-a} dt, \quad (\text{putting values from 2.3.9}) \\ &= \frac{\theta - a}{b - a} \end{aligned} \quad (2.3.10)$$

Hence, I get the CDF as follows in three cases.

$$F_{\Theta}(\theta) = \begin{cases} 0 & , \theta \leq a \\ \frac{\theta - a}{b - a} & , a < \theta \leq b \\ 1 & , \theta > b \end{cases} \quad (2.3.11)$$

In the third case of equation 2.3.11, when $\theta > b$, though the PDF is 0 for this case, however, as the integral is taken over the interval $(-\infty, \theta)$ which includes the interval (a, b) where the PDF is non zero. Hence, the CDF will take the total probability value as 1. Therefore, when $a = 0, b = 1$ for $U(0, 1)$, the CDF becomes as follows.

$$F_{\Theta}(\theta) = \begin{cases} 0 & , \theta \leq 0 \\ \theta & , 0 < \theta \leq 1 \\ 1 & , \theta > 1 \end{cases} \quad (2.3.12)$$

Theorem 1. *If $\Theta = \theta$ is sampled from a function $g(\cdot)$, s.t. $f(\theta) \leq cg(\theta)$ and accept the sample if $u \leq \frac{f(\theta)}{cg(\theta)}$, $u \sim U[0, 1]$, then the probability distribution of the sample θ will be equal to $f(\cdot)$.*

Proof. If I want to get $P(U \leq \theta)$, that probability would be θ by equation 2.3.12, giving the following equation.

$$P(U \leq \theta) = \theta \quad (2.3.13)$$

However, here θ is the value of a random variable (r.v.) Θ , which means that if I want to find the probability of U less or equal to any such value that the r.v. Θ can take, then I have to get it in the form of average i.e. I need to find the expectation, $E_{\Theta}(P(U \leq \Theta))$. Now, $E_{\Theta}(P(U \leq \Theta))$ is equal to the integral of $P(U \leq \Theta)f_{\Theta}(\theta)$ where $f_{\Theta}(\theta)$ is the PDF of Θ i.e. $\Theta \sim f_{\Theta}(\cdot)$. Here, $f_{\Theta}(\cdot)$ is $g(\cdot)$ which the proposal density from which I pluck the samples. Therefore,

$$\begin{aligned} E_{\Theta}(P(U \leq \Theta)) &= \int_{-\infty}^{\infty} P(U \leq \Theta)f_{\Theta}(\theta) d\theta \\ &= \int_{-\infty}^{\infty} P(U \leq \Theta)g(\theta) d\theta \end{aligned} \quad (2.3.14)$$

When $\Theta = \frac{f(\theta)}{cg(\theta)}$, then the following is obtained.

$$\begin{aligned} E_{\Theta}(P(U \leq \frac{f(\theta)}{cg(\theta)})) &= \int_{-\infty}^{\infty} P(U \leq \frac{f(\theta)}{cg(\theta)})g(\theta) d\theta \quad (\text{using equation 2.3.14}) \\ &= \int_{-\infty}^{\infty} \frac{f(\theta)}{cg(\theta)}g(\theta) d\theta \quad (\text{using equation 2.3.13}) \\ &= \frac{1}{c} \int_{-\infty}^{\infty} f(\theta) d\theta \\ &= \frac{1}{c} \end{aligned} \quad (2.3.15)$$

Thus, equation 2.3.15 gives the probability of a plucked sample from $g(\theta)$ being accepted following the acceptance criteria of the algorithm i.e

$$P(\Theta = \theta \text{ being accepted}) = \frac{1}{c} \quad (2.3.16)$$

As I am plucking values from $g(\cdot)$ and accept the value on satisfaction of the acceptance criteria (i.e. ratio of $f(\cdot)$ and $g(\cdot)$) exceeds a uniform random number) designed in the algorithm, I need to consider how I am plucking values from $g(\cdot)$ in order to show that the PDF of an accepted sample (that have been generated by satisfying the acceptance criteria) is equal

to the PDF of $f(\cdot)$. Therefore, I want to find the probability of $\Theta = \theta$ lying in any interval $[a, b]$ that is defined on the domain of $g(\cdot)$ i.e. I want to find $P(\Theta = \theta \in [a, b] | \theta \text{ is accepted})$. Now, this conditional can be written in terms of joint and marginal as follows.

$$P(\Theta = \theta \in [a, b] | x \text{ is accepted}) = \frac{P(\Theta = \theta \in [a, b], \theta \text{ is accepted})}{P(\theta \text{ is accepted})} \quad (2.3.17)$$

Now, using the law of total probability in continuous domain, I obtain $P(\Theta = \theta \in [a, b], \theta \text{ is accepted})$ as the integral of $(P(\Theta = \theta \in [a, b] | \theta^* \text{ is accepted})P(U \leq \frac{f(\theta^*)}{cg(\theta^*)}))$ in the interval $[a, b]$ as follows.

$$\begin{aligned} P(\Theta = \theta \in [a, b], \theta \text{ is accepted}) &= \int_a^b g(\theta^*) P(U \leq \frac{f(\theta^*)}{cg(\theta^*)}) d\theta^* \\ &= \int_a^b g(\theta^*) \frac{f(\theta^*)}{cg(\theta^*)} d\theta^* \\ &= \frac{1}{c} \int_a^b f(\theta^*) d\theta^* \end{aligned} \quad (2.3.18)$$

Putting the value obtained from equation 2.3.18 and 2.3.16 in equation 2.3.17, I get,

$$\begin{aligned} P(\Theta = \theta \in [a, b] | \theta \text{ is accepted}) &= \int_a^b f(\theta^*) d\theta^* \\ &= F_\Theta(b) - F_\Theta(a) \end{aligned} \quad (2.3.19)$$

Thus I can see that, no matter from which interval of the domain of $g(\cdot)$ the sample is plucked, if I accept the sample satisfying the acceptance criteria of the algorithm, the probability distribution of the sample will be equal to $f(\cdot)$. $\square$

Need for better sampling algorithm

Rejection sampling gives an approach for sampling from the known posterior density function instead of simply optimizing an objective function. However in general, not all parameters of the target density are known - in fact, the target density that I need to sample from, namely, the posterior probability density of random variables $X_1, X_2, \dots$ given the data, is not known since its parameters are unknown. indeed, when the target posterior includes (multiple) unknown parameters that are to be learnt from the data, I need to perform rejection sampling sequentially from each trial candidate that is proposed for the target posterior, until I am sure that I am sampling from the correct target density. also, when there are multiple unknowns to be learnt, I will be required to compute the marginal posterior of each unknown given the data, so that inference on each parameter can be made, along with uncertainties. For

example, if the target (posterior) distribution is the product of several Normal distributions, then for a large volume of the support, the resulting density will be very close to zero, with multiple isolated peaks manifest. to optimise such a posterior is very difficult, with the computational difficulty increasing with increasing dimensionality of the state space over which the inference is taking place, i.e. with increasing number of unknowns.

2.4 MCMC introduction

2.4.1 Markov Chain in the context of MCMC

A sequence $X_1, X_2, \dots$ of random variables is called a stochastic process as long as each such random variable is defined as a mapping from the sample space of the same probabilistic experiment, to the space that the random variable takes values in. A Markov chain is a sub-sequence of the set of random variables that constitutes a Markov process, such that the probability of any element of this chain is conditional upon no other variable, except the one previous to the currently considered variable, i.e. it follows Markov property. To be precise, the conditional distribution of the $n + 1$ -th element of the chain, namely the random variable X_{n+1} , conditional on all previously-occurring elements in the chain, namely, $\{X_1, X_2, \dots, X_n\}$, is really the conditional probability of X_{n+1} on X_n . The space in which the X_i take values is called the state space of the Markov chain. The conditional distribution of X_{n+1} given X_n is called the transition probability distribution of a Markov chain. A Markov Chain having stationary transition probabilities is one in which the conditional distribution of X_{n+1} given X_n does not depend on n , i.e. is same for all n , so that transition probability at any state is a fixed probability. The joint distribution of a Markov chain is determined by, 1) the marginal distribution of X_1 , named as initial distribution and 2) the transition probability distribution. Due to the assumption of stationary transition probabilities, for a chain that admits a stationary distribution, this does not depend on n . However, for a Markov Chain having non-stationary transition probabilities, the conditional distribution of X_{n+1} given X_n does depend on n i.e. it changes over time. Most of the Markov chains in the context of MCMC have uncountable state space, hence, the initial distribution and the transition probability distribution cannot be represented as vector and matrix respectively, but only as probability distributions. An extensive literature on Markov chains can be found in Bowerman (1974); Ching and Ng (2006).

Stationarity: A stochastic process is stationary if the joint distribution of the vector $(X_n, X_{n+1}, \dots, X_{n+k})$ does not depend on n for each fixed k , where k is any positive integer. A Markov chain is stationary, i.e. the marginal distribution of X_n does not depend on n , iff

the chain is irreducible, ergodic. A Markov chain is said to be irreducible if any two of its states are accessible from each other. A state is said to be ergodic if it is positive recurrent and aperiodic. A state is said to be positive recurrent if it is recurrent (probability to return to the same state is one) and starting from the state, if the expected time until the process returns to the same state is finite. If each of the states of a Markov chain has period one, then the Markov chain is called aperiodic.

Ergodicity: The theorem for stationary stochastic processes is often called the Birkhoff ergodic theorem (Gray, 1997) which states that under a certain technical condition called “ergodicity”, it has the following property (analogous to the Strong Law of Large Numbers). If $Y_1, Y_2, \dots$ is a stationary real-valued stochastic process that is ergodic, and $E[Y_i] = \mu$, then

$$\bar{Y}_n \xrightarrow{a.s.} \mu, \quad n \rightarrow \infty \quad (2.4.1)$$

2.4.2 Markov Chain Monte Carlo (MCMC)

MCMC algorithms were originally developed by the physicists who were interested in modelling the probabilistic behaviour of collections of atomic particles at Los Alamos in the 1940’s. However, when they could not achieve the modelling analytically, as another option they were able to simulate the behaviour of the particles by constructing a Markov chain for which the desired distribution was the stationary distribution of the Markov chain and then simulate the Markov chain until stationarity is achieved. Having done this, they have introduced Metropolis algorithm (Metropolis et al., 1953) which has a huge impact, and which was then generalized with the Metropolis-Hastings (MH) algorithm (Hastings, 1970). Now a days, there is a huge number of applied work using this algorithm in several domains such as Finance, Biology, Medicine, Meteorology, Computer vision etc. particularly it has now become standard computing tools for analysing Bayesian complex models.

As the name suggests, MCMC relies on Markov chain theory to produce simulations and Monte Carlo approximations from a complex target distribution. MCMC simulation method produces a sequence of samples $X_1, X_2, \dots, X_n$ which is Markov chain, and the (ergodic) theory guarantees that X_i converges (in distribution) to the target distribution as $i \rightarrow \infty$ (Robert and Changye, 2020).

Generating samples with Metropolis-Hastings Algorithm: Although literature on MCMC includes a vast amount of techniques for generating samples, among which the Metropolis-Hastings Algorithm is the most fundamental algorithm and many other MCMC algorithm use this algorithm as a basic building block. Hence, this paragraph presents the intuition behind the design of the acceptance criteria of the Metropolis-Hastings Algorithm, then

shows the overall algorithm in Algorithm 6 and presents the detail proof in subsequent sub-sub-section 2.4.3.

I want a way to generate new sample θ_{i+1} having the current sample θ_i such that the distribution of the final samples, denote $\rho(\theta)$ for $n \rightarrow \infty$ is stationary (i.e. converges to something) and is equal to the posterior, $\pi(\theta)$. The stationarity condition satisfies by invoking the detailed balance property which says that the probability is conserved when moving from one state to another (i.e. the process is reversible). This implies that whether I jump from θ_i to θ_{i+1} or θ_{i+1} to θ_i , the joint probability for either of the case will be equal, and I know that joint probability is equal to (*conditional* $\times$ *marginal*), so I have the following equation where $Pr(\theta_{i+1}|\theta_i)$ is the probability of moving from state θ_i to state θ_{i+1} .

$$\begin{aligned} Pr(\theta_{i+1}|\theta_i) \cdot Pr(\theta_i) &= Pr(\theta_i|\theta_{i+1}) \cdot Pr(\theta_{i+1}) \\ \implies \frac{Pr(\theta_{i+1}|\theta_i)}{Pr(\theta_i|\theta_{i+1})} &= \frac{Pr(\theta_{i+1})}{Pr(\theta_i)} = \frac{\pi(\theta_{i+1})}{\pi(\theta_i)} \end{aligned} \quad (2.4.2)$$

The final equality comes as I am trying to generate samples from the posterior $\pi(\theta)$. I now figure out the way of moving to a new state by computing this probability. This can be achieved through two steps, 1) proposing a new state θ_{i+1}^* based on a proposal distribution $q(\theta_{i+1}^*|\theta_i)$ given the current sample (it is to be noted that the samples that are generated are dependent), 2) then, based on a transition probability $T(\theta_{i+1}^*|\theta_i)$ either I accept the new state ($\theta_{i+1} \leftarrow \theta_{i+1}^*$) or reject the new state ($\theta_{i+1} \leftarrow \theta_i$). Combining these terms gives the probability of moving from one state to another as follows.

$$Pr(\theta_{i+1}|\theta_i) = q(\theta_{i+1}|\theta_i) \cdot T(\theta_{i+1}|\theta_i) \quad (2.4.3)$$

I then need to determine the transition probability $T(\theta_{i+1}^*|\theta_i)$ of accepting or rejecting the new sample θ_{i+1}^* . Substituting into the expression for detailed balance, I find that our form for the transition probability must satisfy the following constraint which gives an intuition of the acceptance criteria of the Metropolis-Hastings Algorithm.

$$\frac{T(\theta_{i+1}|\theta_i)}{T(\theta_i|\theta_{i+1})} = \frac{\pi(\theta_{i+1})}{\pi(\theta_i)} \cdot \frac{q(\theta_i|\theta_{i+1})}{q(\theta_{i+1}|\theta_i)} \quad (2.4.4)$$

2.4.3 Why Metropolis-Hastings (MH) algorithm works

Now, the proof will be presented that establishes why the Metropolis-Hastings algorithm works.

Algorithm 2: Metropolis-Hastings Algorithm

Input : Posterior pdf $\pi(\cdot)$, x_0 is seed value; proposal distribution q , number of maximum iterations N

Output : Traces and marginals of the simulated values

```

1 Set  $X_0 \leftarrow x_0$ 
2 for  $i \leftarrow 1$  to  $N$  increment by 1 do
3   Sample  $X^* \sim q(x^*|X_{i-1})$  and  $u \sim U[0, 1]$ 
4   Accept  $X_i \leftarrow X^*$  if  $\min[1, (\frac{\pi(x^*)}{\pi(X_{i-1})} \cdot \frac{q(X_{i-1}|x^*)}{q(x^*|X_{i-1})})] > u$ 
5   Take  $X_i \leftarrow X_{i-1}$  otherwise
6 end

```

Theorem 2. *Metropolis-Hastings (MH) algorithm leads to stationary distribution that is equal to target density*

Proof. The proof consists of showing two aspects as follows, 1) the chain will have unique stationary distribution and 2) the unique stationary distribution is equal to the target distribution.

MH algorithm generates a stochastic process which is a Markov chain. The Markov property is satisfied as the distribution of the next state depends only on the current state.

The chain is irreducible as at each step there exist a positive probability to reach anywhere in the support of the proposed distribution (denoted as Q) which mainly defines the state space of the chain.

Also, for some state there is a positive probability that the chain will stay in the same state, hence the period of the state is 1. For an irreducible Markov chain all states share same period, hence the chain is aperiodic. So, by using the basic limit theorem, the chain will have a stationary distribution.

Now, due to the random nature of the MH algorithm, the chain becomes positive recurrent (i.e. the average time to return to a state starting from the same state is finite). I also have shown that the chain is aperiodic, so the chain becomes ergodic. Hence, it will have a unique stationary distribution

As the detailed balance condition is a sufficient condition (stated below) for the chain to have stationary distribution, I will show that the MH algorithm achieves the detailed balanced condition by its design with respect to our target density (denoted as f) i.e. the chain is reversible. It ensures that for each pair of states (i, j) , the flux from i to j is equal to the flux from j to i .

$$\pi_i P_{ij} = \pi_j P_{ji} \quad (2.4.5)$$

where, π is the probability distribution over the states of the chain and the P_{ij} is the conditional probability given that the chain is in state i at time n , that the chain jumps to the state j at time $n + 1$.

$$P_{ij} = P\{X_{n+1} = j | X_n = i\} \quad (2.4.6)$$

Now, by design of the acceptance probability of the MH algorithm, the transition probability (denoted by $P(x_{i+1}|x_i)$) is as follows where i is the current and x_{i+1} is the proposed.

$$\begin{aligned} P(x_{i+1}|x_i) &= Q(x_{i+1}|x_i) * \min\left\{1, \frac{f(x_{i+1})Q(x_i|x_{i+1})}{f(x_i)Q(x_{i+1}|x_i)}\right\} \\ &= \min\left\{Q(x_{i+1}|x_i), \frac{f(x_{i+1})Q(x_i|x_{i+1})}{f(x_i)}\right\} \end{aligned} \quad (2.4.7)$$

Hence, I get,

$$f(x_i)P(x_{i+1}|x_i) = \min\{f(x_i)Q(x_{i+1}|x_i), f(x_{i+1})Q(x_i|x_{i+1})\} \quad (2.4.8)$$

Similarly,

$$\begin{aligned} P(x_i|x_{i+1}) &= Q(x_i|x_{i+1}) * \min\left\{1, \frac{f(x_i)Q(x_{i+1}|x_i)}{f(x_{i+1})Q(x_i|x_{i+1})}\right\} \\ &= \min\left\{Q(x_i|x_{i+1}), \frac{f(x_i)Q(x_{i+1}|x_i)}{f(x_{i+1})}\right\} \end{aligned} \quad (2.4.9)$$

Hence, I get,

$$f(x_{i+1})P(x_i|x_{i+1}) = \min\{f(x_{i+1})Q(x_i|x_{i+1}), f(x_i)Q(x_{i+1}|x_i)\} \quad (2.4.10)$$

From 2.4.8 and 2.4.10, I get 2.4.11 which satisfies the detailed balance condition with respect to our target density, hence the MH algorithm leads to a unique stationary distribution equal to our target density.

$$f(x_i)P(x_{i+1}|x_i) = f(x_{i+1})P(x_i|x_{i+1}) \quad (2.4.11)$$

□

2.5 Various MCMC algorithms

In my warm-up phase, I have implemented four MCMC algorithms in the context of the small real-world data (O-ring data from Space shuttle challenger blast) mentioned in detail in the next subsection 2.6 where the model contains two unknown parameters and the algorithms are written with two parameters, thus providing guidance to tackle multiple parameters. The first one is Random Walk Metropolis-Hastings (RWMH) depicted in Algorithm 3. The algorithm shows the acceptance ratio when using the log-likelihood and Normal distribution as proposal distribution q with the current sample as mean. The proposal is a symmetric i.e. $q(\text{proposed}|\text{current}) = q(\text{current}|\text{proposed})$, hence the acceptance probability only involves the target density.

Algorithm 3: Random Walk Metropolis Hastings Sampler

Input : Posterior pdf $\pi(\alpha, \beta|data)$, α_0, β_0 are seeds value; $\sigma_\alpha, \sigma_\beta$ are jump scales for α and β respectively, number of maximum iterations N

Output : Traces and marginals for parameters α, β

```

1 Set  $\alpha[0] \leftarrow \alpha_0, \beta[0] \leftarrow \beta_0, \pi[0] \leftarrow$  assign a small initial value
2 for  $i \leftarrow 1$  to  $N$  increment by 1 do
3    $\pi_{current} \leftarrow \pi[i-1]$ 
4    $\alpha_{proposed} \sim \mathcal{N}(\alpha[i-1], \sigma_\alpha), \beta_{proposed} \sim \mathcal{N}(\beta[i-1], \sigma_\beta)$ , and  $u \sim U(0, 1)$ 
5   if  $(\log(\pi(\alpha_{proposed}, \beta_{proposed}|data)) - \log(\pi_{current})) > \log(u)$  then
6      $\alpha[i] \leftarrow \alpha_{proposed}, \beta[i] \leftarrow \beta_{proposed}, \pi[i] \leftarrow \pi(\alpha_{proposed}, \beta_{proposed}|data)$ 
7   else
8      $\alpha[i] \leftarrow \alpha[i-1], \beta[i] \leftarrow \beta[i-1], \pi[i] \leftarrow \pi[i-1]$ 
9   end
10 end

```

The 2nd one is Independent Sampler (IS) presented in Algorithm 4 which uses log-likelihood and Normal distribution as proposal distribution q with fixed mean, so that $q(\text{proposed}|\text{current}) = q(\text{proposed})$.

The 3rd one is Random Walk Metropolis Hastings within Gibbs Sampler (RWG) shown in the Algorithm 5 using the log-likelihood and Normal distribution as proposal distribution q with the current sample as mean. Parameters are proposed and updated one by one.

The 4th one is the Metropolis Hastings algorithm using the log-likelihood and truncated Normal distribution as proposal distribution q with the current sample as mean, depicted in the Algorithm 6. One parameter is left truncated at 0 and the other one is right truncated at 0. Due to the truncated nature, the proposal is not symmetric here.

Algorithm 4: Independent Sampler

Input : Posterior pdf $\pi(\alpha, \beta | data)$, α_0, β_0 are seeds value; μ_α, μ_β are fixed proposal mean; $\sigma_\alpha, \sigma_\beta$ are jump scales for α and β respectively, number of maximum iterations N

Output : Traces and marginals for parameters α, β

- 1 Set $\alpha[0] \leftarrow \alpha_0, \beta[0] \leftarrow \beta_0, \pi[0] \leftarrow$ assign a small initial value
- 2 **for** $i \leftarrow 1$ **to** N **increment by 1 do**
- 3 $\pi_{current} \leftarrow \pi[i-1]$
- 4 $\alpha_{proposed} \sim \mathcal{N}(\mu_\alpha, \sigma_\alpha), \beta_{proposed} \sim \mathcal{N}(\mu_\beta, \sigma_\beta)$, and $u \sim U(0, 1)$
- 5 $\alpha_Q \leftarrow (\log(\mathcal{N}(\alpha_{proposed} | \mu_\alpha, \sigma_\alpha)) - \log(\mathcal{N}(\alpha_{current} | \mu_\alpha, \sigma_\alpha)))$
- 6 $\beta_Q \leftarrow (\log(\mathcal{N}(\beta_{proposed} | \mu_\beta, \sigma_\beta)) - \log(\mathcal{N}(\beta_{current} | \mu_\beta, \sigma_\beta)))$
- 7 **if** $(\log(\pi(\alpha_{proposed}, \beta_{proposed} | data)) - \log(\pi_{current}) + \alpha_Q + \beta_Q) > \log(u)$ **then**
- 8 $\alpha[i] \leftarrow \alpha_{proposed}, \beta[i] \leftarrow \beta_{proposed}, \pi[i] \leftarrow \pi(\alpha_{proposed}, \beta_{proposed} | data)$
- 9 **else**
- 10 $\alpha[i] \leftarrow \alpha[i-1], \beta[i] \leftarrow \beta[i-1], \pi[i] \leftarrow \pi[i-1]$
- 11 **end**
- 12 **end**

2.6 Application with real-world data

I have implemented the above mentioned four types of MCMC algorithms on a small scale real-world data named as O-ring data (example taken from Dalal et al. (1989); Lavine (1991); Robert and Casella (2005)).

2.6.1 Data

In 1986, the Space Shuttle Challenger exploded very soon after launch, killing all seven astronauts on board. The explosion was the result of an O-ring failure. An O-ring is a rubber ring that seals parts of the Shuttle. The O-rings are made of a material that stiffen at the ambient coldness (Dalal et al., 1989), thereby causing these O-rings to fail to provide the required sealing, resulting in the explosion of the Shuttle. The accident was believed to have been caused by the unusually cold weather (with ambient temperatures of about 31°F) at the time of the launch. Table 2.1 depicts the failure/success (1/0) of the O-ring for the flight at the given temperature.

2.6.2 Model for the O-ring data

For a binary response variable, $\mathbf{Y}$ (failure/success), with a predictor variable, $\mathbf{X}$ (temperature), I want to model the probability of failure given the temperature assuming that it follows

Algorithm 5: Random Walk Metropolis Hastings within Gibbs

Input : Posterior pdf $\pi(\alpha, \beta | data)$, α_0, β_0 are seeds value; $\sigma_\alpha, \sigma_\beta$ are jump scales for α and β respectively, number of maximum iterations N

Output : Traces and marginals for parameters α, β

```

1 Set  $\alpha[0] \leftarrow \alpha_0, \beta[0] \leftarrow \beta_0$ 
2 for  $i \leftarrow 1$  to  $N$  increment by 1 do
3    $\alpha_{proposed} \sim \mathcal{N}(\alpha[i-1], \sigma_\alpha), u \sim U(0, 1)$ 
4   if  $(\log(\pi(\alpha_{proposed}, \beta[i-1] | data)) - \log(\pi(\alpha[i-1], \beta[i-1] | data))) > \log(u)$ 
5     then
6        $\alpha[i] \leftarrow \alpha_{proposed}$ 
7     else
8        $\alpha[i] \leftarrow \alpha[i-1]$ 
9   end
10   $\beta_{proposed} \sim \mathcal{N}(\beta[i-1], \sigma_\beta)$  if
11     $(\log(\pi(\alpha[i], \beta_{proposed} | data)) - \log(\pi(\alpha[i], \beta[i-1] | data))) > \log(u)$  then
12     $\beta[i] \leftarrow \beta_{proposed}$ 
13  else
14     $\beta[i] \leftarrow \beta[i-1]$ 
15  end
16 end

```

Bernoulli distribution. The probability of failure given the temperature is modelled with the following parametric function with two parameters α, β .

$$P(Y = 1 | X = x) = \frac{e^{(\alpha + \beta x)}}{1 + e^{(\alpha + \beta x)}} \quad (2.6.1)$$

Using the above mentioned assumption of the Bernoulli, the likelihood is depicted as follows with n being the number of data points which is 23 here.

$$L(\alpha, \beta | y) \propto \prod_{i=1}^n \left(\frac{e^{(\alpha + \beta x)}}{1 + e^{(\alpha + \beta x)}} \right)^{y_i} \left(\frac{1}{1 + e^{(\alpha + \beta x)}} \right)^{1-y_i} \quad (2.6.2)$$

I take Gaussian priors for both α and β as follows.

Algorithm 6: Vanilla MCMC

Input : Posterior pdf $\pi(\alpha, \beta | data)$, α_0, β_0 are seeds value; $\sigma_\alpha, \sigma_\beta$ are jump scales for α and β respectively, a_α, b_α are lower and upper truncation in proposal distribution for α , a_β, b_β are lower and upper truncation in proposal distribution for β , number of maximum iterations N

Output : Traces and marginals for parameters α, β

- 1 Set $\alpha[0] \leftarrow \alpha_0, \beta[0] \leftarrow \beta_0, \pi[0] \leftarrow$ assign a small initial value
- 2 **for** $i \leftarrow 1$ **to** N **increment by** 1 **do**
- 3 $\pi_{current} \leftarrow \pi[i-1], a_\alpha \leftarrow 0, b_\alpha \leftarrow +\infty, a_\beta \leftarrow -\infty, b_\beta \leftarrow 0$
- 4 $\alpha_{proposed} \sim \mathcal{N}_T(\alpha[i-1], \sigma_\alpha, a_\alpha, b_\alpha), \beta_{proposed} \sim \mathcal{N}_T(\beta[i-1], \sigma_\beta, a_\beta, b_\beta)$, and $u \sim U(0, 1)$
- 5 $\alpha_Q \leftarrow (\log(\mathcal{N}_T(\alpha_{current} | \alpha_{proposed}, \sigma_\alpha, a_\alpha, b_\alpha)) - \log(\mathcal{N}_T(\alpha_{proposed} | \alpha_{current}, \sigma_\alpha, a_\alpha, b_\alpha)))$
- 6 $\beta_Q \leftarrow (\log(\mathcal{N}_T(\beta_{current} | \beta_{proposed}, \sigma_\beta, a_\beta, b_\beta)) - \log(\mathcal{N}_T(\beta_{proposed} | \beta_{current}, \sigma_\beta, a_\beta, b_\beta)))$
- 7 **if** $(\log(\pi(\alpha_{proposed}, \beta_{proposed} | data)) - \log(\pi_{current}) + \alpha_Q + \beta_Q) > \log(u)$ **then**
- 8 $\alpha[i] \leftarrow \alpha_{proposed}, \beta[i] \leftarrow \beta_{proposed}, \pi[i] \leftarrow \pi(\alpha_{proposed}, \beta_{proposed} | data)$
- 9 **else**
- 10 $\alpha[i] \leftarrow \alpha[i-1], \beta[i] \leftarrow \beta[i-1], \pi[i] \leftarrow \pi[i-1]$
- 11 **end**
- 12 **end**

Failure	1	1	1	1	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0
Temperature	53	57	58	63	66	67	67	67	68	69	70	70	70	70	72	73	75	75	76	76	78	79	81

Table 2.1 O-ring data

$$\begin{aligned}
\pi_0(\alpha) &= \mathcal{N}(\mu_\alpha, \sigma_\alpha^2) = \frac{1}{\sigma_\alpha \sqrt{2\pi}} e^{-(\alpha - \mu_\alpha)^2 / 2\sigma_\alpha^2} \\
\pi_0(\beta) &= \mathcal{N}(\mu_\beta, \sigma_\beta^2) = \frac{1}{\sigma_\beta \sqrt{2\pi}} e^{-(\beta - \mu_\beta)^2 / 2\sigma_\beta^2}
\end{aligned} \tag{2.6.3}$$

Hence, our posterior PDF, denoted by $\pi(\alpha, \beta)$ becomes as follows.

$$\pi(\alpha, \beta) = L(\alpha, \beta | y) * \pi(\alpha, \beta) = \prod_{i=1}^n \left(\frac{e^{(\alpha + \beta x)}}{1 + e^{(\alpha + \beta x)}} \right)^{y_i} \left(\frac{1}{1 + e^{(\alpha + \beta x)}} \right)^{1-y_i} * \pi_0(\alpha) * \pi_0(\beta) \tag{2.6.4}$$

Now, the computation of the likelihood (RHS of Equation 3.3.1) is computationally expensive. Even for 23 data points it is computationally heavy, so, if I take log likelihood then the form will be simplified. Then the likelihood becomes,

$$\begin{aligned}
 \log(L(\alpha, \beta|y)) &= \log\left(\prod_{i=1}^n (p(x_i)^{y_i} * (1 - p(x_i))^{1-y_i})\right), \text{ where } p(x) = \frac{e^{(\alpha+\beta x)}}{1 + e^{(\alpha+\beta x)}} \\
 &= \sum_{i=1}^n (y_i \log(p(x_i)) + (1 - y_i) \log(1 - p(x_i))) \\
 &= \sum_{y_i=0, i=1}^n \log(1 - p(x_i)) + \sum_{y_i=1, i=1}^n \log(p(x_i)) \\
 &= \sum_{y_i=0, i=1}^n -\log((1 + e^{(\alpha+\beta x_i)})) + \sum_{y_i=1, i=1}^n (\log(e^{(\alpha+\beta x_i)}) - \log((1 + e^{(\alpha+\beta x_i)})))
 \end{aligned} \tag{2.6.5}$$

I can use this expression from 2.6.5 and rewrite the log posterior pdf in 2.6.4 as,

$$\begin{aligned}
 \log(\pi(\alpha, \beta)) &= \sum_{y_i=0, i=1}^n -\log((1 + e^{(\alpha+\beta x_i)})) + \sum_{y_i=1, i=1}^n (\log(e^{(\alpha+\beta x_i)}) - \log((1 + e^{(\alpha+\beta x_i)}))) + \\
 &\quad \log(\pi_0(\alpha)) + \log(\pi_0(\beta))
 \end{aligned} \tag{2.6.6}$$

After performing inference on model parameters through posterior sampling with various kinds of MCMC to compute the marginal posterior of each parameter given the data, then the 95% Highest Probability Density credible region (HPDs) (Kruschke, 2015) on each parameter is learned. The mean of the HPD region is considered as the learned value of the parameters, though, in principle, any value lying in the HPD region can be used as the learned value of the parameters. Using mean of the HPD region, the values of the parameters are learned as follows, $\alpha \sim 15$, $\beta \sim -0.23$. Hence, applying these learnt values of the parameters in the model, the failure probability is computed at the launch temperature 31F, giving 95% of chance of failure at the launch time. If this kind of analysis could be done before hand at that time, the fatal accident could have been avoided, thus saving lives.

2.6.3 Results

Table 2.2 shows the results of different MCMC methods that I have implemented, chains with different configurations and corresponding observations pertaining to the chains. Several experimentations are done in order to understand the effect of jump scales, effect of priors,

number of iterations on getting the converged chain. The algorithms are also pushed by using challenging seeds to see them working even with some challenging starting points. 0 is used as the seed for random number generation. For the convergence test, I have used 3-window test i.e. have plotted the histograms of the few samples pertaining to three window from the distinct location of the post-burn-in chain. Overlapping histograms from the three windows indicates convergence. Following are the abbreviations used to represent the implemented algorithms - RW: Random Walk Metropolis Hastings, IS: Independent Sampler, RWG: Random Walk Metropolis Hastings within Gibbs, VMH: Vanilla MH. Results for each algorithm are presented in a block ending with three horizontal lines. A block between double horizontal lines denotes a chunk of analysis starting from one configuration. Figures for these results are presented in Appendix B.2 as a separate pdf file which is included in the main report, hence the page number of the pdf with the figures starts from, to differentiate the page number from the page number of the main report, the page number of the figure file contains a *. In the table, the page number with * is mentioned in each row, thus connecting the figures of the chain whose configuration is given in that row along with the observation (if any). Result pertaining to a chain is presented in a single page which consists of trace plots of the parameters and the log posterior, the histograms of the marginals of the parameters, histograms of the log posterior values, 3-window convergence test for both the parameters.

For RW, I have done a pilot run consisting of 6 chains because I don't know the prior and need to figure out a good enough prior which will not lead the convergence to the wrong result, starting with the chain-A depicted in the page-1*, looking at the model, it can be understood that α will be positive and β should be a small negative number, hence the configuration of the chain-A with a small jump scale. The next chain, chain-B, starts with the end of the chain-A i.e. the end value of chain-A becomes the starting value of chain-B, and likewise chain-C to F are generated and the end values of the parameters are being noted. The results of chain-B to F are presents in page-2* to page-6*. With these 6 pilot chains, it can be seen that alpha value ranges in 14, 15, 16 with mean 15 and beta value ranges in -0.22, -0.23, -0.24 with mean -0.23. So, now I set the prior mean for alpha as 15 and prior mean for beta -0.23 and run the chain with a little flat prior resulting chain-G presented in page-7* from which I can see that it is approaching convergence, so I need to strengthen the prior. It is to be noted that the correct value of the parameters are $\alpha=15$, $\beta=-0.235$ and from the pilot chain I already get an intuition of the priors close to the correct value. In case, I do not get good estimate from the end values of the parameters of the pilot chains, I then opt to take the mean of the chain. The next chain with strengthened prior is chain-H presented in page-8* where I got convergence, however, the convergence of α needs to be better, so in the next chain, I have increased the jump scale of α and got good convergence whose results are

presented in the page-9*. For a converged chain, the marginal of the parameters are unimodal Gaussian. Other five chains with several other configurations including challenging seed value of the parameters are experimented with and the results are presented from page-10* to page-14*.

Page-15* to page-17* presents the chains for three different configurations of the IS. Due to the fixed mean of the proposal density, the chain convergence when I sample from a proposal density centred near to the correct values of the parameters.

Page-18* shows the chain for RWG with small jump scales and flat priors which is far from convergence, however, looking at the end portion of the traces it might be the case that both the jump scales of α, β should be increased. So, the next chain presented in page-19* follows this experimentation from which it appears that convergence is approaching. Hence, keeping the same configuration, I run the next chain with longer iteration presenting the plots for the chain in page-20*, however, convergence is not achieved rather the end portion of the traces of the parameters indicates that the α jump scale needs to be decreased and the β jump scale needs to be increased. So, the next two chains depicted in the page-21* and page-22* does the experimentation and concludes that the chain will not converge with this approach, hence, I need to increase the jump scales of the parameters. The next chain is with the increased jump scales and converged, the results are shown in page-23*. The next chain experiments with the same configuration as the previous one, but with out using the prior i.e. using the uniform prior. And I see that results (depicted in page-24*) indicates that convergence is approaching but with high error. I then experimented with the chain presented in page-19*, from the prior perspective, by using stronger prior which is close to the correct values of the parameters (shown in page-25*), and I got convergence. Then in the next chain, I have used a prior which is not close to the correct values of the parameters in the same chain of page-19*, and I got wrongly converged results, depicting that it is prior driven, plots are presented in page-26*. The chain of page-19* is again used to experiment with using a challenging seed keeping the other configurations same, the results are shown in page-27*.

Next chains pertain to the results of VMH. The chain presented in page-28* uses challenging seed and uniform prior i.e. equivalent to using no prior, and there is no convergence, so in the next chain, depicted in the page-29*, uses priors with the same configuration, and achieves convergence. Then I increased the jump scale along with the change in seed with a strong prior close to the correct value and convergence is achieved, these results for the chain is shown in page-30*. The next chain uses the same configuration as before but with a somewhat flat prior than before, and the convergence is also achieved, results are given in page-31*. The last chain which uses the same configuration but with a prior which is

not close to the correct values of the parameters, wrongly converged results are achieved indicating the impact of prior, these results are presented in page-32*.

2.7 Conclusion

In this chapter I present various aspects of MCMC including state of the art, different algorithms that I pursued on a real word small-scale data set and results. I have presented why a simple yet fundamental sampling technique such as rejection sampling works, which forms the base of the current day's MCMC methods. I have also included the basic probabilistic and mathematical background that is required to perform inference with MCMC techniques. The algorithms pertaining to each type of the MCMC methods are included that are implemented in Java programming language in this chapter.

I have performed a comprehensive experimentation (running various chains with different configurations) with each type of the MCMC inference for learning the unknown parameters of the model, to investigate the quality of convergence, effect of - priors, variance of the proposal density, sensitivity on priors, various seeds etc. The figures pertaining to the experimentation performed with MCMC in this chapter are included in Appendix B.

Alg.	α_0	β_0	α_{μ}^{prop}	β_{μ}^{prop}	α_{σ}^{prop}	β_{σ}^{prop}	α_{μ}^{pri}	α_{σ}^{pri}	β_{μ}^{pri}	β_{σ}^{pri}	n	Convergence	AR%	Note	HPD 95%	page
RW	10	-0.3	-	-	0.027	0.00405	-	-	-	-	10^6	No, need work	84.1	Pilot run, C-A: $\alpha[n]:15.42, \beta[n]:-0.232$	-	1*
RW	15.42	-0.232	-	-	0.027	0.00405	-	-	-	-	10^6	No	84.4	C-B:from end of C-A $\alpha[n]:16.07, \beta[n]:-0.24$	-	2*
RW	16.07	-0.24	-	-	0.027	0.00405	-	-	-	-	10^6	No	84.5	C-C:from end of C-B $\alpha[n]:15.5, \beta[n]:-0.233$	-	3*
RW	15.5	-0.233	-	-	0.027	0.00405	-	-	-	-	10^6	No	84.6	C-D:from end of C-C $\alpha[n]:16.4, \beta[n]:-0.24$	-	4*
RW	16.4	-0.24	-	-	0.027	0.00405	-	-	-	-	10^6	No	84.3	C-E:from end of C-D $\alpha[n]:14.1, \beta[n]:-0.21$	-	5*
RW	14.1	-0.21	-	-	0.027	0.00405	-	-	-	-	10^6	No	84.3	C-F:from end of C-E $\alpha[n]:15.4, \beta[n]:-0.232$	-	6*
RW	From 6 pilot chains, α range: {14,15,16}, mean=15; β range: {-0.22,-0.23,-0.24} mean=-0.23. Use as prior															-
RW	15.43	-0.232	-	-	0.027	0.00405	15	2	-0.23	2	10^6	Approaching	84.5	C-G: strengthen prior	-	7*
RW	15.43	-0.232	-	-	0.027	0.00405	15	0.5	-0.23	0.5	10^6	Yes	84.3	C-H: do better for α $\uparrow$ JS for α, β	-	8*
RW	15	-0.22	-	-	0.14	0.0154	15	0.5	-0.23	0.5	2×10^5	Yes, Good	50.1	-	$\alpha:(14.04, 15.96)$ $\beta:(-0.255, -0.21)$	9*
RW	15	-0.23	-	-	0.013	0.0156	15	5	-0.23	5	5×10^5	No	50.1	Approaching Need more work	-	10*
RW	1	-0.2	-	-	0.045	0.00495	-	-	-	-	60000	Close	80.2	Approaching Need prior	-	11*
RW	1	-0.2	-	-	0.045	0.00495	15	1	-0.235	1	60000	Close	80.9	Approaching Need more work	-	12*
RW	1	-0.2	-	-	0.045	0.00495	15	0.5	-0.235	0.5	2×10^5	Yes	80.9	Choose better seed next seed=end of this	-	13*
RW	15	-0.22	-	-	0.045	0.00495	15	0.5	-0.235	0.5	2×10^5	Yes, Better	80	beta is good Do better for α	$\alpha:(14.05, 15.897)$ $\beta:(-0.253, -0.212)$	14*
IS	13	-0.2	15	-0.235	0.3	0.03	-	-	-	-	10^5	Yes	28	-	$\alpha:(14.55, 15.44)$ $\beta:(-0.252, -0.215)$	15*
IS	13	-0.2	14	-0.2	2	0.07	-	-	-	-	30000	Yes, wrong	11	prop μ driven	-	16*
IS	13	-0.2	14	-0.2	2	0.07	15	1	-0.235	5	30000	Yes	11.9	-	$\alpha:(11.34, 16.68)$ $\beta:(-0.257, -0.174)$	17*
RWG	10	-0.2	-	-	0.032	0.0036	15	2	-0.23	2	3×10^5	Far from conv.	98	Need Work	-	18*
RWG	10	-0.2	-	-	0.08	0.0096	15	2	-0.23	2	3×10^5	Approaching	95	Need Work	-	19*
RWG	10	-0.2	-	-	0.08	0.0096	15	2	-0.23	2	5×10^5	May be	95.2	$\downarrow \alpha_{JS}, \uparrow \beta_{JS}$	-	20*
RWG	10	-0.2	-	-	0.072	0.0108	15	2	-0.23	2	5×10^5	May be	95.7	$\downarrow \alpha_{JS}, \uparrow \beta_{JS}$	-	21*
RWG	10	-0.2	-	-	0.055	0.01375	15	2	-0.23	2	5×10^5	Will not	96.7	Increase JS	-	22*
RWG	10	-0.2	-	-	0.155	0.02015	15	2	-0.23	2	5×10^5	Yes	90.8	-	$\alpha:(11.3, 18.76)$ $\beta:(-0.29, -0.174)$	23*
RWG	10	-0.2	-	-	0.155	0.02015	-	-	-	-	5×10^5	Approaching	90.8	High error	-	24*
RWG	10	-0.2	-	-	0.08	0.0096	15	0.5	-0.23	0.5	3×10^5	Yes	95	Strong prior	$\alpha:(14.1, 15.9)$ $\beta:(-0.25, -0.21)$	25*
RWG	10	-0.2	-	-	0.08	0.0096	12	2	-0.23	2	3×10^5	Yes, wrong	95	Prior driven	-	26*
RWG	1	-0.2	-	-	0.08	0.0096	15	2	-0.23	2	5×10^5	Yes	95	-	$\alpha:(11.3, 18.6)$ $\beta:(-0.288, -0.176)$	27*
VMH	1	-0.1	-	-	0.025	0.000625	-	-	-	-	2×10^5	No	86	NLR	-	28*
VMH	1	-0.1	-	-	0.025	0.000625	15	0.5	-0.23	0.5	2×10^5	Yes	87	-	$\alpha:(14.1, 15.9)$ $\beta:(-0.254, -0.079)$	29*
VMH	15	-0.22	-	-	0.35	0.0455	15	0.5	-0.23	0.5	2×10^5	Yes, Good	20	-	$\alpha:(13.63, 15.76)$ $\beta:(-0.256, -0.204)$	30*
VMH	15	-0.22	-	-	0.35	0.0455	15	1.5	-0.23	1.5	2×10^5	Yes, Good	21.5	-	$\alpha:(11.95, 19.4)$ $\beta:(-0.308, -0.198)$	31*
VMH	15	-0.22	-	-	0.35	0.0455	13	1.5	-0.23	1.5	2×10^5	Yes, wrong	20.9	Prior mean driven	-	32*

Table 2.2 Different MCMC methods, configurations and observations (Correct value of $\alpha=15$, $\beta=-0.235$). α_0 is seed, *prop* denotes proposal, μ is mean, σ is std. deviation, *pri* denotes prior, *n* is iteration, AR is acceptance rate, C-A denotes Chain-A, NLR: need longer run, JS: Jump Scale, RW: Random Walk MH, IS: Independence Sampler, RWG: Random Walk MH within Gibbs, VMH: Vanilla MH. A block between double horizontal line denotes a chunk of analysis starting from one configuration. Figures for these results are presented in Appendix B.2

Chapter 3

Efficient Uncertainty Quantification for Under-constraint Supervised Learning using Gaussian Processes

3.1 Introduction

Learning the functional relationship between the response and the predictor variables, constitutes a supervised mechanistic learning set up. Automated and reliable learning of this function is sought, given a training data set, with the ultimate aim being the automated and reliable prediction of the response at test values of the predictors. Probabilistic learning provides the framework for undertaking inference on models that explain the data under consideration, while also quantifying uncertainties objectively. Despite the importance of uncertainty quantification motivated in chapter 1, uncertainty quantification is still an emerging area and it requires careful addressing, in order to achieve high quality intelligence from the data (Ghahramani, 2015). A poorly chosen representation of uncertainty can affect the learning and prediction task greatly.

This chapter illustrates what is implied by a poorly chosen way of addressing uncertainties, and shows the effect of the same on the quality of predictions; in contrast, efficient uncertainty learning, enhances prediction performance within supervised learning using Gaussian Processes (GP), with inference undertaken using Markov Chain Monte Carlo (MCMC). The effect of variously propagating the uncertainty is discussed below, on predictions undertaken on the output variable, at test inputs, subsequent to the learning of the functional relationship between the input and the output, where this functional relation is modelled as a realisation from a GP. Comparative results for regression is also undertaken

with Deep Neural Network (DNN) systems (which do not account for uncertainties), with varying architecture. Such a comparative exercise will indicate the sensitivity of the results to the chosen architectural details. This chapter starts with a brief introduction to GP Regression with a toy example.

GP-based regression was introduced in O’Hagan (1978); however, it gained popularity as a non-parametric modelling approach since the publication of (Neal, 1998). Rasmussen (1997) states that GP predictive performance is comparable to several other state-of-the-art modelling approaches including neural networks. Predictive distributions are computed for each test point in GP regression, and the mean of these predictive is considered as the predicted output value, while the closed-form prediction of the variance provides the uncertainty on the prediction. Treating a function - of unknown form, and therefore treated as a random function in the Bayesian setting - as a sample from a GP implies that the joint probability of a finite number of realisations of this function - if scalar-valued - is multivariate Normal with a mean vector and variance-covariance matrix, which constitute the parameters of this multivariate Normal density. This covariance matrix, if parametrised using a covariance kernel may be of a chosen parametric form, with hyper-parameter(s) assigned to this chosen form. The type of structure that can be learnt using a GP, depends on the selection of the kernel. Duvenaud (2014) shows an extensive literature on kernels, how base kernels can be added and/or multiplied to produce a large range of models including generating topological manifolds through different compositions of kernels. However, again this selection of base kernels and the composition is a matter of choice which might not be appropriate according to the data at hand, hence, the learning of the values of the hyperparameters of the kernels from the data is needed rather than imposed by hand (because they are unknown quantities), to avoid erroneous predictions. Wilson et al. (2020) presents techniques for efficiently sampling functions from GP posteriors as sampling from GPs is essential to the estimation of quantities of interest in several complex algorithms such as guiding reinforcement learning agents (Kuss and Rasmussen, 2003), black-box optimizers (Snoek et al., 2012).

Basics of Gaussian Process based regression is included in the Section 3.2 of this chapter. The data and the model is presented in Section 3.3. Section 3.4 depicts two different ways of generation of training datasets that is required to show the various ways of uncertainty percolation including learning parameters with under-constraint MCMC and results obtained with various DNN architectures. The efficient uncertainty percolation has been presented in Section 3.5. Finally the chapter is concluded in Section 3.6.

3.2 Gaussian Process Regression

A Gaussian Process (GP) is a collection of random variables, any finite number of which have a joint multivariate Normal distribution (Rasmussen and Williams, 2006). A function of unknown form is treated as random in the Bayesian setting, so, a sample from a GP implies that the joint probability of a finite number of realisations of this function - if scalar-valued - is multivariate Normal with a mean vector and variance-covariance matrix. A GP is a distribution over functions, where a function can be thought of as infinitely long vector consisting of the value of this function at every input. While a multivariate Gaussian distribution is fully specified by mean vector and covariance matrix, a Gaussian Process is fully specified by a mean function $m(\cdot)$ and covariance function, also known as kernel $k(\cdot, \cdot)$:

$$f(\mathbf{x}) \sim GP(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$$

Squared exponential (SQE) kernel is one of the widely used kernels whose form is as follows, where ℓ is known as length-scale hyperparameter that controls the smoothness of the function. Informally, ℓ can be thought of as the distance one has to move in input space before the function value can change significantly.

$$\kappa(x, x') = \exp\left(-\frac{1}{2\ell^2}(x - x')^2\right) \quad (3.2.1)$$

The kernel function must be symmetric and positive definite, i.e. $k(x, x') = k(x', x)$, and the kernel matrix K induced by k for any set of input, is a positive definite matrix. Typically, a zero-mean GP is considered.

GP is largely used in practice due to some of its advantages. These are cited below:

- GP works quite well even when there is small amount of data.
- GP is highly interpretable.
- GP gives closed-form predictive distribution
- GP provides prediction uncertainty.
- GP-based learning and particularly, the subsequent output prediction, is fast.

Given a test point, GP does not provide a deterministic predicted value, rather it gives a random variable that follows a Normal distribution with a mean and variance, the prediction variance serves as an indicator of the prediction uncertainties. Prediction uncertainties are

helpful in understanding the reliability of the prediction and also one can get intuition for intelligently providing training data to enhance the model performance.

Suppose there are n points $X = \{x_1, x_2, \dots, x_n\}$ and corresponding outputs,

$$Y = \{f(x_1), f(x_2), \dots, f(x_n)\}.$$

For a new test points X_* , one wants to predict $f(X_*)$ i.e. f_* by learning the functional relationship between X and Y as $Y = f(X)$. By assuming that the regression function f is a random realisation of a Gaussian process, one gets the property that the distribution of the values of f at any finite number of points is a Gaussian distribution. Assuming zero-mean, one has,

$$\begin{pmatrix} \mathbf{y} \\ f_* \end{pmatrix} \sim \mathcal{N}\left(0, \begin{pmatrix} K(X, X) & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{pmatrix}\right), \quad (3.2.2)$$

where $K(X, X)$ is the covariance matrix for the training points, $K(X_*, X)$ is the covariance vector between the new test point and the training points, and $K(X_*, X_*)$ is the inherent measurement noise. To get the posterior distribution over functions one needs to restrict this joint prior distribution to contain only those functions which agree with the observed data points. Then,

$$\begin{aligned} p(f_* | X_*, X, \mathbf{y}) &\sim \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*) \text{ where,} \\ \boldsymbol{\mu}_* &= (X_*, X) K(X, X)^{-1} \mathbf{y} \\ \boldsymbol{\Sigma}_* &= K(X_*, X_*) - (X_*, X) K(X, X)^{-1} K(X, X_*) \end{aligned} \quad (3.2.3)$$

where $\boldsymbol{\mu}_*$ gives the mean of the predictive density and the covariance part i.e. $\boldsymbol{\Sigma}_*$ gives predictive uncertainty. For noisy data the function f becomes $f + \varepsilon$ where ε is a Gaussian noise with variance σ^2 . Hence, the prior on noisy training data becomes $K(X, X) + \sigma^2 I$ and subsequently the joint distribution becomes as follows.

$$\begin{pmatrix} \mathbf{y} \\ f_* \end{pmatrix} \sim \mathcal{N}\left(0, \begin{pmatrix} K(X, X) + \sigma^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{pmatrix}\right), \quad (3.2.4)$$

With the following predictive distribution.

$$\begin{aligned} p(f_* | X_*, X, \mathbf{y}) &\sim \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*) \text{ where,} \\ \boldsymbol{\mu}_* &= (X_*, X) (K(X, X) + \sigma^2 I)^{-1} \mathbf{y} \\ \boldsymbol{\Sigma}_* &= K(X_*, X_*) - (X_*, X) (K(X, X) + \sigma^2 I)^{-1} K(X, X_*) \end{aligned} \quad (3.2.5)$$

Generating random realizations from GP: Sometimes, samples from GP are required in many complex algorithms. Algorithm 7 presents an algorithm to sample from a multivariate Normal $\mathcal{MN}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. The sampling is done by evaluating the sampled functional values at a finite set of points. Once this method is at hand, one can sample from GP prior by invoking Algorithm 7 with mean zero and $K(X_*, X_*)$ as covariance. Similarly, sampling from GP posterior can be done by invoking Algorithm 7 with mean $\boldsymbol{\mu}_*$ and covariance $\boldsymbol{\Sigma}_*$.

Algorithm 7: Sampling from GP

Input : mean $\boldsymbol{\mu}^{n \times 1}$, covariance matrix $\boldsymbol{\Sigma}^{n \times n}$, sample size p
Output : Each column of $\mathbf{f}^{n \times p}$ as sample function

- 1 $LL^T \leftarrow \text{CholeskyDecomposition}(\boldsymbol{\Sigma}^{n \times n})$
- 2 $\mathbf{u}^{n \times p} \sim \mathcal{N}(0, 1)$
- 3 $\mathbf{f}^{n \times p} \leftarrow \boldsymbol{\mu}^{n \times 1} + L^{n \times n} \mathbf{u}^{n \times p}$

3.2.1 A toy example

Figure 3.1a shows a toy example having seven training points consisting of $\{(x_i, y_i)\}_{i=1}^7, x_i, y_i \in \mathbb{R}$ where x_i, y_i are input and output respectively. Now, GP regression is applied on this toy example with SQE kernel presented in equation 3.2.1 having its length-scale hyperparameter ℓ set to 0.3 by hand. The GP predictive results are shown in Figure 3.1b where black circles represent the training data, red stars are predictive means with their uncertainties shown in the pink shaded region. Ten samples from GP prior are shown in Figure 3.2a and ten samples from GP posterior are shown in Figure 3.2b. Figure 3.3a and 3.3b show GP predictions on the same example with different values of ℓ . Hence, it can be seen that the value of the hyperparameters are unknown and needs to be learned from the data to avoid erroneous predictions, rather than imposed by hand which is presented in the subsequent sections.

3.2.2 Constrained prediction and Gaussian Process

Generally a GP does not pose any constraint on its predictive computation; however, when the application demands the outputs to abide by certain physically-motivated constraints, then said constraints need to be taken on board to render predictions correct. For example, if in the application, the output is a probability, then one needs the GP output prediction to lie between 0 to 1. Now, I present an illustration of a method to ensure reliable uncertainty percolation, within supervised learning performed using Gaussian Processes (GP), and Markov Chain Monte Carlo based inference. I show the effect of variously propagating the uncertainty, on predictions undertaken on the output variable, at test inputs, subsequent to the learning of

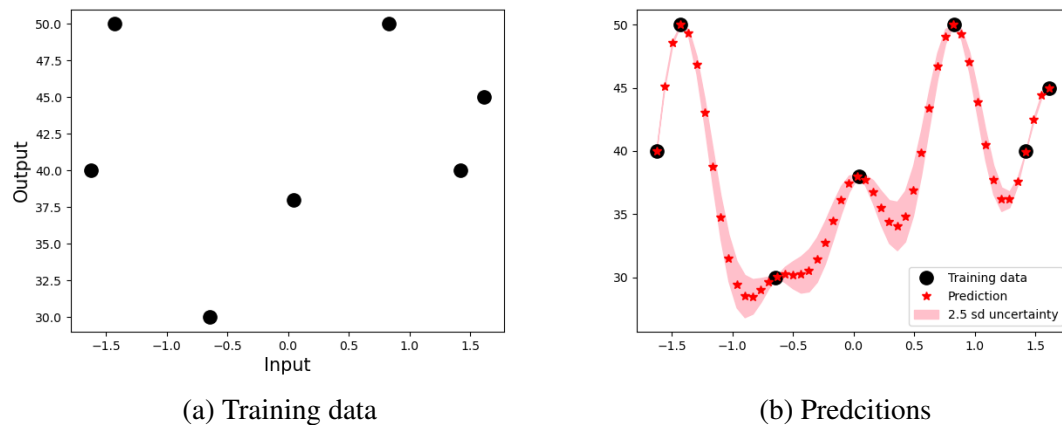


Fig. 3.1 Toy example of GP modeling a one-dimensional function with $\ell = 0.3$ with training data and GP predictive mean.

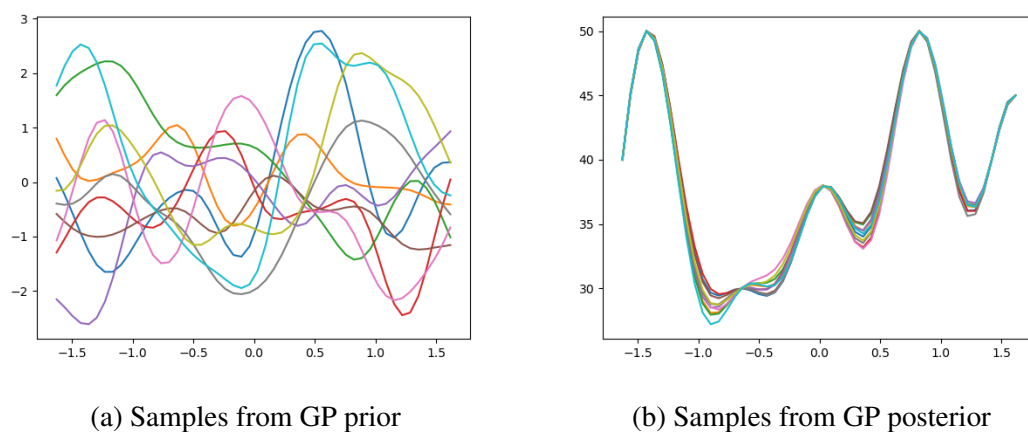


Fig. 3.2 Ten samples from GP prior and GP posterior from the GP modeling a one-dimensional function with $\ell = 0.3$ on the toy example.

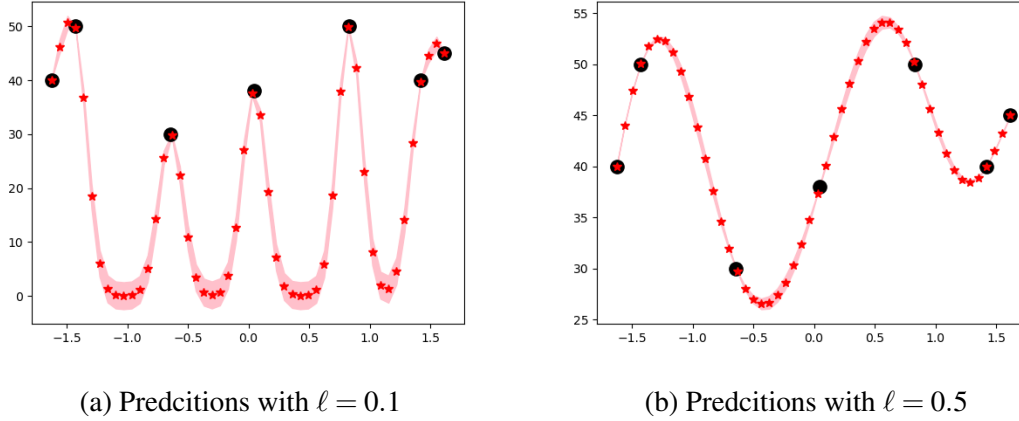


Fig. 3.3 Example of GP modeling a one-dimensional function with different ℓ on the toy example.

the functional relationship between the input and the output, where this functional relation is modelled as a realisation from a GP. The efficiency of imposing a physically motivated constraints on the output - via priors imposed on the GP covariance kernel hyperparameters - is compromised under certain strategies adopted to propagate uncertainty.

3.3 Data and model

I use the real test flight data on O-ring failure status Y presented in the previous chapter. Here the input is a scalar, namely, temperature $X \in \mathbb{R}$ at which test flights of the Space Shuttle were undertaken, The failure status variable, i.e. the output variable, is binary, and is coded to attain 1 if the O-ring fails; else $Y = 0$. Table 2.1 depicts this test flight data, I refer to this data as D . Given that the output is binary, its regression against the predictor variable (temperature X) will be modelled with the logistic regression model (Robert and Casella, 2004; Young, 2018). Thus, given that I have only one predictor variable, the linear predictor in this regression model has two regression coefficients, namely, the slope $\beta \in \mathbb{R}$ and the intercept parameter $\alpha \in \mathbb{R}$. Under the assumption of *iid* data points, likelihood is the product of the probability of each value of Y , at the corresponding temperature. This probability is the Bernoulli probability mass function, with parameter that is a function of temperature. I use Gaussian priors on each parameter, and ultimately write down the joint posterior of the slope and intercept parameters given the test flight data. Inference on model parameters is undertaken by performing posterior sampling with MCMC to compute the marginal posterior

of each parameter given the data D , using which I learn the 95% Highest Probability Density credible region (HPDs) on each parameter.

Our learning of the parameters of this logistic regression model will allow us to compute the probability for O-ring failure at given temperatures. Such computation is relevant to our supervised learning endeavour, in which this $\Pr(Y = 1|X = x)$ is the output and X is the input variable. The set $D_{train} := \{(x_i, \Pr(Y = 1|x_i))\}_{i=1}^M$ of M pairs of value of this design input $X = x_i$ and the corresponding output will constitute the training data using which I will learn the functional relationship between this output and input, by modelling the said relationship as a random function that is realisation from a GP. Thus, our supervised learning endeavour will reduce to the learning of the parameters of the covariance structure of this GP. As I will parameterise the covariance structure using a parametric kernel, our supervised learning will basically reduce to our learning of the hyperparameters of this covariance kernel. There is in fact only one such hyperparameter that is relevant - this is the length scale ℓ . So I will learn ℓ using D_{train} , where D_{train} is arrived at from the learning of α and β , as motivated above.

In my work, I know the parametric form of how output values - $\{\Pr(Y = 1|X = x_i)\}_{i=1}^M$ - are distributed across the range of the input temperature values; this is the logistic function in X , with parameters α and β . This knowledge will benefit the inspection of the learnt input-output functional relation that is modelled with a GP, (and learnt using generated training data D_{train}). The point is that the GP-based learning of this function permits prediction of the output at test temperatures, and as I know the theoretical distribution of the output against temperatures, I can check for the veracity of our predictions. Also, the uncertainty on the output value computed at a design temperature is available from the uncertainty in our learning of the slope and intercept parameters, (β and α), of the logistic regression model. This offers us a tool to check how noise in the training set affects the prediction performance.

As stated in the previous section, I could compute the output, i.e. probability of O-ring failure at a design temperature, by using a summary (eg. mean, median, etc.) of the marginal of the slope and intercept parameters - as learnt using the MCMC. I could alternatively compute the output using any value of the slope and intercept parameters from within the 95% HPDs on each such parameter, that are learnt by MCMC. I will show that with different choices from within the 95% HPD range, the parametrisation of the GP covariance structure varies, thereby affecting the final prediction. This method of computing the probability of O-ring failure at a design temperature then appears unsatisfactory.

In order to avoid this, a second way of computing this probability is to design a pipe-lined algorithm for prediction, in which I learn the slope and intercept parameters at each iteration of MCMC, given the test flight data D , and then proceed in that same iteration, to compute the probability $\Pr(Y = 1|X = x_i)$ of O-ring failure at each design temperature x_i , at the

values of α and β that are current as of this iteration. This is repeated across iterations and uncertainties in each probability value accrued across the (post-burnin) iterations of MCMC. Then by using this pipe-line, uncertainties in our learning of parameters of the original (logistic) model are best propagated into the generation of the training set D_{train} .

In the next section I will present the results obtained using the former approach towards training set generation, namely, by computing output using summaries of the parameters. I display results of predictions made following the GP-based learning of the relation between probability of O-ring failure and temperature, as well as with the DNNs. Then, in Section 3.5 I will present the pipe-lined architecture for prediction.

3.3.1 Learning parameters of the logistic regression model

For binary response variable $Y \in \{0, 1\}$ that represents O-ring failure status, and the temperature $X \in \mathbb{R}$, I want to model the probability of failure given a temperature, assuming that Y follows a Bernoulli distribution. The probability of failure given the temperature $X = x$ is modelled within logistic regression using the slope and intercept parameters $\beta \in \mathbb{R}$ and $\alpha \in \mathbb{R}$ in general.

$$P(Y = 1|X = x) = \frac{e^{(\alpha+\beta x)}}{1 + e^{(\alpha+\beta x)}}.$$

Then likelihood is depicted as follows, (with n being the number of data points which is 23 here).

$$\mathcal{L}(\alpha, \beta|D) \propto \prod_{i=1}^n \left(\frac{e^{(\alpha+\beta x_i)}}{1 + e^{(\alpha+\beta x_i)}} \right)^{y_i} \left(\frac{1}{1 + e^{(\alpha+\beta x_i)}} \right)^{1-y_i} \quad (3.3.1)$$

I use Normal priors for both α and β :

$$\pi_0(\alpha) = \mathcal{N}(\mu_\alpha, \sigma_\alpha^P) = \frac{1}{\sigma_\alpha^P \sqrt{2\pi}} e^{-(\alpha - \mu_\alpha)^2 / 2\sigma_\alpha^{P2}}; \quad \pi_0(\beta) = \mathcal{N}(\mu_\beta, \sigma_\beta^P) = \frac{1}{\sigma_\beta^P \sqrt{2\pi}} e^{-(\beta - \mu_\beta)^2 / 2\sigma_\beta^{P2}}.$$

Thus, the unscaled joint posterior $\pi(\alpha, \beta|D, \pi_0(\alpha), \pi_0(\beta))$ of the model parameters, given the test flight data D is given as the product of the aforementioned likelihood and priors.

Figure 3.4 displays results of learning α and β given data D .

The 95% HPDs on the parameters are: [14.8, 15.19], (with a mean of 14.995) for α and [-0.249, -0.216], (with a mean of -0.2325) for β .

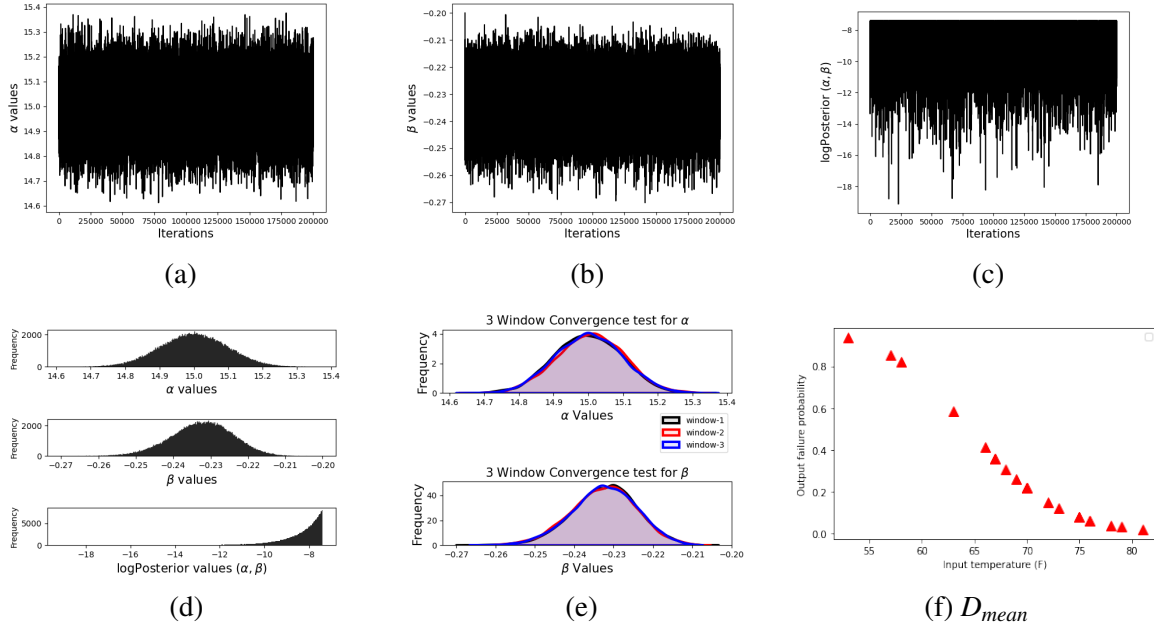


Fig. 3.4 Results of learning α, β using test flight data D . The top row depicts the trace (variation with iteration index) of α , β and logarithm of the joint posterior of these 2 parameters, given data D , as I move from the left to the right panel. In the lower rows, at the extreme left are the histogram representations of the marginal posterior of α and β given data D , and that of the log of the joint posterior. The middle column of panes in the lower row represent the histograms of α (upper) and β (lower panel) values sampled during the MCMC chain, over three distinct, equally-wide (covering 20000 consecutive iterations), non-overlapping iteration windows. the three histograms are coloured in blue, red and black, and overplotted. The right most panel in the lower row depicts the $\Pr(Y = 1|X = x_i)$ computed at 16 different temperatures using the mean of the α and β learnt in this chain, using data D . This MCMC chain was run using a Metropolis-within-Gibbs algorithm. The total number of iteration in this chain is 200000; the variances of the truncated Normal proposal densities for α and β are 0.074^2 , and 0.008^2 respectively. Normal priors were used on the parameters.

3.4 Learning with training data sets generated at parameter summaries with DNN exposition

In this section, first I discuss the generation of training data D_{train} comprising pairs of values of design input, and the probability for Y to be 1 at that design temperature. As motivated above, I will undertake this generation in two distinct ways - for the D_{train} generated under a given approach, I refer to it by its updated name.

In the first approach, at the end of the MCMC chain that is used to learn the parameters α and β of the logistic regression model, I compute the mean of the sampled α and β , to generate the $\Pr(Y = 1|X = x)$, at the design temperature value x ; such $(x, \Pr(Y = 1|x))$ pairs will constitute a training set D_{mean} (depicted in the bottom right panel Figure 3.4). D_{mean} is one of the training data sets that I will use to learn the functional relationship between this output (i.e. $\Pr(Y = 1|x)$) and the input X . (As stated above, such supervised learning will reduce to our learning of the length scale hyperparameter ℓ of the covariance structure of the GP that I will model our sought function with).

Another training data D_{right} will be built by using the learnt values of α and β at the right edge of the 95% HPD learnt on α and β using data D . In principle, any values from within the 95% HPDs on α and β - inferred upon using data D - can be used in this context.

I will perform with-uncertainty learning of the kernel hyperparameter ℓ , to undertake predictions on the output variable - which is $\Pr(Y = 1|X = x)$ - at test inputs. When making this learning of the relationship between the output $\Pr(Y = 1|x)$ and input X , there is no information available to our learning mechanism to have the output predictions - at test temperatures - to remain confined to the interval $[0,1]$. In other words, the supervised learning algorithm does not distinguish on the basis of the nature of the output variable. I will impose such a constraint on the learning to ensure meaningful predictions, using MCMC-based inference.

3.4.1 Learning ℓ under the constraint given D_{mean}

I attempt learning the function $f(\cdot)$, where $W = f(X)$, with the output $W \equiv \Pr(Y = 1|X = x)$, s.t. the unknown $f : \mathbb{R} \rightarrow [0, 1]$. I do this by treating $f(\cdot)$ as an unknown, which in the Bayesian paradigm translates to $f(\cdot)$ being modelled as a random variable, or rather a random function in this case. Then by definition, $f(\cdot)$ is ascribed a probability distribution. A probability distribution on the space of functions is a stochastic process. So I model $f(\cdot)$ as a realisation from a process. For maxima generalisability and ease of computation, I choose this process to be a GP, s.t. minimal constraints are imposed on the sample function $f(\cdot)$.

So $f \sim GP(\mu(x), cov(x, x'))$, where $\mu(\cdot)$ and $cov(\cdot, \cdot)$ are mean and covariance functions of this GP. Then by definition, the joint probability of a finite number of realisations of $f(\cdot)$ - such as the M realisations of $f(\cdot)$ at each of the design points in the training data D_{train} - is Multivariate Normal, with the M -dimensional mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}^{(M \times M)}$. I parameterise matrix $\boldsymbol{\Sigma}$ by saying that the ij -Th element of this matrix that represents the covariance between W_i and W_j , is modelled as a declining function $K(\cdot, \cdot)$ of the distance between the inputs s_i and x_j at which these outputs are realised. Here $i, j = 1, \dots, M$. This distance is chosen to be Euclidean, and $K(\cdot, \cdot)$ a Squared Exponential (or SQE) kernel function, s.t. $\boldsymbol{\Sigma} = [cov(W_i, W_j)] = [K(x_i, x_j)]$, with

$$K(x_i, x_j) := \exp\left(-\frac{(x_i - x_j)^2}{\ell^2}\right), \quad \ell > 0.$$

I choose to model the covariance structure of this GP using an SQE kernel with a fixed length scale hyperparameter $\ell > 0$. I compute the empirical mean $\bar{w}$ of the output values $f(x_1) = w_1, \dots, f(x_M) = w_M$ that live in the training data D_{Train} , as each component of $\boldsymbol{\mu}$.

So as the joint of $f(x_1), \dots, f(x_M)$ is Multivariate Normal, with mean $\boldsymbol{\mu}$, and covariance matrix $\boldsymbol{\Sigma} = [K(x_i, x_j)]$, it implies that the joint of $w_1, \dots, w_M$ is this Multivariate Normal. But the joint of these M values of the output W that live in training data, is the probability of the data on W . In fact, it is a conditional probability - conditional on the parameters of the mean and covariance, i.e. on ℓ . But the probability of the data conditional on model parameters is the likelihood. Thus, the likelihood is Multivariate Normal with mean $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma}$ that is kernel parameterised as above, with hyperparameter ℓ , i.e. the likelihood is

$$\mathcal{L}(\ell|D_{train}) = \frac{1}{\sqrt{(2\pi)^M |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{w} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{w} - \boldsymbol{\mu})\right),$$

where $\mathbf{w} = (w_1, \dots, w_M)^T$, and $\boldsymbol{\Sigma}$ is kernel parameterised using the SQE kernel with hyperparameter ℓ . I impose Gaussian priors on ℓ . Then the unscaled posterior on ℓ given the training data is this likelihood times the Gaussian prior. I perform learning with Metropolis Hastings in which ℓ is proposed from a truncated Normal density, and training data that is generated as the data D_{mean} , using the mean of the sampled α and β that are learnt using D . I update ℓ in each iteration of the chain, and subsequently, predict the mean and variance of the output W at test values of the input. However, these predictions do not necessarily offer output values to remain in $[0, 1]$.

To remedy this, I impose the constraint that I learn only those ℓ that predict outputs at any test input, to lie in $[0, 1]$. I accomplish this by inputting a prior on ℓ that offers a density

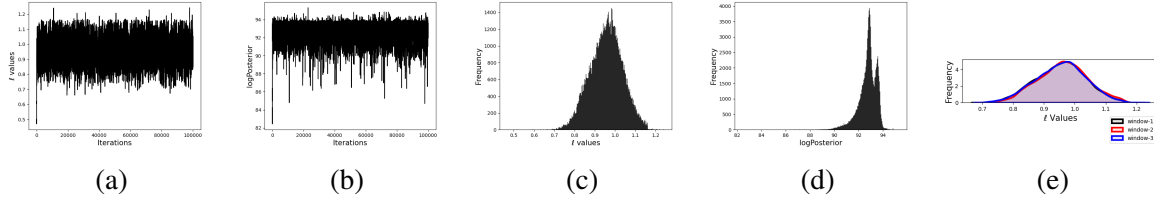


Fig. 3.5 Results of learning ℓ using the generated training data D_{mean} in which, at design temperature x_i , the output ($w_i \equiv \Pr(Y = 1|X = x_i)$) is computed using the mean of the learnt α and β . Here the 2-stepped prior is imposed on ℓ . The top left and second-to-left panels respectively depict traces of learnt values of ℓ and logarithm of posterior of ℓ given training data D_{mean} . The 2 further panels to the right present the histogram representations of the marginal of ℓ given D_{mean} and the histogram of the posterior values. The bottom row depicts the 3-window convergence test, (see caption of Figure 3.4) to confirm convergence of the MCMC chain.

of $\rho_{hi} > 0$ if the predictions on an arbitrarily chosen set of test temperatures are within $[0,1]$, and the prior is set at the value ρ_{lo} .

In the t -th iteration of the MCMC chain, after proposing ℓ as $\ell^{*,t} \sim \text{Truncated Normal}(0, \ell^{(t-1)}, 0.03^2)$, the step prior function is computed using a 2-stepped prior as discussed below. Here $\ell^{(t-1)}$ is the current value of ℓ in the $t - 1$ -th iteration of the MCMC chain.

$$f_{stepPrior}(\ell_{prop}) = \begin{cases} \rho_{hi}\pi_{0,Normal}, & \text{if output } W \text{ predicted at test inputs using } \ell = \ell_{prop} \in [0, 1] \\ \rho_{lo}\pi_{0,Normal}, & \text{otherwise,} \end{cases} \quad (3.4.1)$$

where $\pi_{0,Normal}$ is the Normal $\mathcal{N}(0.9, 0.08^2)$ prior placed on ℓ . I choose $\rho_{hi} = 1$, and $\rho_{lo} \leq e^{-10^3}$.

The MCMC chain that I run comprises 100000 iterations; and the 95% HPD learnt on the ℓ using D_{mean} is $[0.79509924, 1.10960904]$, with mean 0.9523 approximately. Figure 3.5 shows results obtained by running this chain. Figure 3.8a shows the GP prediction for this set up using $\ell \approx 0.9523$.

3.4.2 Learning with DNNs

I have also implemented Deep Neural Network systems with varying architectures to make prediction of output $W \equiv \Pr(Y = 1|X = x_{test})$ at test temperatures. Here the DNN architecture varies in number of hidden layers (one to four), with 64 neurons in each layer. I make these predictions using Tensorflow, following the code that is available in the official documentation/tutorial cite of Tensorflow (Tensorflow, 2022). To allow for compatible comparison, here I perform predictions at the same test data, as used when output predictions

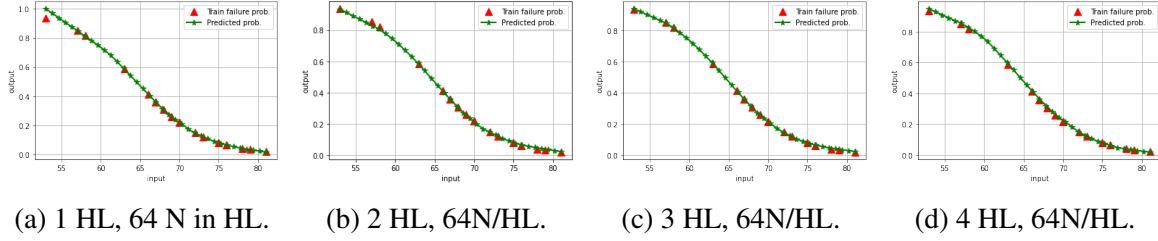


Fig. 3.6 DNN predictions with varying architectures. “HL”: Hidden layer, “N”: Neurons, “64N/HL”: 64 neurons in each hidden layer.

were made following the learning of ℓ using MCMC, as in subsection 3.4.1. I perform these DNN-based predictions using Adam optimiser that has a learning rate=0.001, loss function='mean_absolute_error', epoch=500.

All the four predictive plots from DNN are shown in Figure 3.6 The comparative results of DNN and GP are presented in the Table 3.1 and it is to be noted that DNN results are sensitive to the choice of architecture, even for this simplistic data set. Such sensitivity is cause for worry when reliable and robust ML implementation is sought. For more complex real-world data sets with higher noise, such sensitivity is likely to be more pronounced. Also, DNN does not produce the (probability) predictions within the strict range of 0 and 1, while predictions following GP-based learning could be successfully constrained to lie in this interval.

3.4.3 Learning ℓ under constraint, with MCMC, using D_{right}

Instead of using the training data D_{mean} one could potentially use any value of α and β from within 95% HPDs learnt on these parameters, in the chain run with test flight data D , to compute $\Pr(Y = 1|X = x_i)$ where x_i is the i -th design temperature; $i = 1, \dots, M$. The training data that will then result, will not be D_{mean} since the output at any design temperature in D_{mean} is computed at the mean of the α and β samples generated in the MCMC chain run with test flight data D . As stated above, when I populate a training data by computing $\Pr(Y = 1|X = x_i)$ using the value of $\alpha=15.19$ and of $\beta=-0.216$ from the right-most edge of their respective 95% HPD that is learnt in this MCMC chain, for $i = 1, \dots, M$, get the training set that I refer to as D_{right} .

Results of GP-based learning of ℓ undertaken with training data D_{right} are presented in Figure 3.7. From this chain, I learn the 95% HPD on ℓ to be [0.7999134, 1.11657139], with mean ℓ of 0.9582 approximately. Figure 3.8b shows the GP prediction for this set up using $\ell = 0.9582$.

Table 3.1 Prediction of output W at test inputs - with DNNs of varying architectures, and following GP-based learning of the length scale hyperparameter (ℓ) of the covariance kernel that parametrises the GP covariance structure. Below, “HL” stands for number of Hidden Layers in the DNN, and “N” for number of Neurons.

1 HL, 64N	2 HL, 64N each	3 HL, 64N each	4 HL, 64N each	GP predictive mean
1.0310167	0.9359667	0.938025	0.9486213	0.9323123
0.9939897	0.91862655	0.91935253	0.9286917	0.934071
0.9569628	0.90128636	0.90068007	0.9093453	0.926627
0.9199359	0.8839463	0.8820075	0.8906931	0.911141
0.88290906	0.86660624	0.863335	0.87204087	0.888929
0.8458822	0.84823	0.8443388	0.8533887	0.861319
0.8088551	0.8216592	0.818619	0.82862586	0.82952
0.7718283	0.79057	0.78710705	0.80185705	0.794533
0.7348015	0.7537405	0.75483435	0.77277607	0.757104
0.6977744	0.7143501	0.7184024	0.73662823	0.717731
0.6607475	0.6749598	0.6793759	0.69309384	0.676709
0.62372065	0.6355695	0.63738376	0.6467888	0.634208
0.5862205	0.5937305	0.59471357	0.599609	0.590366
0.5407498	0.5457597	0.54709804	0.55195147	0.545373
0.4952766	0.49778882	0.4992079	0.5042205	0.49954
0.44980314	0.44990715	0.45125672	0.45653468	0.453327
0.40433016	0.4031423	0.40382358	0.40979505	0.407338
0.3588567	0.35748953	0.35848558	0.36351866	0.362288
0.31549928	0.31303966	0.31533292	0.32044703	0.318933
0.27524063	0.2705538	0.27200228	0.2804457	0.277996
0.23979539	0.23742433	0.24069557	0.2418064	0.240096
0.20099688	0.19988464	0.20456474	0.2145113	0.205685
0.16196822	0.16679865	0.17303263	0.18275131	0.175016
0.14389752	0.14620048	0.14776252	0.15101965	0.148136
0.12673585	0.12363774	0.12680508	0.12768927	0.124907
0.10957434	0.10223782	0.1055288	0.10772515	0.105044
0.09241267	0.08653921	0.09068373	0.08748056	0.088173
0.07525112	0.07084078	0.07638919	0.0738593	0.073881
0.06405788	0.05796587	0.06551494	0.06084543	0.06177
0.05543705	0.04907831	0.0565593	0.0475015	0.05149
0.04681628	0.04019085	0.04897907	0.04064081	0.042759
0.03819548	0.03227264	0.04272956	0.03435563	0.035364
0.02957465	0.02511932	0.03683984	0.02806261	0.029157
0.02095388	0.0179661	0.03095011	0.0229567	0.024029
0.0123331	0.01081278	0.02506035	0.01791653	0.0198959

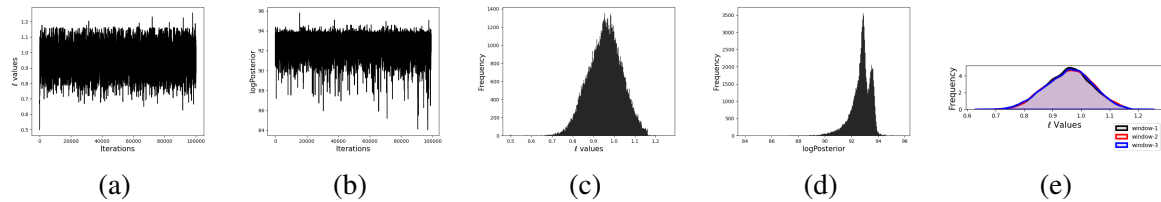


Fig. 3.7 As in Figure 3.5, except these depict results obtained using training data D_{right}

Uncertainty-included Output Prediction

Once ℓ is updated in any iteration of the MCMC chain run with training data D_{train} - which could be D_{mean} or D_{right} - I predict the output W at a test input. These predicted mean outputs at each of the considered test temperatures are plotted against temperature, in green, in Figure 3.8; W from the training data at a design temperature is depicted in these figures in red triangles, while the uncertainty predicted at a test input, on the output, is depicted as the salmon-pink shaded region. This depicted “uncertainty” is 2.5 times the standard deviation that is predicted - in a closed form way, along with the mean - at any test temperature. This is an advantage of performing learning with GPs; the output mean and variance predictions are closed-form. One concern that I have about this learning is that the choice of the training set that was employed in undertaking this learning appears ambiguous. In fact, percolation of the uncertainties in the learning of α and β is not correct in the formulation of these training data. In particular, when I examine the values of the predicted outputs and the uncertainties predicted on the outputs at any test temperature, I can firstly see that the result varies depending on which training data I have considered in the learning, and secondly, the result shows that some probabilities within the uncertainty bands, are predicted as being in excess of 1. This is in spite our 2-stepped prior that I imposed to ensure that all output values stay restricted to $[0,1]$.

In the left panel of Figure 3.8, the result of predictions using D_{mean} are shown, while on the right, the same is shown when D_{right} is used.

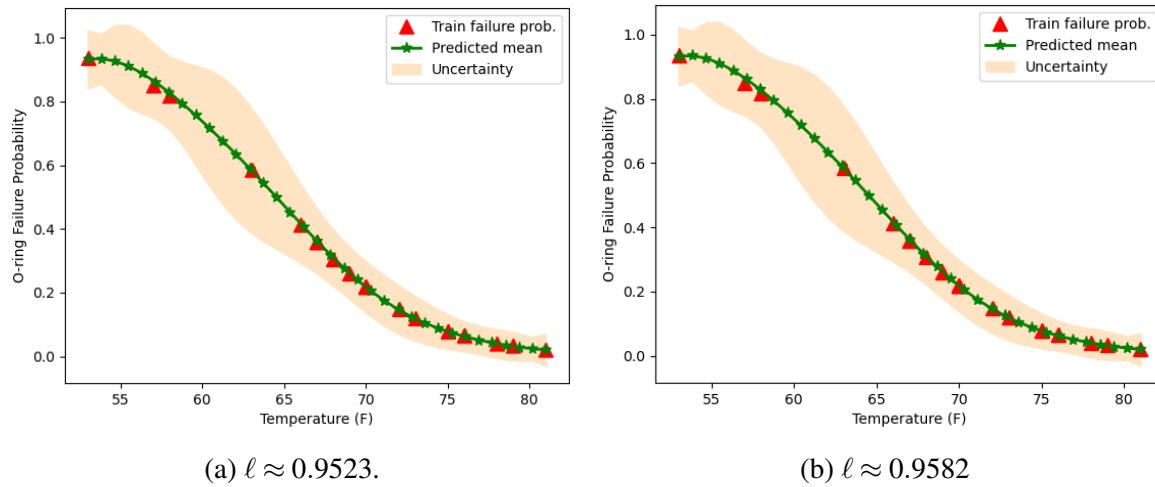


Fig. 3.8 GP predictions with 2.5 standard deviation of uncertainties with different ℓ s

3.5 Efficient uncertainty percolation using pipe-lined architecture

As I have seen in the previous section, predictions following GP-based learning can result in unacceptable uncertainties, and ambiguous results stemming from generation of training sets based on ambiguity in summarising uncertainties in the earlier stages. In this section I will illustrate a pipeline architecture for efficient handling of uncertainties which can also be extended to higher dimensional data. Figure 3.9 represents the two approaches of uncertainty modelling and percolation as flow charts. The pipeline architecture consists of three blocks within the MCMC chain that I run. These blocks are executed sequentially within each iteration of the MCMC chain, so that uncertainty gets propagated through the stages without incorporating further errors that can creep in via arbitrary summarisation of learning outcomes in previous stages.

Broadly, inside an iteration of the MCMC chain, in the first block, α, β are updated - as within an iteration of Metropolis Hastings - using the test flight data D . In the second block of this iteration, the training data D_{train} is then computed at the current α and β values, and subsequently the GP kernel hyperparameter ℓ is updated using this current training data D_{train} , under the 2-stepped prior to ensure that output predictions remains within 0 and 1 at all considered temperatures. Then in the third block of this iteration, with the current value of ℓ , closed form mean and variance predictions of output values are undertaken, at each test temperatures. Hence, after the full iteration is over, I get traces of the predictive means at each test temperature. I use the range of values of the output $W \equiv \Pr(Y = 1 | X = x_{test})$ sampled across the iterations, at each test input x_{test} , to compute the 95% HPD credible region. I compute the mean and the standard deviation of this sample, as the central prediction at test temperature x_{test} , with uncertainties of 2.5 times this sample standard deviation on either side of the mean, as the uncertainty in the prediction. The predicted variance at each test temperature is also recorded, and this is compared to the uncertainty of predictions obtained using the variation across MCMC samples.

First block In an iteration of the MCMC chain, in the first block α, β are learnt using data D , with Metropolis Hastings, with the same configuration that is used in Subsection 3.3.1. The log of the posterior defined in this subsection is used. Gaussian priors are used for both α, β . Once, the burnin phase is over, then next stages are switched on.

Second block In this current iteration of the MCMC chain, the O-ring failure probability is computed at design temperatures $x_1, \dots, x_M$ using the current values of α, β as in Sec-

tion 3.3.1. This newly computed failure probability now becomes the output computed at a design input. When performed over the whole set of design inputs, the training data D_{train} is generated. this is employed in learning the GP kernel hyperparameter ℓ , as depicted in the earlier Subsection 3.4.1.

Third block In the third block of this iteration, predictive mean and variance of the output at each test temperature is computed, using the current ℓ .

Figure 3.10 represents the results of learning the parameters α and β , using the test flight data, within the MCMC chain - to update the training set $\{(x_i, w_i)\}_{i=1}^M$ - that simultaneously learns the hyperparameter (ℓ) of the covariance structure of the GP that models the functional relation between output W and input X . Here, the output W here is the probability of the O-ring failure and the input X is the ambient temperature to which such an O-ring is exposed. At each iteration of the MCMC chain, the uncertainty-included outputs are predicted at each test temperature. Thus, at the end of the MCMC chain, I predict values of mean and standard deviation of the output at each x_{test} ; such predictions are shown in Figure 3.11. Traces of predicted values of the output W at few of the considered test temperatures are depicted in Figure 3.12, along with uncertainties.

Algorithm 8 depicts the algorithm of the pipeline with the following inputs. $\alpha_0, \beta_0, \ell_0$ are seeds value for α, β, ℓ ; $\sigma_\alpha, \sigma_\beta, \sigma_\ell$ are jump scales for α, β, ℓ respectively; μ_α, μ_β prior mean and $\sigma_\alpha^p, \sigma_\beta^p$ are prior standard deviation for α and β respectively, number of maximum iterations N ; N_ℓ is starting iteration for learning ℓ , n number of test data points, ε constant noise for GP.

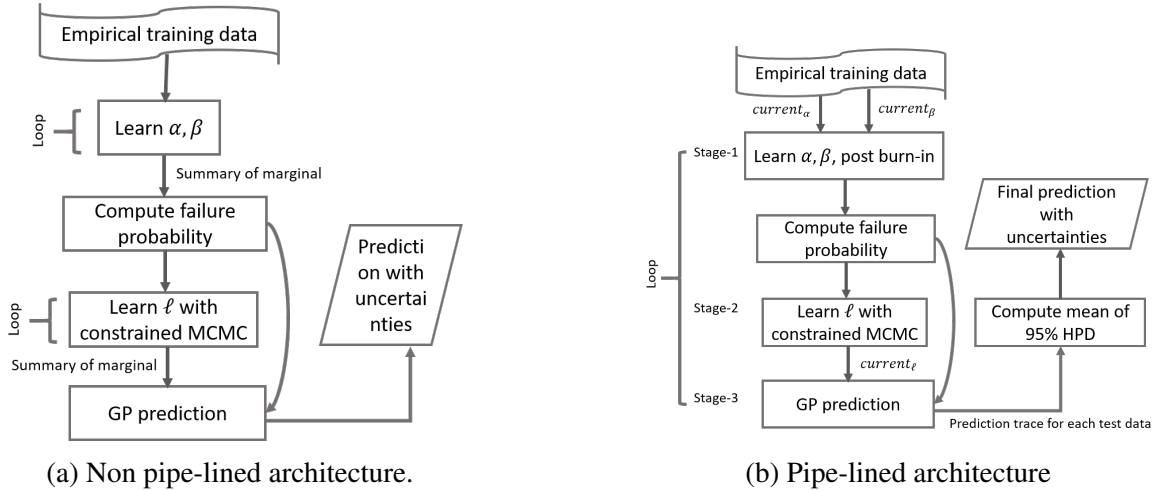


Fig. 3.9 Two approaches of dealing with uncertainty

Algorithm 8: Pipeline for Forward Prediction

```

1 Set  $\alpha[0] \leftarrow \alpha_0, \beta[0] \leftarrow \beta_0, \ell[0] \leftarrow \ell_0$ 
2  $\mathbf{x} \leftarrow$  Non-standardised input vector from D
3  $\mathbf{x}^{\text{std}} \leftarrow \text{Standardise}(\mathbf{x})$ 
4  $\mathbf{x}_{\text{test}} \leftarrow$  generate n test data within range of  $\mathbf{x}^{\text{std}}$ 
   /* Block-1: MH for learning  $\alpha, \beta$  */
5 for  $i \leftarrow 1$  to  $N$  increment by 1 do
6    $\alpha_{\text{current}} \leftarrow \alpha[i-1], \beta_{\text{current}} \leftarrow \beta[i-1]$ 
7    $\alpha_{\text{proposed}} \sim \mathcal{N}(\alpha[i-1], \sigma_\alpha), u \sim U(0, 1)$ 
8    $a \leftarrow \log(\pi^I(\alpha_{\text{proposed}}, \beta[i-1] | D, \pi_0^\alpha, \pi_0^\beta)) - \log(\pi^I(\alpha[i-1], \beta[i-1] | D, \pi_0^\alpha, \pi_0^\beta))$ 
9   if  $a > \log(u)$  then
10    |  $\alpha[i] \leftarrow \alpha_{\text{proposed}}$ 
11   else
12    |  $\alpha[i] \leftarrow \alpha[i-1]$ 
13   end
14    $\beta_{\text{proposed}} \sim \mathcal{N}(\beta[i-1], \sigma_\beta)$ 
15    $b \leftarrow \log(\pi^I(\alpha[i], \beta_{\text{proposed}} | D, \pi_0^\alpha, \pi_0^\beta)) - \log(\pi^I(\alpha[i], \beta[i-1] | D, \pi_0^\alpha, \pi_0^\beta))$ 
16   if  $b > \log(u)$  then
17    |  $\beta[i] \leftarrow \beta_{\text{proposed}}$ 
18   else
19    |  $\beta[i] \leftarrow \beta[i-1]$ 
20   end
   /* Block-2: Vanilla MCMC for learning  $\ell$  */
21  $\mathbf{y}_p \leftarrow \text{ComputeFailureProbability}(\alpha_{\text{current}}, \beta_{\text{current}}, D)$  as in Subsection 3.3.1
22  $\mathbf{y}_p^{\text{std}} \leftarrow \text{Standardise}(\mathbf{y}_p)$ 
23 if  $i \geq N_\ell$  then
24    $j \leftarrow (i - N_\ell), \ell_{\text{current}} \leftarrow \ell[j-1], \pi_{\text{current}}^{\text{II}} \leftarrow \pi^{\text{II}}[j-1], a_\ell \leftarrow -\infty, b_\ell \leftarrow +\infty$ 
25    $\ell_{\text{prop}} \sim \mathcal{N}_T(\ell_{\text{current}}, \sigma_\ell, a_\ell, b_\ell)$ 
26    $p \leftarrow f_{\text{stepPrior}}(\mathbf{x}^{\text{std}}, \mathbf{y}_p, \mathbf{x}_{\text{test}}, \ell_{\text{prop}}, \varepsilon)$ 
27   if  $p = 1$  then
28      $\ell_Q \leftarrow$ 
29        $(\log(\mathcal{N}_T(\ell_{\text{current}} | \ell_{\text{prop}}, \sigma_\ell, a_\ell, b_\ell)) - \log(\mathcal{N}_T(\ell_{\text{prop}} | \ell_{\text{current}}, \sigma_\ell, a_\ell, b_\ell)))$ 
30      $\pi_{\text{prop}}^{\text{II}} \leftarrow \log(\pi^{\text{II}}(\ell_{\text{prop}} | \mathbf{y}_p^{\text{std}}, \mathbf{x}, \pi_0^\ell))$ 
31     if  $(\pi_{\text{prop}}^{\text{II}} - \pi_{\text{current}}^{\text{II}} + \ell_Q) > \log(u)$  then
32       |  $\ell[j] \leftarrow \ell_{\text{prop}}, \pi^{\text{II}}[j] \leftarrow \pi_{\text{prop}}^{\text{II}}$ 
33     else
34       |  $\ell[j] \leftarrow \ell_{\text{current}}, \pi^{\text{II}}[j] \leftarrow \pi_{\text{current}}^{\text{II}}$ 
35     end
36   else
37     |  $\ell[j] \leftarrow \ell_{\text{current}}, \pi^{\text{II}}[j] \leftarrow \pi_{\text{current}}^{\text{II}}$ 
38   end
   /* Block-3: prediction following GP-based learning */
39    $\{\mathbf{y}_o, \sigma_{\text{gp}}\} \leftarrow \text{GPprediction}(\ell[j], \mathbf{x}^{\text{std}}, \mathbf{y}_p, \mathbf{x}_{\text{test}}, \varepsilon)$ 
40 end

```

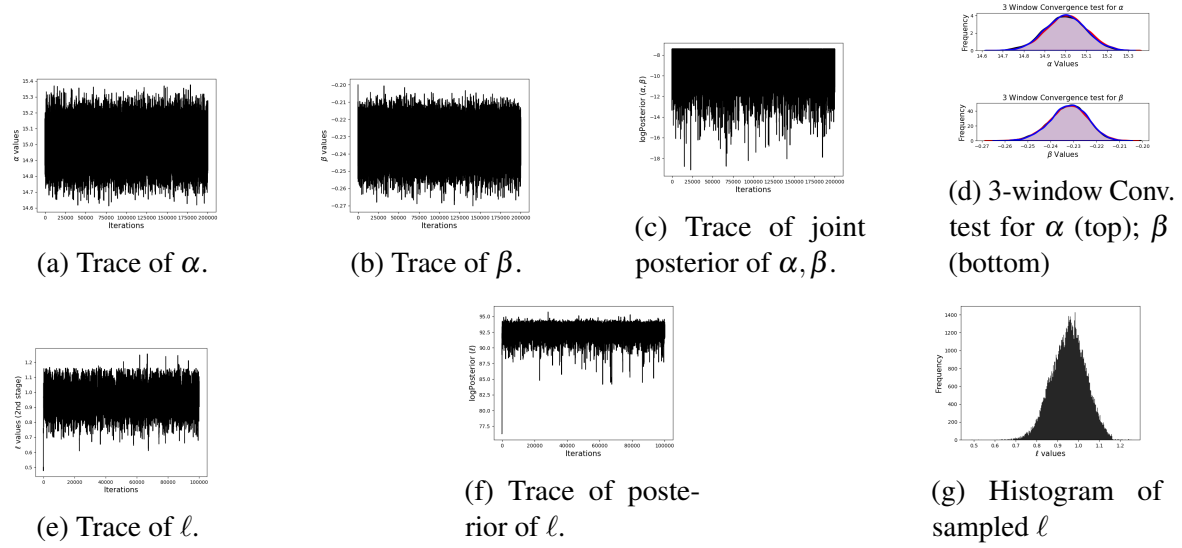


Fig. 3.10 Traces and plots for learning α, β, ℓ in pipe-lined architecture; above, “Conv.” substitutes for “convergence”

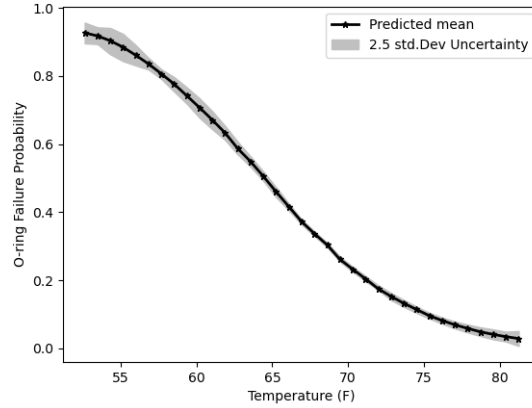


Fig. 3.11 In the MCMC chain that updates the α and β parameters, while learning ℓ , predicted mean output $W \equiv \Pr(Y = 1|X = x_{test})$, at each test input x_{test} , is depicted in filled circles. Said prediction follows learning of the input-output functional relationship by modelling this sought function with a GP, the covariance structure of which is parameterised by ℓ . 2.5 times the predicted standard deviation of the output value is depicted on each side of the predicted mean, within the grey shaded region.

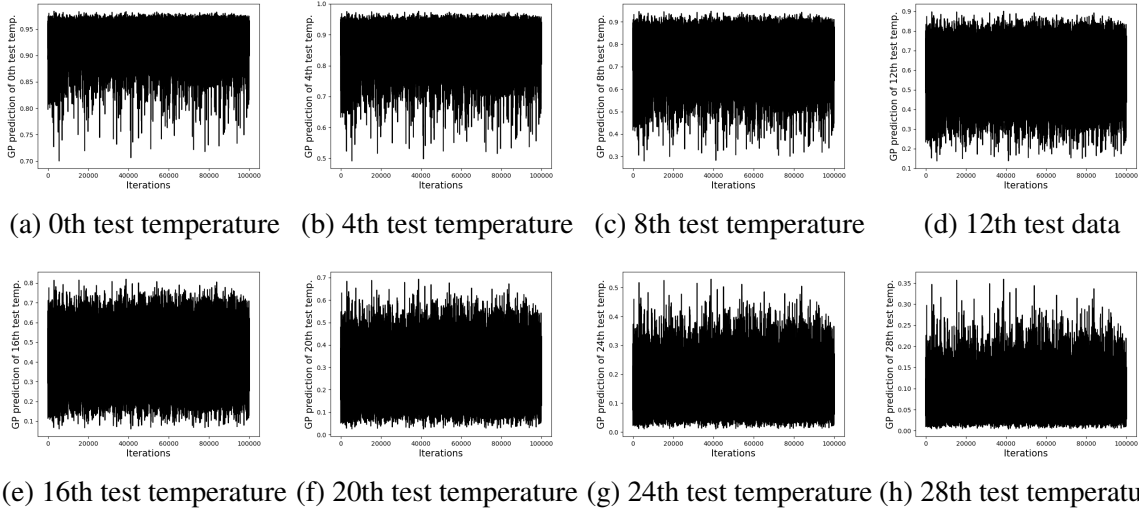


Fig. 3.12 Variation of predicted mean of the output $W \equiv \Pr(Y = 1|X = x_{test})$ with iteration index, at few values of x_{test}

3.6 Conclusion

This chapter presents the background related to GP-based regression, following a examples and depict how to sample from a GP. Then I present a robust method for correct propagation of uncertainty in the context of a small real-world data where both input and output is scalar-variate. As the presented method is generic, it can be applied to the regression setup with any complex, sensitive, precision-aware real-world data situation where prediction of uncertainty interval is necessary. I have undertaken two different Bayesian approaches of uncertainty percolation to compare the quality of predictions made in cases and showed an efficient way of quantification of uncertainty. This chapter also presents inference with under-constraint MCMC by designing appropriate prior in order to achieve the constraint while learning the parameters.

I have also performed predictions with DNN models for comparison. The results obtained from the DNN models are sensitive to the parameters of the architecture. It is possible that with further parameter tuning, better DNN model and architecture can be found by trial-and-error to produce better predictions, however, the need for such tuning is worrying in the context of the reliability of the predictions, specially when the true value at the new test points are not known in real-world problems and thus the optimal configuration of the DNN architecture is unknown, given the training data. In later chapters, I will also show that DNNs can yield diversely incorrect predictions, based on the architecture of the DNN and other model parameters.

Chapter 4

A Fully Non-parametric, Non-stationary, Parsimonious, Kernel for Supervised Learning

4.1 Introduction

In supervised learning, one needs to learn the relationship between two variables X and Y where these two variables are associated. When this functional relationship is nonlinear, it poses a special challenge to the learning with the small-to-medium sized data that bears inhomogeneities in its correlation structure. Let this relationship be denoted by $Y = f(x)$. Learning this sought function $f(\cdot)$ facilitates further analytical tasks such as prediction, imputing missing values etc. This sought inter-variable functional relation $f(\cdot)$ can be modelled as a random function. In general, X is the system parameter vector with single to multiple parameters and Y is the observable. Y can be a scalar or vector or be tensor-valued. This chapter deals with the situation when both X and Y are scalar-valued, which is a common situation in many real-world data sets. In the next chapter, I will consider the case of a multivariate X and scalar-valued Y .

If a parametric form is assigned to the sought function $f(\cdot)$ - such as fitting splines/wavelets to the data - then this modelling strategy becomes infeasible in high-dimensional situations and moreover, this parametric fitting strategy ignores the measurement or observational uncertainties on the values of the outputs that are realised at the respective design inputs. Additionally, parametric modelling does not capture the correlation structure of the $f(\cdot)$. Under these circumstances, non-parametric learning of the sought function $f(\cdot)$ is desirable, where the very form of the sought function is learnt.

In the probabilistic learning set up, the sought function $f(\cdot)$ can be modelled as a random function, given the training data that comprises values in pairs, one of which is the design input X and another is the value of the observable Y realised at that design input. In the Bayesian paradigm, a random variable or a random function takes a value or form respectively with a certain probability which is given by the probability distribution of this variable or function. Now, the probability distribution on the space of functions is defined as a stochastic process. Therefore, treating the sought functional relation $f(\cdot)$ as random function is actually same as $f(\cdot)$ as a sample function from a stochastic process.

I choose a Gaussian Process (GP) (Rasmussen and Williams, 2006) from the vast repertoire of the stochastic processes, to model $f(\cdot)$ for reasons such as:

- it imposes minimal restrictions on the sought function,
- it incurs easy computation,
- it allows for generalisability across dimensions.

As GP-based models learn the sought function non-parametrically, these models yield high quality of prediction and thereby GP-based learning has become a popular choice in the domain of Machine Learning and Statistics.

It is to be noted that the model here is $Y = f(x)$, unlike other models such as linear regression, where error term ζ is added to the output $Y = f(x) + \zeta$. Instead, the noise in the output training data Y and uncertainties in the learnt function $f(\cdot)$ are considered during the computation of the closed-form, uncertainty-included prediction of output values at a given test input.

By definition (Rasmussen and Williams, 2006), a GP is a collection of random variables and any finite number of those random variables are jointly distributed as a Multivariate Normal. This implies that the joint probability density of $f(\cdot)$ computed at the design input points is Multivariate Normal with a mean and covariance matrix. Therefore, the joint probability density of $Y_1, Y_2, \dots, Y_n$, - where $Y_i = f(x_i), i = 1, \dots, n$ - is Multivariate Normal, conditional on the parameters of the multivariate Normal density, i.e. its mean and covariance matrix. However, this joint probability density of output training data Y , conditional on its parameters mean and covariance matrix, is nothing but the likelihood of these parameters in the data $\{y_1, \dots, y_n\}$. Thus, in the Bayesian paradigm, with this likelihood and priors ascribed to the model parameters, the unscaled joint posterior of the model parameters can be obtained, given the empirical data, using Bayes rule.

Now, inferential techniques such as MCMC can be performed on this joint posterior density, to learn the marginal posterior density of the unknown parameters of the model

along with uncertainty, given the training data. Hence, the mean and covariance matrix of this Multivariate Normal density; which are the parameters of the model - need to be learnt in order to learn the sought function $f(\cdot)$. When zero-mean GP is invoked to model the sought function $f(\cdot)$, then covariance matrix is the only model parameter, that is unknown and needs to be learnt. When the covariance structure of a GP is learnt given the empirical data, then this learning will yield the correlation structure of any random function (such as the sought function $f(\cdot)$) sampled from that GP. Now, I will discuss the important aspect of several ways to learn this covariance matrix.

A covariance matrix is a positive-definite, symmetric matrix. Therefore, the elements of the upper triangle will be exactly the same as the lower triangle of the covariance matrix. So, to learn the covariance matrix, learning any one of the upper or lower triangle of the symmetric covariance matrix will suffice. Now, for a covariance matrix with dimension $N \times N$, the size of the upper or lower triangle is $(N^2 - N)/2$ - which implies that, to learn the elements of the covariance matrix, I need to learn only $(N^2 - N)/2$ number of unique elements. When $(N^2 - N)/2$ is a small number, such as less than 200, then MCMC can be used to learn all the $(N^2 - N)/2$ elements. However, when N is large, it is practically infeasible to learn all the elements of the upper or lower triangle of the covariance by directly using MCMC.

A covariance matrix can also be learnt using parametrisation of the covariance matrix, using a kernel function $k(\cdot, \cdot)$. A kernel function or covariance kernel is a positive-definite function that represents the correlation between the outputs Y_i , and Y_j , realised at the corresponding input points x_i , and x_j . A kernel function is typically expressed in terms of a decaying function of the distance between these input values x_i , and x_j . This decaying function ensures that more the distance between the two input points x_i , and x_j , smaller the correlation between Y_i , and Y_j .

Definition 4.1.1. *The ij -th element of the covariance matrix represents the covariance between the output points Y_i , and Y_j , $\text{cov}(Y_i, Y_j)$ which is parametrically chosen using $k(\cdot, \cdot)$ and modelled as $\text{cov}(Y_i, Y_j) = k(x_i, x_j)$.*

Remark 4.1.1. *The kernel function $k(\cdot, \cdot)$ comprises of hyperparameter(s), which are much fewer than N and needs to be learnt in order to learn the covariance matrix. Thus using the kernel parametrisation approach, the learning of the sought function $f(\cdot)$, reduces to the learning of the few hyperparameter(s) of the kernel function $k(\cdot, \cdot)$.*

In this chapter, I will first motivate the need for a non-parametric non-stationary kernel over stationary kernels in the context of real-world data, which often bears inhomogeneities in its correlation structure (in Section 4.2). Then in Section 4.3, I will present the literature

on the existing kernels along with their limitations and scope, that demands the development of a new kernel addressing the limitations and challenges. Section 4.4 will present a newly developed kernel that addresses the challenges identified in Section 4.3, by being fully non-parametric and parsimonious. Section 4.5 will depict that this developed non-parametric kernel is equivalently non-stationary by introducing a non-stationary version of this non-parametric kernel which yields input location-pair specific length scales. Aspects of this fully non-parametric and non-stationary models are Blended to create a Blended model in Section 4.6.

Then, I will present predictive results of the newly developed models using a real-world univariate data, that is inhomogeneously-correlated, along with the comparative results from other similar models including two non-stationary models from the literature. Altogether, I will present results obtained from nine models, in Section 4.7. These nine models include two non-stationary kernels from the literature, two popular stationary kernels from literature within a single GP layer based model to illustrate the limitations of such models. All the nine models are illustrated with the same univariate, training data and test data. The comparative results show that the newly developed, fully non-parametric kernel, with its requirement of learning very few hyperparameters i.e. a parsimonious model, outperforms the other models that has been considered and the prediction of the non-stationary version of the non-parametric kernel outperforms other non-stationary kernels from the literature. Finally, I will conclude this chapter in Section 4.8.

4.2 Need for non-stationary, non-parametric kernels for real-world data

Kernel parametrisation can fail with real-world data for many reasons, yielding erroneous learning of the hyperparameters of the kernel $k(\cdot, \cdot)$ and thus producing wrong prediction that uses the wrongly learnt kernel hyperparameters. This means that the $cov(Y_i, Y_j)$ is not correctly captured by the $k(x_i, x_j)$. This may happen when an inappropriate parametric form has been ascribed to the kernel which does not truly capture the correlation structure of the empirical data. After all, a chosen parametric form of the kernel also imposes a covariance structure on the sought function $f(\cdot)$.

Often real world data sets bear inhomogeneous correlation structure due to multiple reasons such as when the data is obtained from a non-linear system, or when the dataset is built up using multiple data sources. In such situations, outputs realised at different subsets of the input space are differently correlated. When, the correlation structure of the data is

uneven throughout the input space, the same values of the kernel hyperparameters cannot be used at all the input points, because, the values of the hyperparameters determine how quickly the correlation between the outputs Y_i , and Y_j will fade. Therefore, the correlation between outputs Y_i , and Y_j can be different from the correlation between outputs Y_a , and Y_b over respective values of X where $i \neq a, j \neq b$ within the dataset.

In such circumstance, where the correlation structure of the training data is uneven, the values of the hyperparameters of the kernel need to be learnt accordingly, so that the correlation fades away differently in different subspace of the output space. Such a kernel is referred as non-stationary kernel. Non-stationary kernels are not shift-invariant over the input space X . Thus, correlation function of a GP which is being parametrised using such non-stationary kernel becomes shift-variant over the input space X and the GP is then referred as a non-stationary GP.

Therefore, when a real-world data bears inhomogeneities in its correlation structure, i.e. correlation structure is non-uniform throughout the input space, then modelling of the functional relationship between the input and output variables should invoke a non-stationary GP. The covariance function of such invoked non-stationary GP is parametrised with a non-stationary kernel $K(\cdot, \cdot; \theta)$, where the kernel hyperparameter vector $\theta \in \mathbb{R}^H$ is not a constant at all points in input space X . To ensure that the hyperparameters are not constant, the kernel hyperparameter vector θ is usually modelled using not just the distance $d(x_i, x_j)$ between the two input points (x_i, x_j) , $x_i, x_j \in X$, but instead, as a function of the relevant input locations. Then for the input pair x_i, x_j , I express the k -th component θ_k of the vector θ as $\theta_k = g_k(x_i, x_j)$, for $k = 1, \dots, H$. Here the function $g_k(\cdot)$ is an unknown function. Modelling θ_k as $g_k(x_i, x_j)$ will ensure that every pair of input locations will have its own unique value of hyperparameter θ_k . For the input pair x_i, x_j , I refer to this value as $\theta_{i,j}^{(k)}$.

This model of the kernel then correctly captures the correlation between two outputs realised at any two inputs. However, such modelling of non-stationarity would require learning $O(n^2/2)$ number of hyperparameters for a training set with n design points. Since this model varies the kernel hyperparameters at one pair of input points to another pair of input points, the form of the kernel varies from one pair of input values to another pair of input values. This is then a fully non-parametric kernel that is more desirable than assigning a non-stationary kernel with parametric form. Therefore, a fully non-parametric kernel would reflect the true correlation structures of the empirical data when learning $f(\cdot)$.

At the same time, it is desirable that the kernel has very few hyperparameters to be learnt with the small-to-medium sized data. In the next section, I will discuss the literature on kernels and talk about the challenges that remain unaddressed till date.

4.3 State-of-the-art kernels

Genton (2001) presents a work on the fundamentals of a class of kernels including stationary and non-stationary kernels. Apart from the fundamental base kernels, work has been done to design new stationary kernels composed of base kernels. Duvenaud et al. (2013) presented such an automated kernel composition over a space of kernel structures that are built compositionally by adding and multiplying some base kernels, though the computational burden of such an approach limits the complexity of the selected kernels (Tobar et al., 2015). Wilson and Adams (2013) proposed a stationary spectral mixture (SM) kernel, where the power spectral density (PSD) of the GP is parametrised by a mixture of Gaussians. In spite of the expressivity of the spectral kernels, SM kernel gets limited by its optimisation procedure because the form of such kernel includes multiple modes in the marginal likelihood which poses special challenges to the optimisation (Simpson et al., 2021).

As mentioned before, these kernels are given a chosen parametric form, thus bringing the possibility of incorrect learning of the correlation structure. Because, a chosen parametric form to the kernel also imposes a covariance structure on the sought function. Therefore, in order to avoid incorrect predictions using GP, it needs to be ensured that the correlation between the output points are correctly represented through the correlation between the corresponding input points. To address this challenge, this chapter presents a fully non-parametric way of kernel parametrisation for GP regression, which is data driven and overcomes the limitation of erroneous parametric modelling. This non-parametric kernel is data driven by design because it involves learning of the kernel flexibly to reflect the correlation structure of the data by learning the correct representation of the correlation structure of the underlying stochastic process that generates the sought function as its random sample.

Now, I will present literature on the work that has been carried out in the past in the area of non-stationary kernels. Higdon et al. (1999) developed a non-stationary kernel based on the convolution of the kernels centred at the input locations. Using the work of Higdon et al. (1999) as foundation, Paciorek and Schervish (2003) have proposed a closed form non-stationary correlation function that can be built from any stationary correlation function (Paciorek, 2003). Adams and Stegle (2008) introduced the input dependent variability of the amplitude using the product of two stationary Gaussian Processes. Ba and Joseph (2012) achieve non-stationary behaviour through the summation of two stationary Gaussian Processes independent of each other with Gaussian correlation functions and they call it composite Gaussian Process (CGP) model. Davis et al. (2021) further extends the CGP model with modification by providing Bayesian composite Gaussian Process (BCGP) model. Heinonen et al. (2016) also used two Gaussian Processes to achieve non-stationarity by

modelling the length scale, signal variance and noise variance with latent or inner Gaussian Process. The latent GP uses squared exponential covariance kernel. They learn nine hyperparameters for each input dimension. Remes et al. (2017) proposed non-stationary spectral kernel where they also have two layers of Gaussian Processes and hyperparameters of the kernel, that models the covariance function of the outer GP, are modelled with latent GPs. They have used location-dependent length scale as in Gibbs kernel (Gibbs, 1998) within their newly designed non-stationary kernel.

Heinonen et al. (2016) modelled the length scale; signal variance; and noise variance, with latent GPs within a two-layered model, to achieve non-stationarity, where the latent GPs are ascribed squared exponential covariance kernels. Remes et al. (2017) proposed a non-stationary spectral kernel that again invokes two layers of GPs in which each hyperparameter of the kernel is modelled with a latent GP. In fact, they have used the Gibbs non-stationary kernel (Gibbs, 1998) where the length scale is modelled as an input-dependent function. Q number of mixture components are used. The number of hyperparameters to be learnt depends on Q and the number of input dimensions, which yields large number of hyperparameters for higher dimensions. A two-layered learning structure is also used by (Paciorek and Schervish, 2003). They constructed a non-stationary covariance function, the input-dependence of which is manifest in a pair of (generally) matrix-valued parameters defined at the pair of inputs, at which the kernel is being computed - to parametrise the correlation between the output variables that are realised at these inputs. The evolution of these kernel matrices determines how frequently the covariance structure changes in input space. For a d -dimensional input, the number of hyperparameters that need to be learnt is $O(d^2)$. The complexity of the parametrisation technique discourages this method for $d > 5$.

Thus, the existing two-layered strategies appear to challenge our overarching need to learn few parameters. It is also unknown, why further layers are not added to model the hyperparameters of the GPs in the inner layer, in these existing models. Thus, correctness of the model, as well as parsimony of the strategy need to be understood.

Remark 4.3.1. *In the literature, many non-stationary covariance kernels of the outer GP models the functional relationship between input and output of the empirical data, by varying hyperparameters. These hyperparameters vary from one input location to another. These hyperparameters are further modelled using other lower level i.e. inner GPs. However, these lower level GPs are assumed to be stationary without any justification.*

In general, no information is available pertaining to values of the hyperparameters of the kernels that parametrise the covariance function of the GPs in the inner layer, for a new application. Therefore, manually setting the values of these hyperparameters based on prior knowledge or “using grid-search over suitable values” as suggested in literature (Heinonen

et al., 2016) does not solve the problem given that a protocol for such a suggested search is not available to assess computational complexity of the learning. Usually, prior information on kernel hyperparameters does not exist, although, information on the prior probability distribution of the output at a given input can be available, but that does not inform on the mapping from the space of a kernel hyperparameter to the space of the output. So, essentially the hyperparameters of the lower level of latent GP are modelled with stationary, parametric covariance kernel. That means the hyperparameters of the non-stationary kernel are learnt parametrically. Hence, these non-stationary kernels are parametric and with a requirement of learning large number of hyperparameters per input dimension without explaining why further layers are not added to model the hyperparameters of the GPs in the inner layer.

Moreover, the assumption of the lower level GPs to be stationary, may not be correct given the real-world data at hand, in such a case, one needs further lower level GPs in order to correctly capture the correlation structure of the data and literature does not show how to address this issue. Therefore, there is a need for a new non-parametric kernel which is at the same time non-stationary. The kernel must be computationally easy, equipped with a well-defined algorithm for easy implementation, and involve the learning of only a few parameters (Noack et al., 2024a).

4.3.1 Discussion on hierarchical GP models

In the context of regression: Park and Choi (2010) present a two layered GP-based non-parametric regression model. This work is fundamentally different from the learning method used to undertake the various applications in this thesis, and said difference is manifest in multiple ways. Firstly, the method discussed in this thesis uses a fully non-parametric kernel that is two layered, where the kernel hyperparameters of the non-stationary GP (that is used in the outer layer) has been modelled with stationary (as proven, and not assumed) GPs in the inner layer. There are no approximations made to the GPs, nor is input space partitioned into clusters - unlike the work done by Park and Choi (2010). Indeed, Park and Choi (2010) partitioned the input space using a clustering algorithm, and the correlations amongst the partitions are then modelled using a GP in the outer layer, while the correlation between the data points within each partition is again modelled using GPs in the inner layer. Stationary, parametric (SQE) kernels (that the authors identify as radial basis function kernels) are used to model the covariances in both the outer and inner layers, with the hyperparameters of these kernels estimated by maximizing the marginal likelihood. Apart from the fundamental modelling differences with our work - which does not undertake any enforced clustering analysis, but learns all correlation structures robustly and non-parametrically - performance

of this method depends on the quality of clustering (and therefore partitioning) of the input space, on which, often real-world situations does not provide any empirical insight.

In the context of Bayesian inverse problems: Inverse problems have been copiously undertaken within the Bayesian paradigm (Dashti and Stuart, 2013; Stuart, 2010). In the inverse problem setup, typically the output $y \in Y$ realised at the design inputs $u \in X$ are observed, where the output is a function of the design input, expressed as: $y = \mathcal{G}(u)$. The aim here is to solve for the input u . In the Bayesian setting, the aim is to find a posterior probability density on u given a belief on u in terms of prior. Literature includes several types of priors, among which hierarchical Bayesian approach (Dunlop et al., 2017) is popular. Inverse problems are often ill-posed and exhibit properties such as solutions that are not unique, or non-existent solutions, or solutions that are sensitive to y . Regularization is a popular way to deal with such problems (Cotter et al., 2010).

In the Bayesian mindset, regularization can be attained through the priors that provide a priori regularization of the posterior density. For example, Giordano and Nickl (2020) have applied 'rescaled' Gaussian Process priors (as did Monard et al. (2021)) and hierarchical priors in the Karhunen-Lo  ve expansion of the Gaussian base prior, in their inverse problem setup. Gaussian priors are an important class of priors, given that regularization can be achieved through a covariance matrix or a covariance operator, via which, regularity properties are ascribed on the random draws. A covariance matrix is specified by a kernel function and the properties of that kernel function introduces the regularization; on the other hand, the rate of decay of the eigenvalues of the covariance operator provides insights on the regularization (Stuart, 2010).

Now, the 'rescaled' Gaussian Process prior mentioned above, uses a stationary (Whittle-)Mat  rn covariance kernel (Monard et al., 2021). Neither is the learning of hyperparameter(s) of that stationary kernel discussed/undertaken, nor is the hierarchical modelling of those hyperparameter(s) with other GPs discussed, or is even relevant to such approaches. Indeed, the possibility of a non-stationary GP prior is not been pertinent to these problems, and therefore precluded from discussions. Hence, the covariance structure of the GP prior is not derived using the given empirical data. Here lies the fundamental difference between the fully non-parametric, parsimonious kernel used in this chapter, and the work done within inverse problem approaches, such as that by Giordano and Nickl (2020). Furthermore, the learning method used in this thesis does not undertake determining the covariance structure of any GP in terms of a covariance operator.

An important aside is that although this thesis deals with forward and direct learning, the invoked non-parametric, parsimonious kernel can be used in inverse learning setup as well (Chakrabarty, 2025; Chakrabarty et al., 2023).

In the context of Gaussian Random Field: One of the fundamental tools in developing Bayesian hierarchical learning models within the non-parametric modelling paradigm and spatial statistics, to learn the sought function, is Gaussian Random Fields (GRF), where a scalar-variate GRF is also referred as a GP (Abrahamsen and regnesentral, 1997). However, high-dimensional GRFs in this thesis are referred as vector-valued GP, matrix-valued GP. Choosing priors for these learning models remains challenging, as a poorly chosen prior can decrease the predictive performance by affecting the behaviour of the posterior, specially in high dimensions, and destabilising the inference leading to incorrect learning of unknown model parameters. GRF can be considered as a prior for learning sought functions (can be multidimensional) in various applications.

For example, van der Vaart and van Zanten (2009) constructed a fixed prior based on a rescaled GRF where the choice of the scaling factor is limited. van der Vaart and van Zanten (2009) has rescaled the sample paths by a random variable, inverse of which is termed as bandwidth parameter, and this bandwidth parameter is assigned a Gamma distribution. Through the constructed rescaled prior, van der Vaart and van Zanten (2009) depicts that the model fits the data better when the parameters are adapting i.e. varying. They also proved that the resulting posterior density contracts to the true function at a certain minimax rate and up to a logarithmic factor. Although they have shown a well behaved inference with stationary Squared Exponential covariance function, they have not provided any guideline on how to select the hyperparameters i.e. whether to select the hyperparameters manually or to be learnt.

Yang and Tokdar (2015) points out that the adaptivity and near minimax optimality of regression presented in van der Vaart and van Zanten (2009) are suitable for lower dimensional applications. The presented non-parametric, non-stationary kernel in this chapter can be used to define the covariance structure of any GRF prior and provides a guideline on how to learn hyperparameters instead of using stationary kernel functions to define covariance structure of the GRF as in van der Vaart and van Zanten (2009). The later chapters in this thesis has shown how the kernel presented in this chapter can be applied in high-dimensions, hence, this kernel can also be used to specify high-dimensional GRF priors.

4.3.2 Discussion on Multiple Kernel Learning

The area of Multiple Kernel Learning (MKL) provides algorithms to combine multiple kernels to represent the covariance structure of empirical data, instead of choosing a particular kernel from the vast repository of the kernels. Genton (2001) presents literature on classes of kernels from the statistical point of view. Gönen and Alpaydın (2011) presents an extensive literature on MKL which identifies some key properties in the context of MKL, that includes: functional form; learning method; target function; and computational complexity.

The pre-specified finite set of parametric kernels are combined using a combination function while both the kernels and the combination function have their own parameters. This combination function can be linear (Bach et al., 2004; Cortes et al., 2010; Sonnenburg et al., 2006), nonlinear (Cortes et al., 2009; Gönen and Alpaydın, 2013; Zhuang et al., 2011) or non-stationary (Lewis et al., 2006; Wu et al., 2014). A linear combination typically comprises of the weighted sum of the kernels, while a nonlinear combination uses nonlinear functions such as multiplication, power, and exponentiation to combine the kernels having different weights. In non-stationary combination setting, the weights of the kernels are made input data dependent, although the kernels which are being combined still have parametric form.

The parameters of the kernels and the weights of the individual kernels are learnt using several approaches such as optimisation, Bayesian approach (Girolami and Rogers, 2005; Gönen, 2012). While similarity-based functions are a popular choice for defining target functions for optimisation, Bayesian formulation uses posterior density function to learn the parameters. Different MKL algorithms have different types of computational complexities based on several factors such as the number of kernels used, training time etc., however, the survey on MKL (Gönen and Alpaydın, 2011) did not find significant difference among various MKL algorithms in terms of accuracy, though computational complexities varies.

Although some recent work on MKL have been reported in (Ghanizadeh et al., 2024; M Ghari and Shen, 2022), despite the availability of such methods, still there does not exist any universal method that guide the choice of kernels and their combination given the data, especially given the inhomogeneous correlation structures of the real-world data. This is because through the combination (no matter how complex) of the kernels, the final kernel is still being ascribed a parametric form i.e. an algebraic expression which may represent the empirical correlation structure of the given data incorrectly, this is more prominent in high-dimensions. On the other hand, the kernel used in this chapter is fully non-parametric and the smoothness adapts according to the inhomogeneities present in the correlation structure of the data, avoiding the need for selecting and combining kernels. The non-parametric kernel is parsimonious at the same time, requiring smaller number of hyperparameter learning, while

in MKL approach, the number of kernel parameters along with the kernel weight parameter grows with the number of kernels considered.

4.3.3 Discussion on vector-valued kernels

In recent applications, supervised learning deals with multiple outputs in addition with the traditional scalar-valued outputs. In probabilistic setting, when the output is scalar valued, then the sought function is modelled as a random realisation from a scalar-valued GP. Whereas, the sought function is modelled as a random realisation from a vector-valued GP when the output is a vector. A kernel function that models the covariance structure of the vector-valued GP is also vector-valued. Alvarez et al. (2012) presents an extensive survey on choices for vector-valued kernel functions that are either stationary or non-stationary. Other works on this topic can be found in Alvarez and Lawrence (2011); Hutchinson et al. (2021); Kondor and Jebara (2003).

Probabilistic modelling using GPs for multi-output learning has been largely developed in the area of spatial statistics, where the prediction of vector-valued output has been termed as “cokriging” (Alvarez et al., 2012). “Linear model of coregionalization” (LMC) is a popular choice for cokriging (Alie et al., 2024; Goulard and Voltz, 1992; Journel and Huijbregts, 1976). LMC is constructed by linearly combining independent processes which essentially means that the sought function is modelled as a linear combination of independent random functions (Alvarez et al., 2012; Marcotte, 2012). For example, if the sought function is modelled as a linear combination of k number of independent random functions, then it means that there are k number of underlying processes with their own covariance structures, where there is no inter-correlation between these random functions $f_i(\cdot)$, $i = 1, \dots, k$.

For the simplicity of the discussion, if I assume that each of the k independent processes use Squared Exponential (SQE) kernel with length scale ℓ , then the value of ℓ will be different for each of the k processes. However, the challenge is to determine k and the corresponding correct ℓ for each of the k processes given the empirical data. This is more challenging when the correlation between a pair of inputs is different than another pair within the input space. Also, when a parametric kernel function is used to model the covariance structure of the k processes, it implies that a covariance structure has been imposed which may not reflect the true covariance structure of the given empirical data. However, the non-parametric kernel presented in this chapter avoids such overhead of ad hoc choices of k and corresponding lengthscales, by a complete non-parametric modelling, where the lengthscales vary from one input pair to another input pair yielding a flexible way to learn the sought function with varying smoothness according to the correlation presents in the empirical data. This flexible way of learning the varying smoothness of the sought function by varying the lengthscales

mimics the correlation structure of the given data and thus avoids the incorrect learning and prediction thereby that stems from ascribing pre-chosen form to the correlation structure.

This chapter includes illustrations on scalar-valued non-parametric kernel, where as later chapters of this thesis includes illustrations on vector-valued non-parametric kernels in both multivariate learning and high-dimensional learning setup.

In the next Section 4.4, I present a new non-parametric kernel addressing the challenges mentioned above. The hyperparameters of the kernel of the outer GP are learnt non-parametrically - the lower level GPs that model the hyperparameters of the kernel of the outer GP are designed in a way that they have been shown to be stationary without any assumption of stationarity imposed by hand (Roy and Chakrabarty, 2024). This ensures that there is no requirement of further hierarchies of the lower level or latent GPs. Also, the stationarity of the inner layer has been proved to imply the sufficiency of the 2-layered learning strategy. This is achieved through modelling the hyperparameters of the outer GP in terms of another variable that is not generated by the inner GPs. Hence, this newly developed non-parametric kernel is by design dual layered, unlike dual layered by assumption from the state of the art. At the same time, it is parsimonious. For example, an SQE-looking non-parametric kernel (this model is detailed in the following sections) would require $O(2d)$ number of hyperparameters to be learnt for a d -dimensional input vector pertaining to the empirical training data.

4.4 Methods, non-parametric model and inference

Let $\mathbf{D}_{train}$ be the training data that consists of the set of M number of data pairs $\{(x_i, y_i)\}_{i=1}^M$ and I would like to learn the function $f(\cdot)$ where $Y = f(X)$ where $X \in \mathcal{X} \in \mathbb{R}$ is the input variable, $Y \in \mathcal{Y} \in \mathbb{R}$ is output variable and $f : \mathcal{X} \rightarrow \mathcal{Y}$. In this chapter, I will learn a univariate, scalar-valued sought function $f(\cdot)$. However, in the later chapters, I will be extending this learning methodology to learn multivariate and high-dimensional sought function $f(\cdot)$ in a fully non-parametric manner.

As mentioned before, I seek the sought function $f(\cdot)$ by treating $f(\cdot)$ as an unknown, which in the Bayesian paradigm translates to $f(\cdot)$ being modelled as a random function. Then by definition, $f(\cdot)$ is assigned a probability distribution. A probability distribution on the space of functions is given by a stochastic process. So, $f(\cdot)$ is modelled as a random realisation from a stochastic process. This process is chosen to be a GP, called $\mathcal{GP}_{outer}$ such that $f(\cdot) \sim \mathcal{GP}_{outer}(\mu(\cdot), cov(\cdot, \cdot))$, where $\mu(\cdot)$ and $cov(\cdot, \cdot)$ are mean and covariance functions of this GP. The justification for the choice of GP has already included in Section 4.1.

The covariance function of this $\mathcal{GP}_{outer}$ is kernel parametrised where each kernel hyperparameter is again modelled as a random realisation from another distinct GP. This GP is nested within the outer GP $\mathcal{GP}_{outer}$. For H number of kernel hyperparameters of $\mathcal{GP}_{outer}$, there will be $\mathcal{GP}_1, \mathcal{GP}_2, \dots, \mathcal{GP}_H$ distinct inner GPs nested with the $\mathcal{GP}_{outer}$.

Remark 4.4.1. *This newly built parsimonious kernel is based on the theory that the average effect of drawing different sample functions from a given GP is equivalent to that of drawing a sample function from each of a set of GPs that are rendered different, as updated during the equilibrium stage of the undertaken inference (via MCMC).*

Novelty of this newly developed kernel lies in as follows:

1. The inner GPs $\mathcal{GP}_1, \dots, \mathcal{GP}_H$ are not assumed to be stationary, instead proven to be stationary by its design. Thus the sufficiency of two layers of GPs in learning the sought function can be proven.
2. The covariance function of the inner GPs $\mathcal{GP}_i, i = 1, \dots, H$ are learnt instead of specifying manually or using *ad hoc* strategies.
3. For such learning of covariance function of $\mathcal{GP}_i$, datasets pertaining to each $\mathcal{GP}_i$ has been generated such that this data is solely relevant to the i -th hyperparameter of the kernel of $\mathcal{GP}_{outer}$. This data is different from the empirical data.
4. The new kernel is fully non-parametric while achieving non-stationarity and most importantly parsimony. A parsimonious non-stationary kernel is an important and required contribution in the current context of real-world data while producing reliable prediction.
5. This kernel is computationally fast, equipped with methodical flow diagram and detailed algorithm for ease of implementation suitable for real-word data which often bears inhomogeneities in its correlation structure.
6. This parsimonious kernel can be used efficiently for different data situations such as when the variable X and/or Y is a scalar variate, multivariate or high-dimensional.

4.4.1 Model

A Multivariate Normal density $\mathcal{MN}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ has a mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. When the joint probability density of M variables is Multivariate Normal, the mean vector has dimension M and the dimension of the $\boldsymbol{\Sigma}$ is $M \times M$. As the sought function $f(\cdot)$ is modelled

as a random realisation from the outer GP $f(\cdot) \sim \mathcal{GP}_{outer}(\mu(\cdot), cov(\cdot, \cdot))$, by definition of GP (Rasmussen and Williams, 2006), the joint probability of the outputs $f(x_1), \dots, f(x_M)$ is a Multivariate Normal density, where $f(x_i)$ realised at design inputs x_i , $\forall i = 1, \dots, M$. So,

$$[f(x_1), f(x_2), \dots, f(x_M)] = \mathcal{MN}(\boldsymbol{\mu}, \boldsymbol{\Sigma}).$$

The ij -th element of the covariance matrix $\boldsymbol{\Sigma}$ represents $cov(Y_i, Y_j)$, i.e. the covariance between the two random variables Y_i and Y_j , where Y_i is the output $f(x_i)$ realised at $X = x_i$, $\forall i = 1, \dots, M$.

Now, the $cov(Y_i, Y_j)$ of $\mathcal{GP}_{outer}$ is modelled with the covariance kernel $K(x_i, x_j; \boldsymbol{\theta})$ where $\boldsymbol{\theta} \in \mathbb{R}^H$ is a vector valued variable which contains all the values of the hyperparameters of the kernel $K(\cdot, \cdot; \boldsymbol{\theta})$. This implies, similarity between a pair of input points are mapped to the similarity between the corresponding pair of output points.

When the data on the outputs is standardised by the mean and standard deviation of the observations, $\boldsymbol{\Sigma}$ becomes the correlation matrix.

Definition 4.4.1. *Non-stationary kernel parametrisation of the correlation function of the $\mathcal{GP}_{outer}$ using $K(\cdot, \cdot; \boldsymbol{\theta}_{ij})$ is achieved through defining the correlation matrix $\boldsymbol{\Sigma}$ as: $[corr(Y_i, Y_j)] \equiv [corr(f(x_i), f(x_j))] = [K(x_i, x_j; \boldsymbol{\theta}_{ij}^{(1)}, \boldsymbol{\theta}_{ij}^{(2)}, \dots, \boldsymbol{\theta}_{ij}^{(H)})]$. Here $\boldsymbol{\theta}_{ij}^{(k)} = g_k(x_i, x_j)$, $\boldsymbol{\theta}_{ij}^{(k)} \in \mathbb{R} \forall k = 1, \dots, H$, and $i, j = 1, \dots, M$.*

Defining the input dependent kernel hyperparameters $\boldsymbol{\theta}_{ij}^{(k)}$, $\forall k = 1, \dots, H$ according to Definition 4.4.1 would require learning of $O(M^2/2)$ distinct hyperparameters for a training data with M data points. This is practically infeasible to learn since this scheme requires us to learn too many parameters, even for M as small as 10. This is undesirable and instead, we want to design a parsimonious model.

I can avoid the overhead of learning huge number of hyperparameters to achieve non-stationarity in a smarter way in which the kernel $K(x_i, x_j; \boldsymbol{\theta})$ is itself a random function which varies from one sample function to another sample function of the underlying stochastic process, where the sample functions are drawn independently. In other words, the kernel must be flexible enough so that it correctly represents the correlation structure of $\mathcal{GP}_{outer}$, given the data, as the sought function $f(\cdot)$ is modelled as a random sample of $\mathcal{GP}_{outer}$.

To ensure that the kernel $K(x_i, x_j; \boldsymbol{\theta})$ truly represents the correlation structure of $\mathcal{GP}_{outer}$ given the data, and thereby the correlation structure of the sample function $\tilde{f}(x) \sim \mathcal{GP}_{outer}$, $x \in \mathcal{X}$, I follow the suggestion Chakrabarty (2025) and Roy and Chakrabarty (2024), I treat the hyperparameters of this kernel as a random function of the sample function i.e.

$$\boldsymbol{\theta}_i = z_i(\tilde{f}(\cdot)), \forall i = 1, \dots, H, \boldsymbol{\theta}_i \in \mathcal{H} \subseteq \mathbb{R}, z_i: \mathcal{Y} \longrightarrow \mathcal{H}. \quad (4.4.1)$$

Hence, θ_i changes with the input points X as the random function $z_i(\cdot)$ changes with X . So, the kernel $K(\cdot, \cdot; \boldsymbol{\theta})$ is by definition non-stationary.

Equation 4.4.1 presents a generalised model and hence is a good model. However, there exists difficulties with the learning of $z_i(\cdot)$ and therefore with the prediction of the θ_i at a given sample function. To learn this $z_i(\cdot)$, the data needed is,

$$\mathbf{D}_{latent} = \{((\tilde{f}(x))_c, (\theta_i)_c)\}_{c=1}^{N_{LB}}$$

where N_{LB} is the number of data points in the data set $\mathbf{D}_{latent}$. The difficulties arise with this data $\mathbf{D}_{latent}$ are as follows.

As I can sample one $\tilde{f}(\cdot)$ for a given θ_i imposed on the $\mathcal{GP}_{outer}$ model, it appears that I can generate $\mathbf{D}_{latent}$. However, I can generate infinite realisations of $\tilde{f}(\cdot)$ at a given θ_i . Also multiple θ_i values can be implied by the same $\tilde{f}(\cdot)$. So, $\mathbf{D}_{latent}$ is such a training data that there is maximal departure from the one-to-one-ness in $\mathbf{D}_{latent}$.

Other problem with learning $z_i(\cdot)$ using Equation 4.4.1 is that, I will need to deal with function valued kernels (which can be substituted by a high-dimensional vector). So the task reduces to learning using a (very high-dimensional) vector-valued kernel. This implies learning a very large number of kernel hyperparameters, which violets the aim of developing parsimonious learning model.

To avoid these difficulties and at the same time to achieve non-stationarity, I modify the model presented in Equation 4.4.1 following the methodology advanced by Chakrabarty (2025); Wang and Chakrabarty (2018) that invoke the following result.

Average effect of drawing different sample functions from a given GP is equivalent to that of drawing a sample function from each of a set of GPs that are different, as updated during each time step of the equilibrium stage of the undertaken inference (via MCMC), as long as the system is ergodic.

Here the time step variable is $T \in \mathcal{T}$. More details about ergodicity can be found in Birkhoff (1931, 1942); Neumann (1932); van Lith (2001); Von Plato (1988). Equation 4.4.1 is modified with the time step variable T , as T is unique for each of the independently drawn sample function $\tilde{f}(\cdot)$ from $\mathcal{GP}_{outer}$. Therefore, I can treat $\tilde{f}(x)$ as an unknown function $\tau(\cdot)$ of t such that,

$$\tilde{f}(x) = \tau(t), \tau : \mathcal{T} \longrightarrow \mathcal{Y}.$$

Using this,

$$\theta_i = z_i(\tilde{f}(x)) = z_i(\tau(t)), \forall i = 1, \dots, H$$

Denoting the composition of the function $z_i(\cdot)$ and $\tau(\cdot)$ as $g_i(\cdot)$, I can write

$$\theta_i = g_i(t), \forall i = 1, \dots, H, t > N_0, g_i : \mathcal{T} \longrightarrow \mathcal{H}$$

after running a large number (N_0) of iterations of the inference (MCMC in this thesis and N_0 denotes iteration number at which burnin ends).

It is to be noted that each θ_i is learnt in the inner layer. θ_i is modelled as the output of an unknown function $g_i(\cdot)$ of T . Here $g_i(\cdot)$ is a sample function of the inner GP $\mathcal{GP}_i$ such that

$$g_i(\cdot) \sim \mathcal{GP}_i(\mu_i(\cdot), \text{cov}_i(\cdot, \cdot)), \forall i = 1, \dots, H,$$

which makes the learning of θ_i computationally feasible.

Now I will present the proof that $\mathcal{GP}_i$ can be a stationary GP. Thereafter, I will discuss the construction of the training data $\mathbf{D}_i^{(t)}$ required to learn $g_i(\cdot)$ in the inner layer.

Proof that inner layer can be stationary

1. The mapping from $\mathbb{N} \longrightarrow \mathbb{R}$ is Lipschitz continuous (more can be seen in Knapp (2007) and Roy and Chakrabarty (2024)). Therefore $\mathcal{T} \longrightarrow \mathcal{H}$ is Lipschitz, i.e. $g_i(\cdot)$ is Lipschitz $\forall T = t$. This is true $\forall i = 1, \dots, H$.
2. Let H_0 denote the hypothesis that $\nexists$ any finite interval in $\mathcal{H}$, s.t. $g_i(t)$ is bounded within this interval, $\forall t \in \mathcal{T}$.
3. However, by statement-1, this hypothesis is false.
4. The alternative to this hypothesis is then true, i.e. it is true that $\exists$ a finite interval in $\mathcal{H}$, s.t. $g_i(t)$ is bounded within this interval, $\forall t \in \mathcal{T}$.
5. By Theorem 1 of Cambanis (1973), sample functions of a stationary GP are either all continuous or $\nexists$ a finite interval s.t. the function is bounded in this interval, at all values of the domain variable.
6. But by statement-4, all sample functions of $\mathcal{GP}_i$ that model $g_i(\cdot)$ are Lipschitz continuous.
7. Therefore, $\mathcal{GP}_i$ can be a stationary process.

It is to be noted that if $g_i(\cdot)$ is not a mapping: $\mathcal{T} \subseteq \mathbb{N} \longrightarrow \mathcal{H} \subseteq \mathbb{R}$, but is a different kind of mapping, this proof of continuity of all sample functions of $\mathcal{GP}_i$ will not hold. Then we will not be able to prove that $\mathcal{GP}_i$ can be stationary. So, when existing non-stationary kernels

(Heinonen et al., 2016; Paciorek and Schervish, 2003; Remes et al., 2017) use the empirical data $\{(x_i, y_i)\}_{i=1}^M$ to learn hyperparameters of $\mathcal{GP}_i$, for any i , the above proof that $\mathcal{GP}_i$ is stationary, does not hold. If these researchers wish to prove that $\mathcal{GP}_i$ is stationary, they will have to undertake an independent proof. However, they do not undertake any corroboration or confirmation of this results. They simply assume $\mathcal{GP}_i$ to be stationary for all $i = 1, \dots, H$. Hence, their assumption of stationarity of the inner GPs is not justified, while this thesis uses the existing proof that $\mathcal{GP}_i$ can be stationary.

$\mathcal{GP}_i$ can be treated as non-stationary, but as proven above, it is sufficient to prove that it is stationary. Thus, I will hold $\mathcal{GP}_i$ to be stationary $\forall i = 1, \dots, H$. Therefore all hyperparameters of $\mathcal{GP}_i$ will be treated as constants, that I will learn given the training data relevant to the learning of $g_i : \mathcal{T} \subset \mathbb{N} \longrightarrow \mathbb{R}$.

Therefore in the presented model, two layers of GPs are sufficient to learn the sought function $f(\cdot)$ using the empirical training data. The GP $\mathcal{GP}_{outer}$ in the outer layer of this two-layered learning strategy will be non-stationary and all the H kernel hyperparameters of this $\mathcal{GP}_{outer}$ will again be modelled with stationary GPs $\mathcal{GP}_1, \dots, \mathcal{GP}_H$ in the inner layer. But any hyperparameter of a kernel that parametrises the correlation function of an inner-layer GP is a constant since this GP can be stationary.

Now, I will discuss how to learn the constant hyperparameter(s) of the covariance function of $\mathcal{GP}_i$, using a training data that is being generated dynamically within the inferential setup.

Generating training data to learn $g_i(\cdot)$

Learning of $g_i(\cdot)$ cannot be achieved with the empirical training data $\mathbf{D}_{train}$. As my model is, $\theta_i = g_i(t)$, $\forall i = 1, \dots, H$, the training data $\mathbf{D}_i$ used to learn $g_i(\cdot)$ is built of values of the iteration index T , and the value of θ_i learnt using the sample function drawn from $\mathcal{GP}_{outer}$ at this time step. If the hyperparameters of the covariance function of the inner layer $\mathcal{GP}_i$, $\forall i = 1, \dots, H$ are learnt using the empirical training data $\mathbf{D}_{train}$ then the likelihood is modelled using the values of the outputs, where correlation amongst pairs of outputs is modelled using values of X . This does not correctly capture the covariance structure of $\mathcal{GP}_i$ which is a collection of variables representing kernel hyperparameters generated at different values of T . We then need to generate training dataset $\mathbf{D}_i$ that is different from the data that contains (x_i, y_i) pairs.

$\mathbf{D}_i$ is generated in each iteration of the inference, and consists of the iteration index of the past few (namely, N_{LB} number of) iterations as input and the updated value of the hyperparameter θ_i as the output. Let N_{LB} be the size of the training dataset $\mathbf{D}_i$ that is being used to learn $g_i(\cdot)$ with $\mathcal{GP}_i$; $\theta_i^{(k)}$ denotes the current value of the hyperparameter θ_i updated in the k -th iteration $\forall i = 1, \dots, H$.

Definition 4.4.2. At t -th iteration, $\mathbf{D}_i := \{(t^{(k)}, \theta_i^{(k)})\}_{k=t-N_{LB}}^{t-1}, \forall i = 1, \dots, H; t^{(k)} \in \mathbb{N}, \theta_i^{(k)} \in \mathbb{R}$ and $t^{(k)} < t$. We look back N_{LB} number of iterations before the t -th iteration, to obtain the iteration index and the corresponding value of the hyperparameter θ_i as updated in that iteration.

In this thesis, I have used MCMC i.e. Markov Chain Monte Carlo (Neal (1997)) inferential framework and SQE kernel to model the covariance function of the stationary GPs (proven to be stationary by design) in the inner layer.

Sought function $f(\cdot)$ is learnt using standardised data where each output $y_i \in \mathcal{Y}$ of the training data $\mathbf{D}_{train}$ is standardised as $y_i^{(std)} = (y_i - \bar{y})/\hat{\sigma}$, $\forall i = 1, \dots, M$ using the sample mean $\bar{y} = \sum_{i=1}^M y_i/M$ and unbiased estimate of sample variance $\hat{\sigma}^2 = \sum_{i=1}^M (y_i - \bar{y})^2/(M-1)$. Then predictions made with the learnt $f(\cdot)$ are again unstandardised using the same value of the sample mean $\bar{y}$ and the unbiased estimate of sample variance $\hat{\sigma}^2$ computed using $\mathbf{D}_{train}$. This implies that, $\boldsymbol{\mu} = \mathbf{0}$ and $\boldsymbol{\Sigma}$ is a correlation matrix denoted by $\boldsymbol{\Sigma} = [corr(Y_i, Y_j)]$.

Outline of the non-parametric model

The finalised model is as follows. From the training data $\mathbf{D}_{train}$, the sought functional relationship $f(\cdot)$ is learnt though a non-stationary GP $\mathcal{GP}_{outer}$ in the outer layer of the learning scheme s.t.

$$f(\cdot) \sim \mathcal{GP}_{outer}(\boldsymbol{\mu}(\cdot), cov(\cdot, \cdot)) \quad (4.4.2)$$

Now, the i -th kernel hyperparameter that parametrises the covariance function of the $\mathcal{GP}_{outer}$ is again modelled as a function $g_i(t)$ of the time-step variable (i.e. the iteration index), where $g_i(\cdot)$ is modelled as a random realisation of inner GP $\mathcal{GP}_i$, $\forall i = 1, \dots, H$ s.t.

$$g_i(\cdot) \sim \mathcal{GP}_i(\boldsymbol{\mu}_i(\cdot), cov_i(\cdot, \cdot)), \forall i = 1, \dots, H \quad (4.4.3)$$

$g_i(\cdot)$ is learnt using GP $\mathcal{GP}_i$ in the inner layer with the data $\mathbf{D}_i^{(t)}$ that is a dynamically varying lookback data generated at each iteration for each kernel hyperparameter of the outer layer. Hence, the GP of outer layer $\mathcal{GP}_{outer}$ holds within itself, $\mathcal{GP}_1, \dots, \mathcal{GP}_H$. Also, I have discussed above that $\mathcal{GP}_i$ can be stationary and the hyperparameters of the kernel that parametrise the covariance function of $\mathcal{GP}_i$ are learnt using data $\mathbf{D}_i^{(t)}$. Thus two-layers are sufficient in this fully non-parametric learning strategy yielding a parsimonious, non-parametric kernel.

Non-parametric kernel adapts to local smoothness of the underlying function

Below, I discuss the correlation function of the GP that is used to model a scalar-valued univariate function, where this correlation function is parametrised by a kernel $K(\cdot, \cdot; \ell)$ with a single hyperparameter ℓ . This explanation can then be generalised to higher-dimensions.

A kernel will adapt to the local smoothness of the sought function implies that the hyperparameter of the kernel will represent the true correlation of the function. This means that if the correlation between two outputs Y_i and Y_j , realised at inputs x_i and x_j respectively, varies from one input pair to another, the kernel hyperparameter ℓ will vary accordingly, implying that ℓ is not constant and the outer GP is non-stationary.

In other words, the correlation $\text{corr}(Y_i, Y_j)$ between Y_i and Y_j which is kernel parametrised as $K(x_i, x_j; \ell)$ with kernel hyperparameter ℓ , is different than the correlation $\text{corr}(Y_a, Y_b)$ between outputs Y_a, Y_b (that are realised at inputs x_a and x_b respectively), with kernel parametrisation $K(x_a, x_b; \ell')$ where $\ell \neq \ell'$. We recall that the kernel hyperparameter ℓ is a function $g(\cdot)$ of the temporal index t of the undertaken inference which is generated by the inner GP.

We have assumed that the system is ergodic, which implies:

1. drawing a distinct sample function from a GP at each of a large number of input-pairs at a single time point t (or iteration of the undertaken inference), is equivalent on the average to
2. sampling once during an iteration, from each of multiple GPs that vary slightly over the large number of iterations.

So, in the second situation, a sample function $\hat{f}(\cdot)$ drawn from the outer GP, at the two pair of inputs points (x_i, x_j) and (x_a, x_b) (following our previous discussion), in the t_k -th iteration with kernel hyperparameter ℓ and another sample function drawn at the t_q -th iteration with kernel hyperparameter ℓ' , where $\ell \neq \ell'$, will be on average (over the course of large number of iterations) equivalent to the sample function drawn at (x_i, x_j) with kernel hyperparameter ℓ and at (x_a, x_b) with kernel hyperparameter ℓ' from the outer GP at t_q -th iteration as in the first situation.

Hence, treating ℓ as output in the t -th iteration yields on an average the same information onto ℓ , as in the case of ℓ being different at different input pairs. Therefore, dual-layered GP model allows the kernel hyperparameter ℓ to acknowledge the non-stationarity of the outer GP by varying or adapting to the local smoothness of the sought function.

In this dual-layered GP model, if the parameters of the correlation function of outer GP and inner GP are learnt "correctly", given the relevant training sets (both the empirical

training set that is used for learning in outer layer and the lookback data which is used for learning in the inner layer) - prediction will be correct. This is corroborated by the results shown in Figures 4.27, 5.8, 5.9, 5.10, 6.14, 6.16 in the thesis. However, when the number of such parameters is high in high-dimensional application then the inference by MCMC becomes difficult and there is a possibility that not all the parameters are learnt correctly introducing some error in predictions.

4.4.2 Example of non-parametric kernels

When I build the non-parametric kernel from a basal shape called “Basal”, then I will call the non-parametric kernel as “Basal-looking”, because the hyperparameters $\theta_1, \dots, \theta_H$ of the “Basal” kernel are constant, however, in the non-parametric version, these hyperparameters $\theta_1, \dots, \theta_H$ are not constant, but treated as a random function and modelled as a random realisation of the inner GPs $\mathcal{GP}_1, \dots, \mathcal{GP}_H$ respectively. Now, I will present two examples of non-parametric kernels “SQE-looking” and “Matérn-looking” that are built using the basal shape SQE and Matérn respectively. It is to be noted that these basal shapes are stationary kernels, and thus, from any stationary basal shape, the corresponding non-parametric kernel can be built using the protocol presented in this chapter.

“SQE-looking” non-parametric kernel: When the basal shape is “SQE” i.e. “Basal” $\equiv$ “SQE” (depicted in Equation 4.4.6) with length scale ℓ i.e. $H = 1$, the non-parametric “SQE-looking” kernel will model this ℓ with the inner GP $\mathcal{GP}_1$ as $\ell = g_1(t)$, where t is the time step or the iteration index of the inference. It is to be noted that as the amplitude hyperparameter is not relevant when the output of the training data is standardised, the only hyperparameter that needs to be learnt to parametrise the covariance function of the outer GP $\mathcal{GP}_{outer}$ is ℓ . Definition 4.4.3 gives the non-parametric “SQE-looking” kernel.

Definition 4.4.3.

$$\text{SQE-looking kernel: } K(x_j, x_k; g_1(\cdot)) = \exp\left(-\frac{(x_j - x_k)^2}{2(g_1(t))^2}\right);$$

where $\ell = g_1(t)$ and $g_1(\cdot) \sim \mathcal{GP}_1(\mu_1(\cdot), \text{cov}_1(\cdot, \cdot))$.

δ is the single hyperparameter of the SQE kernel that parametrises the covariance function $\text{cov}_1(\cdot, \cdot)$ of the inner GP $\mathcal{GP}_1$. Therefore, in the inner layer,

$$\text{cov}(\ell_i, \ell_j) = [K(t_i, t_j)] = \exp\left(-\frac{(t_i - t_j)^2}{2\delta^2}\right), \quad \delta > 0, i, j = t - 1, \dots, t - N_{LB}$$

Hence, this SQE-looking non-parametric kernel requires learning of two hyperparameters when the sought function $f(\cdot)$ is univariate and scalar-valued. When the sought function $f(\cdot)$ is multivariate with d number of input dimensions, then the SQE-looking non-parametric kernel would require learning $O(2d)$ hyperparameters.

“Matérn-looking” non-parametric kernel: Similarly when “Basal” $\equiv$ “Matérn” (the basal Matérn is presented in Equation 4.4.7), the non-parametric kernel built using this basal Matérn shape will be referred as “Matérn-looking” non-parametric kernel. As the basal Matérn consists of two hyperparameters ℓ and ν , hence, $H = 2$, and these hyperparameters will again be modelled as random realisation from inner GPs $\mathcal{GP}_1$ and $\mathcal{GP}_2$ as $g_1(t)$ and $g_2(t)$ respectively, where t is the time step or the iteration index of the inference. Definition 4.4.4 gives the non-parametric “Matérn-looking” kernel.

Definition 4.4.4.

Matérn-looking kernel :

$$K(x_j, x_k; g_1(\cdot), g_2(\cdot)) = \frac{2^{1-g_2(\cdot)}}{\Gamma(g_2(\cdot))} \left(\frac{\sqrt{2g_2(\cdot)}}{g_1(\cdot)} d(x_j, x_k) \right)^{g_2(\cdot)} K_\nu \left(\frac{\sqrt{2g_2(\cdot)}}{g_1(\cdot)} d(x_j, x_k) \right),$$

where $\theta_i = g_i(t)$ for $i = 1, 2$, and $g_i(\cdot) \sim \mathcal{GP}_i(\mu_i(\cdot), cov_i(\cdot, \cdot))$, with $\theta_1 \equiv \ell$, $\theta_2 \equiv \nu$.

Since $cov_i(\cdot, \cdot)$ of the inner GPs $i = 1, 2$ is parametrised using a stationary SQE kernel, a single hyperparameter δ_i with $i = 1, 2$ is learnt for each of the two GPs in the inner layer. Therefore, in the inner layers,

$$cov(\ell_i, \ell_j) = [K(t_i, t_j)] = \exp \left(-\frac{(t_i - t_j)^2}{2\delta_1^2} \right), \quad \delta > 0, i, j = t - 1, \dots, t - N_{LB}$$

$$cov(\nu_i, \nu_j) = [K(t_i, t_j)] = \exp \left(-\frac{(t_i - t_j)^2}{2\delta_2^2} \right), \quad \delta > 0, i, j = t - 1, \dots, t - N_{LB}$$

Thus, it requires learning of four hyperparameters in when the sought function $f(\cdot)$ is univariate and scalar-valued. learning $O(4d)$ hyperparameters are required using kernel when the sought function $f(\cdot)$ is multivariate with d number of input dimensions.

4.4.3 Inference

I have used Random-Walk-within-Gibbs MCMC inference to implement the non-parametric learning where the output of both the empirical training data $\mathbf{D}_{train}$ and lookback training

data $\mathbf{D}_i^{(t)}$ is standardised using sample mean and unbiased estimate of sample variance. The predictions made using both of these standardised data are then unstandardised using the same sample mean and unbiased estimate of sample variance that are used for standardisation of the respective training set.

The inference comprises two blocks. Detailed steps performed within these blocks for the t -th iteration of the MCMC chain are as follows.

First block of inference:

1. For each θ_i where $i = 1, \dots, H$, θ_i^* value is proposed from the proposal density $p_i(\cdot, \sigma_i)$ as $\theta_i^* \sim p_i(\theta_i^{(t-1)}, \sigma_i)$. $\theta_i^{(t-1)}$ is the current value of θ_i at the end of the $t - 1$ -th iteration, proposal variance σ_i is experimentally chosen.
2. Likelihood is computed at the empirical training data $\mathbf{D}_{train}$, using both proposed values and the current values of $\theta_1, \dots, \theta_H$. likelihood of $\theta_1, \dots, \theta_H$ in the data $\mathbf{D}_{train}$ is Multivariate Normal:

$$\mathcal{L}(\theta_1, \dots, \theta_H | \mathbf{D}_{train}) = \frac{1}{\sqrt{(2\pi)^M |\boldsymbol{\Sigma}|}} \exp \left(-\frac{1}{2} (\mathbf{y} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{y} - \boldsymbol{\mu}) \right), \quad (4.4.4)$$

where y_j is the standardised output realised at input x_j defines $\mathbf{y} = (y_1, \dots, y_M)^T$, for $j = 1, \dots, M$, $\boldsymbol{\mu} = \mathbf{0}$; and $\boldsymbol{\Sigma} = [\text{corr}(Y_k, Y_j)] = [K(x_k, x_j; \theta_1, \dots, \theta_H)]$, with $K(\cdot, \cdot; \theta_1, \dots, \theta_H)$ either SQE-looking, or Matérn-looking non-parametric kernel defined in previous paragraph.

3. Having computed the likelihood, the posterior $\pi(\theta_1, \dots, \theta_H | \mathbf{D}_{train})$ is computed using priors on $\theta_1, \dots, \theta_H$.
4. Acceptance ratio A_t is computed at t -th iteration:

$$A_t = \frac{\pi(\theta_1^{(*)}, \dots, \theta_H^{(*)} | \mathbf{D}_{train}) \prod_{i=1}^H p_i(\theta_i^{(t-1)}, \sigma_i)}{\pi(\theta_1^{(t-1)}, \dots, \theta_H^{(t-1)} | \mathbf{D}_{train}) \prod_{i=1}^H p_i(\theta_i^{(*)}, \sigma_i)},$$

5. $\forall i = 1, \dots, H$, if $A_t \geq u$ where $U \sim \text{Uniform}[0, 1]$, then update θ_i as $\theta_i^{(t)} = \theta_i^{(*)}$, else update θ_i as $\theta_i^{(t)} = \theta_i^{(t-1)}$.
6. When $0 < t \leq N_{LB}$, only this one block is being executed to update θ_i as $\theta_i^{(t)}$ which acts as current value of θ_i in the $t + 1$ -th iteration. However, when $t > n_0$, with

$N_{max} > n_0 > N_{LB}$, second block is executed to further modify the value $\theta_i^{(t)}$ of θ_i which has been updated in the first block.

Second block of inference: Second block implements $\theta_i = g_i(\cdot)$ where $g_i(\cdot) \sim \mathcal{GP}_i$, learns the hyperparameter δ_i of $\mathcal{GP}_i$ using the lookback data $\mathbf{D}_i^{(t)}$ and once again update the current value $\theta_i^{(t)}$ of θ_i which was previously updated through accept-reject in the first block. So, in this second block (when first executed at t -th iteration), the lookback data $\mathbf{D}_i^{(t)}$ is constructed with a fixed size of N_{LB} using the current values $\theta_i^{(t)}$ of θ_i , $\forall i = 1, \dots, H$. As the lookback data is of fixed size, so it is constructed only once and afterwards it is only updated. When new data point is inserted at the rear end of $\mathbf{D}_i^{(t)}$, automatically one element is removed from the front end of $\mathbf{D}_i^{(t)}$, i.e. $\mathbf{D}_i^{(t)}$ is always full and follows “First-In-First-Out” principle.

1. The output of the lookback data $\mathbf{D}_i^{(t)} = \{(t^{(k)}, \theta_i^{(k)})\}_{k=t-N_{LB}}^{t-1}$ is standardised.
2. The likelihood of δ_i in the lookback data $\mathbf{D}_i^{(t)}$ is Multivariate Normal:

$$\mathcal{L}_i(\delta_i | \mathbf{D}_i^{(t)}) = \frac{1}{\sqrt{(2\pi)^{N_{LB}} |\boldsymbol{\Sigma}_i|}} \exp\left(-\frac{1}{2}(\mathbf{y}_i - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_i)\right), \quad (4.4.5)$$

where $\mathbf{y}_i = (\theta_i^{(t-N_{LB})}, \dots, \theta_i^{(t-1)})^T$ and $\boldsymbol{\Sigma}_i = [\text{cov}(\theta_i^{(k)}, \theta_i^{(j)})] = [K(t^{(k)}, t^{(j)})]$, with

$$K(t^{(k)}, t^{(j)}) = \exp\left(-\frac{(t^{(k)} - t^{(j)})^2}{2\delta_i^2}\right), \quad \delta_i > 0, \forall i = 1, \dots, H.$$

3. $\delta_i^{(*)} \sim \mathcal{N}(\delta_i^{(t-1)}, \sigma_\delta^{(i)})$, $\forall i = 1, \dots, H$ value is proposed using the current value $\delta_i^{(t-1)}$ of δ_i , compute the likelihood and thereby compute the posterior $\pi(\delta_i^{(*)} | \mathbf{D}_i^{(t)})$ of δ_i given the relevant lookback data, $\forall k = 1, \dots, H$ by invoking prior on δ_i .
4. Acceptance ratio $A_t^{(\delta_i)}$ is computed as:

$$A_t^{(\delta_i)} = \frac{\pi(\delta_i^{(*)} | \mathbf{D}_i^{(t)})}{\pi(\delta_i^{(t-1)} | \mathbf{D}_i^{(t)})}$$

5. Based on the acceptance ratio $A_t^{(\delta_i)}$ update δ_i . $\forall i = 1, \dots, H$, if $A_t^{(\delta_i)} \geq u$, then update δ_i as $\delta_i^{(t)} = \delta_i^{(*)}$, else update δ_i as $\delta_i^{(t)} = \delta_i^{(t-1)}$.
6. Once, δ_i is updated, now the value of θ_i is predicted as $\bar{\theta}_i$ at t using $\mathcal{GP}_i$ with the current value of kernel hyperparameter $\delta_i^{(t)}$ given the lookback data $\mathbf{D}_i^{(t)}$. This predicted $\bar{\theta}_i$

becomes the current value $\theta_i^{(t)}$ of θ_i in the t -the iteration s.t.

$$\theta_i^{(t)} = \bar{\theta}_i, \forall i = 1, \dots, H$$

7. This predicted mean value $\bar{\theta}_i$ is:

$$(e^{-((t-1)-t)^2/2(\delta_i^{(t)})^2} \dots e^{-(t-N_{LB}-t)^2/2(\delta_i^{(t)})^2}) \mathbf{\Sigma}_i^{(t)} (\theta_i^{(t-1)}, \dots, \theta_i^{(t-N_{LB})})^T,$$

where $\mathbf{\Sigma}_i^{(t)} = [\exp(-(t^{(k)} - t^{(j)})/2(\delta_i^{(t)})^2)]$, for $k, j = t - N_{LB}, \dots, t - 1$.

8. Computation of the expected mean of the output realised at the test inputs are done using this current $\theta_i^{(t)}$, $\forall i = 1, \dots, H$ given the empirical training data $\mathbf{D}_{train}$ with $\mathcal{GP}_{outer}$.

This learning scheme is depicted with a flow diagram in the Figure 4.1 and as an algorithm in the Algorithm 9.

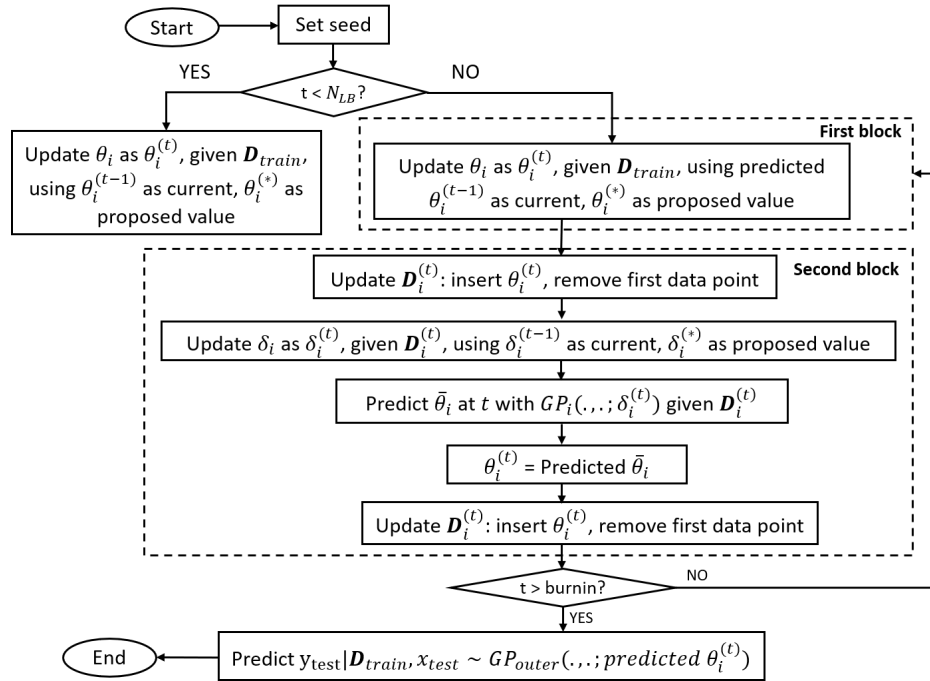


Fig. 4.1 Flow of the dual-layered GP-based learning using fully non-parametric kernel.

Algorithm 9: Implementation of the non-parametric kernel

```

1   $\forall i = 1, \dots, H$ 
2  At  $t=0$ , set seed for  $\theta_i$ 
3  for  $t \leftarrow 1$  to  $N_{iter}$  increment by 1 do
4      if  $t < N_{LB}$  then
5           $\theta_i^{(t)} \leftarrow$  Update  $\theta_i$  given  $\mathbf{D}_{train}$ , using  $\theta_k^{(*,t)}$  as proposed,  $\theta_i^{(t-1)}$  as current
            value
6      else
7          if  $t = N_{LB}$  then
8              Repeat line 5 in Block-1
              /* Block-2 */
9              Construct  $\mathbf{D}_i^{(t)} = \{\{t^{(i)}\}_{i=t-N_{LB}}^{t-1}, \{\theta_i^{(t-N_{LB})}, \dots, \theta_i^{(t-1)}\}\}$ 
10              $\delta_i^{(t)} \leftarrow$  Update  $\delta_i$  given  $\mathbf{D}_i^{(t)}$ , using  $\delta_i^{(*,t)}$  as proposed,  $\delta_i^{(t-1)}$  as current
              value
11             Predict  $\bar{\theta}_i$  at  $t$  with  $\mathcal{GP}_i(\mu_i(\cdot), cov_i(\cdot, \cdot; \delta_i^{(t)}))$  given  $\mathbf{D}_i^{(t)}$ 
12              $\theta_i^{(t)} \leftarrow$  Predicted  $\bar{\theta}_i$ 
13             Modify  $\mathbf{D}_i^{(t)}$  as: insert  $\theta_i^{(t)}$  at end, remove first item, s.t.  $\mathbf{D}_i^{(t)} =$ 
               $\{\{i\}_{i=t-N_{LB}}^{t-1}, \{\theta_i^{(t-N_{LB}+1)}, \dots, \theta_i^{(t-1)}, \theta_i^{(t)}\}\}$ 
14         else
15             Repeat line 5 in Block-1
              /* Block-2 */
16             Modify  $\mathbf{D}_i^{(t)}$  as: insert  $\theta_i^{(t)}$  at end, remove first data point
17             Repeat line 10, 11, 12 and 16 in order
18         end
19     end
20     Post burnin, predict at input test points using predicted  $\theta_i^{(t)}$  as:
         $f(\mathbf{x}^*) | \mathbf{D}_{train}, \mathbf{x}^* \sim \mathcal{GP}_{outer}(\mu(\cdot), cov(\cdot, \cdot; (\theta_1^{(t)}, \dots, \theta_H^{(t)})))$ 
21 end

```

Inference when implementing SQE-looking kernel and Matérn-looking kernel

SQE-looking implementation: In this case, $H = 1$ and $\theta_1 \equiv \ell$, hence there will be a single inner layer GP to model ℓ . δ is such one hyperparameter that parametrises the covariance function of the stationary GP in the inner layer using SQE kernel.

- The inference starts with proposing ℓ in the first block of the t -th iteration from $\mathcal{N}(\ell^{(t-1)}, \sigma_\ell)$, $\forall t = 1, \dots, N_{iter}$ and then updating ℓ using the accept-reject criterion of the Random Walk Metropolis Hastings (depicted in step-4 of the first block of inference in section 4.4.3) as the first block of the Metropolis-within-Gibbs inferential strategy presented in section 4.4.3.
- Then in the second block of the same iteration, lookback data $\mathbf{D}^{(t)}$ for $t > N_{LB}$ for ℓ is constructed and if constructed previously then it is updated with the current value of ℓ obtained at the end of first block.
- Then δ is updated as follows: it is proposed from $\mathcal{N}(\delta^{(t-1)}, \sigma_\delta)$, posterior of δ given $\mathbf{D}^{(t)}$ is computed for both the current and proposed values of δ in order to compute the acceptance criterion (depicted in step-4 of the second block of inference in section 4.4.3).
- Once δ is updated, using this updated δ and $\mathbf{D}^{(t)} = \{(i, \ell^{(i)})\}_{i=t-N_{LB}}^{t-1}$ as training data, the value of ℓ at the t -th iteration is then predicted which now becomes the current value of ℓ .
- The lookback data is then updated with this current value of ℓ obtained at the end of the second block.

Matérn-looking implementation: In this case, $H = 2$ and $\theta_1 \equiv \ell, \theta_2 \equiv \nu$. Therefore, there are two hyperparameters, δ_1, δ_2 to parametrise the covariance function of the stationary GPs in the inner layer using SQE kernel. δ_1, δ_2 are updated in the second block of the t -th iteration where $t > N_{LB}$, where δ_1 and δ_2 are proposed from $\mathcal{N}(\delta_1^{(t-1)}, \sigma_{\delta_1})$ and $\mathcal{N}(\delta_2^{(t-1)}, \sigma_{\delta_2})$ respectively. I have used Gaussian priors on ℓ and ν where these priors are experimentally chosen.

4.4.4 Single GP layer based models

While I have motivated a fully non-parametric model above, it is also possible to include a single stationary GP to generate sample functions that will be used to model the sought

function $f(\cdot)$. Such a stationary kernel could be an SQE or a Matérn - the one or two (respectively) constant hyperparameters of which I will learn, given the training dataset. For a literature on kernels see Genton (2001), on automatically designing new stationary kernels composed from base kernels see Duvenaud et al. (2013), on stationary spectral mixture kernel see Wilson and Adams (2013) - though the computational burden of automatically designing new stationary kernels limits the complexity of the selected kernels (Tobar et al., 2015) and in spite of the expressivity of the spectral kernels, Simpson et al. (2021) suggests that the spectral kernels get limited by its optimisation procedure because the form of such kernel includes multiple modes in the marginal likelihood which poses special challenges to the optimisation. I will display results from SQE and Matérn stationary single-layer models to showcase their shortcomings when modelling a real-world functional relationship between X and Y .

In this model the sought function $f(\cdot)$ is modelled as a random realisation from a Gaussian Process whose covariance structure has been modelled using a parametric, stationary kernel. Therefore, $f \sim \mathcal{GP}(\mu(x), \text{cov}(x, x'))$, where $\mu(\cdot)$ and $\text{cov}(\cdot, \cdot)$ are mean and covariance functions of this GP. Then by definition, the joint probability of a finite number of realisations of $f(\cdot)$ - such as M realisations of $f(\cdot)$ at M design points $x_1, \dots, x_M$ in the training data $\mathbf{D}_{train}$ - is Multivariate Normal, with the M -dimensional mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}^{(M \times M)}$.

I parametrise matrix $\boldsymbol{\Sigma}$ by saying that the ij -th element of this matrix that represents the covariance between the variable Y_i - that represents the output at $X = x_i$ - and Y_j , is modelled as a covariance function $K_S(\cdot, \cdot)$ of the distance between the inputs x_i and x_j at which these output variables are realised. But the joint of these M output variables, is the probability of the data on Y . In fact, it is a conditional probability - conditional on the parameters of the mean and the covariance matrix. Hence this Multivariate Normal becomes the likelihood of the inferential scheme. I will present results of this Single GP layer based parametric model where stationary SQE and Matérn kernel is used as $K_S(\cdot, \cdot)$ respectively while modelling with a 0-mean GP.

Equation 4.4.6 shows the stationary SQE kernel with length scale hyperparameter ℓ that has been invoked in this thesis with the training data where all outputs are standardised and hence I have not included and subsequently learnt the amplitude hyperparameter.

$$K_S(x_i, x_j) = \exp \left(-\frac{(x_i - x_j)^2}{2\ell^2} \right), \quad \ell > 0 \quad (4.4.6)$$

Equation 4.4.7 shows the stationary Matérn class of covariance kernels with positive hyperparameters ν and ℓ , where $d(\cdot, \cdot)$ is the Euclidean distance between x_i and x_j ; $K_\nu(\cdot)$ is

the modified Bessel function; and $\Gamma(\cdot)$ is the Gamma function.

$$K_S(x_i, x_j) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}}{\ell} d(x_i, x_j) \right)^\nu K_\nu \left(\frac{\sqrt{2\nu}}{\ell} d(x_i, x_j) \right) \quad (4.4.7)$$

4.5 Non-stationary extension of the non-parametric kernel

In the introduction of this chapter, I have already mentioned that for a kernel $K(\cdot, \cdot; \boldsymbol{\theta})$ where $\boldsymbol{\theta} \in \mathbb{R}^H$ which models the correlation structure of a stochastic process $\mathcal{GP}_{outer}$, to capture the non-stationarity, the kernel hyperparameters $\boldsymbol{\theta}$ need to be varied through out the input space. This has been achieved by modelling the kernel hyperparameters as a function of the input location such that, $\theta_k = g_k(x)$, $\forall k = 1, \dots, H$, in many of the standard literature of non-stationary kernels. However, this kind of modelling bears some ambiguities, because, when I am modelling the correlation between the outputs Y_i and Y_j realised at the corresponding inputs x_i and x_j , $\theta_i = g_i(x)$ can be written in several ways as follows.

1. θ_k can be expressed in terms of x_i only - $\theta_k = g_k(x_i)$.
2. θ_k can be expressed in terms of x_j only - $\theta_k = g_k(x_j)$.
3. θ_k can be expressed in terms of mean of x_i, x_j - $\theta_k = g_k((x_i + x_j)/2)$.
4. θ_k can be expressed in terms of maximum of x_i, x_j - $\theta_k = g_k(\max(x_i, x_j))$.

Many other statistical summaries obtained from x_i, x_j can be used to write $\theta_k = g_k(\cdot)$. Each of the models will have their own significance but conveys very less interpretation on the correctness of the modelling true covariance between the outputs Y_i and Y_j .

Therefore, in this thesis, I have modelled the hyperparameter θ_k that is unique to the index pair i and j of the inputs, i.e. for every input pair x_i and x_j , their will be a unique hyperparameter θ_{ij} . The correlation structure of the $\mathcal{GP}_{outer}$ then will be presented as,

$$\text{Corr}(Y_i, Y_j) = K(x_i, x_j; (\theta_{ij}^{(1)}, \dots, \theta_{ij}^{(H)})).$$

However, learning of θ_{ij} for all input pairs, is a computation intensive task and may be impractical for large datasets, because for M number of data points, the learning would require $O(M^2/2)$ number of θ_{ij} to be learnt.

Therefore, in this chapter, I have presented the non-parametric model that leverages the equivalence of sampling across time-step and sampling across input pairs in which $\theta_k = g_k(t)$ with $g_k(\cdot) \sim \mathcal{GP}_k(\mu_k(\cdot), \text{cov}_k(\cdot, \cdot))$ where $\mathcal{GP}_k$ is proven to be stationary, $\forall k = 1, \dots, H$. In

addition, I am also interested in learning θ_{ij} while escaping the humongous amount of learning by replacing with minimal learning that incurs lesser computation. So, in this section, I will present the non-stationary equivalent of the non-parametric kernel, where the non-stationary kernel is constructed as SQE-looking kernel. This SQE-looking non-stationary kernel consist of the length scale hyperparameter ℓ_{ij} pertaining to the input pair $X = x_i$ and $X = x_j$, where x_i and x_j are the i -th and j -th design points in training set $\mathbf{D}_{train}$.

Modelling correlation between each input pair: Modelling the correlation between a input pair x_i and x_j , may these inputs be design points or test input points, needs modelling the correlation at x_i and x_j . Now, this requirement encourages me to look around a small neighbourhood of each of x_i and x_j in the input space, such that, I define a small width ε to construct multiple neighbouring points around x_i within the width of ε and then generate closed form predictive means at those neighbouring points.

Now in a different approach, multiple sample functions can be generated from the outer GP and outputs at the multiple input locations around x_i can be obtained, however, this approach poses much computational burden for high-dimensional outer GP, hence, I opted to choose multiple neighbouring points around x_i in the input space and then generate closed form predictions at those neighbouring points.

To avoid any bias in the modelling of the correlation at x_i , the V number of neighbouring points D_{x_i} are constructed within the vicinity of $(x_i - \varepsilon)$ and $(x_i + \varepsilon)$ that are uniformly distributed within the ε -neighbourhood. It is to be noted that the ε should be much smaller than the minimum inter-input distance i.e. $|x_i - x_j|$, $\forall x_i, x_j \in (D_{train} \cup \mathbf{x}^*)$ because using this ε -neighbourhood, the correlation at x_i is being modelled. Similarly, V number of neighbouring points D_{x_j} can be generated for x_j .

So, now I compute the closed form predictive means at each of the neighbouring points of D_{x_i} and D_{x_j} using GP_{outer} with the non-parametric SQE kernel (see Definition 4.4.3) to yield D_{y_i} and D_{y_j} respectively.

Definition 4.5.1. D_{x_i} is the neighbouring input points $\{x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(V)}\}$ s.t. $|x_i^{(k)} - x_i| \leq \varepsilon$, $\forall k = 1, 2, \dots, V$. D_{y_i} is the predicted mean output $y_i^{(k)}$ at each neighbouring input point $x_i^{(k)}$, where, $D_{y_i} = \{y_i^{(1)}, y_i^{(2)}, \dots, y_i^{(V)}\}$, and,

$$y_i^{(k)} | \mathbf{D}, x_i^{(k)} = (e^{-((x_i^{(k)} - x_1)^2 / 2g_1(t)^2)} \dots e^{-(x_i^{(k)} - x_M)^2 / 2g_1(t)^2}) (\boldsymbol{\Sigma})^{-1} (y_1, \dots, y_M)^T$$

.

$\boldsymbol{\Sigma} = [\exp(-(x_i - x_j)^2 / 2(g_1(t))^2)]$ is the correlation matrix in the non-parametric kernel model that is used to generate the predictions.

Definition 4.5.2. The empirical correlation $\text{Corr}(D_{y_i}, D_{y_j})$ computed using D_{y_i} and D_{y_j} is,

$$\text{Corr}(D_{y_i}, D_{y_j}) = \frac{(D_{y_i} - \bar{D}_{y_i})(D_{y_j} - \bar{D}_{y_j})}{V - 1} = \frac{\sum_{k=1}^V ((y_i^{(k)} - \bar{D}_{y_i})(y_j^{(k)} - \bar{D}_{y_j}))}{(V - 1)}$$

where $\bar{D}_{y_i}, \bar{D}_{y_j}$ are the mean of the D_{y_i} and D_{y_j} respectively.

Corollary 4.5.1. Input location-pair specific hyperparameter ℓ_{ij} is estimated by equating the empirical correlation $\text{Corr}(D_{y_i}, D_{y_j})$ to the non-stationary kernel-parametrised correlation using the SQE-looking kernel as:

$$\text{côrr}(Y_i, Y_j) = \exp\left(-\frac{(x_i - x_j)^2}{(\ell_{ij})^2}\right),$$

s.t.

$$\ell_{ij} = \sqrt{\frac{-(x_i - x_j)^2}{\ln(\text{côrr}(Y_i, Y_j))}}.$$

This holds $\forall i \neq j, i, j = 1, \dots, M$. For $i = j$, the length scale of the correlation function is unity.

Non-stationary prediction: In order to generate non-stationary prediction with the model developed in this chapter, I construct evenly distributed neighbouring points for each input point within its ε -neighbourhood followed by the correlation computation for each pair of inputs x_i and x_j using their corresponding $D_i^{(y)}$ and $D_j^{(y)}$. By this way, I will get length scale hyperparameters at pair of inputs (x_i, x_j) where each input belongs to:

1. Training data s.t. $\forall x_i, x_j \in \mathbf{D}_{\text{train}}$.
2. Test data s.t. $\forall x_i, x_j \in \mathbf{x}^*$ where $\mathbf{x}^*$ is the test input.
3. One of which is a test input point and the other is a design (training) input s.t. $\forall x_i, x_j \in (\mathbf{x}^* \cup \mathbf{D}_{\text{train}})$.

Once, these ℓ_{ij} hyperparameters are estimated, then closed form predictions are performed at the test input points $\mathbf{x}^*$ with GP_{outer} .

Definition 4.5.3. The closed for prediction of the expectation and variance of the output Y^* realised at a test input at $X = x^*$, $x^* \in \mathbf{x}^*$ are computed as:

$$\mathbb{E}(Y^* | \mathbf{D}_{\text{train}}, \mathbf{x}^*) = (e^{-((x^* - x_1)^2 / (\ell_{*1})^2)} \dots e^{-(x_{\text{test}} - x_M)^2 / (\ell_{*M})^2}) \mathbf{\Sigma}^{-1} (y_1, \dots, y_M)^T, \quad (4.5.1)$$

$$\begin{aligned} \text{Var}(Y^* | \mathbf{D}_{\text{train}}, \mathbf{x}^*) &= 1 - (e^{-((x^* - x_1)^2 / (\ell_{*1})^2)} \dots e^{-(x^* - x_M)^2 / (\ell_{*M})^2}) \mathbf{\Sigma}^{-1} \\ &\quad (e^{-((x_1 - x^*)^2 / (\ell_{*1})^2)} \dots e^{-(x_M - x^*)^2 / (\ell_{*M})^2}). \end{aligned} \quad (4.5.2)$$

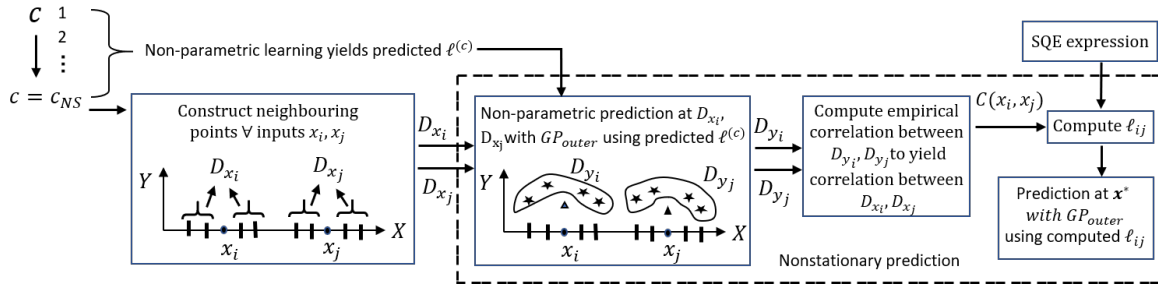


Fig. 4.2 The methodical flow of non-stationary extension of the SQE-looking non-parametric kernel Cartoon graphical representations. The cartoon graphs represents inputs along X axis and outputs along Y axis, four neighbouring points of two training inputs x_i and x_j are depicted in short vertical lines along with its corresponding predictive means in stars, triangle shows the training output realised at x_i and x_j .

$\Sigma = [\exp(-(x_i - x_j)^2 / 2(\ell_{ij})^2)]$ is the correlation matrix in the non-stationary model.

Remark 4.5.1. For a multivariate d -dimensional input, the construction of the non-stationary kernel follows the similar scheme. The construction of neighbouring points for an input point in X will be performed along each direction in X . Prediction of the outputs at the neighbouring points of input points along each i -th input dimension will be used to generate length scale hyperparameters for the i -th dimension where $i = 1, \dots, d$.

All the outputs are standardised and inputs remain unstandardised for both inference and GP predictions. Using this protocol, equivalent Matérn-looking non-stationary kernel can be built from the Matérn-looking non-parametric kernel.

Implementation and numerical issues of this non-stationary kernel: At first, the non-parametric model is built, then on top this non-parametric model, the non-stationary extension of this non-parametric model is implemented. I start with, learning the hyperparameter of the non-parametric kernel pertains to the outer GP, using MCMC inference and thereby, update the value of the kernel hyperparameter in its each iteration c . Post convergence, at iteration $c = c_{NS}$, using the learnt value of the kernel hyperparameter, the non-stationary model is built.

Figure 4.2 presents the methodological flow for this non-stationary extension of non-parametric kernel with some graphical cartoons to depict the construction of the neighbouring points, computation of correlations, estimation of the location-pair specific lengthscales etc. for only two training inputs x_i and x_j . The dotted box in this figure mainly depicts the non-stationary prediction given the neighbouring points, $data_{\ell}^{(c)}$ and the form of the $K(.,.)$ which is SQE in this illustration.

The correlation matrix computed using the ℓ_{ij} can have some numerical aspects as the estimation of the $\{\ell_{ij}\}_{i<j; i,j=1}^M$ may incur some numerical instability.

- The correlation matrix Σ can be unnormalised - it is then normalised by dividing each element by the largest correlation present in it.
- The normalisation step may incur another numerical issue further when the normalised Σ will be treated as the covariance matrix to compute the input location-pair specific lengthscales. During normalisation process, if the largest element of Σ , is not a diagonal element then after normalisation, Σ will contain unity as a non-diagonal element. That will mean that $Cov(x_i, x_j) = 1$ when $x_i \neq x_j$. Hence, to maintain the properties of the covariance matrix Σ , it needs to be ensured that the very non-diagonal unity element is less than unity by subtracting a small noise from it.
- As $corr(Y_i, Y_j)$ is approximated by the empirical estimate of correlation computed at the output values predicted at the neighbouring input points within the ε -neighbourhood of x_i, x_j , $\forall x_i, x_j \in \mathcal{X}$, the empirical correlation estimate may not approximate $corr(Y_i, Y_j)$ correctly, especially when the sample functions of $\mathcal{GP}_{outer}$ are very rough (this may happen when data bears high amount of inhomogeneities in its correlation structure) within the ε -neighbourhood of x_i, x_j . Subsequently, the computation of ℓ_{ij} will be incorrect as computation of ℓ_{ij} is dependent on the $corr(Y_i, Y_j)$.

Given these numerical issue, I will now explore another model, that I call Blended model where the non-parametric and the non-stationary kernel models are jointly fed.

4.6 Blended model: a non-stationary non-parametric kernel

This new kernel is designed in a way that blends the non-parametric kernel and its equivalent non-stationary kernel - hence it is called Blended kernel model. I will discuss this Blended model in the context of SQE-looking kernel with length scale ℓ for the univariate situation. The likelihood function $\mathcal{L}(\cdot)$ is defined as a Multivariate Normal of the variable $\mathbf{y}^*$ that is the difference in the prediction obtained from the non-parametric model $\mathbf{y}^*$ and the non-stationary model $\mathbf{y}^{*'} at test inputs $\mathbf{x}^*$ with mean as null vector i.e.$

$$\mathbf{y}^* = |\mathbf{y}^* - \mathbf{y}^{*'}|.$$

Broadly there are three stages to achieve this Blended kernel.

1. In the first stage, the non-parametric learning is performed and the chain is run until convergence is achieved. Post-convergence, the learnt hyperparameter, i.e. the current value of the ℓ , is used to perform the non-parametric prediction on the test data $\mathbf{x}^*$ and then propagated to the second stage.
2. In the second stage, the propagated current value of the ℓ from the first stage, is used to construct the non-stationary kernel model to yield location-pair specific hyperparameters ℓ_{ij} to generate the non-stationary prediction at the test inputs $\mathbf{x}^*$.
3. In the third stage, a new learning is employed to update the current value of the ℓ that has been propagated from the first stage using the likelihood function $\mathcal{L}(\cdot)$ using a new MCMC chain. The updated length scale in this stage is referred as ℓ_B .

As the current value of the ℓ , from the first stage, is obtained after the Markov chain achieves convergence, still the current value of the ℓ_B will change as the new chain will change due to the in-built variance within the new chain. Hence, the non-parametric prediction will change as well as the non-stationary prediction will also change where the predictions are made on the same test inputs $\mathbf{x}^*$. Therefore, the difference variable $\mathbf{y}^*$ will also change along with the likelihood defined in terms of the $\mathbf{y}^*$. This will enable the sampling from the posterior and design another Markov chain in this third stage where the length scale ℓ_B will vary with having its seed set to the current value of the ℓ that is propagated from the first stage. This implies that now, I will vary the current value of the ℓ , that is propagated from the first stage, in this newly designed chain of the Blended model, by proposing it, then updating it through accept-reject, and show that the difference of the non-parametric prediction and the non-stationary prediction remains within a small scale while the Blended model converges. So the likelihood becomes,

Definition 4.6.1. *The likelihood within the Blended kernel given the training data $\mathbf{D}_{train}$ is defined as*

$$\mathcal{L}(\ell_B | \mathbf{D}_{train}, \mathbf{x}^*) = \frac{1}{\sqrt{(2\pi)^{M'} |\boldsymbol{\Sigma}^*|}} \exp \left(-\frac{1}{2} (\mathbf{y}^*)^T (\boldsymbol{\Sigma}^*)^{-1} (\mathbf{y}^*) \right),$$

where M' is the cardinality of the test inputs $\mathbf{x}^*$, $\boldsymbol{\Sigma}^*$ is the variance-covariance matrix which tells the scale of the difference between the predictions.

This scale reflects the numerical error introduced, while computing the predictions, which might not be the same for all the inputs, because performing predictions for some inputs can be harder. However, I do not want to introduce any bias in the predictions at any particular input, so the scale at which the difference lies between the non-parametric model and the

non-stationary model is same and fixed manually. So it is a diagonal matrix with a constant value in its diagonals. The inference for Blended model is as follows where there is no inner layer but only learning ℓ_B . This inference can be extended for other kind of non-parametric kernel, apart from the SQE-looking non-parametric kernel.

- Perform non-parametric learning and prediction at test inputs $\mathbf{x}^*$, using the current value of ℓ , in the first stage, as mentioned previously; propagate the current value of the ℓ to later stages. This current value of ℓ , acts as seed in the third stage.
- Perform non-stationary learning, using the learnt non-parametric kernel in the first stage and generate non-stationary prediction at test inputs $\mathbf{x}^*$ in the second stage.
- Propose the value of ℓ_B , at t -th iteration, form a proposal density $\mathcal{N}(\cdot, \sigma_B)$ as $\ell_B^* \sim \mathcal{N}(\ell_B^{(t-1)}, \sigma_B)$ where variance of the proposal Normal density σ_B is experimentally chosen.
- In the same iteration, construct the non-stationary kernel using the proposed ℓ_B^* and compute the likelihood $\mathcal{L}(\ell_B | \mathbf{D}_{train}, \mathbf{x}^*)$. After invoking prior on the ℓ_B , the joint posterior is computed.
- The proposed ℓ_B^* is either accepted or rejected based on the accept-reject criterion of the chosen flavour of MCMC that has been implemented. In this work, I have performed learning with Metropolis Hastings.
- Post-convergence, non-stationary predictions at test inputs $\mathbf{x}^*$ are generated with the updated ℓ_B .

Remark 4.6.1. *The construction of the neighbouring points $D_i^{(x)}$ for all the input points is performed only once and used all through.*

Figure 4.3 depicts the flow of the three stages along with the Blended model that uses SQE-looking non-parametric kernel, in univariate situation.

4.7 Results

I will now illustrate the results of the developed models along with the comparisons using some of the state-of-the-art kernels with the Brent crude oil price data obtained from <https://datasource.kapsarc.org>. Although this is a time series data, but the newly developed models are not restricted for time series data only, rather the developed kernels are well suited for

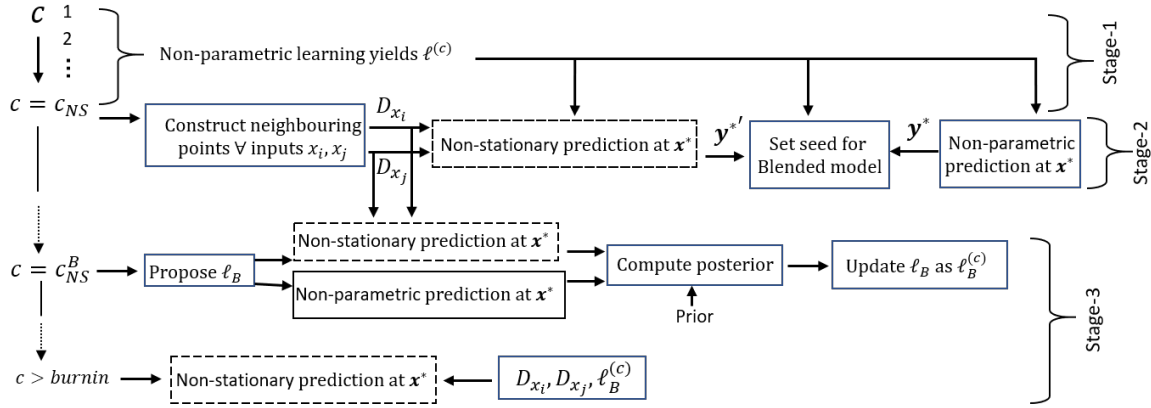


Fig. 4.3 The methodical flow of three stages along with Blended model using SQE-looking non-parametric kernel. The dotted box presents the whole flow of non-stationary prediction presented in Figure 4.2.

diverse real-world applications to learn the functional relationship between the output and input variable. This dataset was chosen for the illustration purpose because it is univariate, it has inhomogeneous correlation structure along with lots of missing data and the input temporal points (at which, an output oil price is realised) are not uniformly, evenly spaced. As stated before, in my implementation, only outputs are standardised for inference and predictions obtained using GP regression, while inputs remain unstandardised. In this section, I will first describe the dataset, then present the results with the newly developed models along with the comparative models from literature, on the same empirical training dataset using same test input points. I have computed mean squared error (MSE) for both the predicted mean and the uncertainties (95% Highest Probability density credible region or HPD (Kruschke, 2015)) for all the models as performance metric and presented at the end of this section in the Table 4.1. In this thesis, the convergence diagnostic for the learning of the hyperparameters has been performed by sampling the values of the hyperparameter over three distinct, equally-wide (covering k consecutive iterations taken from k_1, k_2, k_3 iterations for the three windows respectively), non-overlapping iteration windows during the MCMC chain (post burnin), where $k_1 \neq k_2 \neq k_3$. The three histograms are coloured in blue, red, black, and the windows are overplotted. All the Implementations are done in Python programming language <https://www.python.org/>.

4.7.1 Data

The Brent oil prices are taken for the period of 5th January, 2018 to 8th March 2022, however, it does not contain oil prices for every day, there are missing days all along the data. From

the whole data set, oil prices for the periods 5th January, 2018 to 29th December 2021 are considered as training data comprising of 184 training points. From the time range of training data, randomly and more or less uniformly 20 data points are chosen as test data for prediction task. I have transformed the dates into the days in running total i.e. the date of the first data point is denoted as day zero, the second data points corresponding date is denoted as day one and so on. In this thesis, all the prediction results will be presented against the days in running total. The training data pertaining to this oil price data used all through this thesis is shown in Figure 4.4.

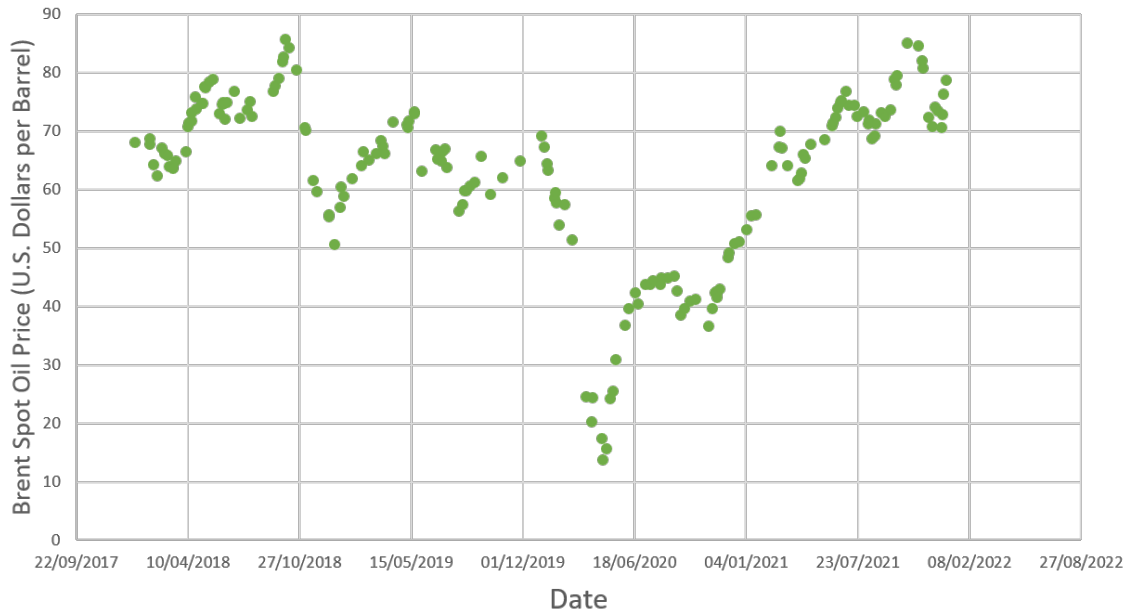


Fig. 4.4 Brent crude oil price training data. Along x axis is dates and along y is the oil price in U.S Dollars.

4.7.2 Results from SQE-looking non-parametric kernel:

I have used 2-block Random walk MCMC to learn the hyperparameters of the SQE-looking non-parametric kernel using dual-layered learning model. The mean of the 95% HPD credible region on the learnt length hyperparameters, ℓ in the first block, the predicted ℓ at the end of second block and δ in this two-layered SQE-looking non-parametric kernel model are 23.9369, 23.9368 and 0.5989 respectively. Figure 4.5 shows the traces of learnt ℓ , predicted ℓ , δ along with the logarithm of the posterior and convergence diagnostics of the learnt ℓ , predicted ℓ , δ . Final prediction of the oil prices using this model is depicted in the left panel of the top row in the Figure 4.27. Figure 4.6 shows prediction traces of the test points for

this learning. This MCMC chain was run for 50000 iterations; the variances of the Normal proposal densities for ℓ and δ are 0.01^2 , and 0.01^2 respectively. Normal priors were used for ℓ and δ . Prior mean for ℓ is 24 with variances 0.01^2 , prior mean for δ is 0.6 with variances 0.01^2 . Seed for ℓ and δ are 23.93 and 0.75 respectively. Lookback window size is set to 16. Prediction started from 30000 iteration.

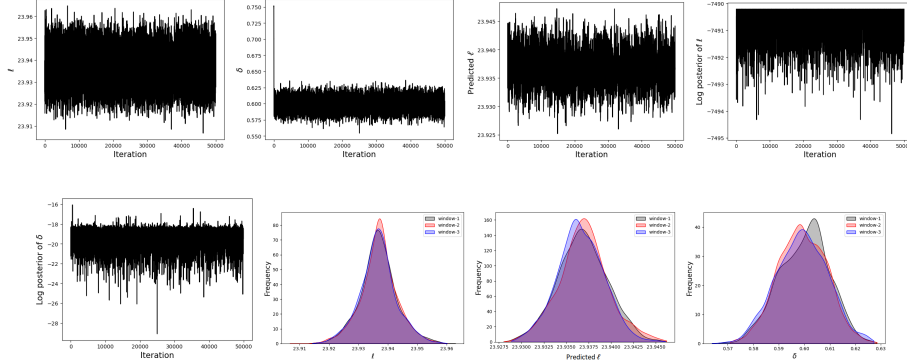


Fig. 4.5 Results of learning hyperparameters with dual staged model using SQE-looking non-parametric kernel using Brent crude oil price data. The panels from left, in the top row depict the trace of ℓ that is updated in the first block, δ learnt in the second block, predicted ℓ using the inner GP at the end of second block, the logarithm of posterior in the outer GP, given the training data. The panels from left, in the bottom row depict the trace of logarithm of the posterior of δ in the inner GP, given the dynamically-varying lookback data and the convergence diagnostics pertaining to the ℓ , predicted ℓ and δ respectively.

4.7.3 Results of Matérn-looking non-parametric kernel:

2-block Random walk MCMC has been used to learn the hyperparameters of the Matérn-looking non-parametric kernel. I have learnt both the kernel hyperparameters ℓ, ν . The kernel hyperparameter of the inner stationary GP $\mathcal{GP}_1$ that further models ℓ , is δ . Similarly, δ_ν is the kernel hyperparameter of the inner stationary GP $\mathcal{GP}_2$ that models ν .

The mean of the 95% HPD credible region on the learnt length scale hyperparameters, ℓ in the first block, the predicted ℓ at the end of second block and δ in this non-parametric model are 30.0001, 29.9999 and 0.6001 respectively.

The mean of 95% HPD credible region on the learnt length hyperparameters, ν at the end of first block, predicted ν in the second block and δ_ν in this model 0.9669, 0.9666 and 0.2981 respectively.

Results of learning ℓ and ν using dual-layered learning model that invokes Matérn looking non-parametric kernel, with the Brent crude oil price data is shown in Figure 4.7. This MCMC chain was run for 50,000 iterations; variances of the Normal proposal densities

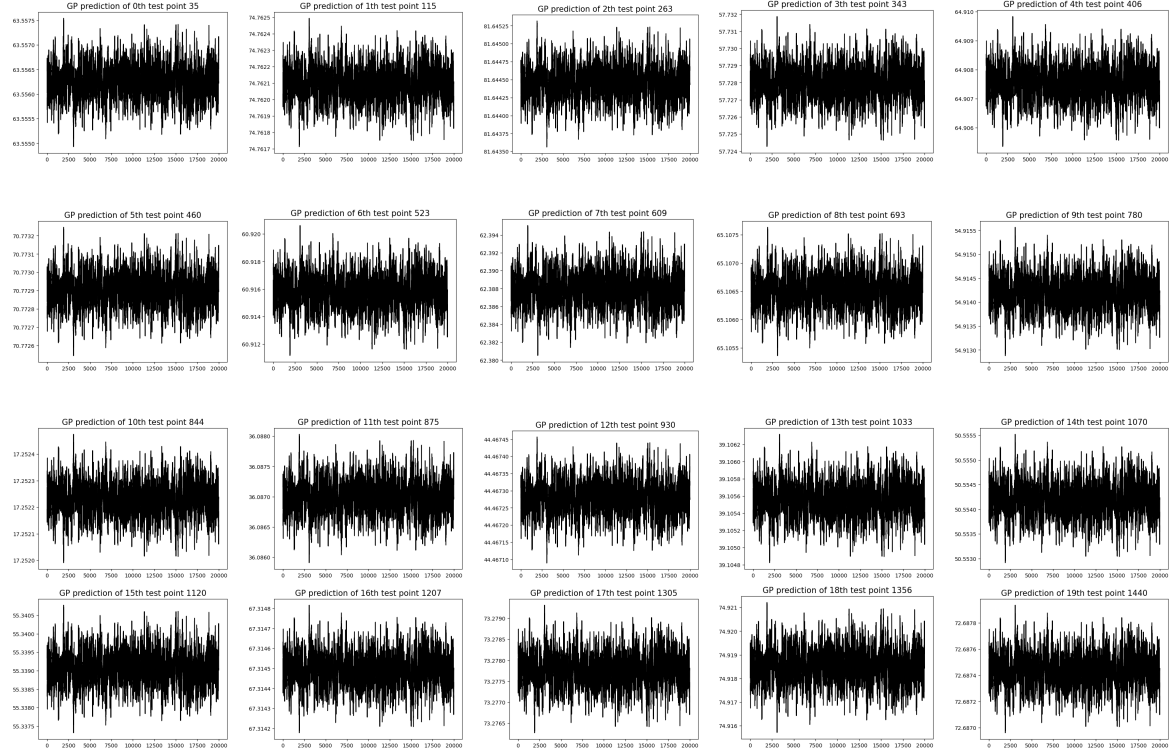


Fig. 4.6 Traces of prediction of the test points with SQE-looking non-parametric kernel. The x-axis and y-axis of each plot denotes the iteration and the oil prices respectively.

for ℓ , δ_ℓ , v , δ_v are 0.01^2 , 0.01^2 , 0.02^2 , 0.01^2 respectively. Normal priors were used on all sought hyperparameters; prior mean for ℓ is 30 with variance of 0.01^2 ; prior mean for δ_ℓ is 0.6 with variance of 0.02^2 ; prior mean for v is 0.96 with variance of 0.02^2 ; prior mean for δ_v is 0.3 with variance of 0.02^2 . Seed for ℓ , δ_ℓ , v and δ_v are 30, 0.6, 0.96 and 0.3 respectively. The width of the lookback window, i.e. $N_{LB} = 16$; and prediction of outputs at the identified test inputs were initiated at the 30000-th iteration. Traces of prediction at the test points for dual staged learning with Matérn-looking non-parametric kernel are presented in the Figure 4.8. Final prediction of the oil prices using this model is depicted in the middle panel of the top row in the Figure 4.27.

4.7.4 Results of learning with a non-parametric model, given noisy data

I have undertaken learning with noisy data, where the learning was undertaken with an SQE-looking non-parametric kernel. Indeed, the output in the Brent crude oil price data (considered in this thesis), is a noise-free variable. Therefore, to showcase the capacity in the learning to work with noisy data, I generate said noise, assuming Gaussian errors, I superimpose the error variable $\varepsilon_e \sim \mathcal{N}(0, \sigma_\varepsilon^2)$ on each of the M number of outputs of the

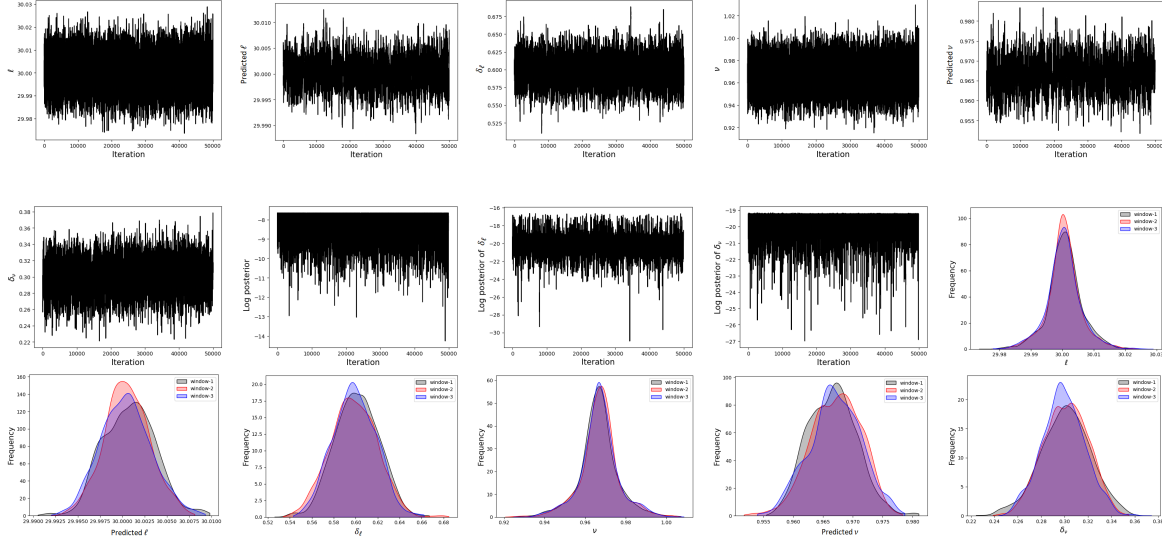


Fig. 4.7 Results of learning ℓ and v using the non-parametric model with Matérn-looking kernel, given Brent crude oil price data. *Top row, from left:* trace of ℓ updated in the first block of the conducted Metropolis-within-Gibbs chain; trace of ℓ as predicted from the $\mathcal{GP}$ in the inner-layer, at the ℓ and v that are updated in the first block; trace of δ ; trace of v updated in the first block of the chain; trace of v value predicted using the $\mathcal{GP}$ in the inner-layer. *Middle row, from left:* trace of δ_v ; trace of logarithm of the joint posterior of ℓ and v given training data; traces of logarithm of posterior of δ_ℓ and δ_v respectively, given relevant (dynamically-varying) lookback data; convergence diagnostic of ℓ . *Bottom row, from left:* convergence diagnostics for predicted ℓ , δ_ℓ , v , predicted v , δ_v .

training set that is constructed using the Brent crude oil prices, where I choose $\sigma_\varepsilon^2 = 0.01$. I refer to the noisy data as $\mathbf{D}/$. In this learning, I add the matrix $\sigma_\varepsilon^2/s^2 I$, to the correlation matrix Σ of the Multivariate Normal likelihood depicted in Equation 4.4.4, where I is a diagonal matrix. Here, s is the sample standard deviation employed in standardising the outputs of the data; $s \approx 14.57$. Subsequent to the learning with data $\mathbf{D}/$ and prediction at test inputs, I unstandardise the learnt values of the error variance with s^2 . Then predicted mean at the test point x^* is given as

$$(e^{-(x^*-x_1)^2/2\ell^2} \dots e^{-(x^*-x_M)^2/2\ell^2})(\Sigma + \frac{\sigma_\varepsilon^2}{s^2} I)^{-1}(y_1, \dots, y_M)^T,$$

and variance will be,

$$1 - (e^{-(x^*-x_1)^2/2\ell^2} \dots e^{-(x^*-x_M)^2/2\ell^2})(\Sigma + \frac{\sigma_\varepsilon^2}{s^2} I)^{-1}(e^{-((x_1-x^*)^2/2\ell^2} \dots e^{-(x_M-x^*)^2/2\ell^2}),$$

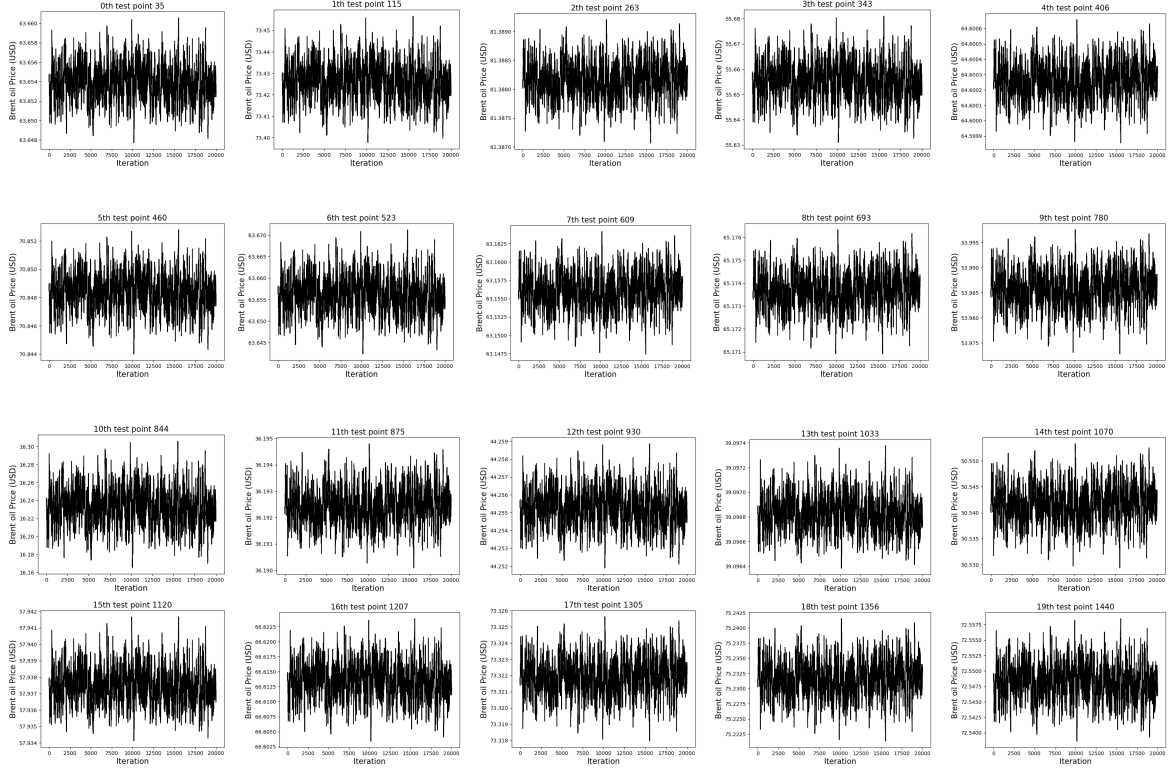


Fig. 4.8 Traces of prediction of the test points for nested learning with Matérn-looking kernel. The x-axis of each plot denotes the iterations and the y-axis shows the oil prices. The left most panel of the top row shows the histogram for test point zero, the second left panel shows the histogram for test point one, and similarly incrementally the right most panel shows the histogram for test point four; in the second row from top, the left most panel shows the histogram for test point five, the second left panel shows the histogram for test point six, and Similarly incrementally the right most panel shows the histogram for test point eight; Similarly with this pattern of increment the bottom most row presents the histogram for test point 17, the second left panel shows the histogram for test point 18 and right most panel shows the histogram for test point 20.

In this learning exercise the error variance σ_ϵ^2 is inferred upon in the first block of the Metropolis-within-Gibbs implementation of the MCMC-based inference. From the first block of the Metropolis-within-Gibbs chain, run with the constructed noisy data $\mathbf{D}'$, the mean of 95% HPD credible region on the learnt length scale hyperparameter ℓ is learnt to be 24.0001. The mean of the 95% HPD on predicted ℓ value obtained from the second block of the chain is again about 24.0003. The mean of the 95% HPD on δ is learnt as 0.2918. The learnt mean of the 95% HPD on the error variance σ_ϵ^2 is learnt to be about 0.011 (it is to be recalled that the true value of this parameter to be 0.01).

Figure 4.9 displays traces of ℓ , δ , σ_ε^2 , and of logarithm of the joint posterior of these variables, given the noisy data. The prior on the error variance has been chosen from pilot runs of the MCMC chain with uniform prior. This MCMC chain was run for 50000 iterations; variances of the Normal proposal densities for ℓ , δ are 0.1^2 , 0.02^2 respectively. Truncated Normal proposal is used for error variance σ^2 with variance 0.01^2 left truncated at 0. Normal priors were used for ℓ , δ and σ^2 . Prior mean for ℓ is 24 with variances 0.1^2 , prior mean for δ is 0.3 with variances 0.1^2 , prior mean for σ^2 is 0.15 with variances 0.1^2 . Seed for ℓ , δ and σ^2 are 23.9, 0.3 and 0.006 respectively. The width of the lookback window, i.e. $N_{LB} = 16$; and prediction of outputs at the identified test inputs were initiated at the 30000-th iteration.

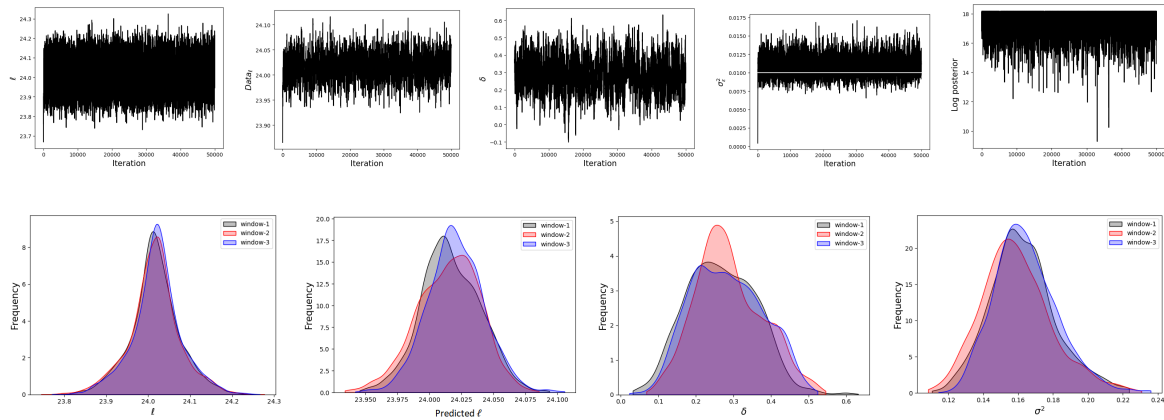


Fig. 4.9 Results of learning with the non-parametric model that uses the SQE-looking kernel, given noisy Brent crude oil price data. The panels from left in the top row depict traces of ℓ ; of ℓ predicted in second block; of δ ; of the error variance σ_ε^2 (with the true value of σ_ε^2 depicted in white line); and the logarithm of the joint posterior of ℓ and σ_ε^2 , given the data noisy data. The panels from left in the bottom row display convergence diagnostics pertaining to the learning of ℓ , predicted ℓ , δ and σ_ε^2 .

Figure 4.10 shows histograms of outputs predicted at five test points, where the predictions are undertaken after learning the temporal-dependence of oil price using the SQE-looking non-parametric kernel with noise-free data and with noisy data. The wider of the two histograms (in broken lines), represents the marginal of the predicted output at the chosen test inputs, given the noisy data.

Prediction of the oil prices using this implementation of the non-parametric kernel with the noisy oil price data, is depicted right panel of the top row in the Figure 4.27. It is to be noted that the uncertainty in the predictions made given the noisy data (presented in the Figure 4.10) is in fact about 0.2 and hence the error bar is not visible in the main prediction plot.

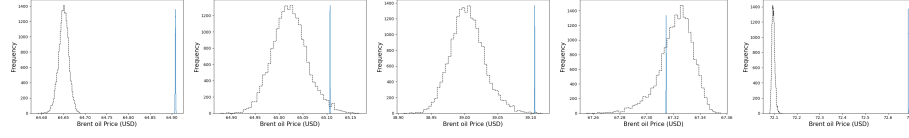


Fig. 4.10 Figure displaying histograms of output values predicted at five test inputs, obtained from the fully non-parametric model - with an SQE-looking kernel that parametrises $\mathcal{GP}_{outer}$ - given noisy data (in black dashed lines), and noise-free data (in blue solid line). The histogram displaying results given the noise-included data, is wider than that depicting prediction results obtained with the noise-free data.

4.7.5 Results of non-stationary model extending from non-parametric model

I have implemented the non-stationary version of the non-parametric model with the SQE-looking kernel as mentioned in Section 4.5. Here, the neighbourhood width ε is set to be 0.092, with $V = 100$ number of neighbouring points constructed within any ε -neighbourhood of each input. Prediction of oil prices using this model is depicted in the left panel of the middle row in the Figure 4.27.

Figure 4.11 and Figure 4.12 respectively present heat maps of the correlation matrices and the contour plots of the length scales ℓ_{ij} for the learning of the non-stationary equivalent of the non-parametric model that uses an SQE-looking kernel. Figure 4.13 shows histograms of output values predicted at five test inputs, obtained from the non-stationary equivalent of the non-parametric model.

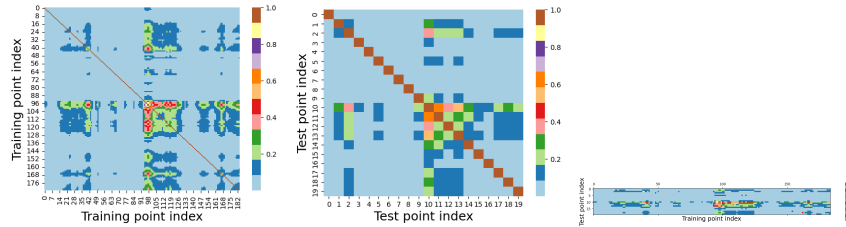


Fig. 4.11 Correlation matrices of the non-stationary equivalent of the non-parametric model that invokes an SQE-looking kernel. The left most plot presents the correlation matrix computed between outputs realised at pairs of design points in the training dataset $\mathbf{D}_{train}$, while the middle plot shows the correlation matrix computed between outputs realised at input pairs that each comprise test inputs in $\mathbf{D}_{test}$. Compared to these panels, the right-most panel depicts the correlation matrix between an output realised at a design point, and another that is realised at a test input.

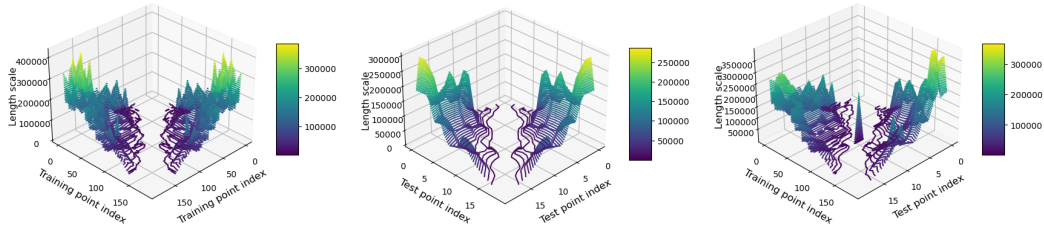


Fig. 4.12 Similar to panels in Figure 4.11, except here, contour plots of the computed length scales - of the non-stationary equivalent of the non-parametric model - are depicted.

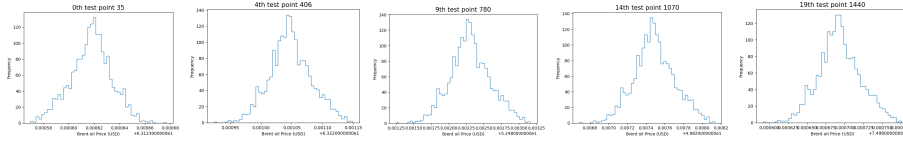


Fig. 4.13 Figure displaying histograms of output values predicted at five test inputs (every fifth), obtained from the non-stationary equivalent of the non-parametric model.

4.7.6 Results of Blended model

In this subsection, I present the results of the Blended model that blends the aspects of the non-parametric learning with SQE-looking kernel and the non-stationary model, hence, $\theta_1 = \ell_B$ is learnt in this Blended model. The mean of the 95% HPD credible region on the learnt length scale hyperparameter, ℓ_B in the Blended model is ≈ 30 . Figure 4.14 presents results of learning in the Blended model including the convergence diagnostic. Figure 4.15 presents traces of few of the input-pair specific length scales ℓ_{ij} . Traces of the mean output predicted at four test points - which happen to be every fifth test input - are included in Figure 4.16 Figure 4.17 and Figure 4.18 present heat maps of the correlation matrices and the contour plots of the length scales ℓ_{ij} for this Blended model. Finally, prediction of the oil prices using this model is depicted in the middle panel of the middle row of the Figure 4.27.

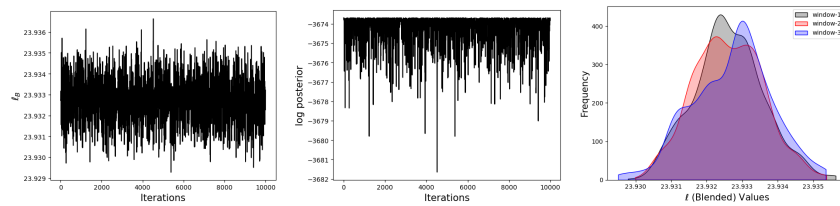


Fig. 4.14 The left panel depicts the trace of the length scale hyperparameter ℓ_B of the Blended model. The middle panel depicts logarithm of the posterior pdf of ℓ_B given data $\mathbf{D}_{train}$. The right panel presents the convergence diagnostic of the learning of ℓ_B .

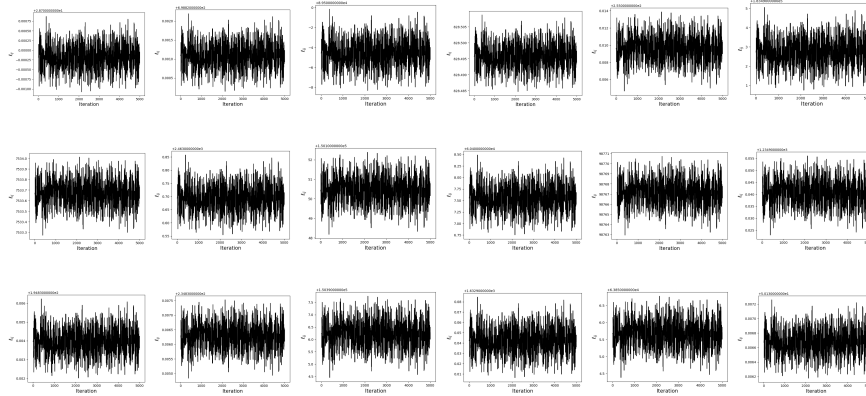


Fig. 4.15 Figure displaying length scale hyperparameters relevant to the implementation of the Blended model. The top row depicts lengths scales computed at input pairs, each of which is a design point. These inputs relevant to the presented plots are (from the left) at the 10-th and 5-th design inputs; the 15-th and the 2nd; the 15-th and the 50-th design points; the 30-th and the 10-th; the 60-th and the 65-th design points; the 180-th and the 10-th. The middle row depicts traces of length scales that are computed at input pairs, each of which is a test input. From the left, panels depict traces of length scale parameters that are relevant to the pair that comprises the 1st and 3rd of the chosen 20 test inputs; the 3rd and 4-th; 2nd and the 14-th test points; the 5-th and the 15-th; the 10-th and the 18-th; and the pair comprising the 19-th and 18-th test points. Bottom row depicts traces of length scales that are computed at input pairs, one of which is a design point, and the other a test input. From the left, the presented traces are pertinent to the pair of the 1st test and the 10-th design inputs; the 1st test and the 25-th design point; the 2nd test and the 100-th design inputs; the 5-th test and the 50-th design; the 9-th test and 25-th design; and the 19-th test and the 180-th design points.

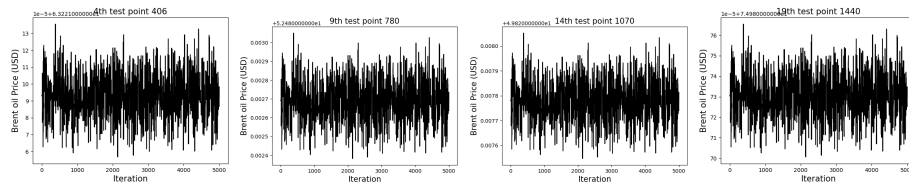


Fig. 4.16 Traces of mean outputs predicted at the test points (every fifth test input), where the predictions are obtained using the Blended model.

Now, I will present comparisons of the predictive capacity of the models I have motivated above and multiple existing models, including two well-known stationary kernels; the non-stationary kernel developed by Remes et al. (2017); and that by Paciorek and Schervish (2003).

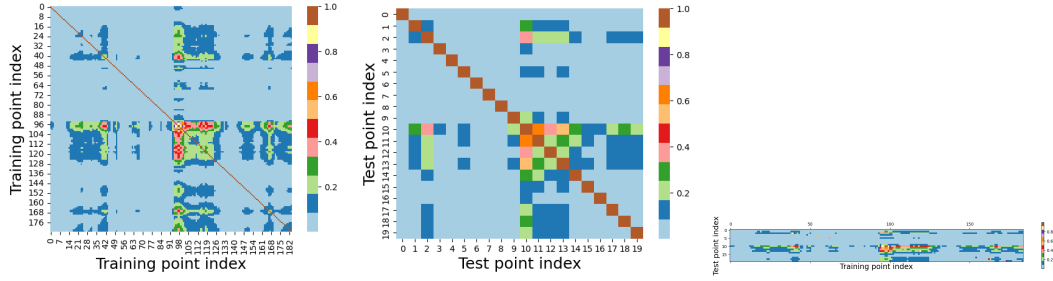


Fig. 4.17 The left-most plot presents a heat map of the correlation matrix, any element of which is the inter-output correlation in the Blended model, where each correlation element is between outputs realised at a pair of design inputs. The heat map in the middle plot represents the correlation computed at inputs pairs, where each of the pair is a test input. The right-most plot depicts the heat map of the correlation matrix, an element of which is the correlation between an output realised that is predicted at a test input, and another that is realised at a design time.

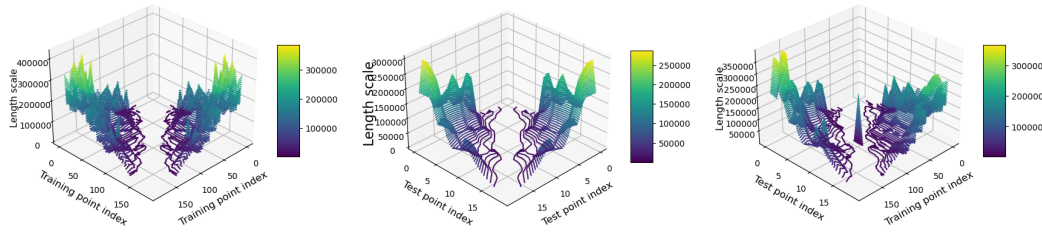


Fig. 4.18 Similar to Figure 4.17, except this plots depicts the contour plots of length scales at which an inter-output correlation is kernel-parametrised by the kernel of the Blended model.

4.7.7 Single GP layer based model using stationary SQE kernel

The only hyperparameter of relevance to this model is the length scale hyperparameter, ℓ (depicted in the Equation 4.4.6). The mean of its learnt 95% HPD is 4.07. Figure 4.19 shows the trace of this learnt ℓ , in addition to the joint log posterior. Predictions of the oil prices (at test times), made using this model are depicted in the right panel of the middle row in the Figure 4.27. Figure 4.20 depicts the traces of mean outputs predicted at the test points with the stationary SQE kernel given the training data.

4.7.8 Single GP layer based model using stationary Matérn kernel

In this model, the learnt hyperparameters of the kernel are ℓ and ν (depicted in the Equation 4.4.7). Means of the learnt values of ℓ and ν in the Matérn kernel are 5.29 and 5.39 respectively. Figure 4.21 displays the learnt traces, in addition to the log posterior. Final predictions using this model are depicted in the left panel of the bottom row in the Figure

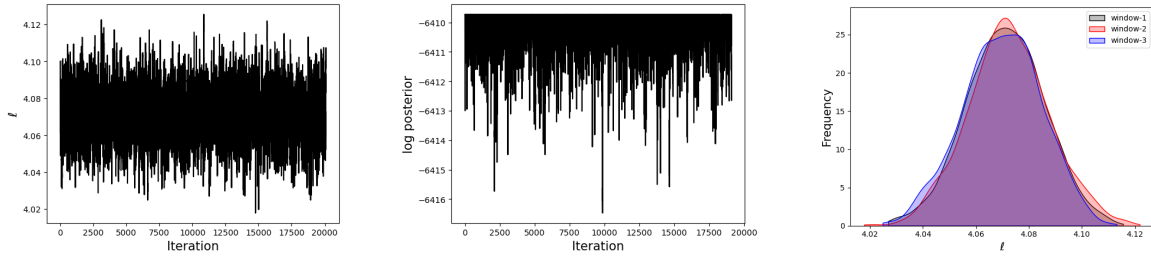


Fig. 4.19 Traces of learnt length scale parameter ℓ (left panel), logarithm of posterior *pdf* of ℓ given the training data (middle panel) and the convergence diagnostics on the learning of ℓ (right panel), using a stationary SQE kernel.

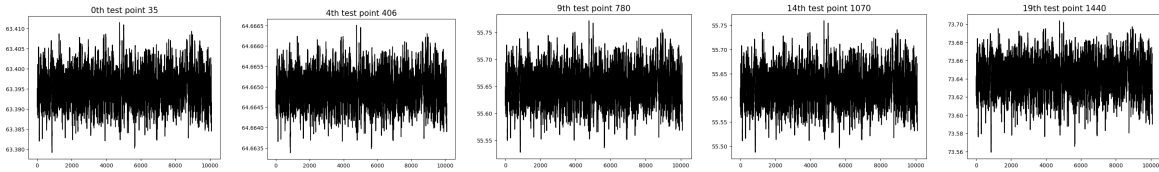


Fig. 4.20 Traces of mean outputs predicted at the test points (every fifth test input), where the predictions are obtained using the stationary SQE kernel.

4.27. Figure 4.22 depicts the traces of mean outputs predicted at the test points with the stationary Matérn kernel given the training data.

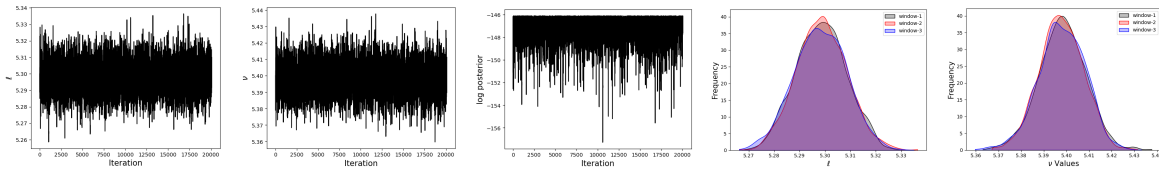


Fig. 4.21 From left: Traces of hyperparameters ℓ and ν learnt with a stationary Matérn kernel, (given the training data on Brent crude oil price data); the trace of the joint log posterior of the hyperparameters given this data; convergence diagnostics on the learning of ℓ and ν respectively.

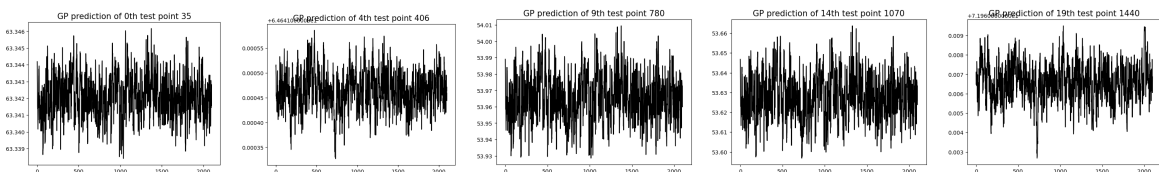


Fig. 4.22 Traces of mean outputs predicted at the test points (every fifth test input), where the predictions are obtained using the stationary Matérn kernel.

4.7.9 Non-stationary kernel from Remes et al. (2017)

Remes et al. (2017) include non-stationarity in their model with a spectral mixture kernel, the number of mixture components in which, can be chosen by hand. However, for each component, three hyperparameters need to be learnt - mixture weight, length scale and frequency, denoted by w, ℓ, μ respectively. In the univariate situation, that is relevant to the training set considered in this chapter, I have used a single component, to maintain the minimal number of hyperparameters to learn. I have used the likelihood suggested by Remes et al. (2017) and inference with MCMC is undertaken. The mean of the 95% HPD credible region on the learnt hyperparameters, w, μ and ℓ in this model are 5.01, 3.8 and 0.29 respectively. Figure 4.23 presents the results of learning these hyperparameters along the convergence diagnostics pertaining to the learning. Predictions at test inputs are shown in the middle panel of the bottom row of the Figure 4.27. Figure 4.24 depicts the traces of mean outputs predicted at the test points with the non-stationary spectral kernel developed by Remes et al. (2017), given the training data.

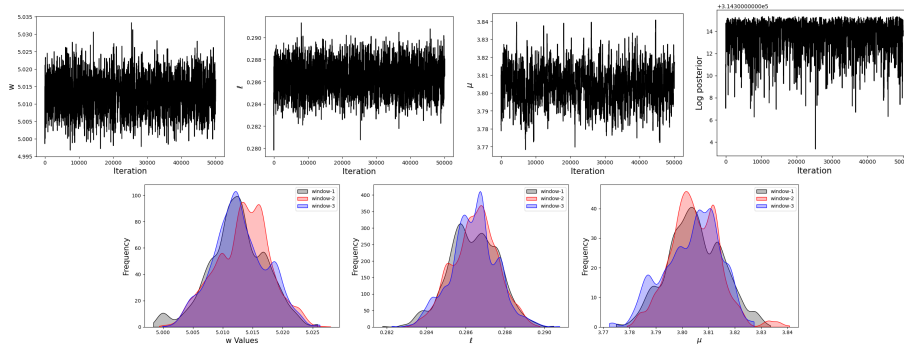


Fig. 4.23 Results of the learning performed with the non-stationary spectral kernel developed by Remes et al. (2017). *Top row, from left:* Traces of the hyperparameters w, ℓ and μ and the logarithm of joint posterior respectively. *Bottom row, from left:* convergence diagnostics of the learning pertaining to the w, ℓ and μ .

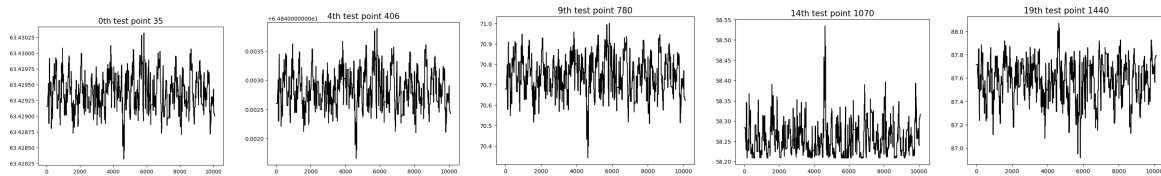


Fig. 4.24 Traces of mean outputs predicted at the test points (every fifth test input), where the predictions are obtained using the non-stationary kernel developed by Remes et al. (2017).

4.7.10 Non-stationary kernel from Paciorek and Schervish (2003)

I have also implemented another non-stationary kernel, that is developed by Paciorek and Schervish (2003). I have learnt the length scale hyperparameter ℓ with a fixed ν hyperparameter of the Matérn kernel used by Paciorek and Schervish (2003). The value of ν is fixed at 5.39, as per the parametric model with Matérn kernel that I have employed earlier, in which I have learnt both ℓ, ν ; (see results in Section 4.7.8. The mean of the 95% HPD credible region on the learnt ℓ is 4.89. Figure 4.25 presents the results of learning with this non-stationary kernel along with the convergence diagnostics, while predictions from this model are displayed in the right panel of the bottom row in the Figure 4.27. Figure 4.26 depicts the traces of mean outputs predicted at the test points with the non-stationary spectral kernel developed by Paciorek and Schervish (2003), given the training data.

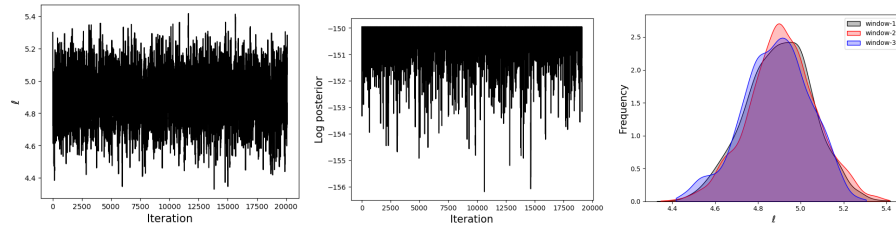


Fig. 4.25 Results of learning performed with the non-stationary kernel developed by Paciorek and Schervish (2003), given the Brent crude oil price data. Trace of the length scale hyperparameter ℓ of the Matérn GP prior - that is assigned a fixed $\nu = 5.3979$ - is displayed in the left panel, while logarithm of the posterior density of this hyperparameter is on the middle and the convergence diagnostics of the learnt ℓ is shown in the right panel.

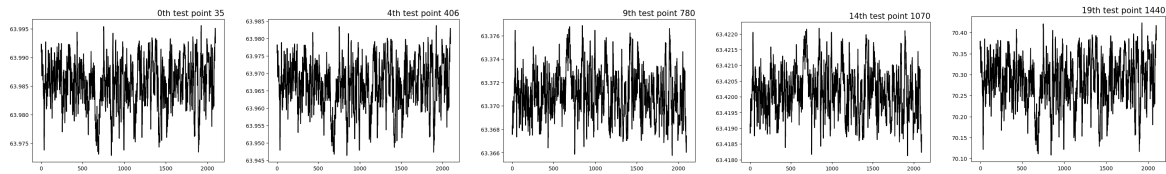


Fig. 4.26 Traces of mean outputs predicted at the test points (every fifth test input), where the predictions are obtained using the non-stationary kernel developed by Paciorek and Schervish (2003).

4.7.11 Comparative performance for predictions made with all models

Table 4.1 presents the performance of the predictions using all the models that are presented. The performance of the predictions is computed in terms of the mean squared error (MSE) computed for both the predicted mean and the uncertainties i.e. 95% HPD of the predictions

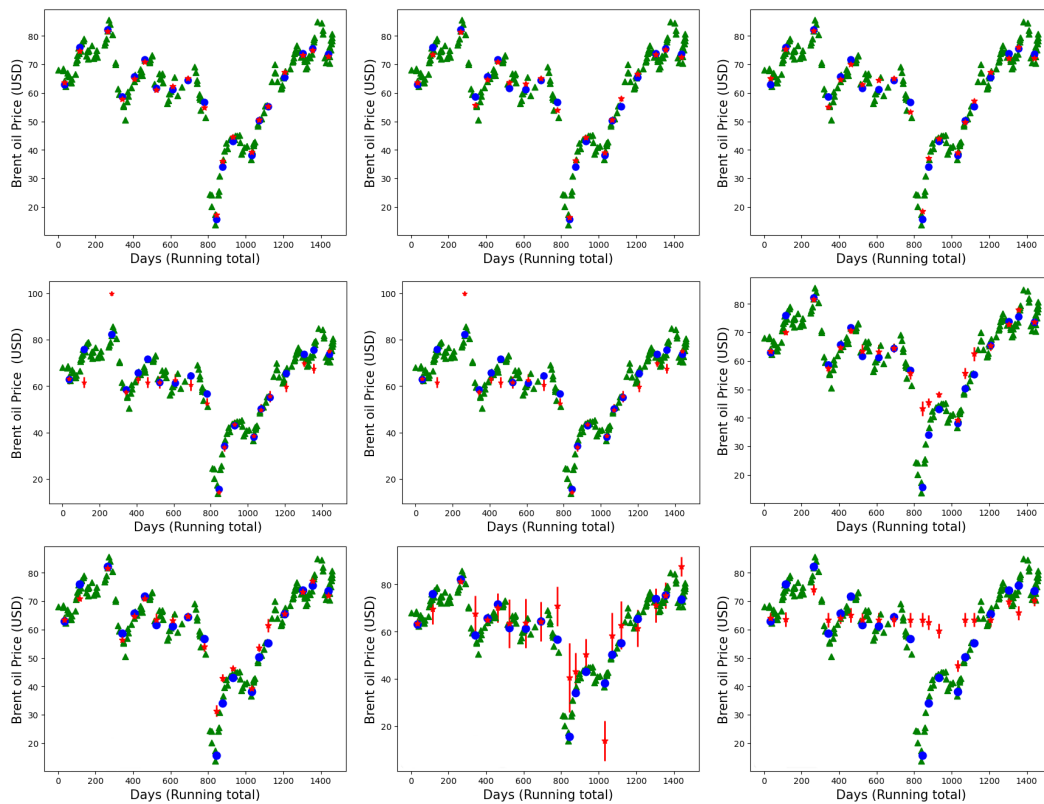


Fig. 4.27 Figure displaying prediction of the oil price output at each of 20 identified test inputs (in red), undertaken following the learning of the temporal variation of the price made using the Brent crude oil price data. 2.5 times the predicted standard deviation of the output is added to either side of the predicted mean, as an error bar (in red). Means of the predicted outputs are presented in red stars; true values of the output are in blue circles; and training points are in green triangles. In the left panel of the top row, predictions made using the fully non-parametric model with the SQE-looking kernel are depicted. The middle panel in this row depicts predictions with the fully non-parametric model using Matérn-looking kernel and the top right panel presents predictions using the non-parametric model with an SQE-looking kernel, given noisy oil price data. In the middle row, the left panel shows the predictions made using the non-stationary version of the fully non-parametric model, (where the non-stationary equivalence is constructed using the non-parametric model with an SQE-looking kernel). The middle panel in this row presents predictions made using the Blended model, while the right panel of the middle row shows predictions made using the single GP layer based model with a stationary SQE kernel. In the bottom row, the left panel shows predictions made using a single GP layer based model with stationary Matérn kernel, while the middle panel depicts predictions made using the non-stationary kernel from Remes et al. (2017) and the right panel shows predictions made using the non-stationary kernel of Paciorek and Schervish (2003).

Model	#Hyp.	MSE(mean)	MSE interval	Time
SQE-looking	2	1.2021	[1.19, 1.21]	2h:4m:15s
Matérn-looking	4	2.509	[2.38, 3.19]	6h:48m:50s
SQE-looking noisy	3	3.7489	[3.63, 4.03]	2h:7m:45s
Non-stationary	2	38.6745	[32.98, 53.02]	5h:39m:21s
Blended	3	38.6731	[32.97, 53.01]	13h:31m:8s
Stationary SQE	1	52.6213	[42.62, 67.47]	1h:48m:2s
Stationary Matérn	2	21.8839	[17.43, 30.03]	9h:35m:17s
Remes et al. (2017)	3	100.694	[102.79, 224.37]	4h:1m:46s
Paciorek and Schervish (2003)	1	206.5252	[189.71, 236.48]	4h:3m:53s

Table 4.1 Table displaying comparative predictive performance and computational times relevant to the different models that I have predicted with. The first column depicts the names of the various models. The second column presents the number of hyperparameter(s) learnt using that model, (typically for each dimension of the input vector). The third column depicts the mean squared error (MSE) computed at the predicted means; the fourth column shows the MSE computed using the predicted 95% HPDs; the fifth and right-most column presents the overall time required to compute the predictions. This includes the time to convergence (achieved by the MCMC chain) and predictions made post convergence.

for all the models against the true values for the test points. The fully non-parametric model outperforms all other models. Also the non-stationary equivalent of the non-parametric model outperforms non-stationary models from the literature. I also report the execution time for computation of the predictions obtained with each of the models to undertake this univariate empirical illustration, along with the number of hyperparameters learnt using each of the models. All the models are run in a laptop having the following configuration: processor is an 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz, 1690 Mhz, 4 Core(s), with 16.0 GB (15.7 GB usable) RAM and 64-bit Windows 10 Enterprise operating system. The implementation of the various model undertaken this chapter, does not employ parallel processing.

It can be seen that the fully non-parametric model performs reasonably faster than other kernels; while the computational time incurred using the Matérn kernel is very high with respect to other models and thus practically infeasible to use when real-time, fast predictions are sought. From the point of view of accuracy of predictions and ease/quickness of computation, the fully non-parametric model is put forward, with the SQE-looking kernel, as adequate in undertaking learning, given a real-world data situation as the one considered in this thesis.

4.7.12 Learning with DNNs

I have also implemented Deep Neural Networks (DNN) with varying architectural details, to predict oil prices at the same test inputs. The purpose of this experiment is to show the sensitivity of the predictions on DNN architecture. The objective is not to identify the architecture that yields the best predictions at the test inputs, but to produce corroboration of the high sensitivity of predictions to architectural markers of DNNs.

Here the DNN architecture is made to vary in the number of hidden layers and in the number of neurons in each hidden layer. these predictions are obtained using TensorFlow, following the code that is available in the official site <https://www.tensorflow.org/tutorials/keras/regression>. I have used the sigmoid activation function for these DNNs along with a learning rate of 0.01. I will now present predictions obtained from three DNN models that are specified as enumerated below.

1. NN-1: DNN with two hidden layers, first and second hidden layer consist of 250 and 184 neurons respectively; 20,000 epochs were noted to be relevant for the loss function to converge, as shown in the left-most panel of Figure 4.28.
2. NN-2: DNN with three hidden layers, each hidden layer has 64 neurons. The loss function has not converged by the 20,000-th epoch (seen in the middle panel of Figure 4.28).
3. NN-3: DNN with three hidden layers, with the first, second and third hidden layer comprising of 64, 184 and 64 neurons respectively. This model does not work.

In Figure 4.28, the panels in the left, middle and right display results obtained using NN-1, NN-2 and NN-3 respectively. The top row depicts predictions at test inputs, and the bottom row show results from a set of epochs - the last 10,000 epochs - to illustrate the convergence of loss. To compute the value of loss at convergence, I have discarded the first 10,000 epochs. From the remaining epochs, 95% confidence interval (CI) and the mean of the loss are computed .

The MSE and 95% CI of the loss, for NN-1, NN-2 and NN-3, respectively are 48.0624 and [3.4144, 4.0294]; 4.5236 and [0.8189, 1.241]; 320.5724 and [10.7665, 10.7679]. It can be noticed that a little perturbation in the architecture causes huge difference in the performance of prediction and as well as the convergence of loss, in this illustration on the learning of a nonlinear function, given a real-world training set comprising about 200 data points. For example, NN-2 and NN-3 have same number of hidden layers and the same number of neurons in the first and third hidden layers, though the numbers of neurons in the middle hidden layer are different. Yet, the MSE from the NN-3 model is about 80 times that from

the NN-2 model, where the loss from the NN-3 model has in fact not converged even after 20,000 epochs. The discrepancies amongst the results from the different DNN models are in spite of usage of the same optimiser with the same activation function, along with same learning rate.

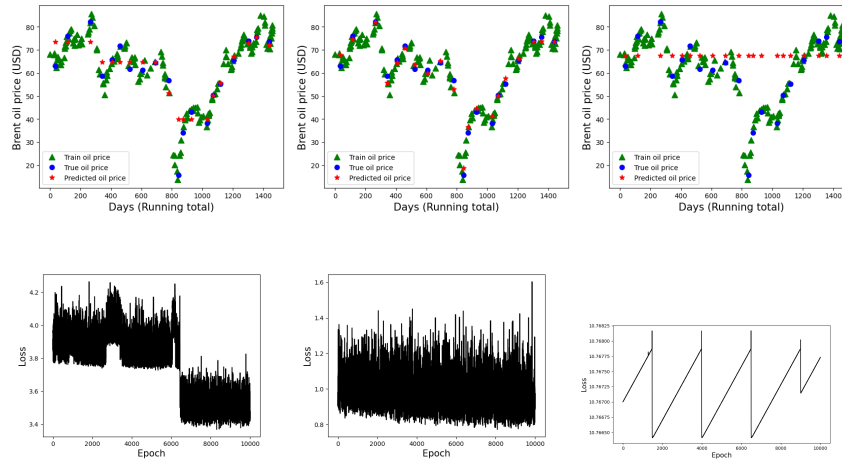


Fig. 4.28 Columns in the left, middle and right depict results obtained using the NN-1, NN-2 and NN-3 models respectively. The top row shows predictions at test inputs; the bottom row presents results from the last 10,000 epochs to illustrate the convergence of loss.

4.8 Conclusion

In this chapter I have presented a fully non-parametric kernel which is equivalently non-stationary and capable of learning the functional relationship between two variables input and output in the presence of inhomogeneities in the correlation structure of the small to medium sized empirical data. The presence of inhomogeneous correlation structure is often a prime phenomena of real-world data sets obtained from diverse application domains.

Many existing non-stationary kernels have used two-layered GPs to address the challenges of learning the sought functional relationship given a non-stationary data. However, the use of two layers in the literature is adopted through assumption of the stationarity of the second layer, which may yield incorrect learning for real-world data. In this thesis, I have proven the sufficiency of two layers by the means of the design of the learning strategy, where the second layer can be stationary without assuming.

One of the most advantageous characteristic of the presented non-parametric kernel is its parsimony i.e. it requires learning of very less number of kernel hyperparameters, while the existing non-stationary kernels in the literature requires learning of large number of hyperparameters, where the number grows for the high-dimensional data. The existing

112A Fully Non-parametric, Non-stationary, Parsimonious, Kernel for Supervised Learning

non-stationary kernels also lack a clear, detailed algorithm for implementation of the kernels, whereas, this thesis included detailed Algorithm for easy implementation.

With the DNN experiments, this chapter also showed that the predictions made with DNNs often can be unreliable given the high level of sensitivity of the DNNs towards its architecture. On the other hand, the presented non-parametric kernel can be used for reliable predictions. The application of this non-parametric kernel with multivariate data and high-dimensional data is included in the later chapters.

Chapter 5

Non-parametric Learning with Multivariate Data and Deep Neural Network Exposition

5.1 Introduction

Machine learning methodologies are constantly evolving and are being used in diverse domains to address many domain-specific challenges. Such domains include healthcare (Alanazi, 2022); finance (Dixon et al., 2020); education (Hilbert et al., 2021); environment (Hino et al., 2018); cybersecurity (Dasgupta et al., 2022); manufacturing (Ademujimi et al., 2017); robotics (Stone and Veloso, 2000). Supervised learning is one of the learning setups which enables to learn the functional relationship between two variables given a training dataset, in order to perform prediction, which is an important requirement within many applications. In these scenarios, an input variable is associated with an output variable, such that the training data has pairs of design inputs and the output realised at this input. The function that links the input and output variables, can be scalar-variate, multivariate or high-dimensional - such as a matrix variate or tensor variate function in general. In any set up, high accuracy of the predictions are desirable while the function learning being reliable and highly generalisable.

Predictions have been undertaken within a wide variety of application domains, nuclear power plants (Rastogi et al., 2017); epidemiology (Lee et al., 2021); industry for mining gold (Yan et al., 2020); energy storage and management systems (Wang et al., 2023); yield of crops (ALI, 2018); industry for the construction of underground structures (Lu et al., 2022) etc. Predictive modelling are also used in accuracy-critical domains such as information

security (Smith et al., 2011); medicine (Begoli et al., 2019) where the learning is required to be robust, interpretable, and uncertainty-included, instead of being point estimates. So the learning must consider the variability or noise present in the training dataset (Lu et al., 2022; Rastogi et al., 2017; Roy and Chakrabarty, 2023; Shen et al., 2022).

When the sought function $f(\cdot)$ (depicted in Chapter 4) is being modelled as a random realisation of a stochastic process or with a probabilistic density function to capture the relation between the input and output variable, given the training data, often these underlying process, density function are considered as non-stationary. So that, the learnt sought function $f(\cdot)$ is non-stationary. A function is stationary, if it is shift-invariant over the input space. Then quantifying non-stationarity is important and it is emerging as an active area of research. Das and Nason (2016) developed a method to measure the degree of non-stationarity, particularly for time series data. Granero-Belinchon et al. (2021) has proposed a method to quantify the non-stationarity of a stochastic process using information theoretic techniques. Noack et al. (2024b) presented a measure of non-stationarity which invokes stationary Gaussian Process (GP) on the subsets (randomly chosen) of the data and estimates the length-scale and the signal variances of the stationary GP using maximum likelihood estimation. Thus, after the end of the iterative inference procedure, a distribution of the length-scales and variances is available to indicate the presence of non-stationarity in the data. However, the covariance function used to model the correlation structure of a stationary GP, is again stationary, having a parametric form - this implies that a chosen form is imposed to the correlation structure of the GP which captures the correlation structure of the data. Hence, assigning a pre-chosen form to the correlation structure of the GP by means of its covariance kernel function, the GP fails to capture the true correlation structure of the empirical data.

Therefore, the non-stationarity or the inhomogeneity also needs to be defined in terms of the empirical data that is at hand. Only then can correct predictive models be built. If the inhomogeneity present in the correlation structure of the data can be defined, then we can build enhanced predictive models which yields reliable predictions given such data. In this chapter I will first present such quantification for the inhomogeneity in the correlation structure of the data, using the methodology presented in Chakrabarty (2025). I will then present the multivariate non-parametric model that invokes an SQE-looking kernel where the learning of the multivariate function is done using a multivariate data that bears inhomogeneity in its correlation structure. Afterwards, I will showcase the impact of inhomogeneities present in the correlation structure of the data on the quality and reliability of predictions accomplished with the popular predictive models from the literature - of both machine learning and probabilistic machine learning.

I will also show that the more inhomogeneous a dataset is, prediction is harder and subsequently the performance of the prediction is poorer with the usage of the same model. This work performs experiments using Deep Neural Networks (DNNs) and a comparative Single layer GP (Rasmussen and Williams, 2006) based learning with a stationary kernel, to achieve predictions on two univariate datasets with varying inhomogeneity in its correlation structure. The GP based learning model uses Markov Chain Monte Carlo (MCMC) (Neal, 1997) inferential framework which enables uncertainty included learning while avoiding finding the "best" model in a trial and error based, brute force search, which may not yield all the "good" models for new test data sets while performing predictions. Neither will the uncertainty in the learning be organically and robustly learnt. Though the GP based learning models yield reliable predictions, still they are also affected by the inhomogeneity present in the correlation structure of the data. The results will show that the performance of the predictions is higher with the dataset having lower inhomogeneity in its correlation structure.

The experiments also depict that the predictive performance of the DNNs are very sensitive to the choice of architecture and the best chosen architecture may not perform well for new test datasets indicating a concern of reliability of the predictions especially when applied to the datasets pertaining to the high-stake, life-threatening areas such as predicting disease risk scores for cancer patients (Chakrabarty et al., 2023). An attempt is made, based on the value of the loss function when the optimiser achieves convergence, to choose the best DNN model and architecture, instead of relying on the trial and error. While it is standard in the literature of DNNs to choose the best DNN architecture or the model using trial and error method that monitors prediction accuracy (Sze et al., 2017; Xu et al., 2021), this work shows that the best DNN architecture (model) chosen using prediction accuracy monitoring, does not prove to be the best DNN that yields best prediction for new test datasets - this is also illustrated using both univariate and multivariate data.

The chapter is organised as follows. The notion of inhomogeneity in correlation structure of data, as formulated by Chakrabarty (2025) is presented in Section 5.2. The application of non-parametric model (presented in Chapter 4) using SQE-looking kernel, with inhomogeneously correlated, real-world multivariate data is given in the Section 5.3. Learning with DNNs are included in the Section 5.4. Finally this chapter is concluded.

5.2 Inhomogeneity in correlation structure of data

Let the training data D comprise a finite (n) number of discrete data points which are as follows, $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n) \forall i = 1, 2, \dots, n$ where x_i is the value of $X \in \mathcal{X}$ which is the input that is associated with the output is $Y \in \mathcal{Y}$, whose realisation is $y_i, \forall i = 1, 2, \dots, n$.

The predictive model is interested to learn the function $f(\cdot)$ where $Y = f(X)$ where I model $f(\cdot)$ as a random realisation from a Gaussian Process (GP), where the choice of a GP is already motivated in Chapter 4. So, the correlation structure of D will have to be mimicked as the correlation structure of the GP that $f(\cdot)$ is treated as sampled from. In other words, a random sample function $\hat{f}(\cdot)$ drawn from the GP is fit to the data D . Here, the correlation structure of the random sample function $\hat{f}(\cdot)$ refers to the correlation function of the underlying stochastic process, which is a GP. The learning aims to learn this correlation function of the underlying GP, conditioned on the data D which bears inhomogeneities in its correlation structure.

As per Chakrabarty (2025), a benchmark sample function $\hat{f}(\cdot)$ of the underlying GP is defined as driven by the data D , to represent the inhomogeneities of the correlation structure of D . A piecewise-linear function - also called the “trivial function” - is chosen to be such benchmark sample function. This function joins every successive pair of points in the training data D with a straight line. Thus, such a sample function is completely data driven to correctly model the correlation structure of the data.

Using this trivial function, Chakrabarty (2025) defined inhomogeneity in correlation structure of the data as follows. First a constant, referred to as “L value” - denoted L_i - which is similar to a Lipschitz constant. L_i is defined such that,

$$d_Y(\hat{f}(x_i), \hat{f}(x_{i+1})) \leq L_i d_X(x_i, x_{i+1}), \forall \hat{f}(x_i), \hat{f}(x_{i+1}) \in Y$$

where $d_Y(\cdot, \cdot)$, $d_X(\cdot, \cdot)$ are the metric for measuring distance in the output and input space respectively. Then the data driven L_i is given by

$$L_i = \frac{d_Y(y_i, y_{i+1})}{d_X(x_i, x_{i+1})}, \quad (5.2.1)$$

Once values of $L_1, L_2, \dots, L_{i-1}$ are all available, a width of δ_i is identified around each L_i to define “similarly-Lipschitz” partitions of the input space, given the data D . Chakrabarty (2025) defines the partitions $[x_i, x_{i+1})$ and $[x_j, x_{j+1})$ of input space X as “similarly-Lipschitz” for chosen $\delta_i, \delta_j \geq 0$ if there is a non-null overlap between the interval $[L_i - \delta_i, L_i + \delta_i]$ and $[L_j - \delta_j, L_j + \delta_j] \forall i, j = 1, 2, \dots, n - 1$.

I have set $\delta_i = \delta_j = \delta$ for all i and j . Here, δ_i is chosen to be the $\varepsilon\%$ of L_i . For example, setting $\varepsilon = 10\%$ to 20% is a decent choice. In this thesis, I have used $\varepsilon = 20\%$. For $d_Y(y_i, y_{i+1})$, I have used absolute difference as $|y_i - y_{i+1}|$.

Then the existence of the inhomogeneities in the correlation structure of D is defined as follows:

If there is at least one interval $[L_i - \delta, L_i + \delta]$, that does not have an overlap with any of the

other intervals $[L_j - \delta, L_j + \delta]$, where $j = 1, \dots, i-1, i+1, \dots, n-1$, then inhomogeneities exists in the correlation structure of D .

Thus, according to this definition, the j -th partition of the input space is then not similarly-Lipschitz with any other partition.

The motivation behind this conceptualisation by Chakrabarty (2025) is that, if inhomogeneities exists in the correlation structure of D in references to the benchmark function $\hat{f}(\cdot)$, then inhomogeneities will also exist with respect to the generic sample functions of the underlying GP because this benchmark function $\hat{f}(\cdot)$ is also the minimally-rough sample function.

So next Chakrabarty (2025) provides a parametrisation η of inhomogeneities in a dataset. For a given dataset D , in order to compute η , we first identify the total number of partition(s) p within the input space of D for which the Lipschitz nature of the minimally-rough function $\hat{f}(\cdot)$ is not similar with other partitions. Then η is defined as,

$$\eta = \frac{p}{n} \quad (5.2.2)$$

where n is the size of the dataset D . So, the design of η ensures that it is unaffected by the size of the data by normalising with n .

5.3 Application of non-parametric Learning with Multivariate Data

Now I will consider the application of non-parametric kernel model developed in Chapter 4 in multivariate data situation for supervised learning of a multivariate, scalar-valued function. So the input variable of the function - that we want to learn - is a vector, and the output is a scalar, i.e. $\mathbf{D}_{train} = \{(\mathbf{x}_i, y_i)\}_{i=1}^M$, where $\mathbf{X} = \mathbf{x}_i$ with $\mathbf{X} \in \mathcal{X} \subset \mathbb{R}^d$, Also, $Y = y_i$ with $Y \in \mathcal{Y} \subset \mathbb{R}$. I would like to learn the functional relationship as follows,

$$Y = f(\mathbf{X}), \text{ where } f : \mathcal{X} \subset \mathbb{R}^d \longrightarrow \mathcal{Y} \subset \mathbb{R}.$$

In this application, I have used SQE-looking non-parametric kernel. Hence, there will be d number of length scales $\ell_1, \dots, \ell_d$ for each input dimension in input space $\mathcal{X}$. The length scale $\ell_i \forall i = 1, \dots, d$ will be modelled using a random function $g_i(t)$ s.t.

$$\ell_i \sim g_i(t), \forall i = 1, \dots, d,$$

where $g_i(t)$ is modelled as the random realisation of an inner stationary GP $\mathcal{GP}_i(t)$, $\forall i = 1, \dots, d$. Hence, the inner layer will constitute of the GPs $\mathcal{GP}_1, \dots, \mathcal{GP}_d$. We recall that the GP pertaining to the inner layer is proven to be stationary. The covariance function of each of the inner layer GPs, namely, $\mathcal{GP}_1, \dots, \mathcal{GP}_d$ is modelled using the SQE kernel function as mentioned in Chapter 4 having length scales $\delta_1, \dots, \delta_d$ respectively. This model learns a total $2d$ number of hyperparameters to achieve fully non-parametric kernel with a d -dimensional multivariate data that bears inhomogeneities in its correlation structure.

The likelihood in this model is again Multivariate Normal and depicted as follows.

$$\mathcal{L}(\ell|D_{train}) = \frac{1}{\sqrt{(2\pi)^M |\Sigma|}} \exp \left(-\frac{1}{2} (\mathbf{y} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{y} - \boldsymbol{\mu}) \right),$$

where $\mathbf{y} = (y_1, \dots, y_M)^T$, $M \times M$ -dimensional covariance matrix $\Sigma^{(M \times M)} = [\text{cov}(Y_i, Y_j)]$. $[\text{cov}(Y_i, Y_j)]$ is modelled as follows:

$$[\text{cov}(Y_i, Y_j)] = [K(\mathbf{x}_i, \mathbf{x}_j)], \forall i, j = 1, \dots, M$$

$K(\cdot, \cdot)$ is a kernel function. In this application, the newly developed SQE-looking non-parametric kernel model presented in Chapter 4 is used as the kernel function $K(\cdot, \cdot)$ for the outer layer GP $\mathcal{GP}_{outer}$ with,

$$K(\mathbf{x}_i, \mathbf{x}_j) := \exp \left(-(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{Q} (\mathbf{x}_i - \mathbf{x}_j) \right),$$

where $\mathbf{Q}$ is the $d \times d$ -dimensional diagonal matrix, having the k -th diagonal element as $1/2\ell_k^2$, $\forall k = 1, \dots, d$. The modelling strategy of ℓ_k in the inner GP remain same as depicted in Chapter 4 with hyperparameter δ_k , $\forall k = 1, \dots, d$.

5.3.1 Application with data

I will illustrate the parsimonious, non-parametric learning with multivariate dataset obtained from Dua and Graff (2017); Salam and El Hibaoui (2018). This is a power consumption data measured in Tetouan city, which is located in north Morocco. The data points are the observations of five input meteorological parameters along with corresponding value of the energy consumption in the three zones of the Tetouan city. The observations are recorded every ten minutes within the period of 1st of January, 2017, to the 31st of December, 2017. The five input meteorological parameters are temperature, humidity, wind speed, general diffuse flows and diffuse flows. Among these five input variables, general diffuse flows and diffuse flows are highly inter-correlated and appear as linear transformations of each other,

hence, I have discarded the input variable general diffuse flows and considered the rest four input variables. I have also added the power consumption values pertaining to three zones to yield a scalar variate “integrated power consumption” as the output variable. Therefore, in this multivariate setup, the training data $\mathbf{D}_{train}$ consists of -

- 4-dimensional input vector variable that consists of the meteorological parameters - temperature, humidity, wind speeds and diffuse flows with length scales pertaining to these input dimensional are $\ell_T, \ell_H, \ell_W, \ell_D$ respectively in the outer layer which are modelled with the inner GPs $\mathcal{GP}_T, \mathcal{GP}_H, \mathcal{GP}_W, \mathcal{GP}_D$. The covariance function of these inner GPs are modelled with stationary SQE kernel with hyperparameters $\delta_T, \delta_H, \delta_W, \delta_D$.
- Scalar variate output variable that presents the integrated power consumption of the Tetouan city.

I construct my training data as a subset from the whole Tetouan City data, I have resampled and considered 203 data points as my training data and 50 data points as test set. In this construction of training and test datasets, no date and time of the observed data points are repeated. Figure 5.1 presents the heat map and the 3D bar plot of the 4-dimensional input of the training data. Figure 5.2 presents the output training data where the integrated power consumption is plotted against the index of the design point in the training set.

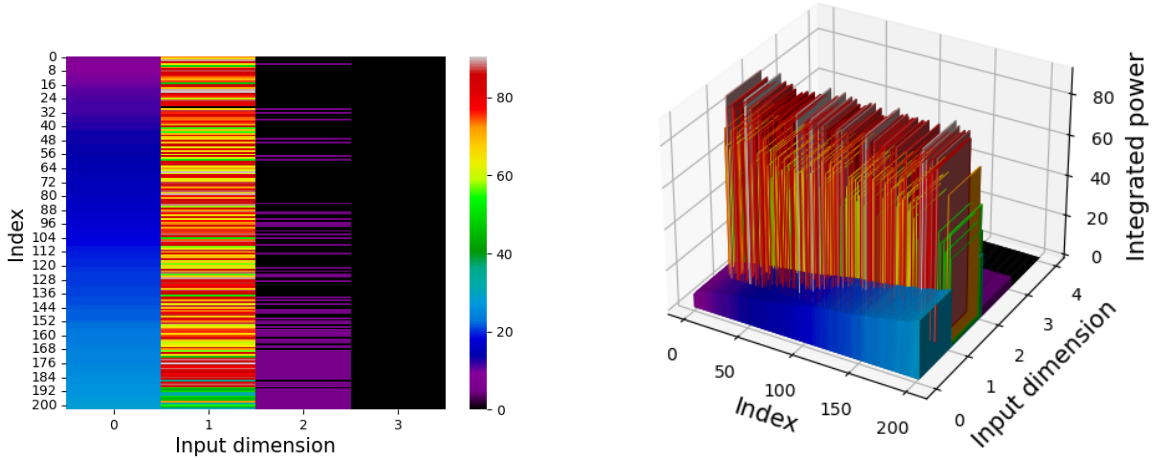


Fig. 5.1 The left most panel shows the heat map and the right panel shows the 3D bar plot of the design training input of the multivariate Tetouan city data.

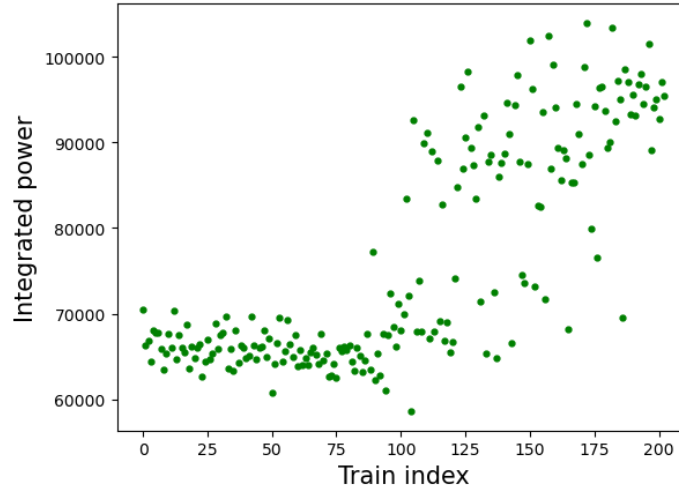


Fig. 5.2 Output training data plotted against index of design training points of the multivariate Tetouan city data.

5.3.2 Results of multivariate learning

In this application, I am learning four length scale hyperparameters $\ell_T, \ell_H, \ell_W, \ell_D$ pertaining to the outer GP that uses the non-parametric model with an SQE-looking kernel. The length scale hyperparameters $\ell_T, \ell_H, \ell_W, \ell_D$ are again modelled with the four stationary inner GPs $\mathcal{GP}_T, \mathcal{GP}_H, \mathcal{GP}_W, \mathcal{GP}_D$ having kernel hyperparameters $\delta_T, \delta_H, \delta_W, \delta_D$ respectively that model the covariance function of the inner GPs. The 95% HPD credible region on the learnt hyperparameters are shown in Table 5.1. This table includes the 95% HPD credible regions learnt on the hyperparameters $\ell_T, \ell_H, \ell_W, \ell_D$ in the first block of the chain; 95% HPD credible regions learnt on the hyperparameters $\delta_T, \delta_H, \delta_W, \delta_D$ in the second block of the chain; and the predicted mean values of the hyperparameters $\ell_T, \ell_H, \ell_W, \ell_D$ obtained at the end of the second block.

Figure 5.3 displays traces of learnt $\ell_T, \ell_H, \ell_W, \ell_D$ in the first block; traces for the predicted $\ell_T, \ell_H, \ell_W, \ell_D$ at the end of second block; and the traces of $\delta_T, \delta_H, \delta_W, \delta_D$. Figure 5.4 presents traces of the logarithm of the posteriors given the training data in the outer layer as well as given the dynamically-varying lookback data. This MCMC chain is run for the 90,000 iterations with the size of the lookback window N_{LB} set to 17. The execution time for this chain is 42 minutes and 35 seconds. Normal proposal densities are used to propose values of the hyperparameters during inference. Normal priors are used on all unknown hyperparameters. Variances of these Normal priors, as well as the Normal proposals are chosen experimentally. The variance of the Normal proposal densities; prior means; prior

	ℓ_T	ℓ_H	ℓ_W	ℓ_D	δ_T	δ_H	δ_W	δ_D
95% HPD	[0.68,0.84]	[8.49,8.51]	[0.01,0.02]	[0.05,0.07]	[0.16,0.24]	[0.42,0.87]	[1,1.18]	[0.3,0.5]
Mean	0.7584	8.4998	0.0128	0.0564	0.1986	0.6474	1.0923	0.3978
Predicted	0.7564	8.4997	0.0135	0.0561	-	-	-	-

Table 5.1 Learnt values of the unknown hyperparameters. For the hyperparameters ℓ_T , ℓ_H , ℓ_W , ℓ_D , “95% HPD” denotes the 95% HPD credible regions learnt in the first block of the chain. For the hyperparameters δ_T , δ_H , δ_W , δ_D , the 95% HPD credible regions are learnt in the second block. “Mean” denotes the mean of the 95% HPD credible regions of the learnt hyperparameters and “Predicted” depicts the predicted mean value of hyperparameters ℓ_T , ℓ_H , ℓ_W , ℓ_D at the end of second block.

variances; and seeds used for the learning of hyperparameters in this MCMC chain are presented in the Table 5.2.

Figure 5.5 presents the convergence diagnostic of the learning of hyperparameters, where values of the hyperparameter are sampled over three distinct, equally-wide (covering 10,000 consecutive iterations) windows, starting at the 60,000-th; 70,000-th; and 80,000-th iterations, during the MCMC chain. Histograms of sampled values of parameters are depicted as coloured in blue, red and black. Figure 5.6 presents heat maps of the correlation matrices used for the learning of the non-parametric model that uses an SQE-looking kernel given the multivariate training data.

	ℓ_T	ℓ_H	ℓ_W	ℓ_D	δ_T	δ_H	δ_W	δ_D
σ_{prop}	0.1^2	0.01^2	0.006^2	0.008^2	0.002^2	0.01^2	0.01^2	0.01^2
μ_{prior}	0.85	8.5	0.03	0.05	0.2	0.6	1.4	0.4
σ_{prior}	0.1^2	0.01^2	0.1^2	0.1^2	0.02^2	0.1^2	0.1^2	0.05^2
Seed	0.1	8.45	0.02	0.015	0.2	0.2	0.2	0.3

Table 5.2 Values of the parameters of the MCMC chain run given the multivariate Tetouan city data. Here σ_{prop} is the variance of the Normal proposal densities for the hyperparameters, μ_{prior} and σ_{prior} are the mean and variance of the Normal priors invoked for the hyperparameters respectively.

Predictive results of the non-parametric multivariate model that uses SQE-looking kernel given the multivariate data has been generated at four distinct test sets referred as “Test-1”, “Test-2”, “Test-3” and “Test-4”. Figure 5.7 depicts the four test datasets where the true values of the test points are plotted against the temperature in blue circle and the output training data points against the input temperature training points are shown in green dots. The top row of the Figure 5.7 displays the true values of the test points against the temperature for

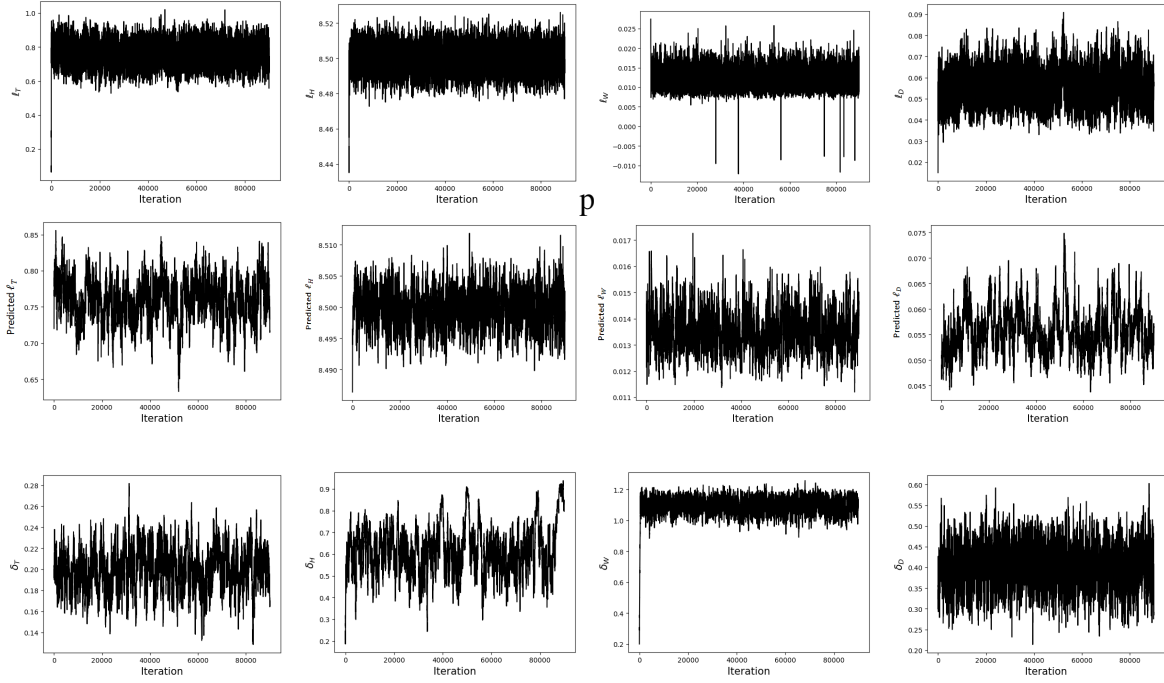


Fig. 5.3 Results of learning $\ell_T, \ell_H, \ell_W, \ell_D$ using the non-parametric model with SQE-looking kernel, given multivariate data. *Top row, from the left:* traces of $\ell_T, \ell_H, \ell_W, \ell_D$ updated in the first block of the Metropolis-within-Gibbs chain undertaken for inference. *Middle row, from the left:* traces of $\ell_T, \ell_H, \ell_W, \ell_D$ as predicted from the $\mathcal{GP}_T, \mathcal{GP}_H, \mathcal{GP}_W, \mathcal{GP}_D$ respectively in the inner layer at the end of second block. *Bottom row, from the left:* traces of $\delta_T, \delta_H, \delta_W, \delta_D$ updated in the second block.

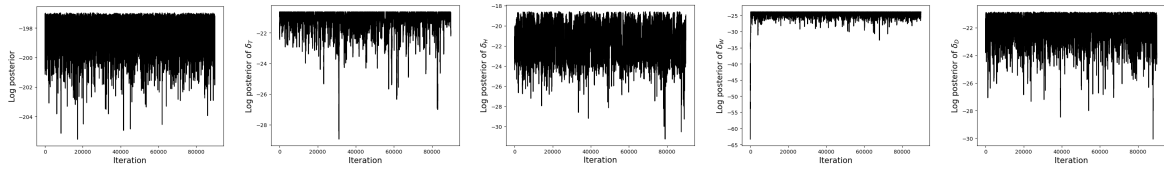


Fig. 5.4 *From left:* trace of logarithm of the joint posterior of $\ell_T, \ell_H, \ell_W, \ell_D$ given training data with the non-parametric model using SQE-looking kernel; traces of logarithm of posterior of $\delta_T, \delta_H, \delta_W, \delta_D$ respectively, given relevant (dynamically-varying) lookback data.

all the test sets and the bottom row shows the same true values of the test points against the temperature, in addition, shows the trivial function (a piecewise-linear-function fitted to the output training data) plotted against the input training temperature data. These test sets have varying levels of difficulty for prediction.

The various levels of difficulties for predicting has been by considering how far the true value at a test point is located from the available training data and similarly, when the true value at a test point lies outside of the trivial function, then the prediction at those test points

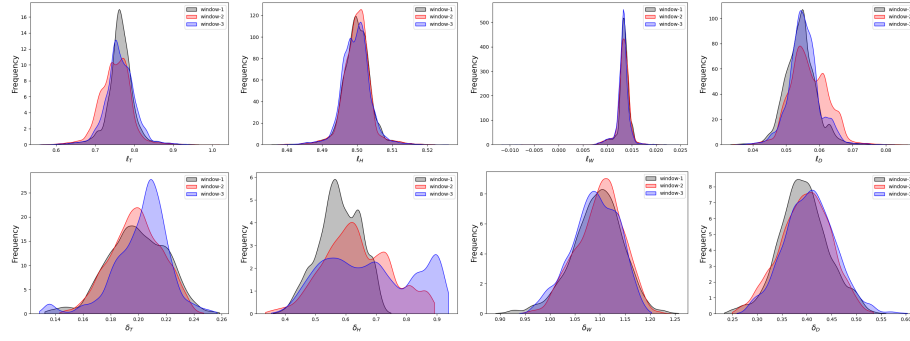


Fig. 5.5 Convergence diagnostic of the learning of the hyperparameters using the non-parametric model with SQE-looking kernel given the multivariate training data where the values of the hyperparameter are sampled over three distinct, equally-wide (covering 10000 consecutive iterations starting at the 60000-th; 70000-th; 80000-th iterations for the three windows respectively), windows during the MCMC chain. Histograms of samples taken in the three windows are coloured in blue, red and black. From extreme left in the top row, the panels pertain to hyperparameters: ℓ_T , ℓ_H , ℓ_W , ℓ_D . From extreme left in the bottom row, the panels pertain to hyperparameters: δ_T , δ_H , δ_W , δ_D .

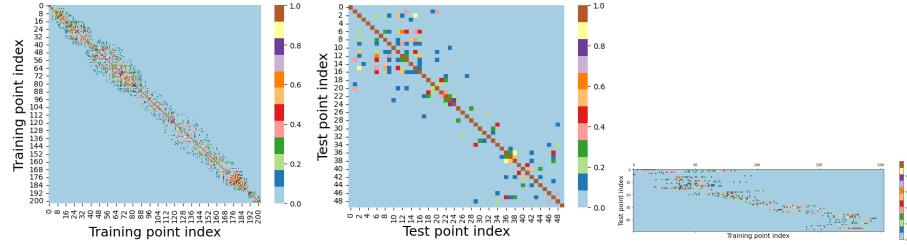


Fig. 5.6 Correlation matrices of the non-parametric model that invokes an SQE-looking kernel. The left most plot presents the correlation matrix computed between outputs realised at pairs of design points in the training dataset, while the middle plot shows the correlation matrix computed between outputs realised at input pairs that each comprise test inputs. The right-most panel depicts the correlation matrix between an output realised at a design point, and another that is realised at a test input.

are hard. For example, in the test set “Test-2”, many of the true values at the test points lie outside of the trivial function with respect to the test points present in the “Test-1”, making it harder to predict at those test points in “Test-2” than to predict with “Test-1”.

Whereas, “Test-3” contains many test points at temperature around 20 to 22 where, there is a significant shift in the output of the training data, indicating a region where one-to-many relation exist. In that region, similar temperatures may have two different kind of output values having large difference in the output values. In such region, training data is sparse and the “Test-3” set contains some of the test points in such region, posing significant challenge to the model for performing predictions at those test points.

	Test-1	Test-2	Test-3	Test-4
RMSE	6547.5	13473.2	12116.9	12725.6
Uncertainty	[6367.8, 6835.9]	[13431.4, 13573.7]	[12093.1, 12196.8]	[12561.5, 12958.5]

Table 5.3 Predictive performance in terms of RMSE where the predictions are obtained using the non-parametric multivariate model with SQE-looking kernel, given the multivariate training data on the four test sets “Test-1”, “Test-2”, “Test-3”, “Test-4”. Top row depicts the root mean square error (RMSE) computed at the mean predicted value, while the bottom row shows the RMSE computed using the predicted 95% HPDs.

In “Test-4”, there are many test points with the true values lie in regions where training data is either absent or very sparse. Predictions are particularly hard to make at such test points.

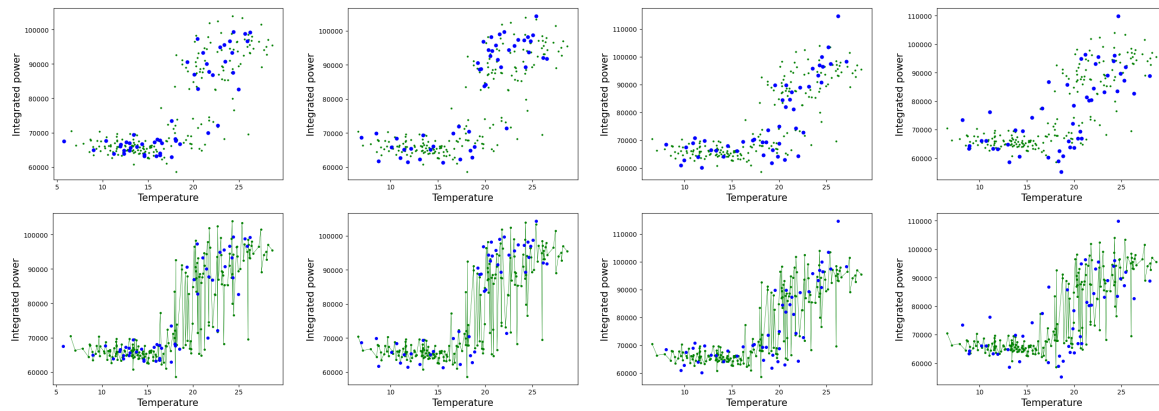


Fig. 5.7 Four test sets - *Top row, from left:* test sets “Test-1”, “Test-2”, “Test-3”, “Test-4” where true values of the test points are plotted against the temperature in blue circle and the training data points against the temperature is shown in green dots. *Bottom row, from left:* test sets “Test-1”, “Test-2”, “Test-3”, “Test-4”, similar to top row, however, in addition it shows the trivial function (a piecewise-linear-function fitted to the output training data) in green solid line.

Figure 5.8 display predictions obtained at the test set “Test-1” while Figure 5.9 shows the predictions obtained at the test sets “Test-2”, “Test-3”, “Test-4” using the non-parametric multivariate model with SQE-looking kernel, given the multivariate training data. In these plots, the true values are shown in blue circles while the predictive means are shown in red stars with the computed uncertainty in terms of the variance of the prediction are plotted in both side of the predicted mean in red straight lines. Table 5.3 presents the root mean square error (RMSE) computed at the predicted means and RMSE computed using the predicted 95% HPDs.

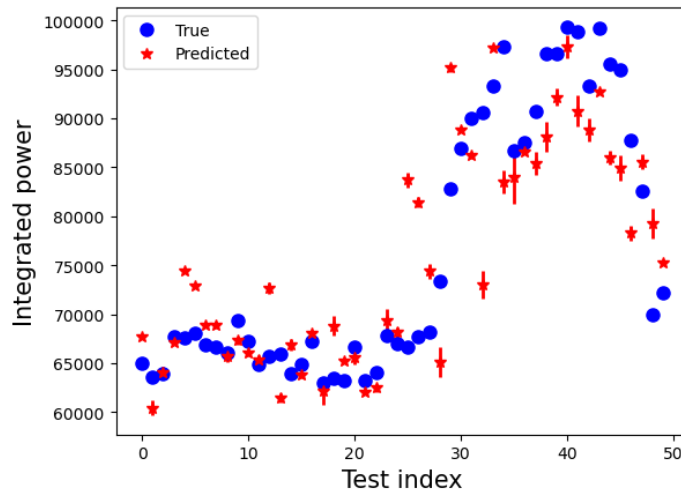


Fig. 5.8 Prediction of integrated power consumption obtained at test set “Test-1” using the non-parametric multivariate model with SQE-looking kernel, given the multivariate training data. Predictions are plotted against the index of the test data points. The true values are shown in blue circles while the predictive means are shown in red stars with the computed uncertainty in terms of the variance of the prediction are plotted in both side of the predicted mean in red straight lines.

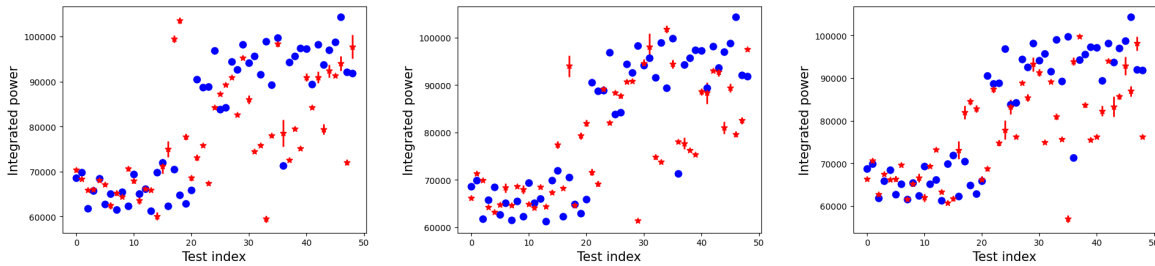


Fig. 5.9 Prediction of integrated power consumption obtained at test sets (from left to right), “Test-2”, “Test-3”, “Test-4”, using the non-parametric multivariate model with SQE-looking kernel, given the multivariate training data. Predictions are plotted against the index of the test data points. The true values are shown in blue circles while the predictive means are shown in red stars with the computed uncertainty given by the variance of the prediction is plotted on both sides of the predicted mean in red straight lines.

Figure 5.10 plots the predictions obtained at the test sets “Test-1”, “Test-2”, “Test-3”, “Test-4” against each of the four input dimensions. The top row presents the predictive plots pertaining to the test set “Test-1”; similarly, the second and third row from the top shows predictive results for “Test-2”, “Test-3”; whereas the bottom row depicts the predictive plots for the “Test-4”. These plots also include the corresponding training data along the relevant input dimension.

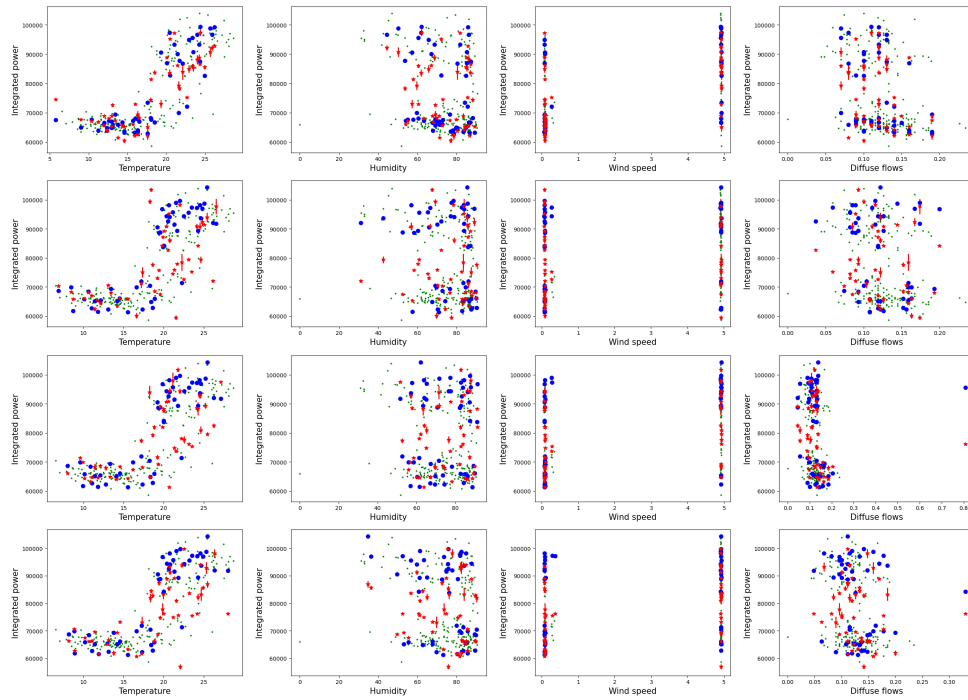


Fig. 5.10 Predictions obtained at the test sets “Test-1”, “Test-2”, “Test-3”, “Test-4” plotted against all the four input dimensions - temperature, humidity, wind power and diffuse flows. Rows from top present: predictive results at the test set “Test-1”, “Test-2”, “Test-3” and “Test-4” respectively. Columns from the left display predictions against: temperature, humidity, wind power and diffuse flows respectively. The true values are shown in blue circles while the predictive means are shown in red stars with the uncertainty of predictions, computed as the variance of the predicted output variable, is plotted on both sides of the predicted mean in red straight lines. Outputs at design points are plotted in green dots.

Figures 5.11, 5.12, 5.13, 5.14 display histograms of outputs predicted at some of the test points, where the predictions are made at the test set “Test-1”, “Test-2”, “Test-3” and “Test-4” respectively.

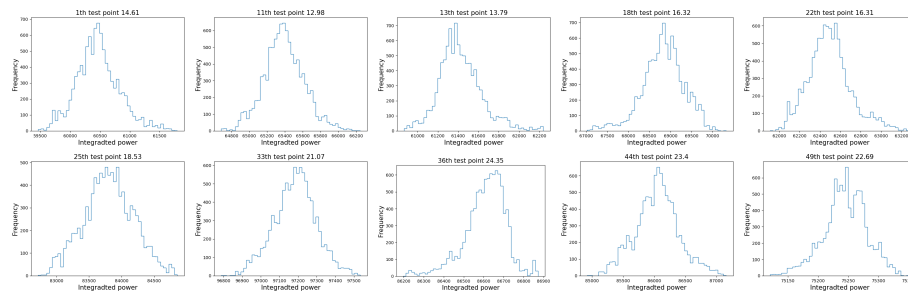


Fig. 5.11 Histograms of predictions made at a few test points, from test set “Test-1” with SQE-looking non-parametric kernel given the Tetouan city multivariate data.

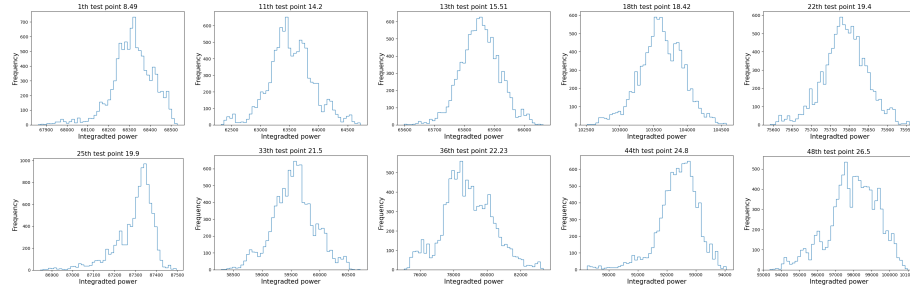


Fig. 5.12 Histograms of predictions made at a few of the test points from test set “Test-2” with SQE-looking non-parametric kernel given the Tetouan city multivariate data.

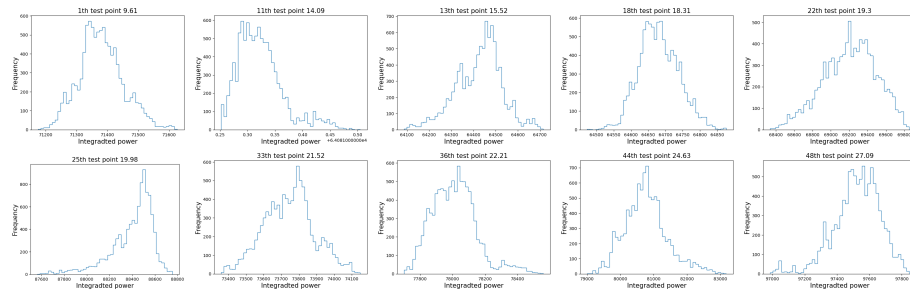


Fig. 5.13 Similar to Figure 5.12 except here, few of the test points from test set “Test-3” are used.

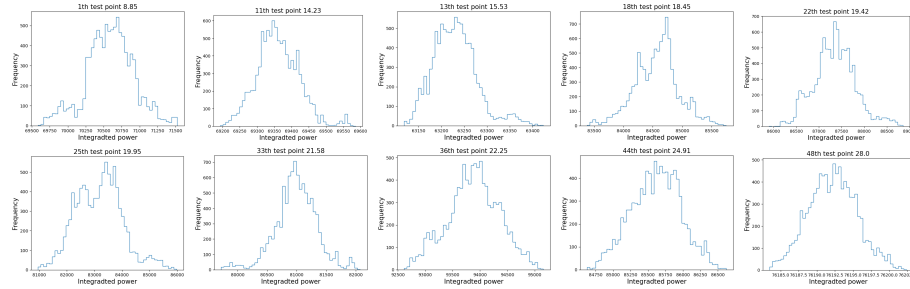


Fig. 5.14 Similar to Figure 5.12 except here, few of the test points from test set “Test-4” are used.

5.4 Learning with Deep Neural Network models

The fundamental unit of a Neural network - which is a supervised learning technique - is an artificial neuron that is designed using the inspiration from the functioning of the human brain as discussed in neuroscience. Although the literature of neuroscience is constantly upgrading our understanding about how the brain works, the network built with artificial neurons has emerged as a popular machine learning model.

5.4.1 DNN model

All neural network (NN) models consists of an input layer that accepts the input values of the training data, and an output layer. The deep neural network (DNN) model involves adding hidden layers between the input and the output layer. Computation of an artificial neuron includes a weighted sum of n number of inputs $x_1, x_2, \dots, x_n$. Then this weighted sum is non-linearly transformed using an activation function $f_n(\cdot)$ to yield the activated output $y_1, y_2, \dots, y_m$ as,

$$y_i = f_n \left(\sum_{j=1}^n w_{ij} x_j + b \right), i = 1, 2, \dots, m$$

where, w_{ij} are weights and b is a bias term.

The complexity of DNN model architecture depends on the number of hidden layers and number of neurons in each layer (Koutsoukas et al., 2017). These are some of the most important hyperparameters of a DNN, and these cannot be learnt or estimated from data, but need to be set before the learning commences. Other essential hyperparameters include a form of the invoked activation function - from a repertoire of such in-built functions that includes “softmax”, “rectified linear unit (ReLU)”, “sigmoid”, “tanh”, etc. Other architectural details include the type of optimizer: stochastic gradient descent (SGD); adaptive moment estimation (Adam); root mean square propagation (RMSprop); etc. (Kuhn et al., 2013; Yang and Shami, 2020).

The performance of the DNN model crucially depends on these hyperparameters. Thence hyperparameter tuning is an important task in learning with DNN models. Manually choosing values of these hyperparameters by trial and error is a standard way in the literature (Xu et al., 2021). Therefore, automated hyperparameter optimisation has emerged as an effort to reduce human effort (Feurer and Hutter, 2019; Luo, 2016). However, it has not gained popularity due to the requirement of heavy computational resources. Some works have considered fixing the number of hidden layer(s) with fixed number of epochs and learning rate to get rid of the tedious task of exploring various architectures and many of the model parameters to achieve the data analytical task with the DNN model, however, the status of the convergence of the optimisation in this case is not known and hence the correctness of learning is questionable (Chinnathambi et al., 2018). Importantly Montavon et al. (2012) shows that the DNN models are very sensitive to the values of its hyperparameters. Hence, I have experimented with DNN models that varies in the number of hidden layers and in the number of neurons in each hidden layer, in order to explore the sensitivity and reliability of the DNN architectures towards prediction with inhomogeneously correlated data. The DNN model selection in this work has been driven by the convergence of the learning. So the

number of epochs and learning rate are also not fixed for all the models. I have considered both the univariate data situation, where both the input and output is a scalar variate, and the multivariate data situation, in which the input is multivariate with the output being a scalar variate.

Once a DNN model with a specific architecture shows convergence of loss upon learning with the chosen optimiser for a sufficiently long epoch, prediction is performed with the DNN model. I have performed these predictions using TensorFlow platform provided by Google and available at <https://www.tensorflow.org/tutorials/keras/regression>, following the code that is available in the official documentation/tutorial site of TensorFlow. Due to the lack of availability of specific guidelines in the literature to choose the best DNN architecture given a dataset - for predictions - I have conducted predictive experiments on several DNN architectures, and based on the predictive accuracy, selected three “best-performing” DNN models. Then using these three best models predictions on three new test datasets are generated with these chosen models.

I have used two univariate datasets D_1 , D_2 and one multivariate dataset D_3 . D_1 , D_2 datasets differ in terms of the inhomogeneity present in their correlation structures. D_3 also bears an inhomogeneous correlation structure. The DNN models used for these datasets are described in the following paragraphs. All the DNN models depicted in this work use Adam optimiser with “mean absolute error” as the loss function.

DNN models for data D_1

$NN2$ denotes DNNs with two hidden layers and varying number of neurons in a layer. For each of the following architectures with two hidden layers, we have performed predictions using three activation functions which are ReLU, softmax and sigmoid with the learning rate 0.01. With five architectures starting with $NN2$, ($NN2_x$, for $x = 0, 1, 2, 3, 4$), we have 15 different DNNs.

1. $NN2_0$: two hidden layers, each layer having 64 neurons.
2. $NN2_1$: two hidden layers, first and second layer consists of 64 and 180 neurons respectively.
3. $NN2_2$: two hidden layers, first and second layer consists of 184 and 64 neurons respectively.
4. $NN2_3$: two hidden layers, first and second layer consists of 184 and 184 neurons respectively.

5. NN_{24} : two hidden layers, first and second layer consists of 250 and 184 neurons respectively.

Similarly, NN_3 denotes DNNs with three hidden layers whose specific details are as follows. With each of these following architectures with three hidden layers, we have performed experiments with three activation functions same as before, thus giving rise to another 15 different DNNs.

1. NN_{30} : three hidden layers, each layer having 64 neurons.
2. NN_{31} : three hidden layers, first, second and third layer consists of 64, 184 and 64 neurons respectively.
3. NN_{32} : three hidden layers, first, second and third layer consists of 184, 184 and 64 neurons respectively.
4. NN_{33} : three hidden layers, each layer having 184 neurons.
5. NN_{34} : three hidden layers, first, second and third layer consists of 184, 64 and 32 neurons respectively.

DNN models for data D_2

NN_1 denotes DNNs with one hidden layer, for each of the following architectures with one hidden layer. where ReLU, softmax and sigmoid activation functions have been used for each of these DNN architectures.

1. NN_{10} : one hidden layer having 200 neurons.
2. NN_{11} : one hidden layer having 199 neurons.
3. NN_{12} : one hidden layer having 64 neurons.

NN_2 denotes DNNs with two hidden layers, for each of the following architectures with two hidden layers. Again, the three activation functions mentioned above are used for each of the NN_{2x} models, for this dataset.

1. NN_{20} : two hidden layers, first and second layer consists of 64 and 180 neurons respectively.
2. NN_{21} : two hidden layers, first and second layer consists of 200 and 64 neurons respectively.

3. NN_{2_2} : two hidden layers, first and second layer consists of 199 and 64 neurons respectively.

Similarly, NN_3 denotes DNNs with three hidden layers whose specific details are as follows. With each of these following architectures with three hidden layers, we have performed experiments with three activation functions same as before, thus giving rise to another 15 different DNNs.

1. NN_{3_0} : three hidden layers, first, second and third layer consists of 64, 184 and 64 neurons respectively.
2. NN_{3_1} denotes DNN with three hidden layers, each layer having 64 neurons.
3. NN_{3_2} : three hidden layers, first, second and third layer consists of 200, 184 and 64 neurons respectively.

DNN models for data D_3

NN_1 denotes DNNs with one hidden layer, for each of the following architectures with one hidden layer, we have performed predictions using ReLU activation functions with learning rate 0.01.

1. NN_{6_0} : six hidden layers, each layer having 64 neurons.
2. NN_{5_0} : five hidden layers, each layer having 64 neurons.
3. NN_{5_1} : five hidden layers, first, second, third, fourth and fifth hidden layer consists of 64, 184, 64, 184 and 64 neurons respectively.

5.4.2 Single layer GP based learning model for univariate data

In this thesis, I will also show that the predictive performance are affected with even single layer GP based models that uses stationary kernels, in the presence of inhomogeneity in the correlation structure of the data. Therefore, I will now present such single layer GP based model that uses stationary kernel. In this model, the sought function, which captures the relationship between the design and target variables of the training data, is modelled as a random realisation of the GP. I use stationary Matérn kernel to model the covariance function of this GP. Equation 5.4.1 shows the Matérn class of covariance kernels with positive

hyperparameters ν and ℓ , where $d(\cdot, \cdot)$ is the Euclidean distance between x_i and x_j ; $K_\nu(\cdot)$ is the modified Bessel function; and $\Gamma(\cdot)$ is the Gamma function.

$$k(x_i, x_j) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}}{\ell} d(x_i, x_j) \right)^\nu K_\nu \left(\frac{\sqrt{2\nu}}{\ell} d(x_i, x_j) \right) \quad (5.4.1)$$

More work on GP based predictions with various kernels can be found in Roy et al. (2023a). The likelihood of the inference is modelled with Multivariate Normal density by the definition of a GP. Vanilla MCMC is used for the inference, to learn the unknown hyperparameters (ν and ℓ) with truncated Normal proposal density and generic priors on the unknowns.

5.4.3 Datasets

I will first consider two univariate datasets D_1 and D_2 where both of the datasets bear varying levels of inhomogeneities in their correlation structure. Dataset D_2 is more inhomogeneously Correlated than dataset D_1 . I will now present details of these datasets along with the inhomogeneity parameters computed with these datasets, to justify my assertion in the first sentence of this paragraph.

Dataset D_1 : This is a subset of Brent crude oil price data that is publicly available at <https://datasource.kapsarc.org>. This data comprises oil prices for the period: 5th January 2018 to 8th March 2022. However, oil prices are not available for every day. I then choose a time interval of 5th January 2018 to 29th December 2021, to randomly sample 184 data points to build the training set D_1 where input variable is the date and the output variable is the oil price. 20 time points are also randomly selected as test inputs, at each of which I will undertake prediction of oil prices, which I will then compare against the true price. Such test data are excluded from the constructed training set. The date stamps are shifted s.t. the time point of the first data point is denoted the 0-th day; the input to the second time point is then the 1-st day, and so on.

The left most panel in Figure 5.15 presents the training data D_1 along with the trivial function discussed above in Section 5.2, The second panel from the left plots the “L values” for which the corresponding interval does not have any overlap with any other interval, in red. The inhomogeneity parameter η has been computed to be 0.1793 with $p = 33$.

Dataset D_2 : The second dataset is the subset of the power consumption data in Tetouan city, that is located in north Morocco (Dua and Graff, 2017; Salam and El Hibaoui, 2018). I have discussed this data in Section 5.3.1. The data includes observations from the 1st of January,

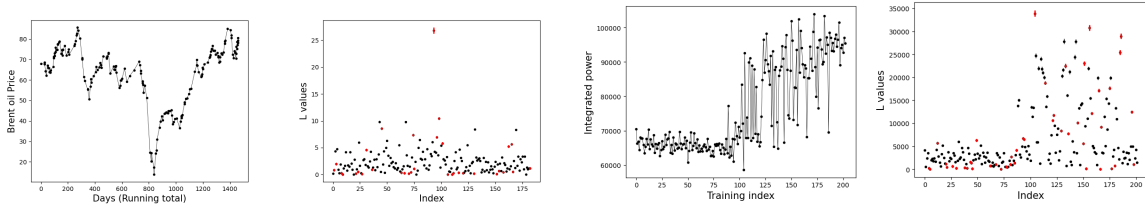


Fig. 5.15 The left most panel presents the training data D_1 along with the minimally-rough sample (trivial) function (with black solid line) where the output oil prices are plotted against days in running total. The second panel from the left plots the computed L-values (in black dots) along with their interval (with black vertical line) from data D_1 against their index; the L-values along with their corresponding interval are marked in red which have no overlap with other L-values. Similarly, the third panel from left depicts plots for dataset D_2 where output integrated power is plotted against index of the training points, while showing the trivial function in black solid line. The right most panel displays the computed L-values (in black dots) along with their interval (with black vertical line) from data D_2 against their index; the L-values along with their corresponding interval are marked in red which have no overlap with other L-values.

2017, to the 31st of December of that year, recorded every 10 minutes. In fact this recorded time series data includes values of five meteorological parameters - including temperature, humidity, wind speed, general diffuse flows and diffuse flows - and the corresponding value of the energy consumption in the three zones that the power distribution in this city is divided into. The power consumption data across these three zones are added, and I refer to the output variable as the “integrated power consumption” in this city. In the univariate setup we model temperature as the input and integrated power consumption as the output variable. The training data size for this dataset is 200.

The third panel from the left in Figure 5.15 presents the training data D_2 where output integrated power is plotted against index of the training points, along with the trivial function (that joins the output values in a piecewise-linear fashion). The right most panel plots the computed L-values (in black dots) along with their interval (with black vertical line) from data D_2 against their index; the L-values along with their corresponding interval are marked in red which have no overlap with other L-values. The inhomogeneity parameter η has been computed to be 0.2315 with $p = 47$.

Hence, it can be seen that the data D_2 is more inhomogeneously correlated than the data D_1 . So, the prediction task is expected to be harder with data D_2 than with data D_1 .

Dataset D_3 : This multivariate data has been constructed using the same power consumption data in Tetouan city that has been used to construct the data D_2 . In the multivariate setup we have used the five-dimensional input vector consisting of temperature, humidity, wind speeds,

diffuse flows along with the same output as in the univariate case, to construct data D_2 . The training data size for this dataset is 203. As the inhomogeneity in the correlation structure of a data is defined in terms of the output values of a training data, the inhomogeneity parameter η for this dataset is the same as that of dataset D_2 since the output of the two datasets D_2 and D_3 are the same.

5.4.4 Results

As the optimisation procedure minimises the loss for learning the DNN model, I have examined the convergence of the loss to ensure correct learning. I also intended to find if a DNN model that has converged to a smaller loss yields greater prediction performance than a model with bigger loss. To do so, I have identified the converged part from the trace of the loss and computed 95% confidence intervals (CI) and then the mean of this 95% CI which is denoted as μ_c . I also compute the mean from the entire trace of the loss. This mean is denoted as μ in order to check if a summary of the loss function correlates with the predictive performance.

Results depict that lower loss does not necessarily yield the best prediction. Also the minimum loss that can be attained through DNN models is unknown in the general case. To monitor prediction performance, I have computed mean square error (MSE) where the MSE is very large, I have also computed the root mean square error (RMSE) which is the square root of the MSE. For each of the datasets, four unique test sets “Test-1”, “Test-2”, “Test-3”, “Test-4” are created by choosing test inputs that are not included in the training data.

Results of DNN models with univariate data

Results with dataset D_1 : Table 5.4 presents predictive performance at the test set “Test-1” along with converged loss from various DNN models to predict oil prices using dataset D_1 . Prediction performance is depicted in terms of MSE. Table 5.4 shows results from using the DNN model named $NN3_1$, which with activation ReLU has achieved best predictive performance (run for 20000 epochs) while $NN2_1$ with sigmoid activation function (run for 20000 epochs) and $NN3_0$ with sigmoid activation function (run for 30000 epochs) yield second and third best predictive performances respectively. The losses attained by these DNN models do not indicate that lower loss can produce better prediction. Also it can be seen that within the setup of a given architecture, just changing the activation function can produce huge differences in predictive performance while similar kind of differences in the loss may or may not be seen. This raises a worry about how to decide on the non-linear transformation through the activation function, given an unseen dataset.

In Figure 5.16, the panels in the left, middle and right display results obtained using $NN3_1$ (with ReLU), $NN2_1$ (with sigmoid) and $NN3_0$ (with sigmoid) respectively. The top row depicts predictions at test set denoted by “Test-1”; in the middle row, the loss function is plotted; and in the bottom row, we show results from the last 2000 epochs, to illustrate the convergence of loss. Predictive performance of these DNN models using data D_1 on three new test sets - “Test-2”, “Test-3”, “Test-4”, are included in Table 5.7, which shows that the best performing DNN model does not yield best predictions for new test inputs.

	$NN2_0$			$NN2_1$			$NN2_2$			$NN2_3$			$NN2_4$		
	ReLU	softmax	sigmoid	ReLU	softmax	sigmoid	ReLU	softmax	sigmoid	ReLU	softmax	sigmoid	ReLU	softmax	sigmoid
MSE	7.849	7.715	35.359	7.704	286.493	4.233	6.909	9.952	57.314	6.761	28.062	23.929	9.3	12.838	48.062
HPD	[1.32, 2.86]	[1.95, 2.36]	[2.22, 2.44]	[1.55, 2.25]	[9.34, 9.47]	[0.88, 1.20]	[1.43, 2.01]	[1.88, 2.34]	[3.67, 3.93]	[1.49, 2.24]	[3.75, 4.03]	[3.01, 3.32]	[1.54, 2.10]	[2.41, 2.77]	[3.41, 4.02]
μ_c	2.095	2.161	2.334	1.9	9.409	1.043	1.723	2.114	3.809	1.868	3.892	3.172	1.823	2.592	3.722
μ	2.917	2.724	2.693	2.715	9.75	1.5	2.506	2.462	4.208	2.497	4.341	3.531	2.51	3.06	4.326
	$NN3_0$			$NN3_1$			$NN3_2$			$NN3_3$			$NN3_4$		
	ReLU	softmax	sigmoid	ReLU	softmax	sigmoid	ReLU	softmax	sigmoid	ReLU	softmax	sigmoid	ReLU	softmax	sigmoid
MSE	6.643	6.432	4.524	3.878	11.569	-	4.425	7.411	-	7.709	6.917	-	8.586	7.155	25.931
CI	[1.27, 1.84]	[1.67, 2.07]	[0.81, 1.24]	[1.03, 1.68]	[2.13, 2.50]	-	[1.04, 1.66]	[0.94, 1.19]	-	[1.17, 1.68]	[0.98, 1.53]	-	[1.50, 2.16]	[1.04, 1.47]	[2.61, 2.92]
μ_c	1.562	1.875	1.029	1.362	2.319	-	1.358	1.071	-	1.434	1.261	-	1.832	1.259	2.767
μ	1.949	3.794	1.386	1.781	3.41	-	1.931	1.944	-	2.066	2.143	-	2.722	2.322	3.2

Table 5.4 Performance of experiments with various DNN architectures to predict oil prices at test set “Test-1”, given dataset D_1 with prediction performance in terms of MSE, ‘CI’ denotes the 95% confidence interval computed from the post burn in trace of loss, μ_c is the mean of the 95% CI, μ is the mean of the entire trace of loss.

Results with dataset D_2 : Table 5.5 presents predictive performance at test set “Test-1” along with converged loss from various DNN models using dataset D_2 . It is to be recalled that this dataset bears more inhomogeneity in its correlation structure than dataset D_1 . The Table 5.5 depicts that many of the considered DNN models did not work for this dataset. Moreover, apart from ReLU activation function, most of the other activation functions did not work well for this dataset. There could be enormous number of DNN architectural configurations, however I have included the architectures that worked for data D_1 as well. I have tried with multiple learning rates to achieve convergence in the optimisation.

The experimentation was harder with this dataset and the predictive performance is quite poor with respect to the dataset D_1 , MSE is also very huge, depicting poor prediction. It can be seen that converged loss is also quite high, though it is unknown what the lowest loss is, that can be achieved with DNN models. $NN3_1$ (with ReLU), $NN2_2$ (with ReLU) and $NN2_0$ (with ReLU) are the best performing DNN models in this case. Even in this situation, it can be noticed that lower loss does not produce better predictions.

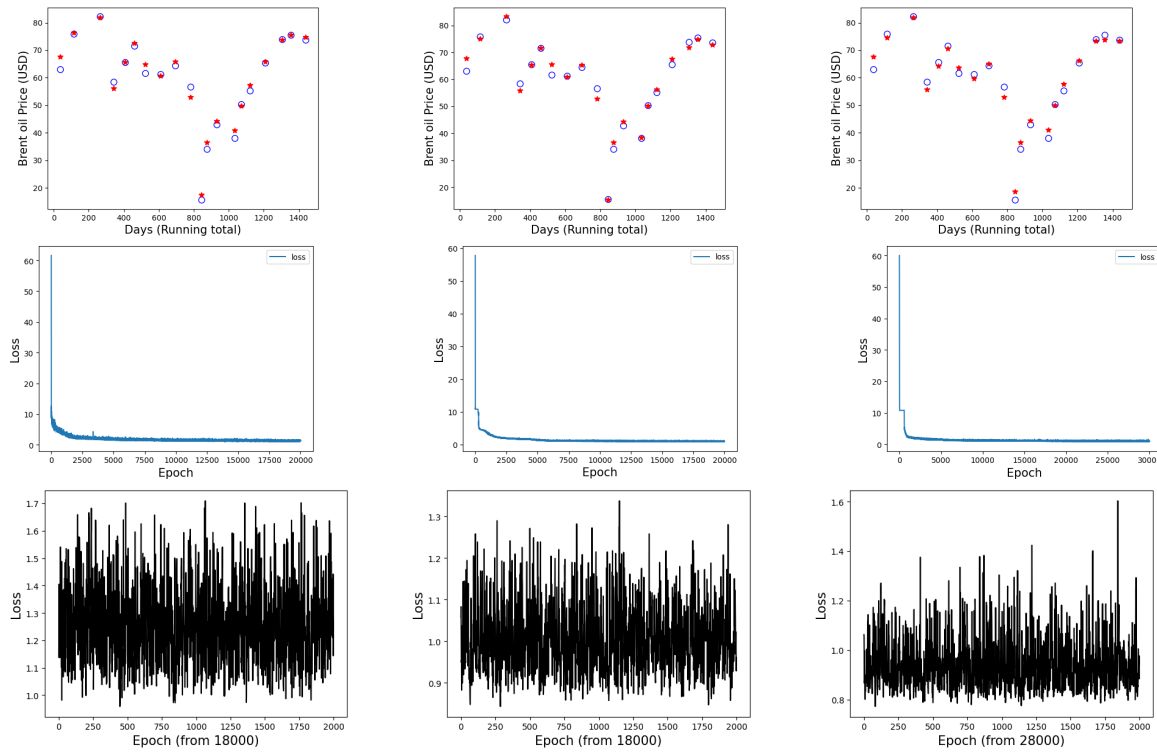


Fig. 5.16 Columns in the left, middle and right depict results pertaining to dataset D_1 with $NN3_1$ (with ReLU), $NN2_1$ (with sigmoid) and $NN3_0$ (with sigmoid) respectively. The top row shows predictions at test inputs “Test-1”; in the middle row the loss function is plotted; and the bottom row presents results from the last 2000 epochs to illustrate the convergence of loss. True values are displayed with blue hollow circle, while the predictions are depicted with red stars.

In Figure 5.17, the panels in the left, middle and right display results obtained using $NN3_1$ (with ReLU), $NN2_2$ (with ReLU) and $NN2_0$ (with ReLU) respectively. The top row depicts predictions at test set “test-1”; in the middle row the loss function is plotted; and in the bottom row, we show results from the last 2000 epochs, to illustrate the convergence of loss. Predictive performance of these DNN models using data D_2 on three new test sets - “Test-2”, “Test-3”, “Test-4”, are included in Table 5.7, which shows that the best performing DNN model does not yield best predictions for new test inputs.

It is to be noted that DNN models with one hidden layer $NN1_0$ and $NN1_1$ differ from each other with only one neuron in their hidden layer. The predictions obtained at the test inputs “Test-1” with these two models are not very different from each other. We get differences in terms of MSE as 87,215.3 with the difference in μ_c as 0.96. The left most and second from left panel in the Figure 5.18 show the predictions at “Test-1” with the data D_2 using

	$NN1_0$			$NN1_1$			$NN1_2$		
	ReLU	softmax	sigmoid	ReLU	softmax	sigmoid	ReLU	softmax	sigmoid
MSE	31724980.7	-	-	31812195.9	-	-	-	-	27557303.7
CI	[4228, 4431.7]	-	-	[4227, 4430.7]	-	-	-	-	[4173.7, 4177.5]
μ_c	4329.8	-	-	4328.8	-	-	-	-	4175.5
μ	5255	-	-	5260.6	-	-	-	-	4854.1
LR	0.2	0.2	0.2	0.2	0.2,0.3	0.2,0.1	0.5	0.5	0.5
	$NN2_0$			$NN2_1$			$NN2_2$		
	ReLU	softmax	sigmoid	ReLU	softmax	sigmoid	ReLU	softmax	sigmoid
MSE	30991905.9	-	-	30876501.9	-	-	30833224.4	-	-
CI	[4219.9, 4331.2]	-	-	[4219.4, 6180.1]	-	-	[4189.4, 4396.5]	-	-
μ_c	4275.5	-	-	5199.7	-	-	4292.9	-	-
μ	4322.9	-	-	4655.6	-	-	4554.9	-	-
LR	0.01	0.01,0.3	0.01	0.3	0.3	0.3	0.3	0.3	0.3
	$NN3_0$			$NN3_1$			$NN3_2$		
	ReLU	softmax	sigmoid	ReLU	softmax	sigmoid	ReLU	softmax	sigmoid
MSE	31905122	-	-	29278017.6	-	-	53900485.3	-	-
CI	[4150.9, 4444.2]	-	-	[4131.3, 4391.9]	-	-	[3309.5, 3656.6]	-	-
μ_c	4297.5	-	-	4261.6	-	-	3483	-	-
μ	4483.7	-	-	4481.3	-	-	3962.5	-	-
LR	0.1	0.1	0.1	0.01	0.01	0.01	0.01	0.01,0.3	0.01,0.3

Table 5.5 Performance of experiments with various DNN architectures to predict integrated power consumption at test set “Test-1”, given dataset D_2 , with prediction performance presented in terms of MSE, ‘CI’ denotes the 95% CI computed from the post burn in trace of loss, μ_c is the mean of the 95% CI, μ is the mean of the entire trace of loss, ‘LR’ denotes the learning rate used and tried as well (for some cases).

DNN model $NN1_0$ and $NN1_1$ respectively. Very little difference can be seen visually in the predictive performance.

The DNN models consisting of two hidden layers that differ from each other with only one neuron in the first hidden layer - $NN2_1$ and $NN2_2$ - the obtained difference in the predictions at “Test-1” in terms of MSE is 43,277.5 with the difference in μ_c computed to be 906.8. The predictions obtained at some of the test inputs using these two models, ($NN2_1$ and $NN2_2$), are quite different from each other (also visually differentiable) which is depicted in the third from left and right most panel of the Figure 5.18. It can be observed that the predictions and the loss are very sensitive to even a little change (change of one neuron) in the DNN architecture, hence, the reliability of the predictions is worrying with an urgent need for explainability in DNN models because there is no clear way of choosing a DNN architecture.

Another interesting result has been depicted in the Figure 5.19 that is obtained using the DNN model $NN1_2$ with Sigmoid activation function. Though the MSE for the prediction at “Test-1” is 27557303.8 which apparently hints towards better predictions than other predictions with higher MSE obtained from other DNN models. The prediction is depicted in the left most panel of the Figure 5.19 whereas loss from long windows of epochs and the last 2000 epochs, have been included in the middle and the right most panel. The loss

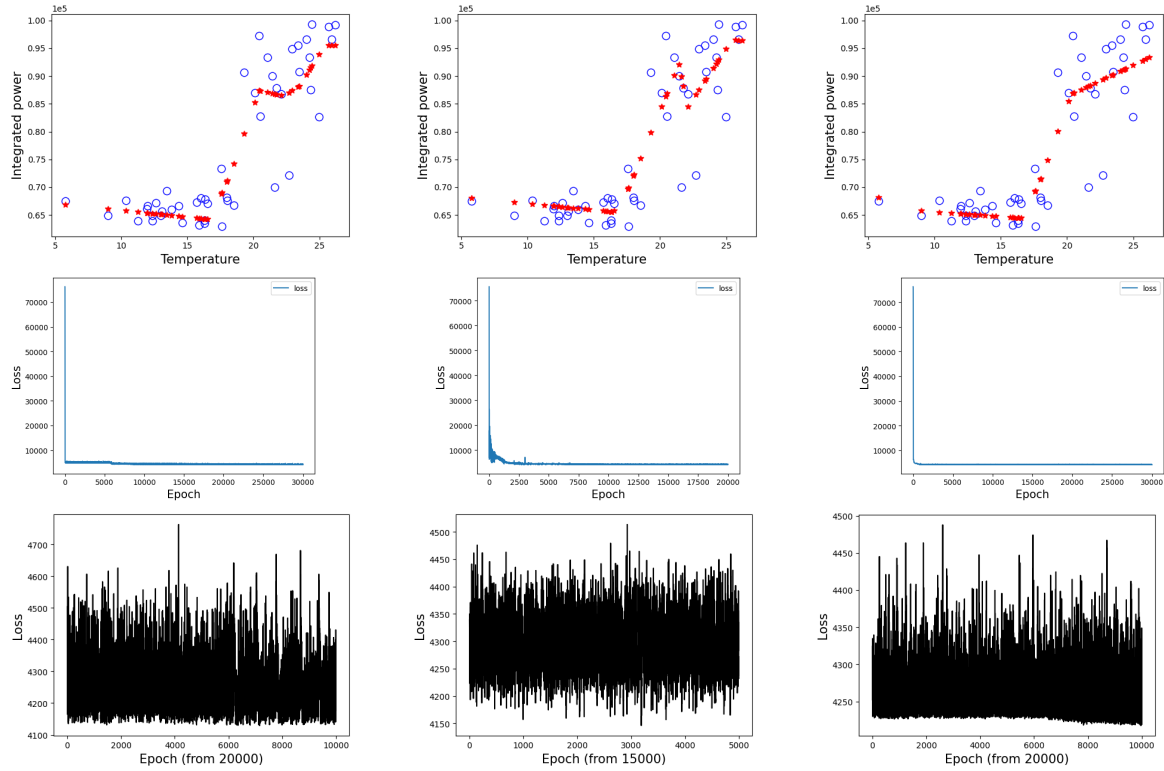


Fig. 5.17 Columns in the left, middle and right depict results pertaining to dataset D_2 using test set “Test-1” with $NN3_1$ (with ReLU), $NN2_2$ (with ReLU) and $NN2_0$ (with ReLU) respectively. The top row shows predictions; in the middle row the loss function is plotted; and the bottom row presents results from the last 2000 epochs to illustrate the convergence of loss. True values are displayed with blue hollow circle, while the predictions are depicted with red stars.

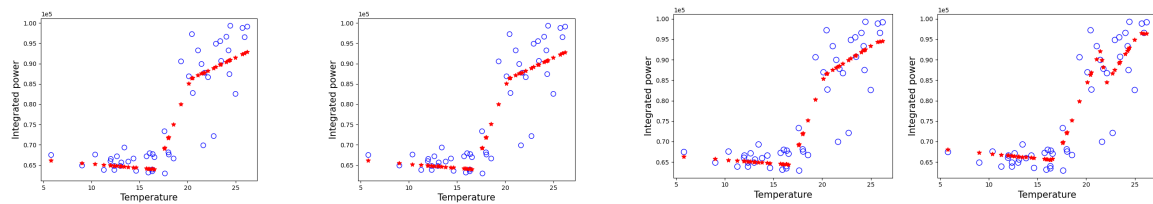


Fig. 5.18 The left most, second from left, third from left and the right most panel presents the predictions with the data D_2 using DNN model $NN1_0$, $NN1_1$, $NN2_1$ and $NN2_2$ respectively. True values are displayed with blue hollow circles, while the predictions are depicted with red stars.

and the quality of predictions (in spite of lower MSE) indicate that this model should not be considered for prediction and MSE for prediction quality should not be the only criteria while selecting the model and judging prediction quality.

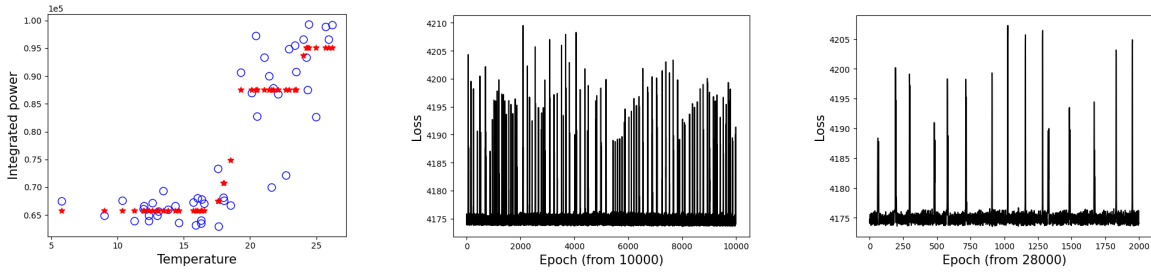


Fig. 5.19 The prediction obtained using DNN model $NN1_2$ with Sigmoid activation function with the dataset D_2 is depicted in the left most panel, loss from long windows of epochs, and the last 2000 epochs, are presented in the middle and right panel respectively. True values are displayed with blue hollow circle, while the predictions are depicted with red stars.

Results of Single layer GP based models with univariate data

Results with dataset D_1 : The sole hyperparameter of relevance to this model is the length scale hyperparameter, ℓ , as we have already set the value of ν to be $3/2$. The mean of its learnt 95% HPD is 16. The left most and second from left panel in the Figure 5.20 depict the traces of learnt length-scale and logarithm of posterior with the training data D_1 . Figure 5.21 shows predictions made on the four test datasets with training data D_1 stationary Matérn kernel having $\nu = 3/2$. Predictive performance of this model using data D_1 on four test datasets, are included in Table 5.7.

Results with dataset D_2 : The sole hyperparameter of relevance to this model is the length scale hyperparameter, ℓ , as we have already set the value of ν to be $1/2$. The mean of the learnt 95% HPD for ℓ is 0.63. It is to be noted that the ν value is the roughness parameter - the lower it is, higher is the roughness of the sought function. For this dataset, ν is required to be set to 0.5 which implies that the sought function is very rough. This also indicates that this data is highly inhomogeneously correlated and no parametric form imposed on the sought function can yield correct learning of the relationship between input and output variable.

The third-from-left panel and the right-most panel in Figure 5.20 depict traces of the learnt length-scale and the logarithm of the posterior with the training data D_2 . Table 5.7 shows predictive performance of this model using data D_2 on four test datasets.

Figure 5.22 shows predictions made on the four sets of test inputs, after learning with training data D_2 and the stationary Matérn kernel having $\nu = 1/2$.

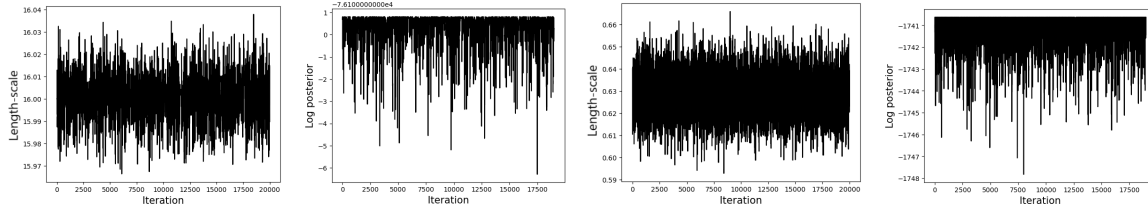


Fig. 5.20 Figure displaying traces of learning made with the GP based model. The left most and second from left panel depict the traces of the length-scale and the logarithm of the posterior with the training data D_1 using stationary Matérn kernel with $\nu = 3/2$. The third from left panel and the right most panel show traces of learnt length-scale and the logarithm of the posterior given data D_2 , where the learning is done using stationary Matérn kernel having $\nu = 1/2$.

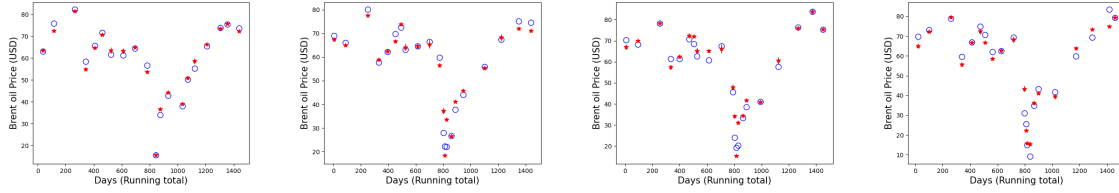


Fig. 5.21 Predictions made on the four test datasets (from left: “Test-1”, “Test-2”, “Test-3”, “Test-4”), given training data D_1 , with stationary Matérn kernel having $\nu = 3/2$.

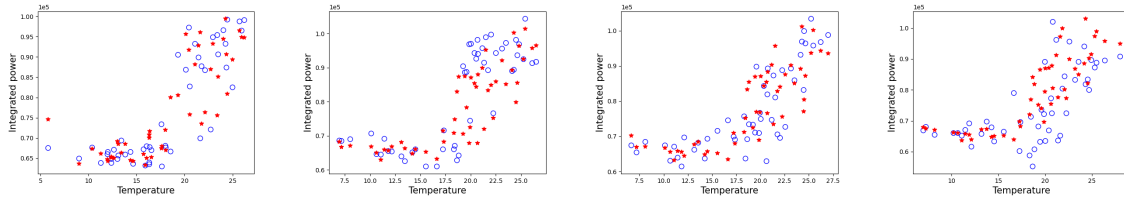


Fig. 5.22 Predictions made on the four test datasets (from left: “Test-1”, “Test-2”, “Test-3”, “Test-4”), given training data D_2 , with stationary Matérn kernel having $\nu = 1/2$.

DNN results with multivariate data D_3

Table 5.6 presents the predictive performance at the test sets “Test-1”, “Test-2”, “Test-3”, “Test-4”, given the data D_3 with three DNN architectures to predict integrated power consumption with predictive performance given in terms of RMSE. Very high predictive errors are noticed for all the test sets and the best DNN did not perform as well, for all the test sets. Figure 5.23 display the predictions obtained at the test set “Test-1” with DNN model NN_{60} that uses ReLU activation function along with the loss function and the quality of convergence. The predictions generated at the test sets “Test-2”, “Test-3”, “Test-4”, using the same model are presented in Figure 5.24, where the predictions are plotted against the index of the test points.

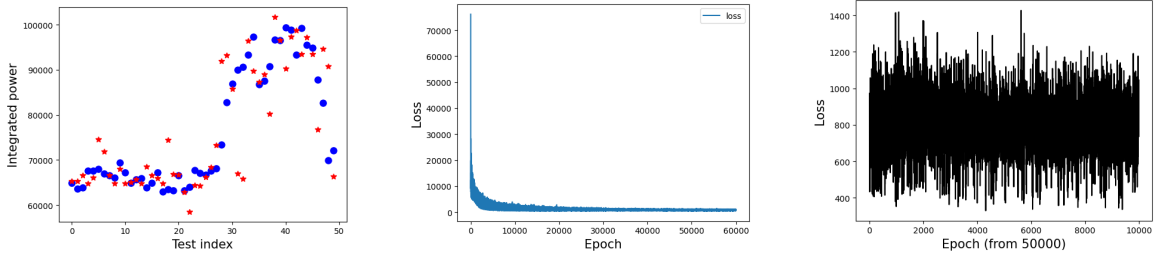


Fig. 5.23 The prediction obtained at test set “Test-1” using DNN model NN_{60} with ReLU activation function with the dataset D_3 is depicted in the left most panel. The loss function and the loss for the last 2000 epochs are presented in the middle and right panels respectively. True values are displayed with blue solid circle, while the predictions are depicted with red stars.

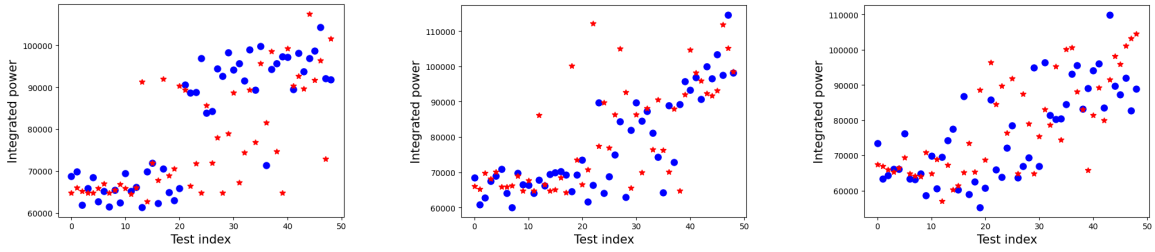


Fig. 5.24 The prediction obtained at test sets “Test-2”, “Test-3”, “Test-4”, given the data D_3 using DNN model NN_{60} with ReLU activation function is depicted the left panel onward respectively. True values are displayed with blue solid circle, while the predictions are depicted with red stars.

DNN Model	Loss			RMSE			
	CI	μ_c	μ	Test-1	Test-2	Test-3	Test-4
NN_{60}	[464.3, 1059.3]	761.76	1228.56	7787.5	13832.8	13645.9	13041.6
NN_{50}	[271.6, 879.6]	575.5	1029.6	7873.7	12988.7	39342.2	18274.7
NN_{51}	[205.3, 831.6]	518.5	911.3	8296.8	12430.4	107799.5	29891.8

Table 5.6 Predictive performance using data D_3 with various DNN architectures that are utilised to predict integrated power consumption with prediction performance in terms of RMSE. ‘CI’ denotes the 95% CI computed from the post burn in trace of loss, μ_c is the mean of the 95% CI, μ is the mean of the entire trace of loss.

For a GP based prediction with stationary kernel using the data D_3 see Roy et al. (2023a).

Data D_1									
		Test-1		Test-2		Test-3		Test-4	
DNN models		MSE	MSE interval	MSE	MSE interval	MSE	MSE interval	MSE	MSE interval
1st	$NN3_1$	3.88	-	5.85	-	5.28	-	23.35	-
2nd	$NN2_1$	4.23	-	4.55	-	5.75	-	26.24	-
3rd	$NN3_0$	4.52	-	4.33	-	6.85	-	23.74	-
GP based model		3.41	[3.23, 4.42]	14.81	[14.44, 16.53]	16	[14.3, 19.57]	19.69	[19.12, 19.73]
Data D_2									
		Test-1		Test-2		Test-3		Test-4	
DNN models		RMSE	RMSE interval	RMSE	RMSE interval	RMSE	RMSE interval	RMSE	RMSE interval
1st	$NN3_1$	5410.92	-	6866.9	-	7464.4	-	10237.1	-
2nd	$NN2_2$	5552.77	-	6780.1	-	7535.6	-	10650.1	-
3rd	$NN2_0$	5567.04	-	6763.1	-	7926.6	-	10447.3	-
GP based model		7768.22	[7766.9, 7769.6]	10290.1	[10288.6, 10291.5]	8988.8	[8988, 8989.6]	11774.4	[11772.2, 11776.6]

Table 5.7 Predictive performances with the three best DNN models and the single layer GP based model, where I have used data D_1 and D_2 with four test sets. For GP based model, RMSE denotes RMSE computed at the predicted means; the RMSE interval shows the RMSE computed using the predicted 95% HPDs.

5.5 Conclusion

The experiments done with DNNs in this section, show that DNN models worked quite well to produce predictions even with medium sized data, when the correlation structure of the data bears low inhomogeneity. The experiments also indicate that the DNN architecture selection is hard for data D_2 than D_1 which are marked by higher inhomogeneities in the correlation structure of the data D_2 than D_1 .

It can be noticed from the experimental results that the DNN models are very sensitive to the architectural choice, since even a little change in one of the hyperparameters can cause significant decline in prediction performance. Indeed, I showed that even changing one neuron in a hidden layer changes the prediction quality. This poses concerns about reliability of prediction especially given inhomogeneously correlated data.

I have also shown that MSE for prediction accuracy is not sufficient always to select the DNN model. Considering the quality of convergence of the optimisation would help selecting more reliable DNN models.

The current formulation of inhomogeneity parameter is useful for comparing various datasets that are inhomogeneously correlated. Future work includes designing a performance metric for prediction, made using data that is inhomogeneously correlated. This metric should consider the subspaces of the input space, outputs over which are differently correlated, yielding difference in the easiness of prediction within multiple subspace of input space.

For example, generating predictions within a particular subspace of $\mathcal{X}$ can be harder than predicting within another subspace. In the next chapter, I will show how the notion of inhomogeneity presented in this chapter can be used for explainability of predictions made in the context of high-dimensional learning.

Chapter 6

Non-parametric Learning in High Dimensions

6.1 Introduction

Tensor (Kolda and Bader, 2009) data is now a days very common within the data analytical tasks in several application domains such as neuroscience, biological applications, clinical applications, any application dealing with image data (Zhou et al., 2024). This is because, high-dimensional data is frequently generated within many real-world situation; those high-dimensional data can be efficiently represented through tensors and a rich set of mathematical techniques are available to deal with such tensors. Therefore, in recent data science applications, high-dimensional data is presented as tensors and various data analytical tasks such as prediction, are performed with the tensors. For recent developments on tensor data analysis, see Liu et al. (2022a).

In this chapter, I will present high-dimensional regression performed with GP that invokes non-parametric SQE-looking kernel (that is presented in Chapter 4). In my learning setup, the input is a vector-variate variable while the output is a matrix-variate variable. As, this chapter is concerned with high-dimensional regression, hence, I will now present the relevant literature on this topic.

Zhou et al. (2013) presents efficient tensor regression methods for tensor-variate high-dimensional input and they use Generalized linear models (Nelder and Wedderburn, 1972) to yield the predictive outputs. They have applied their model to real-world neuroimaging data for predicting clinically relevant response variable that can be discrete or continuous, given 3-dimensional images as input to the model. Although, this method is quite useful for performing regression with real-world high-dimensional data, however, this method lacks

quantification of uncertainties with the prediction, which is a practical requirement for many sensitive application areas.

Guhaniyogi et al. (2017) develops a framework for Bayesian tensor regression where the input can be vector-variate or tensor-variate with the output being scalar-variate. They have developed a new class of prior distributions on the tensor-variate input variable for dimensionality reductions relevant to the vectorization process. This work also includes a computationally efficient MCMC algorithm for performing inference. Extending their method for Multivariate output is a future work as the authors in Guhaniyogi et al. (2017) mention.

Signoretto et al. (2010) have presented a new kernel, suitable for tensor-variate data analysis where both the input and output is tensor-variate, that uses the structural information present in the input tensor data and the kernel has been applied to achieve various high-dimensional supervised task. Tensor-variate GP based regression method has been reported by Zhao et al. (2014) where the input to the model is tensor and output is a scalar and the invokes probabilistic kernel. The probabilistic kernel employs the measure of similarity between two input tensors is based on information divergence that measures the similarity between two probability distributions (such as Fisher kernel (Mika et al., 1999)). However, the probabilistic kernel in this work is still parametric and thus imposes a pre-chosen form to the correlation structure of the GP that might not be able to capture the true correlation structure of the empirical data.

Zhou et al. (2024) recently presented a nonlinear regression model that learns non-parametrically, which is suitable for multivariate, tensor-variate input and scalar output. Although the authors in Zhou et al. (2024) suggested that the learning framework is not restricted to the regression setup with scalar output, but can be generalised to the task of classification and regression with multivariate output. However, generalisability of their method to the regression setup with matrix-variate output is unknown.

Signoretto et al. (2014) has developed learning framework using convex optimization and spectral regularization for high-dimensional tensor input while the output remains as scalar-variate, this method is capable of dealing with missing values in the input space. The regression method developed by Rabusseau and Kadri (2016) is capable of handling tensor-variate output while having vector-variate input.

Risser and Turek (2020) also identifies that the existing non-stationary covariance kernels (Higdon et al., 1999; Paciorek and Schervish, 2003; Risser and Calder, 2015) require learning of large number of hyperparameters with difficulty in implementing them due to lack of the availability of the relevant algorithms. This motivates them to provide a framework to implement existing non-stationary kernels to achieve non-stationary GP, given high-

dimensional data along with Bayesian inference performed with MCMC methods. For moderately large data, the authors adopt an approach that approximates the GP methods. However, the developed framework supports learning of univariate function given vector valued input with size d .

Tensor regression in diverse set up has been reported by the authors: Guo et al. (2011); Yu and Liu (2016); Kossaifi et al. (2020); while further details on tensor regression can be found in Liu et al. (2022b). Tensor regression using DNN models have been reported in Kossaifi et al. (2020).

In this chapter, first I will present the general setup, the required model and inference for the non-parametric learning with high-dimensional data including the methodologies for uncertainty-included prediction of matrix-variate output variable given vector-variate variable and the description of the data in Section 6.2. Section 6.2 also includes the computational issues relevant to the efficient implementation with fast execution of this high-dimensional learning. The single layer GP based model for the high-dimensional learning has been presented in the Section 6.3 for the purpose of comparison. Section 6.4.1 includes all the results including the comparative results and the explanations of the predictive performance using the inhomogeneities present in the correlation structure. Finally the chapter is concluded in Section 6.5.

6.2 Non-parametric learning with high-dimensional data

In this chapter, I will consider the application of the non-parametric kernel model developed in Chapter 4, given high-dimensional data, to undertake supervised learning where the input variable in the training data $\mathbf{D}_{train}$ is vector-valued, while the output variable in $\mathbf{D}_{train}$ is high-dimensional. In this chapter I deal with the case that the output is matrix-valued. The $\mathbf{D}_{train} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^M$, where $X \in \mathbf{X} \in \mathcal{X} \subseteq \mathbb{R}^d$, with $X = x_i \forall i = 1, \dots, m$, and $Y \in \mathbf{Y} \in \mathcal{Y} \subseteq \mathbb{R}^{N \times P}$, with $Y = y_i \forall i = 1, \dots, m$. Here $N, P \in \mathbb{N}$. The output $\mathbf{Y}$ realised at the design input $\mathbf{x}_i$, yields the value y_i . Here $i = 1, \dots, M$. Then $\mathbf{Y}_1, \dots, \mathbf{Y}_M$ are collated to form the 3rd-ordered tensor (Bi et al., 2021)

$$\mathbf{D}_Y := (\mathbf{Y}_1 \dot{,} \dots, \dot{,} \mathbf{Y}_M), \text{ so that } \mathbf{D}_Y \in \mathbb{R}^{N \times P \times M}.$$

Therefore, in this setup, I would like to learn the matrix-valued functional relationship as follows,

$$\mathbf{Y} = \mathbf{f}(\mathbf{X}), \text{ where } \mathbf{f}: \mathcal{X} \subseteq \mathbb{R}^d \longrightarrow \mathcal{Y} \subseteq \mathbb{R}^{N \times P}.$$

As the sought function $\mathbf{f}(\cdot)$ is matrix-valued, to learn this sought function non-parametrically as depicted in Chapter 4, the outer layer GP $\mathcal{GP}_{outer}$ will now be a matrix-variate Gaussian Process $\mathcal{GP}_{outer,matrix}$ s.t.

$$\mathbf{f}(\cdot) \sim \mathcal{GP}_{outer,matrix}(\boldsymbol{\mu}(\mathbf{x}), \text{cov}(\mathbf{x}_i, \mathbf{x}_j)),$$

where $\boldsymbol{\mu}(\cdot)$ and $\text{cov}(\cdot, \cdot)$ are mean and covariance functions of this matrix-variate GP. Then by the definition of a GP, the joint probability density of M realisations of matrix-valued variables $\mathbf{f}(\cdot)$ realised at the M number of design points $\mathbf{x}_1, \dots, \mathbf{x}_M$ of the training data $\mathbf{D}_{train}$, will be a 3rd-order tensor Normal distribution (Kolda and Bader, 2009; Manceur and Dutilleul, 2013; McCullagh, 2018) as,

$$[\mathbf{f}(\mathbf{x}_1), \dots, \mathbf{f}(\mathbf{x}_M)] \sim \mathcal{TN}(\mathbf{T}_\mu, \boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2, \boldsymbol{\Sigma}_3), \quad (6.2.1)$$

where $\mathbf{T}_\mu^{(N \times P \times M)}$ is the 3rd-order mean tensor of this density, the covariance matrices of which are $\boldsymbol{\Sigma}_1^{(P \times P)}, \boldsymbol{\Sigma}_2^{(N \times N)}, \boldsymbol{\Sigma}_3^{(M \times M)}$.

Hence, in order to learn the sought function $\mathbf{f}(\cdot)$ that is a random realisation of the GP $\mathcal{GP}_{outer,matrix}$, $\mathbf{T}_\mu, \boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2, \boldsymbol{\Sigma}_3$ are the unknown model parameters that need to be learnt. As the joint probability of the output $\mathbf{Y}$ is a tensor Normal, the likelihood of the unknown model parameters, given the data is a tensor Normal density.

Definition 6.2.1. *Likelihood of model parameters $\mathbf{T}_\mu, \boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2, \boldsymbol{\Sigma}_3$ given data $\mathbf{D}_{train}$ is a 3rd-order tensor Normal density:*

$$\mathcal{L}(\mathbf{T}_\mu, \boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2, \boldsymbol{\Sigma}_3 | \mathbf{D}_{train}) \propto \exp\left(-\frac{\|(\mathbf{D}_Y - \mathbf{T}_\mu) \times_1 \mathbf{C}_1^{-1} \times_2 \mathbf{C}_2^{-1} \times_3 \mathbf{C}_3^{-1}\|^2}{2}\right),$$

where, the positive definite covariance matrix $\boldsymbol{\Sigma}_i = \mathbf{C}_i \mathbf{C}_i^T, i = 1, 2, 3$, $\mathbf{C}_i$ being the square root of the covariance matrix $\boldsymbol{\Sigma}_i$ and the notation $\times_i, i = 1, 2, 3$, depicts the i -mode product of a matrix and a tensor (Kolda and Bader, 2009; Oseledets, 2011). For a tensor $\mathbf{T}^{(N \times P \times M)} = [a_{ijk}], i = 1, \dots, N; j = 1, \dots, P; k = 1, \dots, M$, $\|\mathbf{T}\|^2$ is the Frobenieous norm and given by $\sum_{i=1}^N \sum_{j=1}^P \sum_{k=1}^M a_{ijk}^2$.

According to Manceur and Dutilleul (2013), a k -th ordered random tensor $\boldsymbol{\Sigma} \in \mathbb{R}^{p_1 \times p_2 \times \dots \times p_k}$ can be factorised into an k -th ordered tensor $\mathbf{T}$ and k number of covariance matrices $\boldsymbol{\Sigma}_1, \dots, \boldsymbol{\Sigma}_k$ by Tucker decomposition (Hoff, 2011; Kolda and Bader, 2009; Manceur and Dutilleul, 2013), such that $\boldsymbol{\Sigma} = \mathbf{T} \times_1 \boldsymbol{\Sigma}_1 \times_2 \boldsymbol{\Sigma}_2 \dots \times_k \boldsymbol{\Sigma}_k$.

Remark 6.2.1. *The mean tensor $\mathbf{T}_\mu \in \mathbb{R}^{N \times P \times M}$ is estimated as $\hat{\mathbf{T}}_\mu$, by using the sample mean $\bar{\mathbf{y}}$ of the sample $\{\mathbf{y}_1, \dots, \mathbf{y}_M\}$.*

I will depict the way I have learnt the covariance matrices $\Sigma_1, \Sigma_2, \Sigma_3$ in the context of this application in the next few sections, after presenting the data that I have made this application with. Once the covariance matrices $\Sigma_1, \Sigma_2, \Sigma_3$ are learnt, the mean and variance of the output $\mathbf{Y}^*$ at a test input $\mathbf{x}^*$ are computed in a closed-form way.

6.2.1 Prediction of matrix-variate variable given vector input

When $\mathbf{Y} \in \mathcal{Y}$ is a matrix-valued variable, the likelihood becomes a 3rd-ordered tensor Normal density $\mathcal{TN}(\mathbf{T}_\mu, \Sigma_1, \Sigma_2, \Sigma_3)$. $\mathbf{y}_i$ can be vectorized to $\text{vec}(\mathbf{y}_i) \in \mathbb{R}^{PN}$.

With the vectorization of $\mathbf{y}_i$, $\forall i = 1, \dots, M$, the vectorised output are now collated to $(\text{vec}(\mathbf{y}_1) : \text{vec}(\mathbf{y}_2) : \dots : \text{vec}(\mathbf{y}_M))$, to form a matrix-variate output variable of the training data.

A tensor Normal density is equivalent to a matrix Normal:

$$\mathcal{TN}(\mathbf{T}_\mu, \Sigma_1, \Sigma_2, \Sigma_3) \equiv \mathcal{MN}(\text{vec}(\mathbf{T}_\mu), \Sigma_3, \Sigma_1 \otimes \Sigma_2),$$

where $\otimes$ denotes the Kronecker product.

The Kronecker product of matrices $\Sigma_1^{(P \times P)}$ and $\Sigma_2^{(N \times N)}$ (in the order) is $\Sigma_1 \otimes \Sigma_2$ with dimensionality $PN \times PN$ s.t. $\Sigma_1 \otimes \Sigma_2 = [a_{j,k} \Sigma_2]$, where $\Sigma_1^{(P \times P)} = [a_{j,k}]$, $\forall j, k = 1, \dots, P$.

Also, the matrix Normal likelihood is equivalent to a Multivariate Normal:

$$\mathcal{MN}(\text{vec}(\mathbf{T}_\mu), \Sigma_3, \Sigma_1 \otimes \Sigma_2) \equiv \mathcal{N}(\text{vec}(\text{vec}(\mathbf{T}_\mu)), (\Sigma_1 \otimes \Sigma_2) \otimes \Sigma_3).$$

Remark 6.2.2. $(\Sigma_1 \otimes \Sigma_2) \otimes \Sigma_3$ is computed using the design and test input points in order to compute the closed-form mean and variance of the predicted output variable $\mathbf{Y}^*$ at a given test input $\mathbf{x}^*$ using $\mathcal{GP}_{outer, matrix}(\mathbf{T}_\mu, (\Sigma_1 \otimes \Sigma_2) \otimes \Sigma_3)$, that generates the sought function

$$\mathbf{f}(\cdot) \sim \mathcal{GP}_{outer, matrix}(\mathbf{T}_\mu, (\Sigma_1 \otimes \Sigma_2) \otimes \Sigma_3).$$

Let $\Sigma_4 = (\Sigma_1 \otimes \Sigma_2) \otimes \Sigma_3$. Let $\Sigma_k^{(\mathbf{x}^*, \mathbf{X})}$, $\forall k = 1, 2, 3$ denote the covariance computed between outputs realised at pairs of inputs, one of which is the test input $\mathbf{x}^*$ and another is a design input $\mathbf{X}$. Then, $(\Sigma_1^{(\mathbf{x}^*, \mathbf{X})} \otimes \Sigma_2^{(\mathbf{x}^*, \mathbf{X})}) \otimes \Sigma_3^{(\mathbf{x}^*, \mathbf{X})}$ is $\Sigma_4^{(\mathbf{x}^*, \mathbf{X})}$.

Similarly, $\Sigma_4^{(\mathbf{X}, \mathbf{X})}$ denotes the covariance computed between outputs realised at pairs of design inputs $\mathbf{X}$ i.e.

$$\Sigma_4^{(\mathbf{X}, \mathbf{X})} = (\Sigma_1^{(\mathbf{X}, \mathbf{X})} \otimes \Sigma_2^{(\mathbf{X}, \mathbf{X})}) \otimes \Sigma_3^{(\mathbf{X}, \mathbf{X})}.$$

Also, $\Sigma_4^{(\mathbf{x}^*, \mathbf{x}^*)}$ denotes the covariance computed between outputs realised at pairs of test inputs $\mathbf{x}^*$ s.t.

$$\Sigma_4^{(\mathbf{x}^*, \mathbf{x}^*)} = (\Sigma_1^{(\mathbf{x}^*, \mathbf{x}^*)} \otimes \Sigma_2^{(\mathbf{x}^*, \mathbf{x}^*)}) \otimes \Sigma_3^{(\mathbf{x}^*, \mathbf{x}^*)}.$$

Definition 6.2.2. *The expected mean and variance of the predicted output variable $\mathbf{Y}^*$ at a given test input $\mathbf{x}^*$ using $\mathcal{GP}_{outer, matrix}(\cdot, \cdot)$ is:*

$$\mathbb{E}[\mathbf{Y}^*] | \mathbf{D}_{train}, \mathbf{x}^* = \mathbf{T}_\mu + \Sigma_4^{(\mathbf{x}^*, \mathbf{X})} \left(\Sigma_4^{(\mathbf{X}, \mathbf{X})} \right)^{-1} \text{vec}(\text{vec}(\mathbf{D}_Y - \mathbf{T}_\mu)) \quad (6.2.2)$$

$$\text{var}(\mathbf{Y}^*) | \mathbf{D}_{train}, \mathbf{x}^* = \Sigma_4^{(\mathbf{x}^*, \mathbf{x}^*)} - \Sigma_4^{(\mathbf{x}^*, \mathbf{X})} \left(\Sigma_4^{(\mathbf{X}, \mathbf{X})} \right)^{-1} \left(\Sigma_4^{(\mathbf{X}, \mathbf{x}^*)} \right)^T \quad (6.2.3)$$

Remark 6.2.3. *The predicted mean $\mathbb{E}[\mathbf{Y}^*]$ of the output realised at the test point $\mathbf{x}^* \in \mathbb{R}^d$ in the Definition 6.2.2, is in the form of the flattened vector of dimension PN . $\mathbb{E}[\mathbf{Y}^*] \in \mathbb{R}^{PN}$ needs to be de-vectorized in order to return to original shape of output $\in \mathbb{R}^{N \times P}$.*

6.2.2 Data

The high-dimensional data for this application has been obtained from astronomical simulations (Chakrabarty, 2007) pertaining to galactic dynamical theory. Said simulations track the evolution of a bunch of stars under the influence of feature parameters of the Milky Way, resulting - at the current time - in the noted velocity vectors of the stars that land in an identified region of the Milky Way Galaxy disk. So the velocities of stars collected in a given region of the galactic disk, inform on the galactic features. This feature parameters of the Milky Way is a vector-valued variable $\mathbf{X} \in \mathcal{X} \in \mathbb{R}^d$. When the velocity vectors of a given 50 stars are collated, then a 50-dimensional matrix-valued stellar velocity matrix $\mathbf{Y}$ is obtained, (given that the tracked velocities are 2-dimensional). Therefore, $\mathbf{Y}$ is affected by $\mathbf{X}$ and in this application I will learn the functional relationship $\mathbf{Y} = \mathbf{f}(\mathbf{X})$ using the training data that is obtained from these astronomical simulations. In this training set, $\mathbf{X}$ is the input and $\mathbf{Y}$ is the output.

In the simulation, the feature parameters of the Milky Way $\mathbf{X} \in \mathbb{R}^d$ is a two-dimensional vector $d = 2$ comprising features X_1 and X_2 . X_1 is the radial location of the Sun from the centre of the Milky Way, and - by astronomical theory - gives the angular speed of the bar where the bar is an elongated (triaxially-shaped) structure built of stars, that rotates, pivoted at the centre of the Galaxy. X_2 is the angular separation of the Sun from an identified axis in the Milky Way disk. Thus $\mathbf{X} = (X_1, X_2)^T$ is then a location vector of the Sun in the two-dimensional Milky Way disc.

The Numerical simulations are conducted in Chakrabarty (2007) with 216 different astronomical models of the Galaxy, where each model is marked by a unique value of the

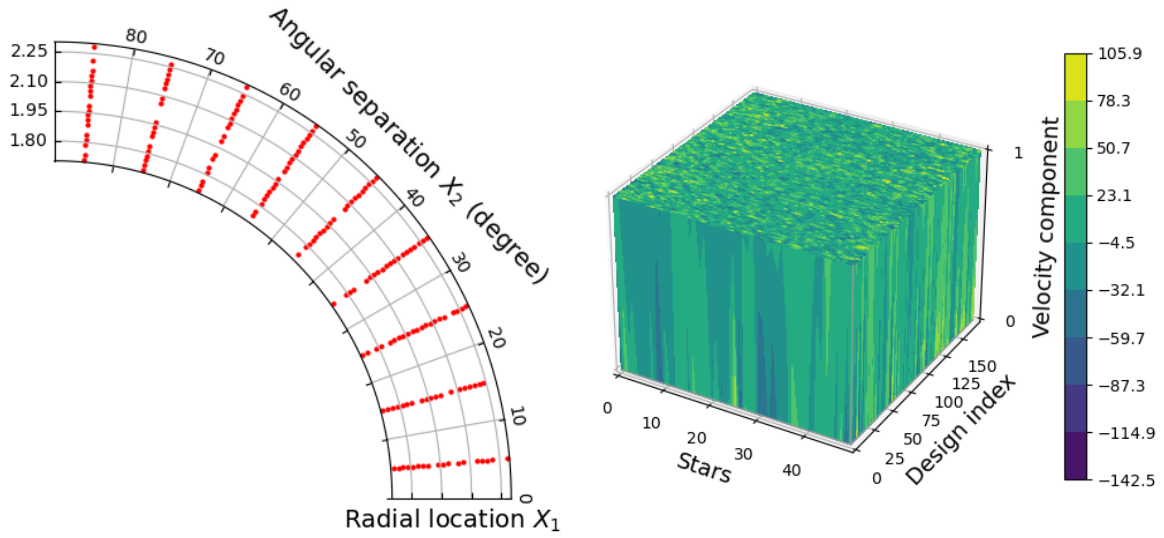


Fig. 6.1 High-dimensional astronomical data obtained from Chakrabarty (2007). In the left panel, input design points $\mathbf{X}$ are depicted in red dots within polar coordinates with the radial component X_1 and the angular component X_2 . The output matrices $\mathbf{Y}$ are depicted in the right panel which are collated to a 3-rd order tensor $\mathbf{D}_Y \in \mathbb{R}^{50 \times 2 \times 173}$ in the training set I use, since my used training set comprises sets of velocity vectors of 50 stars, collected at 173 identified locations (in the galactic disk). The colour coding presents the different values of this tensor. The velocity components are labelled as “0” - for the first velocity component and “1” - for the second velocity component.

Milky Way feature parameter vector $\mathbf{X}$. Here, the feature parameter that is not fixed by hand is the sun’s location in the Milky Way disk. Thus, the sun’s radial location from the centre of the Galaxy, and the angular separation of the Sun-Galactic Centre line (from the bar long axis) are the two feature parameters that are not fixed by hand in the simulations. All other relevant galactic parameters are fixed by hand.

The sun’s 2-dimensional location in the galactic disk is placed on a regular 2-dimensional, galacto-centric polar grid, over a chosen annulus in a 2-dimensional astronomical model of the Milky Way disk. This polar grid in turn is uniformly partitioned into 216 grid cells so that the $\mathbf{x}_i$ joining the centre of the Galaxy to the centroid of the i -th grid cell is the i -th design point representing the 2-dimensional solar location vector (i.e. galactic feature parameter vector) in the i -th model of the Galaxy. The observer in the i -th grid cell measures velocities of multiple stars that land in that grid cell, at the end of a simulated evolution of a sample of stars that are evolved in this i -th model of the Milky Way disk. The velocity vector of 50 stars are chosen in each grid cell. This gives the matrix of two-dimensional velocity vectors of 50 stars in each grid cell. The two-dimensional velocity vector of each star comprise the first velocity component “U” and the second velocity component “V”. Thus, for a single location

vector $\mathbf{x}_i \in \mathbf{X}, \forall i = 1, \dots, M$, the corresponding output $\mathbf{y}_i \in \mathbf{Y}, \forall i = 1, \dots, M$ is a matrix with dimension 50×2 i.e. $N = 50$ and $P = 2$ in this application.

Hence, the simulation data consists of total 216 data points, which I have split randomly s.t about 80% of the data (173 data points) can be used as the training data $\mathbf{D}_{train}$, and the rest of the data can be used as the test set which is depicted in Figure 6.2.

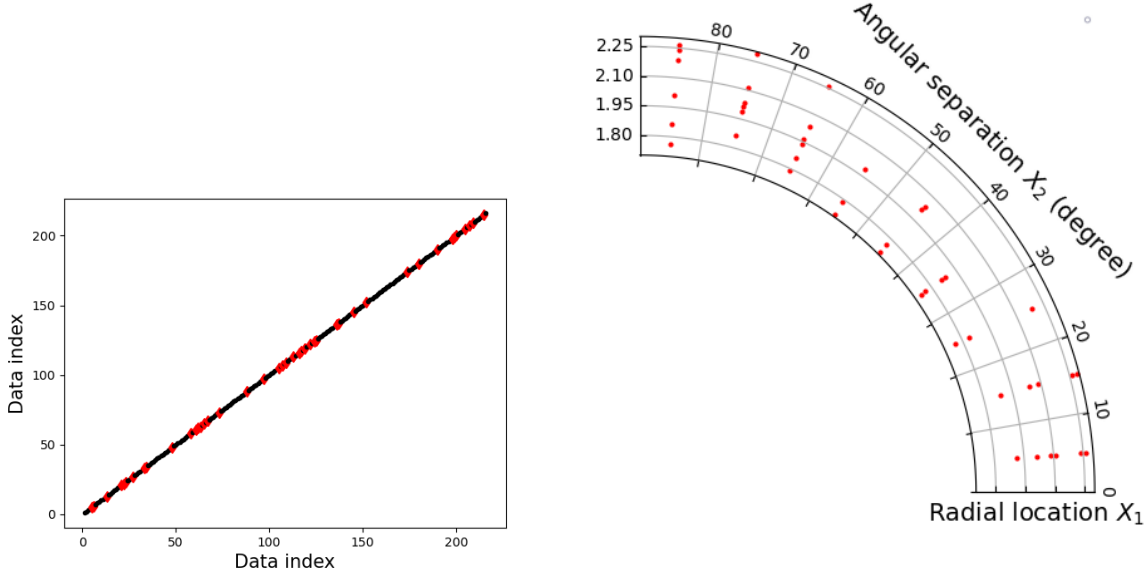


Fig. 6.2 Train-test split (80-20) in the high-dimensional astronomical data obtained from Chakrabarty (2007) that consists of 216 number of data points. In the left panel, the black dots represents the index of 173 training points and the red diamonds show the index of 43 test points. In the right panel input test points $\mathbf{X}$ are depicted in red dots within polar coordinates with the radial component X_1 and the angular component X_2 .

Therefore, the input comprises 173 number of two-dimensional vectors, each of which corresponds to a 50×2 dimensional output matrix. So the full training data is $\mathbf{D}_Y$ which is a tensor of dimensions $50 \times 2 \times 173$. The input training data is visually depicted with red dots in the left panel of the Figure 6.1 while the right panel of this figure presents the output tensor $\mathbf{D}_Y$.

The astronomical simulations generated the output $\mathbf{Y}$ matrices at the different input solar location vectors $\mathbf{X} = (X_1, X_2)^T$ in a way that the correlation structure of data on the generated $\mathbf{Y}$ values bears inhomogeneities in the correlation structure. The inhomogeneity parameter η is computed for each of the 50 stars, for each of its two velocity components across 173 number of design points. The heat map of this matrix is shown in the left panel of the Figure 6.3; the middle and the right panel depict the 3D-bar plot of the same matrix from two different angles for better visualization. Figure 6.3 depicts that the values of the η is highly non-uniform, different from each other, indicating the existence of large inhomogeneities in

the correlation structure of data on the output. In particular, η values of the second velocity component are more discontinuous than the first velocity component, which shows that the output data on the second velocity component is more inhomogeneously correlated than the first velocity component.

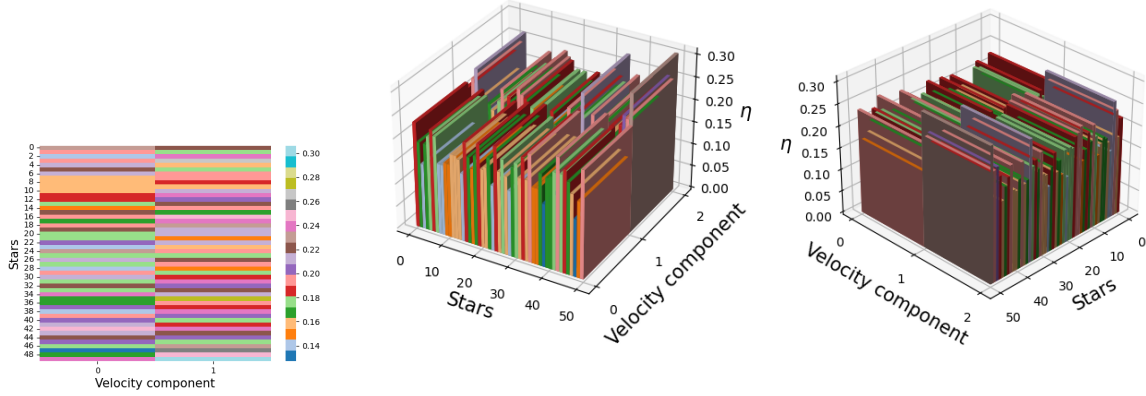


Fig. 6.3 Inhomogeneity parameter η , computed from the output values generated at the design inputs, in the high-dimensional astronomical data. The left most panel shows the heat map of the η values computed for each star for each of the velocity components. The middle and the right panels depict visualisation in 3D-bar plots of the training data, from two different angles. The velocity components are labelled as “0” - for the first velocity component and “1” - for the second velocity component.

6.2.3 Learning of covariance matrices

Now, I will discuss the strategy for learning the inter-component covariances $\Sigma_1^{(P \times P)}$, $\Sigma_2^{(N \times N)}$, $\Sigma_3^{(M \times M)}$ which is now $\Sigma_1^{(2 \times 2)}$, $\Sigma_2^{(50 \times 50)}$, $\Sigma_3^{(173 \times 173)}$ in the context of the current high-dimensional data that is mentioned in the Equation 6.2.1. The high-dimensional learning scheme presented in the context of this application, allows for a clear quantification of the covariances between the components across dimensions. The three covariance matrices $\Sigma_1, \Sigma_2, \Sigma_3$ inform on the following:

- $\Sigma_1^{(2 \times 2)}$ captures the covariance between the two components of the velocity vectors observed in the training set, across all stars and all design inputs (i.e. designed solar locations).
- $\Sigma_2^{(50 \times 50)}$ presents covariance between vectors of different pairs of stars, collected across all design inputs in this training set.
- $\Sigma_3^{(173 \times 173)}$ depicts the covariance amongst velocity matrix pairs observed at different design inputs, (i.e, proposed solar locations).

Three ways to learn covariance matrix: There are three ways through which covariance matrix can be learnt, where each strategy has its own shortcomings. The three types of covariance learning are depicted as follows.

1. Direct learning – each of the distinct elements of the covariance matrix can be directly learnt through an inference technique such as MCMC. For a $K \times K$ covariance matrix, the number of distinct elements that need to be learnt is $O(\frac{K^2}{2})$ which are the elements of either the lower or the upper triangle of the covariance matrix. However, maximally ~ 50 parameters can be learnt directly through MCMC, hence for training dataset with any more than about 10 data points, this approach is not practically possible.
2. Empirical estimate – each of the distinct elements of the covariance matrix is estimated from the empirical data. One way to compute such estimated covariance matrix would be to vectorize the i -th and j -th slices along the desired dimension of the tensor. For example, for a 3-dimensional dataset depicted in the left most panel of Figure 6.4, the slices could be obtained by chopping the 3rd-order tensor data depth-wise (row $\times$ column-dimensional matrix as slices are shown in the second panel from left in the Figure 6.4) or row-wise (depth $\times$ column-dimensional matrix as slices are shown in the third panel from left in the Figure 6.4) or column-wise (row $\times$ depth-dimensional matrix as slices are shown in the right most panel in the Figure 6.4) in order to get vectors corresponding to the i -th and j -th slices. Then the empirical covariance is estimated using these two vectors to yield the estimated covariance between the i -th and j -th slices. However, an empirical estimate of covariance may have the some disadvantages such as:
 - loss of information about correlation between pairs of slices (in the data) that are vectorised.
 - the sample-based estimated covariance can be wrong or misrepresentative of the population covariance, especially when the data size is too small,
3. Kernel parametrisation – The covariance matrix is parametrised using a kernel function $k(\cdot, \cdot)$ such that the covariance $cov(\mathbf{Y}_i, \mathbf{Y}_j)$ of the output variable $\mathbf{Y}_i$ realised at input $\mathbf{x}_i$, and output $\mathbf{y}_j$ realised at $\mathbf{x}_j$ is expressed in terms of the inputs $\mathbf{x}_i$ and $\mathbf{x}_j$, as $k(\mathbf{x}_i, \mathbf{x}_j)$. When a covariance matrix is parametrised using a kernel function $k(\cdot, \cdot)$, then the learning is reduced to the learning of the parameters of the kernel function $k(\cdot, \cdot)$ which are usually substantially fewer in number than $O(\frac{K^2}{2})$ for a $K \times K$ covariance matrix. Kernel parametrisation is possible only when there is an input variable at which the corresponding outputs are seen to be realised.

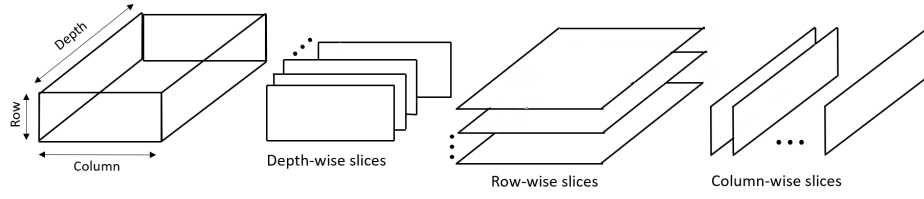


Fig. 6.4 An example 3-dimensional data shown in the left most panel with its depth, row and column depicted. The second panel from left shows the row $\times$ column-dimensional matrix as slices obtained by chopping the 3-dimensional data depth-wise. The third panel from left shows the depth $\times$ column-dimensional matrix as slices obtained by chopping the 3-dimensional data row-wise. The right most panel displays the row $\times$ depth-dimensional matrix as slices obtained by chopping the 3-dimensional data column-wise.

If for a pair of output variables whose covariance needs to be computed, corresponding input values cannot be identified, then in such situations, kernel parametrisation cannot be adopted. Then, direct learning or empirical estimate of the covariances can be used. The choice of the kernel function $k(\cdot, \cdot)$ is an important aspect in order to attain correct learning of the sought $\mathbf{f}(\cdot)$.

Having discussed multiple ways to learn the covariance matrices, I will now present how the covariance matrices $\Sigma_1^{(2 \times 2)}$, $\Sigma_2^{(50 \times 50)}$, $\Sigma_3^{(173 \times 173)}$ are learnt in the context of current application with high-dimensional data.

Learning $\Sigma_1^{(2 \times 2)}$: Three distinct elements of $\Sigma_1^{(2 \times 2)}$ are directly learnt through MCMC. These elements are two diagonal elements of the Σ_1 , referred as $\sigma_{11}^{(1)}$ and $\sigma_{22}^{(1)}$ for the first and second diagonal elements respectively, and the non-diagonal element referred as $\sigma_{10}^{(1)}$.

Learning $\Sigma_2^{(50 \times 50)}$: Direct learning of $\frac{50^2}{2} = 1250$ number of distinct elements of Σ_2 is not feasible practically using MCMC as mentioned in point 1 above. Hence, I can either learn the elements of Σ_2 by kernel parametrisation or empirically estimate. The output tensor $\mathbf{D}_Y \in \mathbb{R}^{50 \times 2 \times 173}$ comprises 173 number of 50×2 -dimensional matrices contain the 2-dimensional velocity vectors of each of the 50 stars, where each such matrix is realised at one of 173 solar locations. Now, the ij -th element of the inter-star covariance matrix Σ_2 represents the covariance between the 173×2 -dimensional i -th and j -th matrix that pertains to the i -th and j -th star. However, no input can be identified at which the 173×2 -dimensional velocity matrix is realised. In other words, the velocity values pertaining to the i -th star does not have any numerical input variable, at which those velocity values are attained. Basically, index labels that can be associated with stars, cannot be used as input variables since the label index cannot be treated to be the numerical-valued input space variable. Hence, kernel

parametrisation cannot be used to learn Σ_2 . Therefore, Σ_2 is empirically estimated according to the Definition 6.2.3 from the data to yield the empirically estimated inter-star covariance matrix $\hat{\Sigma}_2$.

Definition 6.2.3. *Covariance between the 173×2 -dimensional velocity matrix $\mathbf{V}_i := [v_{kq}^{(i)}]$ pertaining to the i -th star and the velocity matrix $\mathbf{V}_j := [v_{kq}^{(j)}]$ pertaining to the j -th star with $k = 1, \dots, 173$; $q = 1, 2$, is estimated as $\widehat{\sigma}_{i,j}^{(2)}$, where:*

$$\widehat{\sigma}_{i,j}^{(2)} = \frac{1}{2} \sum_{q=1}^2 \left[\frac{1}{173-1} \left(\sum_{k=1}^{173} (v_{kq}^{(i)} - \bar{v}_q^{(i)})(v_{kq}^{(j)} - \bar{v}_q^{(j)}) \right) \right], \text{ where } \bar{v}_q^{(i)} = \frac{\left(\sum_{k=1}^{173} v_{kq}^{(i)} \right)}{173}$$

is the sample mean of the q -th column of the matrix $\mathbf{V}_i := [v_{kq}^{(i)}]$ and Similarly for $\bar{v}_q^{(j)}$.

Estimation of the empirical covariance using Definition 6.2.3, avoids the information loss due to vectorization, as the i -th and j -th 173×2 -dimensional velocity matrices are not vectorized in order to compute the empirical covariance. Instead, covariance is estimated for each of the 2 velocity components observed at 173 different solar locations for i -th and j -th star, and thereby taking the mean of the covariances computed for the 2 velocity components.

Learning $\Sigma_3^{(173 \times 173)}$: Σ_3 is kernel parametrised using the SQE-looking non-parametric kernel presented in Chapter 4. The ij -th element $\sigma_{i,j}^{(3)}$ of the covariance matrix Σ_3 represents $cov(\mathbf{Y}_i, \mathbf{Y}_j)$, $\forall i, j = 1, \dots, 173$ where $\mathbf{Y}_i$ and $\mathbf{Y}_j$ are 50×2 -dimensional output matrix realised at each 173 different solar locations. $cov(\mathbf{Y}_i, \mathbf{Y}_j)$ is modelled in terms of the input using SQE-looking non-parametric kernel as:

$$\sigma_{i,j}^{(3)} = [cov(\mathbf{Y}_i, \mathbf{Y}_j)] = A_0 [\exp(-(x_i - x_j)^T \mathbf{Q}^{-1} (x_i - x_j))], \forall i, j = 1, \dots, 173,$$

where $\mathbf{Q}$ is a 2×2 square diagonal matrix, because $\mathbf{x}_i, \mathbf{x}_j$ are 2-dimensional vectors. The diagonal elements of $\mathbf{Q}$ are the length scales $2\ell_1^2$ and $2\ell_2^2$ pertaining to the radial location component X_1 and angular separation component X_2 of the input location vector $\mathbf{X}$. Therefore, the diagonal elements of the $\mathbf{Q}^{-1}$ will be $\frac{1}{2\ell_1^2}$ and $\frac{1}{2\ell_2^2}$. These length scales ℓ_1 and ℓ_2 are learnt from the data. A_0 is the global amplitude which avoids the need for learning multiple amplitudes pertaining to each length scales along each input dimension.

It is to be recalled that the ℓ_1 and ℓ_2 length scales pertain to the outer GP $\mathcal{GP}_{outer}$ which are in turn modelled as a random realisation of the GPs $\mathcal{GP}_1$ and $\mathcal{GP}_2$ respectively in the inner layer as,

$$\ell_1 = g_1(t), \ell_2 = g_2(t); \text{ where } g_1(\cdot) \sim \mathcal{GP}_1, g_2(\cdot) \sim \mathcal{GP}_2,$$

at each iteration $T = t$. The inner layer GPs $\mathcal{GP}_1$ and $\mathcal{GP}_2$ are zero-mean GP with kernel parametrised using SQE kernel as mentioned in Chapter 4 having length scales δ_1 and δ_2 . As the inner GPs are zero-mean GPs and the output of the lookback data is standardised, the amplitude hyperparameters are avoided in the inner layer.

6.2.4 Inference

Thus, the hyperparameters that are learnt using the non-parametric SQE-looking kernel are: $\sigma_{11}^{(1)}, \sigma_{22}^{(1)}, \sigma_{10}^{(1)}, \ell_1, \ell_2, \delta_1, \delta_2$ and A_0 . Among these hyperparameters, $\sigma_{11}^{(1)}, \sigma_{22}^{(1)}, \sigma_{10}^{(1)}, \ell_1, \ell_2$ and A_0 pertain to the outer GP $\mathcal{GP}_{outer,tensor}$ while δ_1 and δ_2 are the kernel hyperparameters of the inner GPs $\mathcal{GP}_1$ and $\mathcal{GP}_2$ respectively.

The inference scheme will be similar to the inference presented in Section 4.4.3 of Chapter 4. The only difference is that, all the kernel hyperparameters in the inference presented in Section 4.4.3 are modelled with inner GPs and their values are predicted at the end of the second block. Thus, the kernel hyperparameters ℓ_1, ℓ_2 are modelled using inner GPs $\mathcal{GP}_1$ and $\mathcal{GP}_2$ respectively and predicted at the end of the second block using the hyperparameters of the two inner GPs δ_1, δ_2 , where these predicted values of the ℓ_1, ℓ_2 become the current values of ℓ_1, ℓ_2 in the next iteration. The other parameters that we seek to learn in this high-dimensional application, are learnt directly from MCMC as stated above. Hence, parameters: $\sigma_{11}^{(1)}, \sigma_{22}^{(1)}, \sigma_{10}^{(1)}, A_0$ of the outer GP are only updated in the first block along with the ℓ_1, ℓ_2 , but no further updating will be undertaken for parameters $\sigma_{11}^{(1)}, \sigma_{22}^{(1)}, \sigma_{10}^{(1)}, A_0$ in the second block. The updated value of the parameters $\sigma_{11}^{(1)}, \sigma_{22}^{(1)}, \sigma_{10}^{(1)}, A_0$ in the first block, become the current value of these parameters in the next iteration. Only the values of δ_1 and δ_2 are updated within the second block in order to predict the values of ℓ_1 and ℓ_2 respectively with the $\mathcal{GP}_1$ and $\mathcal{GP}_2$ given their respective lookback data as training data.

Now, having the estimated mean tensor $\hat{\mathbf{T}}_\mu$ according to Remark 6.2.1, the likelihood of the covariance matrices of the tensor-variate GP $\mathcal{GP}_{outer,tensor}$ given the training data $\mathbf{D}_{train}$ following the Definition 6.2.1 is as follows:

$$\begin{aligned} \mathcal{L}(\sigma_{11}^{(1)}, \sigma_{22}^{(1)}, \sigma_{10}^{(1)}, \ell_1, \ell_2, A_0 | \mathbf{D}_{train}) &= (2\pi)^{-\frac{PNM}{2}} \left(|\mathbf{\Sigma}_1|^{-\frac{PNM}{2P}} |\mathbf{\Sigma}_2|^{-\frac{PNM}{2N}} |\mathbf{\Sigma}_3|^{-\frac{PNM}{2M}} \right) \\ &\times \exp\left(-\frac{\|(\mathbf{D}_Y - \hat{\mathbf{T}}_\mu) \times_1 \mathbf{C}_1^{-1} \times_2 \hat{\mathbf{C}}_2^{-1} \times_3 \mathbf{C}_3^{-1}\|^2}{2}\right), \end{aligned} \quad (6.2.4)$$

where $\mathbf{\Sigma}_i = \mathbf{C}_i \mathbf{C}_i^T, i = 1, 2, 3$ and the empirically estimated covariance $\hat{\mathbf{\Sigma}}_2 = \hat{\mathbf{C}}_2 \hat{\mathbf{C}}_2^T$. It is to be noted that the RHS of the Equation 6.2.4 of the likelihood uses the whole training data in order to learn the sough function $\mathbf{f}(\cdot)$.

Having defined the likelihood in the Equation 6.2.4, the joint posterior probability density of the unknown hyperparameters $\sigma_{11}^{(1)}, \sigma_{22}^{(1)}, \sigma_{10}^{(1)}, \ell_1, \ell_2, A_0$, given the training data $\mathbf{D}_{train}$, invoking the uninformed prior $\pi_0(\cdot)$ on the hyperparameters is as follows:

$$\begin{aligned} \pi(\sigma_{11}^{(1)}, \sigma_{22}^{(1)}, \sigma_{10}^{(1)}, \ell_1, \ell_2, A_0 | \mathbf{D}_{train}) &\propto \mathcal{L}(\sigma_{11}^{(1)}, \sigma_{22}^{(1)}, \sigma_{10}^{(1)}, \ell_1, \ell_2, A_0 | \mathbf{D}_{train}) \\ &\times \pi_0(\sigma_{11}^{(1)}) \pi_0(\sigma_{22}^{(1)}) \pi_0(\sigma_{10}^{(1)}) \pi_0(\ell_1) \pi_0(\ell_2) \pi_0(A_0) \end{aligned} \quad (6.2.5)$$

The likelihood of the δ_1, δ_2 of the inner GPs $\mathcal{GP}_1$ and $\mathcal{GP}_2$ are given in the Equation 4.4.5 of Chapter 4.

6.2.5 Computational issues

A computational burden associated with the modelling of GPs is the computation of the inverse of the covariance matrix Σ . If Σ is $K \times K$ -dimensional, then the computational complexity will be in the order of $O(K^3)$. The computational complexity is worrying when the covariance matrix is formed using the Kronecker product of multiple matrices. Now, the Kronecker product between $a \times b$ -dimensional matrix A and $c \times d$ -dimensional matrix B is a matrix C with dimension $ac \times bd$. Since the covariance matrix is a square matrix, so the Kronecker product between $a \times a$ -dimensional matrix A and $b \times b$ -dimensional matrix B , $C = A \otimes B$, is a matrix C with dimension $ab \times ab$. When, one needs to compute

$$D = C^{-1} = (A \otimes B)^{-1},$$

then the computational complexity for inversion of C will be in the order of $O((ab)^3)$. However, if the property of inverse of Kronecker product is applied, then

$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}, \quad (6.2.6)$$

so that the computational complexity for inversion of C will be in the order of $O(a^3 + b^3)$ which is less than $O((ab)^3)$ for $a, b > 0, a, b \in \mathbb{N}$.

Therefore, in the computation of inverse of Σ_4 where $\Sigma_4 = (\Sigma_1 \otimes \Sigma_2) \otimes \Sigma_3$ in the Definition 6.2.2 to yield the mean of the predictive value of output that is realised at given test input, the usage of the property presented in Equation 6.2.6 will reduce computational complexity to a great extent which will enable the use of the presented predictive methods with real-world small-to-medium sized high-dimensional data efficiently.

Other computational aspects to reduce computation and make the computation faster is as follows:

- Distinct elements of the covariance matrix is only computed during each iteration of the inference because covariance matrix is symmetrical.
- For numerical reasons, a very small amount of noise needs to be added to the diagonals of the covariance matrix during inference, which has very little effect on the statistical properties of the model that is being implemented. More details on this can be found in Section 4.1 of Neal (1998).
- Employing Cholesky decomposition, which is detailed in Section 3.3 of Thisted (1988), for computing inverse and logarithm of the determinant of the covariance matrix is an efficient way of implementation with numerical stability.
- In this high-dimensional application, I have computed the predictive mean and variance (using Equations 6.2.2 and 6.2.3 respectively) at test data only once, instead of computing in multiple iterations of the inference during post-burnin phase, with the mean values of the learnt hyperparameters from the converged part of the MCMC chain. The uncertainty of the predictions are given by the computation of the variance. This approach avoids a huge computational burden incurred by generating high-dimensional predictions in each iteration of the MCMC chain in the post-burnin phase.
- I have achieved further reduction in computational burden by computing the high-dimensional predictive means for all the test points in one go instead of generating predictions one by one at each test input points, thus avoiding extra computations.

6.3 Single layer GP based learning with high-dimensional data

For the purpose of comparison, I have also undertaken the learning $\mathbf{Y} = \mathbf{f}(\mathbf{X})$ with the same high-dimensional training data $\mathbf{D}_{train}$ using parametric kernel SQE in a single layer matrix-variate GP $\mathcal{GP}_{matrix}$ where the sought function $\mathbf{f}(\cdot)$ is a random realisation of the $\mathcal{GP}_{matrix}$ s.t.

$$\mathbf{f}(\cdot) \sim \mathcal{GP}_{matrix}(\mu(\mathbf{x}), cov(\mathbf{x}_i, \mathbf{x}_j)),$$

where $\mu(\cdot)$ and $cov(\cdot, \cdot)$ are mean and covariance functions of this matrix-variate GP $\mathcal{GP}_{matrix}$. Therefore, the joint probability density of the M realisations of the output values of $\mathbf{f}(\cdot)$ at M design inputs, is a 3rd-order tensor Normal distribution given in the Equation 6.2.1.

In this application, the covariance matrices $\Sigma_1, \Sigma_2, \Sigma_3$ are learnt similar to that presented in the section 6.2.3, except the difference is in the kernel parametrisation of the Σ_3 . Σ_3

is parametrised with the parametric SQE kernel whose length scale hyperparameters are constant i.e. the length scale hyperparameters of the SQE kernel will not be further modelled as random realisation of the inner GPs. Thus, the unknown hyperparameters of this model are: $\sigma_{11}^{(1)}, \sigma_{22}^{(1)}, \sigma_{10}^{(1)}, \ell_1, \ell_2, A_0$, that needs to be learnt given the data. The likelihood is used in this application is similar to that presented in the Equation 6.2.4, however, there is no second block in the inference, and all the unknown hyperparameters are updated in the single block of the MCMC chain.

6.4 Results

Now, I will present the results of the high-dimensional learning methodology using the non-parametric kernel that employs dual layered learning strategy with SQE-looking kernel and another single GP layer based model with SQE kernel mentioned earlier in this Chapter. With the non-parametric model, inference is performed using Metropolis-within-Gibbs while the single GP layer based model uses Random-Walk-Metropolis-Hastings based inference. For both the models, MCMC-based inference organically learns the marginal posterior probability density of each learnt hyperparameter, given the training data, resulting the learning of the 95% Highest Probability Density credible regions i.e. 95% HPDs. All the implementations are done in Python programming language (<https://www.python.org/>).

Results of inference with non-parametric model: In this implementation, there are two inner GPS, one is, $\mathcal{GP}_1$ that models the relation between iteration index T and ℓ_1 , another inner GP is $\mathcal{GP}_2$ which models the relation between T and ℓ_2 . Hyperparameters of the stationary kernel that model the covariance function of $\mathcal{GP}_1$ and $\mathcal{GP}_2$ are δ_1 and δ_2 respectively. Rest of the parameters have not been modelled using inner GPs. The 95% HPD credible region on the learnt hyperparameters are as follows.

1. $\sigma_{11}^{(1)} : [0.4577, 0.4939]$ with mean 0.4758
2. $\sigma_{22}^{(1)} : [0.1799, 0.2013]$ with mean 0.1906
3. $\sigma_{10}^{(1)} : [-0.0293, -0.0174]$ with mean -0.0233
4. $A_0 : [2.8727, 3.0111]$ with mean 2.9419

The 95% HPD credible regions on the learnt length scale hyperparameters ℓ_1, ℓ_2 in the first block of the chain are $[-0.0026, 0.0045]$ and $[9.9127, 9.9208]$ respectively with the mean of the 95% HPD credible regions being 0.0009, 9.9168 respectively. The 95%

HPD credible region on the learnt length scale hyperparameters δ_1 and δ_2 of the inner GPs are $[0.1796, 0.2201]$ and $[0.1996, 0.3941]$ respectively with mean of the 95% HPD credible region being 0.1998, 0.2969 respectively. The mean of the 95% HPD credible region of the ℓ_1, ℓ_2 values predicted at the end of second block are 0.00087, 9.9169 respectively.

This MCMC chain is run for 300,000 iterations. The variance of the Normal proposal densities for $\sigma_{11}^{(1)}, \sigma_{22}^{(1)}, \sigma_{10}^{(1)}, \ell_1, \ell_2, \delta_1, \delta_2$ is 0.002^2 . Where as, the variance of the Normal proposal densities for A_0 is 0.01^2 . Normal priors are used for all parameters, VARIANCES OF which are chosen experimentally. Prior mean for $\sigma_{11}^{(1)}$ is 0.4 with variance 0.01^2 , $\sigma_{22}^{(1)}$ is 0.38 with variance 0.01^2 , $\sigma_{10}^{(1)}$ is -0.02 with variance 0.01^2 , ℓ_1 is 0.001 with variance 0.01^2 , ℓ_2 is 10.15 with variance 0.01^2 , δ_1 is 0.2 with variance 0.01^2 , δ_2 is 0.3 with variance 0.05^2 , A_0 is 1 with variances 1^2 . Seed for $\sigma_{11}^{(1)}, \sigma_{22}^{(1)}, \sigma_{10}^{(1)}, \ell_1, \ell_2, \delta_1, \delta_2, A_0$ are 0.2, 0.2, -0.02 , 0.02, 9.95, 0.2, 0.1, 2 respectively. The size of the lookback data N_{LB} is set to 500 in this MCMC chain. Using traces of hyperparameters. the final 1,000,00 iterations are used to compute the mean value of parameters that are used to generate the high-dimensional predictions at the given vector-variate test inputs using the outer GP. Such values for the parameters $\sigma_{11}^{(1)}, \sigma_{22}^{(1)}, \sigma_{10}^{(1)}, \ell_1, \ell_2, A_0$ are 0.4755, 0.1905, -0.0233 , 0.0009, 9.9159, 2.9411 respectively.

The extreme left panel of the Figure 6.13 shows the heat map of the estimated mean tensor $\hat{\mathbf{T}}_\mu$ that I have computed using Remark 6.2.1 to be used within the likelihood-computation of the inference. Figure 6.5 depicts the results of learning with the non-parametric model. Said results include traces of all parameters, namely, of $\sigma_{11}^{(1)}, \sigma_{22}^{(1)}, \sigma_{10}^{(1)}, \ell_1, \ell_2, \delta_1, \delta_2, A_0$ along with values of ℓ_1, ℓ_2 that are predicted at the end of the second block of the inference. The logarithm of the joint posterior probability density given the training data, and the logarithm of the marginal posterior probability density of each learnt hyperparameter of the inner GP, (given the respective lookback dataset), are shown in the Figure 6.6. Figure 6.7 presents the convergence quality of the learning of parameters, where values of a parameter are sampled over three distinct, equally-wide non-overlapping iteration windows (covering 30000 consecutive iterations starting with the $200000 - th, 230000 - th, 260000 - th$ iterations), during the MCMC chain. The three histograms are coloured in blue, red and black, and overplotted. I will discuss the predictive results with this non-parametric model using SQE-looking kernel along with the comparisons with the other single GP layer based model considered in this chapter in the subsection 6.4.1.

Effect of size of lookback data: The choice of the size of the lookback data N_{LB} is an important aspect to consider while performing the learning. In this work I have experimented with various values of N_{LB} with all such values retained as $\sim 10^2$, owing to the presence of high inhomogeneities in the correlation structure of the astronomical data that I am using.

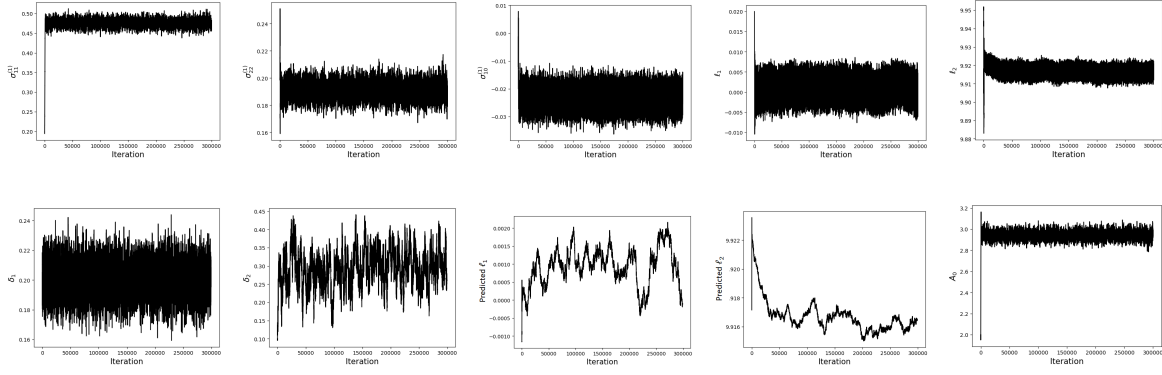


Fig. 6.5 Results of learning all the unknown hyperparameters using the non-parametric model with SQE-looking kernel, given high-dimensional astronomical data. The traces from extreme left in the top row pertain to, $\sigma_{11}^{(1)}, \sigma_{22}^{(1)}, \sigma_{10}^{(1)}, \ell_1$ updated in the first block, ℓ_2 updated in the first block given training data. The traces from extreme left in the bottom row pertain to, δ_1 and δ_2 , given relevant (dynamically-varying) lookback data, ℓ_1 value predicted from the $\mathcal{GP}_1$ in the inner-layer, ℓ_2 value predicted from the $\mathcal{GP}_2$ in the inner-layer, A_0 given training data.

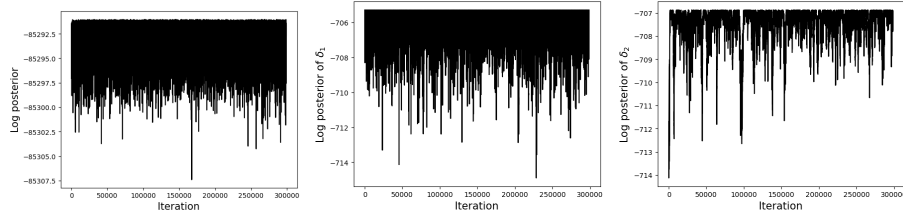


Fig. 6.6 The left panel shows the logarithm of the joint posterior probability density given the high-dimensional astronomical training data considered in this application using the non-parametric model. The middle and the right panel present the logarithm of the marginal posterior probability density of δ_1 and δ_2 of the inner GPs respectively given their relevant lookback data.

Table 6.1 shows the results of learning of the hyperparameters using three different of the lookback window sizes: $N_{LB} = 300, 400, 500$, along with the execution time for running the chains from 300000 iterations on a laptop with configuration mentioned in Section 4.7.11 of Chapter 4. Figure 6.8 shows the traces of the length scales ℓ_1, ℓ_2 with various values set to N_{LB} i.e. $N_{LB} = 300$ is set in one MCMC chain (traces shown in the left most column), then in another chain $N_{LB} = 400$ (traces shown in the middle column) and the third chain takes $N_{LB} = 500$ (traces shown in the right most column).

It can be seen that with higher N_{LB} , the traces show better mixing of the samples with less number of peaks and with lower uncertainty on the learnt value of the hyperparameters. I will now present an explanation of this interesting behaviour of the traces. Different values

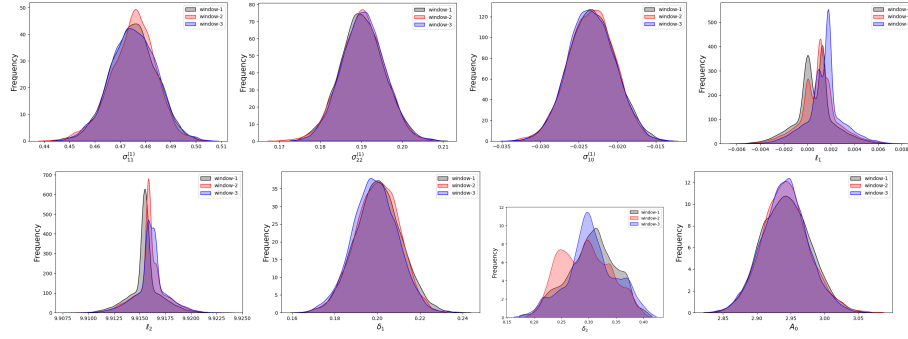


Fig. 6.7 Convergence diagnostic of the learning of the hyperparameters using the non-parametric model with SQE-looking kernel given the high-dimensional data where the values of the hyperparameter are sampled over three distinct, equally-wide non-overlapping iteration windows (covering 30000 consecutive iterations starting at the 200000 – th , 230000 – th , 260000 – th iterations), during the MCMC chain. The three histograms are coloured in blue, red and black, and overplotted. From extreme left in the top row, the panels pertain to hyperparameters: $\sigma_{11}^{(1)}$, $\sigma_{22}^{(1)}$, $\sigma_{10}^{(1)}$, ℓ_1 . From extreme left in the bottom row, the panels pertain to hyperparameters: ℓ_2 , δ_1 , δ_2 , A_0 .

of these length scale hyperparameters imply different forms of the sample functions drawn from the outer GP. Now, the correlation structure of the outer GP captures the true correlation structure of the empirical data by using the non-parametric, non-stationary kernel (from Chapter 4), the more inhomogeneous the correlation structure is, the more diverse forms will be seen in the sample functions drawn from the outer GP. The diverse forms of the sample functions will reflect different peaks present in the empirical data that gives rise to the inhomogeneous correlation structure of the data. Now, each of such different kinds of sample functions will correspond to different values of the length scales, therefore, in general, the trace of length scales will be very wiggly (which can be seen in the traces obtained using the single GP layer based model as well, presented in Figure 6.9).

However, in the non-parametric learning model, the length scales are learnt with the inner GPs using the lookback data. This lookback data is built with the values from the past N_{LB} iterations, therefore, similar to the distributed lag models (Almon, 1965) where the current value is presented by the moving average of the past values, the value of the length scale updated in the current iteration will be represented by the moving average of the values from past N_{LB} iterations. So, for a wiggly trace (due to the higher inhomogeneities present in the correlation structure of the empirical data), the higher the size N_{LB} of the lookback data, the moving average of the past N_{LB} number of values of the length scales, will adjust itself while updating the value of the length scale in the current iteration, so that after certain iterations,

the wiggly-ness of the trace reduces, implying that the learning of the hyperparameter is no longer affected by the inhomogeneities present in the empirical data.

Parameters & time	$N_{LB} = 300$	$N_{LB} = 400$	$N_{LB} = 500$
$\sigma_{11}^{(1)}$	[0.4579, 0.4941]	[0.4576, 0.4934]	[0.4577, 0.4939]
$\sigma_{22}^{(1)}$	[0.1801, 0.2014]	[0.18, 0.2011]	[0.1799, 0.2013]
$\sigma_{10}^{(1)}$	[-0.0292, -0.0173]	[-0.0294, -0.0174]	[-0.0293, -0.0174]
ℓ_1	[-0.0026, 0.0046]	[-0.0029, 0.0044]	[-0.0026, 0.0045]
ℓ_2	[9.9088, 9.9208]	[9.9124, 9.9197]	[9.9127, 9.9208]
δ_1	[0.1797, 0.2202]	[0.1796, 0.22]	[0.179570330.2201]
δ_2	[0.2003, 0.395]	[0.1976, 0.3939]	[0.1996, 0.3941]
A_0	[2.8719, 3.0094]	[2.8727, 3.009]	[2.8727, 3.011]
Time (hr:min:sec)	7:20:53	9:17:30	12:43:24

Table 6.1 95% HPD credible region on the learnt hyperparameters using non-parametric model with SQE-looking kernel having various lookback size.

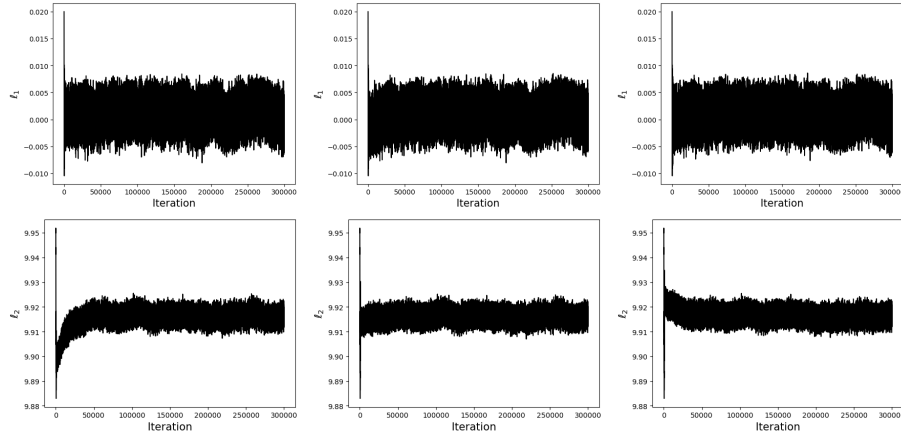


Fig. 6.8 Traces of the length scales ℓ_1, ℓ_2 with various lookback size N_{LB} . The top and bottom rows depict traces of ℓ_1 and ℓ_2 respectively. Traces shown in the left most, middle and right columns pertain to chains with $N_{LB} = 300$, $N_{LB} = 400$ and $N_{LB} = 500$ respectively. These chains are run using the non-parametric model with SQE-looking kernel for 300000 iterations.

Results of inference with single GP layer based model: Now, I will present the results of the learning with the single GP layer based model using SQE kernel, given high-dimensional astronomical data. The 95% HPD credible region on the learnt hyperparameters $\sigma_{11}^{(1)}, \sigma_{22}^{(1)}, \sigma_{10}^{(1)}, \ell_1, \ell_2, A_0$ are [0.4595, 0.4959], [0.1787, 0.2008], [-0.0344, -0.0219], [-0.0059, 0.0081], [11.6539, 11.6932], [4.3702, 4.5962] respectively with the mean of 95%

HPD credible region on the learnt hyperparameters are 0.4777, 0.1897, -0.0282 , 0.0011, 11.6736 respectively.

This MCMC chain is run for 125000 iterations. The variances of the Normal proposal densities for $\sigma_{11}^{(1)}, \sigma_{22}^{(1)}, \sigma_{10}^{(1)}, \ell_1, \ell_2, A_0$ are $0.002^2, 0.002^2, 0.002^2, 0.0004^2, 0.01^2, 0.01^2$ respectively. Normal priors are used for all the unknown hyperparameters which are uninformative and chosen experimentally. Prior mean for $\sigma_{11}^{(1)}$ is 0.4 with variance 0.01^2 , $\sigma_{22}^{(1)}$ is 0.38 with variance 0.01^2 , $\sigma_{10}^{(1)}$ is -0.02 with variance 0.01^2 , ℓ_1 is 0.008 with variance 0.01^2 , ℓ_2 is 12 with variance 0.01^2 , A_0 is 1 with variances 1^2 . Seed for $\sigma_{11}^{(1)}, \sigma_{22}^{(1)}, \sigma_{10}^{(1)}, \ell_1, \ell_2, A_0$ are 0.2, 0.2, -0.02 , 0.002, 12, 2 respectively. From the traces of the parameters, iterations onwards of the 75000-th, are used to compute the mean value of the parameters, which is then used to generate the high-dimensional predictions. These mean values for the parameters $\sigma_{11}^{(1)}, \sigma_{22}^{(1)}, \sigma_{10}^{(1)}, \ell_1, \ell_2, A_0$ are 0.4784, 0.1903, -0.0281 , 0.0021, 11.6736, 4.4753 respectively.

Figure 6.9 depicts the results of learning with the non-parametric model, including traces of all the parameters $\sigma_{11}^{(1)}, \sigma_{22}^{(1)}, \sigma_{10}^{(1)}, \ell_1, \ell_2, A_0$ along with the logarithm of the joint posterior probability density given the training data. Figure 6.10 presents the convergence diagnostic of the learning of the hyperparameters where the values of the hyperparameter are sampled over three distinct, equally-wide non-overlapping iteration windows, (of width 16000 iterations, starting at the $75000 - th, 91000 - th, 107000 - th$ iterations), during the MCMC chain. The three histograms are coloured in blue, red and black, and overplotted. As mentioned earlier, the predictive results are discussed in subsection 6.4.1 along with relevant analysis.

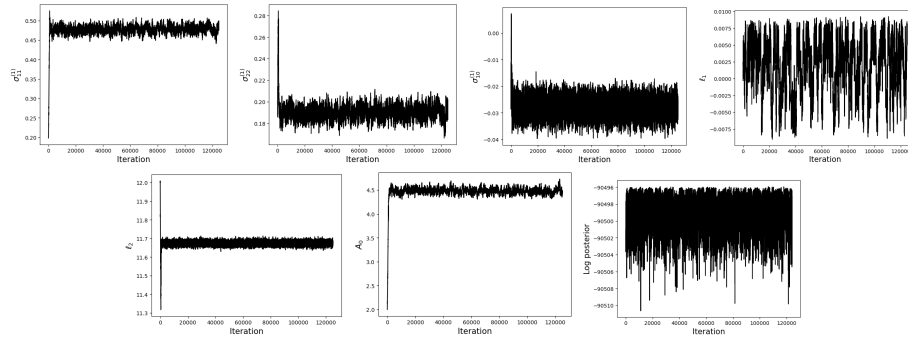


Fig. 6.9 Results of learning all parameters using the single GP layer based model with SQE kernel, given high-dimensional astronomical data. The traces from extreme left in the top row pertain to, $\sigma_{11}^{(1)}, \sigma_{22}^{(1)}, \sigma_{10}^{(1)}, \ell_1$ given training data. The traces from extreme left in the bottom row pertain to, ℓ_2, A_0 and the logarithm of joint posterior given training data.

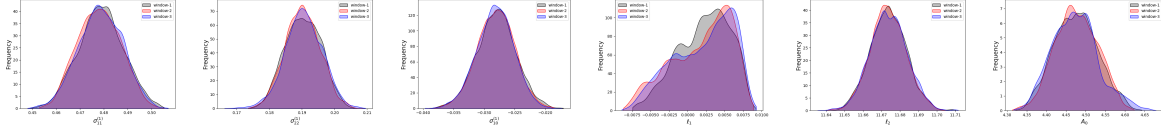


Fig. 6.10 Convergence quality of the learning of the hyperparameters using the single GP layer based model with SQE kernel where the values of the hyperparameter are sampled over three distinct, equally-wide, non-overlapping iteration windows (of width 16000 iterations, starting at the $75000 - th$, $91000 - th$, $107000 - th$ iterations), during the MCMC chain. The three histograms are coloured in blue, red and black, and overplotted. From extreme left, the panels pertain to hyperparameters: $\sigma_{11}^{(1)}$, $\sigma_{22}^{(1)}$, $\sigma_{10}^{(1)}$, ℓ_1 , ℓ_2 , A_0 .

6.4.1 High-dimensional prediction with comparison and analysis

In high-dimensional situations, especially when the predicted output is matrix-variate at the vector-variate test input, depiction of the predictive performance is quite challenging. A scalar summary of predictive performance, such as mean squared error (MSE) or root mean square error (RMSE), does not convey the quality of prediction correctly (Roy et al., 2023b). Prediction quality at test inputs in $[a_i, b_i]$ of the input space $[a, b]$ can be inherently different from that in another sub-volume of the input space, owing to inhomogeneities in the data, but this difference is not properly addressed by a scalar measure of predictive quality. Basically, it may happen that the correlation structure of the observed outputs that are realised in $[a_i, b_i] \subset [a, b]$ of input space, bears more inhomogeneity than the subspace $[a_j, b_j] \subset [a, b]$, $i \neq j$. Then the performance measure should be able to appreciate if a model m_1 performs better in the subspace $[a_i, b_i]$ than another model m_2 which performs better in the subspace $[a_j, b_j]$ but worse in $[a_i, b_i]$ or a model m_3 that performs consistently poor throughout the input space $[a, b]$. Therefore, the predictive measure should be able to highlight the superiority of the model that performs better in that subspace of input space, where predictions are harder to make, compared to other subspaces.

Since the data considered in this application bears varying inhomogeneity over different subspace of input space which is also a phenomenon of real-world data, I have considered two ways of depicting predictive performance given the limitations of scalar summary of predictions discussed above - one is the MSE and another is a compliance score. The compliance measure is a simple binary variable that takes *true* value if the truth lies within the predicted interval. The compliance score is then the number of test inputs at which the truth lies with the interval of the uncertainty-included prediction, out of the total number of test inputs. Thus the compliance score is also a scalar summary of predictions that includes

the correctness and comprehensiveness of the computation of the uncertainty of the prediction which is also a crucial aspect of the learning.

Figure 6.11 and Figure 6.12 present the heat maps of the compliance measure for the predicted output values at each of the 43 test inputs with the non-parametric model using SQE-looking kernel and single layer GP based model respectively. As the output at a given test input is a 50×2 -dimensional matrix, the Figure 6.11 and 6.12 consist of 43 50×2 -dimensional matrix where each of the matrices shows the compliance as a binary value in each of its cell. The binary compliance value is colour coded such that the *true* value is depicted in black, where as the *false* value (i.e. the truth lies outside of the predicted interval) is shown in white. For both of these figures, it is seen that the non-parametric model performs better than the single layer GP based model.

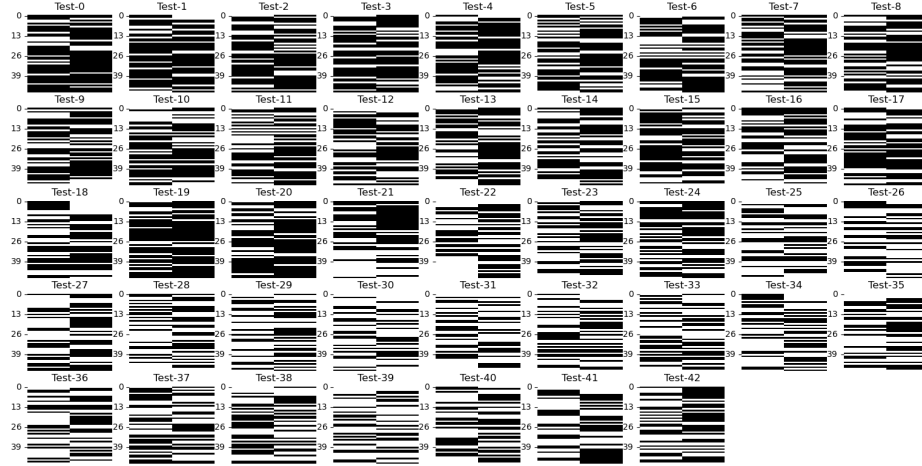


Fig. 6.11 Heat maps of the compliance measure pertaining to the 50×2 -dimensional output matrix predicted at each of the 43 test input using the non-parametric model with SQE-looking kernel, given the training data. Compliance value *true* is depicted in black which means truth lies within the predicted interval and *false* value is shown in white (when truth lies outside of the predicted interval).

Table 6.2 depicts the comparative performance measure in terms of MSE with the two models computed for the two velocity components U and V for each star over all the test points, as each velocity component of a star is a random variable whose value is realised as output at a given test solar location. Here, the two models are: the non-parametric model with SQE-looking kernel and the single layer GP based model with SQE kernel. MSE has been computed with the predicted mean against the true values. From the Table 6.2, it can be seen that the non-parametric model outperforms the single layer GP based model. If I

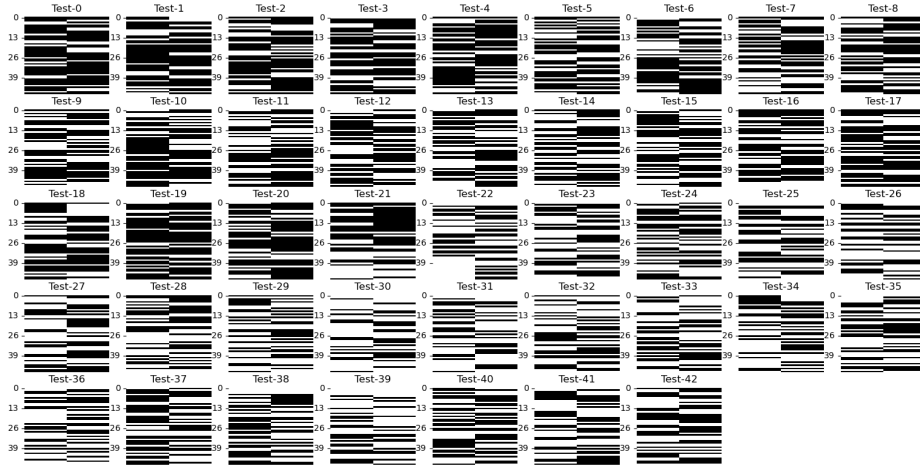


Fig. 6.12 Heat maps of the compliance measure pertaining to the 50×2 -dimensional output matrix predicted at each of the 43 test input using the single layer GP based model with SQE kernel, given the training data. Compliance value *true* is depicted in black which means truth lies within the predicted interval and *false* value is shown in white (when truth lies outside of the predicted interval).

compare the predictive performance between the two velocity components, then it can be seen that overall, the predictions for the V component are better than the predictions for the U component, which is driven by the fact that the data pertaining to the U component is more inhomogeneously correlated than the other velocity component V . Table 6.3 presents the total compliance score obtained from the two velocity components U and V for each test solar locations. The compliance score is presented in %. In this table, it is seen that the non-parametric model outperforms the single layer GP based model.

For better visual representation, Figure 6.13 presents the heat maps of the MSE values depicted in the Table 6.2 pertaining to the non-parametric model, single layer GP based model and the compliance score shown in the Table 6.3.

Now I will present some of the predictive results in detail. In order to do so, I will present the plot for prediction of each velocity component of the stars at the 43 test solar locations. In such plot, the true values are shown in blue circles while the predictive means are shown in red stars with the computed uncertainty in terms of the variance of the prediction are plotted in both side of the predicted mean in red straight lines. Title of each such plot depicts the n -th star where $n \in (0, \dots, 49)$ (as there are 50 stars) for which the predictions are plotted; the values pertaining to the inhomogeneity present in the corresponding output training data i.e. 173-sized vector for each velocity component of each star and the MSE value computed

Table 6.2 Comparative predictive performance using MSE (over all the predicted values at the test points) for each star with the two models.

Non-parametric		Single GP	
U	V	U	V
2880.93	921.406	3681.142	1603.631
3087.296	1030.88	4219.416	2137.26
1608.436	1039.067	2137.475	1647.755
2123.006	1298.036	2654.052	2263.14
1955.575	835.1868	2799.249	1449.005
1219.381	1614.961	1684.109	2952.472
1628.46	1259.888	2266.558	2494.485
2382.469	1152.554	3348.429	2145.761
1895.264	610.3427	2494.566	1172.122
1664.171	1647.262	2346.263	2844.195
2428.876	982.6708	3612.444	1638.532
2560.631	1031.504	3740.257	2209.243
1249.781	1003.528	1924.852	1812.532
1940.516	1054.187	2536.619	1745.792
2037.585	808.0742	2993.442	1683.174
1891.667	1133.96	3033.047	2221.361
1832.225	867.1408	2296.277	1285.4
1718.019	898.4821	2361.518	1576.722
1351.821	1054.499	1806.28	1622.232
1933.928	1217.131	3236.796	2119.496
1210.93	786.5541	2040.957	1383.277
1126.225	1056.643	1553.677	1865.856
1733.318	1183.232	3263.839	2064.884
1905.805	799.8285	2545.448	1208.935
2310.584	1468.545	3263.126	2678.562
2049.092	1258.189	2749.707	2199.787
1553.849	1494.002	2393.895	2420.621
1551.629	1185.631	2251.811	1687.004
1939.561	1134.602	2781.395	2105.491
1564.171	1094.513	2230.54	1677.434
1414.542	1060.045	2326.271	1803.293
2557.497	1050.891	3822.793	1666.41
1769.267	992.7757	2478.977	1621.467
1201.634	1206.355	1959.71	2134.083
1953.984	1265.509	2748.485	2368.278
1639.341	1363.134	2401.507	2502.056
1404.816	1164.808	2206.739	1889.81
2083.297	1006.132	3028.243	1632.643
816.9755	1285.157	1126.152	2346.162
1346.633	1024.619	2030.034	1729.535
1284.169	1326.534	2005.808	2170.701
1504.086	1137.627	2074.242	1774.938
1350.574	1037.484	1995.245	2075.678
1149.402	1427.822	1858.115	2492.823
1011.754	1284.466	1489.191	2281.418
1899.622	1343.541	2789.48	2448.208
2147.442	2123.958	3261.033	3900.503
2400.183	793.443	3626.976	1256.253
1731.222	977.625	2450.853	1762.462
2747.112	1044.421	3539.487	1642.253

Table 6.3 Comparative predictive performance using compliance score (in %) for each test data with the two models.

Non-parametric	Single GP
76	73
73	69
70	67
72	67
70	69
65	59
65	64
63	57
65	62
62	60
61	67
59	58
58	59
61	56
59	56
66	58
59	63
65	66
57	62
77	76
73	70
55	59
50	44
53	47
64	52
43	47
41	42
49	47
44	48
44	45
36	33
39	39
46	46
45	45
43	44
47	48
43	41
52	57
48	51
36	36
47	54
47	48
55	53

out of the predictions generated at the 43 test solar locations. I will include predictive plots of a subset S of ten stars having index 2, 4, 9, 13, 21, 26, 30, 33, 34, 38.

Figure 6.14 presents the prediction of the U component pertaining to the subset S of stars where the predictions are generated using the non-parametric model with SQE-looking

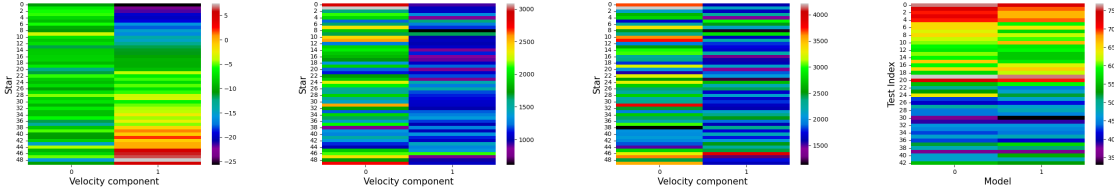


Fig. 6.13 Heat maps that represent, from extreme left: estimated mean tensor $\hat{T}_\mu$ computed using Remark 6.2.1 to be used within the likelihood-computation of the inference; the MSE values depicted in the Table 6.2 pertaining to the non-parametric model; the MSE values presented in the Table 6.2 pertaining to single layer GP based model; the compliance score shown in the Table 6.3 pertaining to the non-parametric model (shown in 0-th column) and single layer GP based model (shown in 1-th column).

kernel. Similarly Figure 6.15 presents the prediction of U component pertaining to the subset of stars S , where the predictions are generated using the single layer GP based model with SQE kernel. It is seen that the quality of prediction is better with the non-parametric model including the uncertainty that is being predicted. The width of the uncertainty predicted using the non-parametric model is in general lower than the single layer GP based model which shows more confident prediction that is yielded by the non-parametric model in spite of the presence of various amount of inhomogeneities in the correlation structure. Figure 6.16 and Figure 6.17 presents the predictive results of the V component of a subset (S) of stars, where the predictions are generated using the non-parametric model and the single layer GP based model respectively.

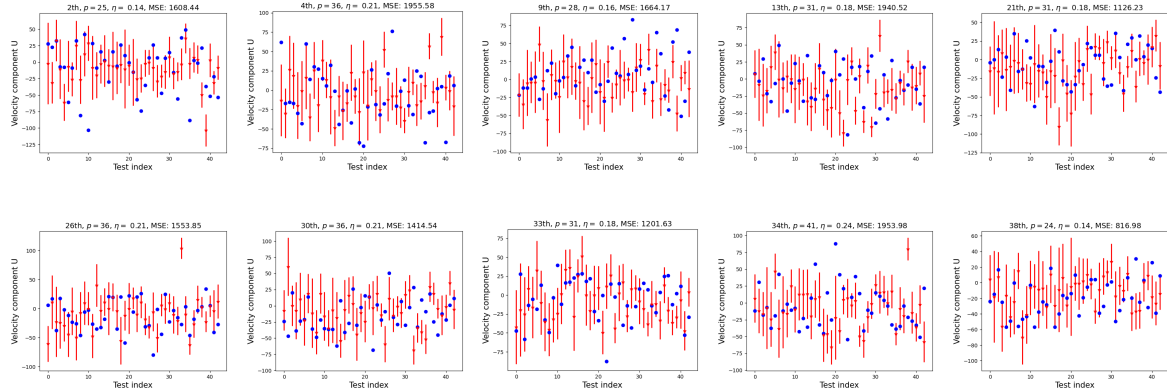


Fig. 6.14 Predictive results of the velocity component U pertaining to the subset of stars S , where the predictions are made with the non-parametric model using SQE-looking kernel, given the training data. True values are shown in blue circles while the predictive means are shown in red stars with the uncertainty plotted as red line along the both side of the predicted mean.

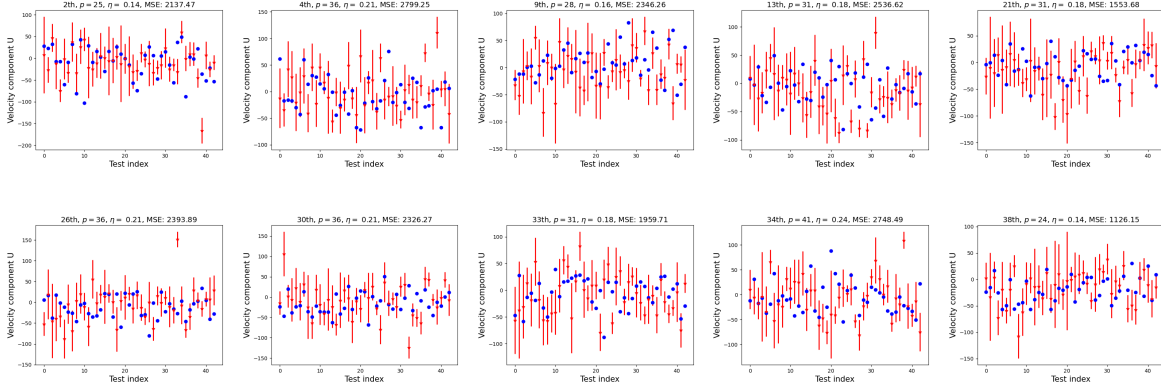


Fig. 6.15 Predictive results of the velocity component U pertaining to the subset of stars S , where the predictions are made with the single layer GP based model using SQE kernel, given the training data. True values are shown in blue circles while the predictive means are shown in red stars with the uncertainty plotted as red line along the both side of the predicted mean.

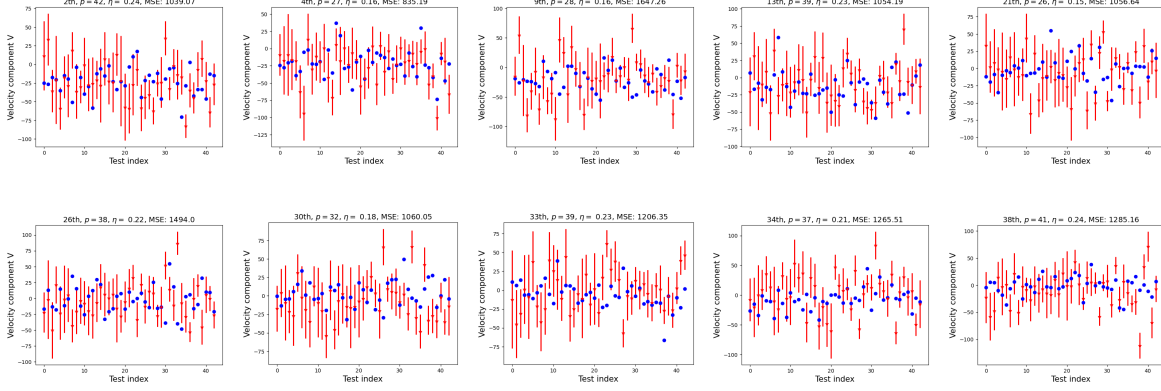


Fig. 6.16 Predictive results of the velocity component V of the subset S of stars, where the predictions are made with the non-parametric model using SQE-looking kernel, given the training data. True values are shown in blue circles while the predictive means are shown in red stars with the uncertainty plotted as red line along the both side of the predicted mean.

Explainability of prediction quality using inhomogeneity in correlation structure: I will now present some analytical plots that will help explain the quality of predictions and demonstrate the superiority of the non-parametric model over the single layer GP based model in the presence of various types and amount of inhomogeneities in the correlation structure. To begin with, I will first present the distribution of the index of the training and test data points in the whole dataset in the Figure 6.18 so that the plots created against training index and test index can be useful in understanding the effect of inhomogeneities present in the subset(s) of the input space.

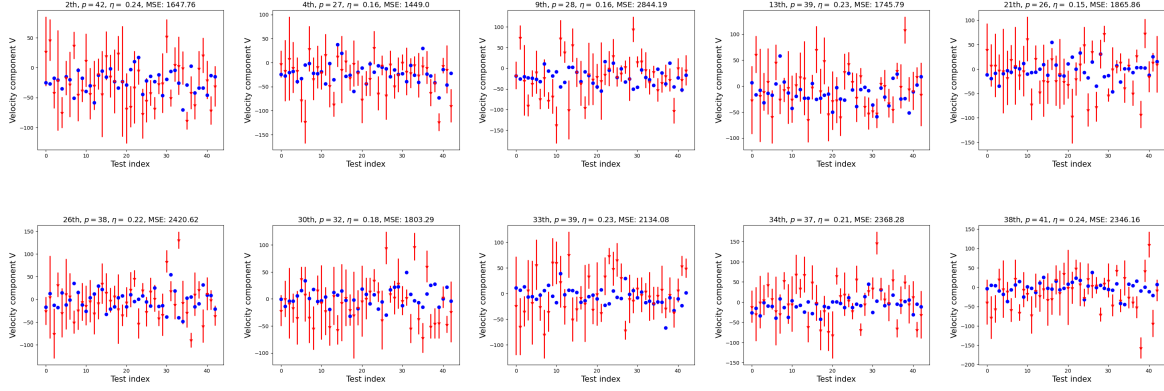


Fig. 6.17 Predictive results of the velocity component V pertaining to the subset S of stars, where the predictions are made with the single layer GP based model using SQE kernel, given the training data. True values are shown in blue circles while the predictive means are shown in red stars with the uncertainty plotted as red line along the both side of the predicted mean.

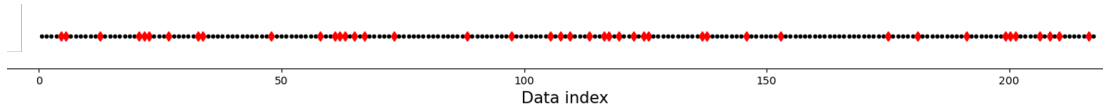


Fig. 6.18 Train-test split (~ 80 -20) in the high-dimensional astronomical data obtained from Chakrabarty (2007) that consists of 216 number of data points. The black dots represents the index of 173 training points and the red diamonds shows the index of 43 test points.

Figure 6.19 shows prediction of the U components given the training data, pertaining to the four stars having index 19, 24, 42 and 49, where the predictions are unsatisfactory for the test inputs in particular subspace of the input. In the predictive plots, the true values are shown in blue circles while the predictive means are shown in red stars with the uncertainty plotted as red line along both sides of the predicted mean. The Figure 6.19 also shows the trivial function plotted corresponding to each such star and the L values computed from those trivial functions. The top row in the Figure 6.19 depicts the predictions along with true value. The middle and bottom rows depict the corresponding trivial function and the L values. These plots indicate that higher the inhomogeneity in the correlation structure within the subspace of the input, harder the prediction at the test inputs within that subspace of input and hence predictions can be bad within that input region. It is to be noted, that the presence of the higher amount of inhomogeneities over one or multiple subspaces of the input space is not explicitly manifest in the inhomogeneity parameter η . η is an important parameter to compare inhomogeneities present in the correlation structure across datasets. However, inspection of the L values along with its dispersion would reveal that the L value(s) that are very high, compared to the mean of all the L values, are more dispersed from the others,

indicating the presence of higher inhomogeneities in the correlation structure in that region of the input space. where the predictions are worse with respect to the other input regions - with lower inhomogeneity in its correlation structure. Now, I will explain the plots of the Figure 6.19 in detail as follows.

1. Consider the left most column in the Figure 6.19 which depicts plots relevant to the 19-th star, where the predictions are visibly bad in the centre region. However, a closer look at the corresponding trivial function and the L values in the plot pertaining to the extreme left panel of the middle and the bottom row, reveal that the central region of the input space is highly inhomogeneously correlated with respect to other regions and accordingly the quality of the predictions is worse than other input regions.
2. Consider the second from the left column in the Figure 6.19 which depicts results relevant to the 24-th star, where the quality of the predictions are good in the central region. From the plots of trivial function and the L values, it is seen that the inhomogeneities in the correlation structure lie in the two sides of the central region, explaining the dis-satisfactory quality of predictions in the both sides of the central region of the input space, especially the right side.
3. The third column from the left in the Figure 6.19 depicts plots relevant to the 42-nd star, where the quality of prediction in the right half of the input space is worse than the left half of the input space. The plots of trivial function and the L values indicate that the right half of the input space is more inhomogeneously correlated that the left half of the input space.
4. The right most column of the Figure 6.19 depicts plots relevant to the 49-th star, where the quality of prediction is dis-satisfactory all over the input space. The plots of trivial function and the L values show that there exist multiple subspace of the input that are spread across the input space, that bear highly inhomogeneous correlation structure.

Figure 6.20 presents the prediction of U components of four stars - with index 19, 24, 42 and 49 - where the predictions are made with the single layer GP based model using SQE kernel, given the training data. Comparing the predictions of U components in Figure 6.19 and Figure 6.20 of the same set of stars, generated using the non-parametric model and the single layer GP based model, it can be said that the non-parametric models is superior than the other model, especially in the subspace of input that is highly inhomogeneously correlated and where predictions are hard to make.

Figure 6.21 presents the trivial function and the L values pertaining to U component of the subset S of ten stars having index 2, 4, 9, 13, 21, 26, 30, 33, 34, 38 - predictive results for

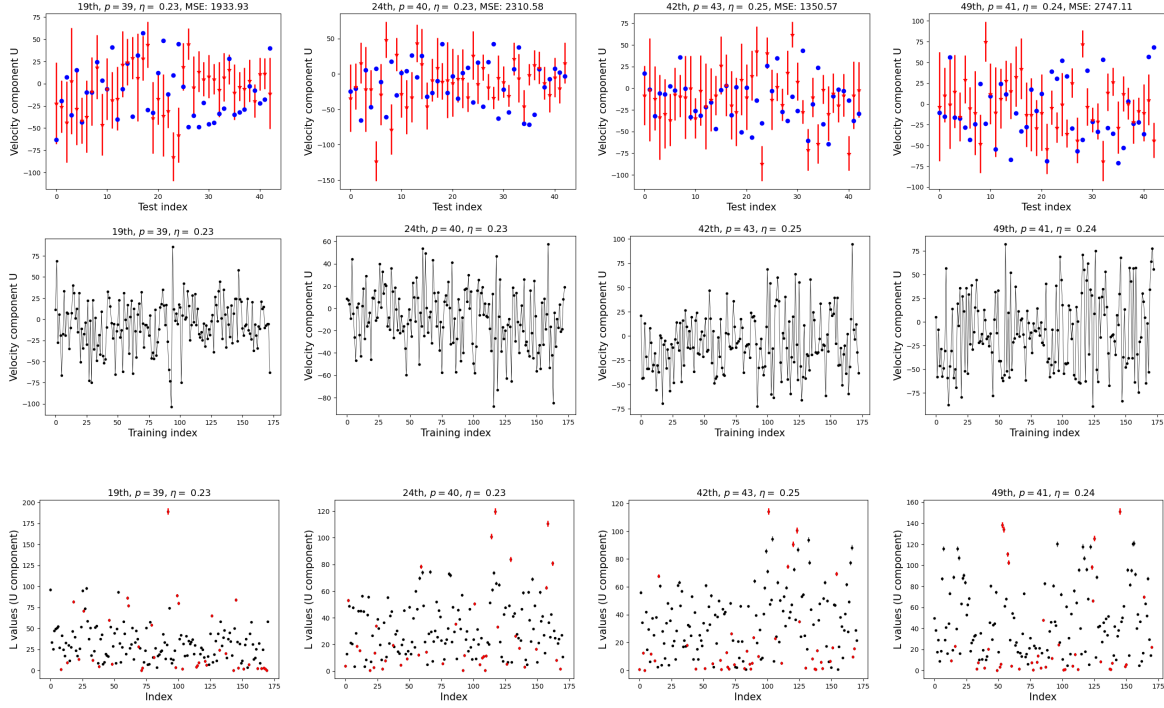


Fig. 6.19 Each column presents the prediction of U components (in top row), trivial function (in middle row) and the L values (in bottom row) pertaining to four stars having index 19, 24, 42 and 49; where the predictions are made with the non-parametric model using the SQE-looking kernel, given the training data. The true values are shown in blue circles while the predictive means are shown in red stars with the uncertainty plotted as red line along the both side of the predicted mean.

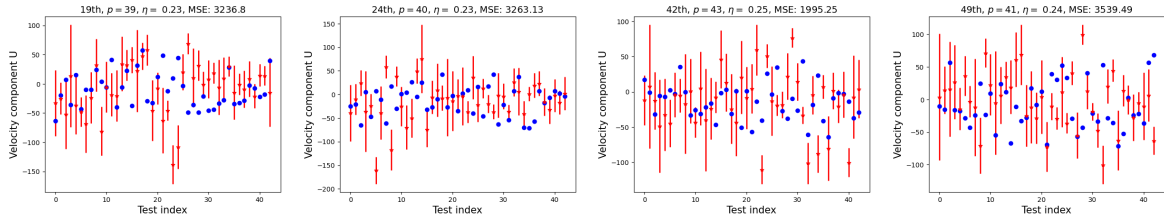


Fig. 6.20 Prediction of U components pertaining to four stars having index 19, 24, 42 and 49 where the predictions are made with the single layer GP based model using SQE kernel, given the training data. The true values are shown in blue circles while the predictive means are shown in red stars with the uncertainty plotted as red line along the both side of the predicted mean.

these which are obtained with the two models (non-parametric and single layer GP based model) are presented in Figure 6.14 and Figure 6.15 respectively. Similarly, Figure 6.22 presents the trivial function and the L values pertaining to the V component of the subset S of stars, predictive results for which using the two models (non-parametric and single layer

GP based model) are given in the Figure 6.16 and Figure 6.17 respectively. These figures are for further supporting the quality and Explainability of the predictions.

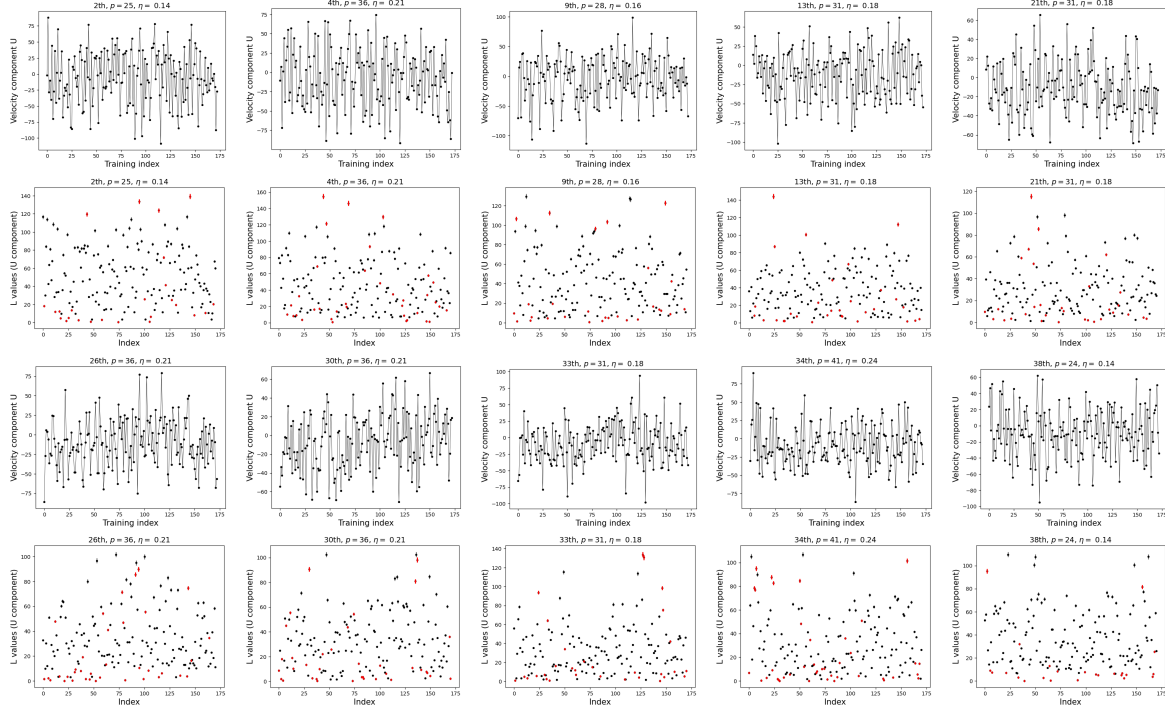


Fig. 6.21 Trivial function and L values pertaining to the U component of the subset S of ten stars having index 2, 4, 9, 13, 21, 26, 30, 33, 34, 38; whose predictive results with the presented two models given the training data are presented in Figure 6.14 and Figure 6.15.

6.5 Conclusion

In this chapter, I have presented the high-dimensional application of the non-parametric kernel developed in Chapter 4. In this setup, I have learnt the functional relationship between vector-variate input variable and matrix-variate output variable in a fully Bayesian way. As, in Bayesian paradigm, an unknown entity is considered as random, therefore, the unknown sought function here has been modelled as a random function. A random function will be distributed according to a probability distribution, but a probability distribution over space of a random function is a stochastic process. Thus, the sought function is modelled as a random sample drawn from stochastic process. I have chosen this stochastic process to be a GP and I have presented the justification for such choice in earlier chapter of this thesis.

In this high-dimensionality setup, the joint probability density of the outputs realised at the design inputs will be the tensor Normal density (in lower dimension, the joint is a

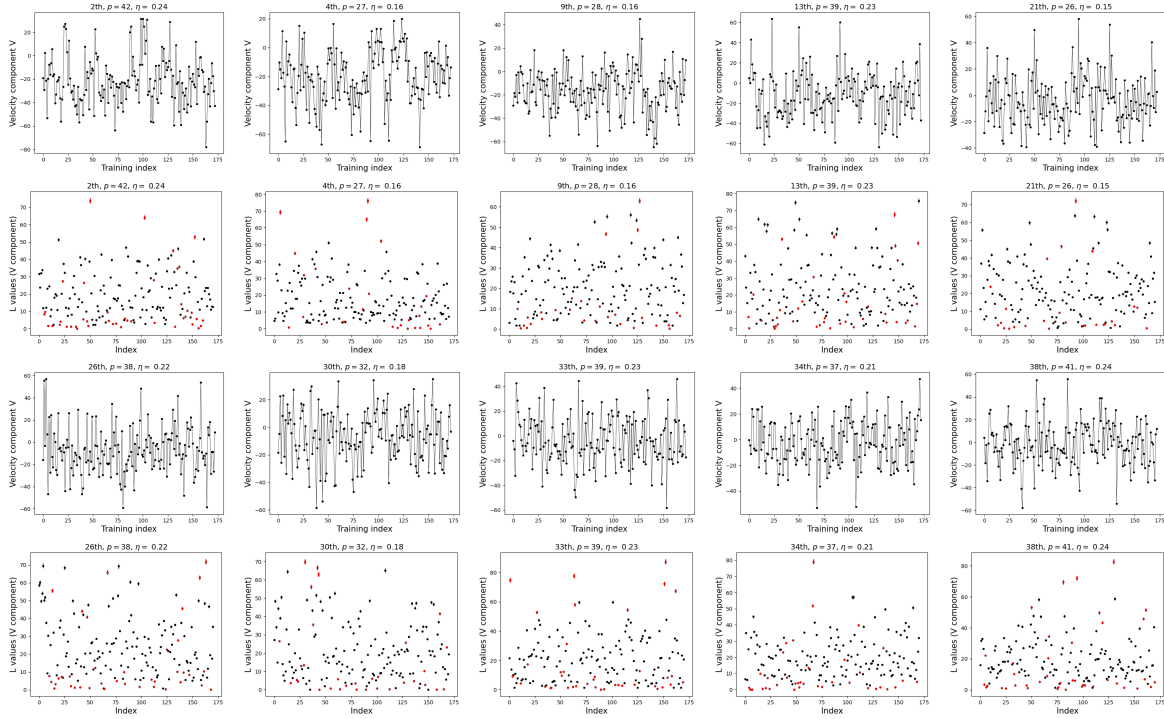


Fig. 6.22 Trivial function and L values of V component of the subset S of ten stars (indexed as 2, 4, 9, 13, 21, 26, 30, 33, 34, 38), whose predictive results with the presented two models given the training data are presented in Figure 6.16 and Figure 6.17.

multivariate Normal density). Thus the likelihood becomes a tensor Normal with its mean and covariance matrices as the model parameters to be learnt. Using the Bayes rule, I then invoke priors on the models parameters and obtain the posterior from the likelihood and the prior. Then I apply Metropolis-within-Gibbs MCMC inference to sample from the posterior to learn the unknown model parameters non-parametrically, given the inhomogeneously correlated data.

Having learnt the unknown model parameters i.e. having learnt the sought functional relationship between the input and output, given the inhomogeneously correlated data, I have performed prediction of the matrix-variate output variable given a vector-variate input variable. The comparative results show that the non-parametric model outperforms the single layer GP based model. I have also included explanations of the predictive performance with the non-parametric model using the inhomogeneities present in the correlation structure which indicates that the presence of higher inhomogeneities in the correlation structure within the subspace of the input space poses more challenges to the prediction than another subspace having lower inhomogeneities.

Chapter 7

Conclusion

In this thesis I have addressed applications of solutions to a common problem, namely, learning a nonlinear function that is inhomogeneously correlated, where the learning is done, given a small to medium sized training data. Here I have used the existing result that sampling multiple times from a GP, yields the same result on the average, as what is achieved by sampling once from each of multiple GPs that are different in the hyperparameters of the kernels that parametrise their covariance functions. This is true, as long as the system is ergodic. This gave rise to the result that a kernel hyperparameter is dependent on a parameter T , where a sample function is sampled from a GP that is uniquely constructed at a value of T . Then T is the iteration index, so the hyperparameter is dependent on this T . Such a model can address the heterogeneity in the correlation structure of available data. So using this model, I have undertaken applications given real-world datasets, to learn functional relationships between variables, and predict the output variable at test inputs. I implemented this idea using two layers of GPs. Given the model that I have used, I had to learn one kernel hyperparameter per dimension of the input vector, for each layer of GPs.

Many existing non-stationary kernels in the literature have used two-layered GPs as their learning strategy with assumption of the stationarity of the second layer. However, this assumption of stationarity in the second layer may not be correct in a given real-world application. In this thesis, the sufficiency of two layers has been proven by the means of the design of the learning strategy, where the second layer can be stationary without assuming it to be.

Within this strategy, I have learnt the scalar-valued univariate function given a univariate data; scalar-valued multivariate function that takes a vector as its input given a multivariate data; matrix-valued function given high-dimensional data.

The thesis has presented the efficient learning of uncertainties while showing that the prediction quality can be poor or incorrect based on the poorly chosen way of addressing uncertainties.

This thesis also presents the internalisation of an existing notion of inhomogeneously correlated data which I have used to explain predictions made on the outputs, following my learning of a high-dimensional function.

This thesis included DNN exposition for univariate and multivariate data and presented the sensitivity of results to DNN architecture, raising concerns over reliability of predictions obtained using DNNs.

7.1 Limitations of current work and future work

Limitations of the current thesis include the following.

- Currently the work presented in this thesis is restricted to the small to medium sized data ($\sim 10^4$).
- The presented methodologies are restricted to the regression task and not readily available for classification task.
- The data that can be handled by the methodologies of this thesis has to be of type numerical.
- Missing data handling or imputation is not a part of the current models.

Future extension of this thesis work can be done in the following directions.

- **Forecasting:** In general GP-based learning of the time dependence on a random variable, leads to poor forecasting, the quality of which can improve only using carefully designed informative priors, in contrast to the high predictive performance that I have demonstrated in this thesis. I would like to extend the forecasting work from this thesis (included in Appendix A) to forecast accurately by developing data-driven, informative priors based on the temporal evolution of the covariance structure.
- **Temporal evolution of covariance structure:** Modelling the temporal evolution of correlation function using Generalised Wishart Process (Wilson and Ghahramani, 2011) captures the covariance structure of the underlying GP from which the sought function (functional relation of two variables) is randomly realised. Modelling the covariance dynamics following (see Chapter-6 of Chakrabarty (2025)) is planned. Such covariance modelling would benefit learning given evolving data.

- **Scalable Bayesian non-parametric model:** Current literature lacks non-parametric, non-stationary, probabilistic learning model which is parsimonious and computationally fast given large and/or high-dimensional datasets. My aim is to develop scalable, non-parametric Bayesian learning models for “Reliable learning with fast computation given large data” by means of techniques such as distributed computing where distributed inference can be performed on the partitioned data based on the second order non-stationarity present in it.
- **Usage of quantified inhomogeneous correlation structure:** The relation between the non-stationarity present in the data and the inhomogeneous correlation structure of a data is also sought for future work and the explainability of predictions can be worked upon using the quantification of the inhomogeneous correlation structure in the empirical data.
- **Extension:** The presented methodologies can be extended for classification task along with capability of handling several other types of the data such as image, audio signals, videos.

Appendix A

Forecasting with Non-parametric Model

A.1 Forecasting with Gaussian Process

Though GP can produce excellent performance for predictive task, it is particularly challenging and sometimes bad in the forecasting task and till date, work available in the literature on forecasting with GP is very limited. There are several reasons for such poor performance of GP in the forecasting task which includes the kernel function used to capture the covariance structure of the data, the distance between the test points for which forecasting needs to be made from the last training data point available i.e. the unavailability of training points around test points has important impact and in this work, I will present novel approaches to enhance GP's performance in forecasting.

A popular approach in literature for forecasting is one-step ahead approach (Cheng and Wei, 2010; Papacharalampous et al., 2018) where the forecasting is not undertaken for all the test points in one go, instead, forecast is generated for the first test point, then the data pair - the test point and the forecast output at that test point - is included to the training data and then this updated training data is used to forecast at the next test data. The One-step ahead strategy is depicted in the following subsection in the context of forecasting with GP using a scalar variate time series data where both the input and output are scalar variate and the input time interval of the training data is not uniformly spaced.

A.1.1 One-step ahead strategy for forecasting

As I mentioned that the forecasting performance depends on the distance between the test points for which forecasting needs to be made from the last training data available. Let, $D_{train} = \{(x_i, y_i)\}_{i=1}^M$ is the training data where the input points of the training data are denoted by the set $D_X^{(M)} = \{x_1, x_2, \dots, x_M\}$, the output points of the training data are denoted

by the set $D_Y^{(M)} = \{y_1, y_2, \dots, y_M\}$ and the test input points are represented by the set $D_{X^*}^{(M^*)} = \{x_1^{(*)}, x_2^{(*)}, \dots, x_{M^*}^{(*)}\}$. So, the forecasting performance at the test points $x_1^{(*)}, x_2^{(*)}, \dots, x_{M^*}^{(*)}$ will depend on the distance $(x_1^{(*)} - x_M), (x_2^{(*)} - x_M), \dots, (x_{M^*}^{(*)} - x_M)$ respectively. More the distance, poorer the performance of GP for forecast. Therefore, a standard approach in the literature is to use an iterative one-step ahead approach (Girard et al., 2002). In the one-step ahead approach of forecasting, once I have learned my kernel hyperparameter, θ where θ denotes the set of hyperparameters of the kernel, $K(., . | \theta)$, used in the GP to compute its covariance matrices, using D_{train} , then I will forecast at the test point x_1^* to generate the forecast output y_1^* using GP.

$$y_1^{(*)} | D_{train}, x_1^{(*)} \sim GP(0, K(., . | \theta)) \quad (\text{A.1.1})$$

Then this $(x_1^{(*)}, y_1^{(*)})$ pair is augmented in the training data D_{train} and now the augmented training data is denoted by $D_{train}^{(M+1)}$. Then, forecasting at the test point $x_2^{(*)}$ is generated using GP with the augmented training data as follows.

$$y_2^{(*)} | D_{train}^{(M+1)}, x_2^{(*)} \sim GP(0, K(., . | \theta))$$

Again, after forecasting, the new pair $(x_2^{(*)}, y_2^{(*)})$ is augmented in the training data $D_{train}^{(M+1)}$ to yield the updated augmented training data which is denoted by $D_{train}^{(M+2)}$. Thus, for the test point $x_i^{(*)}$, the forecast value $y_i^{(*)}$ will be generated using GP with the updated training data $D_{train}^{(M+i-1)}$ as follows,

$$y_i^{(*)} | D_{train}^{(M+i-1)}, x_i^{(*)} \sim GP(0, K(., . | \theta))$$

Data: I have used the same Brent crude oil price training data presented in the Section 4.7.1 of Chapter 4. The data points those are beyond training range i.e. from January 2022 to March 2022, 20 data points are considered as test points for forecast. I have transformed the dates into the days in running total i.e. the date of the first data point is denoted as day one, the second data points corresponding date is denoted as day two and so on. In this work all the forecast results will be presented against the days in running total.

Forecasting with one-step ahead strategy: Once the prediction task is performed with the oil price data using the non-parametric model that invokes SQE-looking kernel, I have performed the forecasting at the test points using the one-step ahead strategy mentioned in the Section A.1.1. The forecast is performed with the learnt values of the hyperparameters i.e. mean of the 95% HPD credible region computed from the marginal of the hyperparameters. The original training data D_{train} is updated by including the test data used for prediction,

predictive results on the test data points for prediction along with its uncertainties resulting an augmented training data D_{train}^+ . D_{train}^+ is used as the training data for the forecast.

Forecasting is not undertaken for all the test points in one go, rather, using one-step ahead strategy, the forecast is generated for the first test point using the augmented training data D_{train}^+ . Once, the forecast result is obtained for the first test point, then the forecast mean and associated uncertainty is again included in the D_{train}^+ and for the second test point, this currently augmented D_{train}^+ is used to generate forecast for the second test point and so on. for all the further test points, this one-step-ahead strategy is used. In this implementation, all inputs remain un-standardised and all outputs are standardised for both MCMC inference and forecast. Figure A.1 depicts results of forecast at the test points given the updated training data with the non-parametric model having SQE-looking kernel and the single layer GP based model having SQE kernel using one-step ahead strategy with the same model presented in the Chapter 4. These results show that in both the cases, the performances of the forecasting do not compare favourably, although, the predictions at the test points are quite satisfactory, however, the forecasts are not acceptable for real-world datasets.

Very recently Corani et al. (2021) have proposed a fixed kernel composition with eight hyperparameters to be learned for forecasting with GP. However, any fixed kernel composition would severely limit the complexity of the function that can be learnt by the GP. Therefore, in the next section, I will present a novel strategy for forecasting which uses a non-parametric kernel presented in Chapter 4 which can yield yield acceptably good performance of the forecast along with very less number of hyperparameter learning while escaping from the restrictions of the parametric, closed forms imposed by the parametric kernels.

A.1.2 A novel strategy for forecasting with One-step ahead approach

In this new strategy of forecasting, after the augmentation of the training data during the one-step ahead iterations approach, the kernel hyperparameter(s) are re-learned. With the re-learned hyperparameter(s), the forecasting is performed iteratively at each test point using one-step ahead approach. The augmentation of the training data also includes noting down the uncertainties of the forecasting at each test point and later on the variance computed from these uncertainties are included in the covariance matrix of the GP as noise.

Let the learnt hyperparameter using the non-parametric kernel $K_N(.,.|\boldsymbol{\theta})$ be $\boldsymbol{\theta}$ with D_{train} as training data when I am going to forecast at the first test point $x_1^{(*)}$. The forecast output $y_1^{(*)}$ is generated at $x_1^{(*)}$ with the non-parametric model, given the updated training data, as follows,

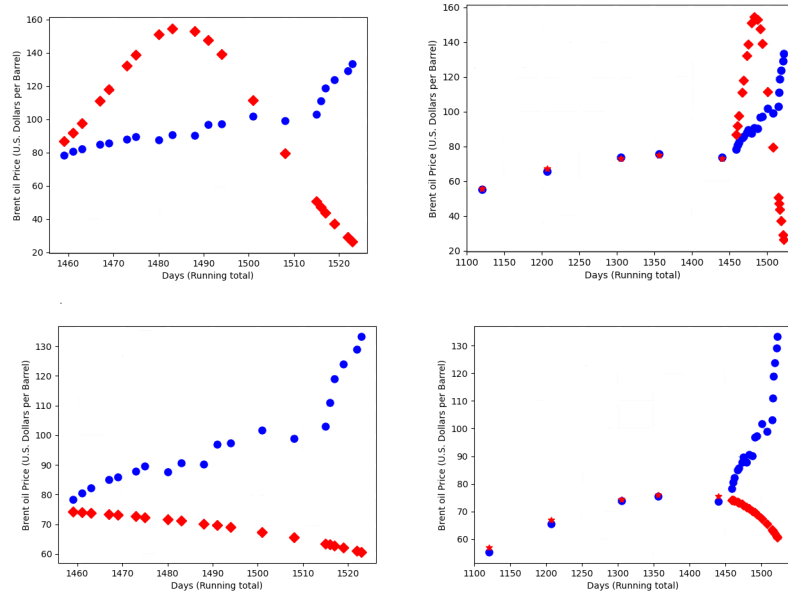


Fig. A.1 Results of forecast at the test points given the updated training data with the non-parametric model having SQE-looking kernel and the single layer GP based model having SQE kernel using one-step ahead strategy with the same model presented in the Chapter 4. *Top row, left panel:* Forecast at the test points, given updated training data with non-parametric model having SQE-looking kernel. *Top row, right panel:* Forecast at the test points, given updated training data with non-parametric model having SQE-looking kernel along with the prediction results at the previous five test points with the same model. *Bottom row, left panel:* Forecast at the test points, given updated training data with single layer GP based model having SQE kernel. *Bottom row, right panel:* Forecast at the test points, given updated training data with single layer GP based model having SQE kernel along with the prediction results at the previous five test points with the same model. True values are displayed with blue solid circle, forecasts are presented with red diamonds, while the predictions are depicted with red stars.

$$y_1^{(*)} | D_X^M, D_Y^M, x_1^{(*)} \sim GP(0, K_N(.,. | \boldsymbol{\theta})) \text{ where}$$

$$y_1^{(*)} = K(x_1^{(*)}, D_X^{(M)} | \boldsymbol{\theta}) [K(D_X^{(M)}, D_X^{(M)} | \boldsymbol{\theta}) + I_1]^{-1} D_Y^{(M)} \text{ where}$$

$$I_1 = \begin{bmatrix} \sigma_{noise} & 0 & \cdots & 0 \\ 0 & \sigma_{noise} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_{noise} \end{bmatrix} \quad (\text{A.1.2})$$

where σ_{noise} is the noise variance added to the diagonal of the covariance matrix of the GP for numerical purpose.

Let the uncertainties of the $y_1^{(*)}$ is $(y_{1-}^{(*)}, y_{1+}^{(*)})$. The variance, $\sigma_1^{(*)}$ is computed from this uncertainty as follows,

$$\sigma_1^{(*)} = \left(\frac{y_{1+}^{(*)} - y_{1-}^{(*)}}{5} \right)^2 \quad (\text{A.1.3})$$

Since, the 95% confidence interval of a Standard Normal distribution is $(-1.96, 1.96)$, i.e. 95% of the area under the curve falls within this interval, so when it is considered that the uncertainty is 5 standard errors wide ($5 \approx 2.5 * 1.96$), then the uncertainty is divided by 5 in the Equation A.1.3. Then, this $(x_1^{(*)}, y_1^{(*)})$ pair is augmented with the training data D_{train} and now the augmented training data is denoted by $D_{train}^{(M+1)}$ in which the set $D_X^{(M+1)} = \{x_1, x_2, \dots, x_M, x_{M+1}^{(*)}\}$ and the set $D_Y^{(M+1)} = \{y_1, y_2, \dots, y_M, y_{M+1}^{(*)}\}$ denote the input and output points of the augmented training data $D_{train}^{(M+1)}$ respectively.

Then, to forecast output $y_2^{(*)}$ at the test point $x_2^{(*)}$, I will re-learn my hyperparameters of the non-parametric kernel $K_N(.,.)$ given the augmented training data $D_{train}^{(M+1)}$. Let, the newly learnt hyperparameter be denoted by θ_1 . The re-learning of the hyperparameter(s) of $K_N(.,.)$ is performed through MCMC where the choice of the parameters pertaining to the prior has been guided in a certain way. Corani et al. (2021) have shown in their work that the forecasting with GP needs prior, and they have designed their prior suitable for their designed fixed kernel composition for forecasting.

However, in this work, the choice for the priors on the hyperparameter(s) is guided by the information redundancy analysis of the training data to yield a data-driven prior which will be presented in the next section.

Once, the hyperparameter θ_1 is re-learned in the MCMC chain with the data-driven prior, then the forecast output $y_2^{(*)}$ at the test input $x_2^{(*)}$ is generated using GP with the augmented training data $D_{train}^{(M+1)}$ as follows.

$$\begin{aligned} y_2^{(*)} | D_X^{(M+1)}, D_Y^{(M+1)}, x_2^{(*)} &\sim GP(0, K_N(.,. | \theta_1)) \text{ where} \\ y_2^{(*)} &= K(x_2^{(*)}, D_X^{(M+1)} | \theta_1) [K(D_X^{(M+1)}, D_X^{(M+1)} | \theta_1) + I_2]^{-1} D_Y^{(M+1)} \text{ where} \\ I_2 &= \begin{bmatrix} \sigma_{noise} & 0 & \cdots & 0 & 0 \\ 0 & \sigma_{noise} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \sigma_{noise} & 0 \\ 0 & 0 & \cdots & 0 & \sigma_1^{(*)} \end{bmatrix} \end{aligned} \quad (\text{A.1.4})$$

Let the uncertainties of the $y_2^{(*)}$ is (y_{2-}^*, y_{2+}^*) . The variance, $\sigma_2^{(*)}$ is computed from this uncertainty as $(\frac{y_{2+}^{(*)} - y_{2-}^{(*)}}{5})^2$. Now, the training data $D_{train}^{(M+1)}$ is again augmented with the input-output pair $(x_2^{(*)}, y_2^{(*)})$ and the updated augmented data is now called $D_{train}^{(M+2)}$.

Thus, for the test point $x_i^{(*)}$, the hyperparameter will be re-learnt using the nested model with the updated augmented training data $D_{train}^{(M+i-1)}$. Let θ_{i-1} denotes the re-learnt hyperparameter. Also, let us assume that the set $D_X^{(M+i-1)} = \{x_1, x_2, \dots, x_M, x_{M+1}^{(*)}, x_{M+2}^{(*)}, \dots, x_{M+i-1}^{(*)}\}$ and the set $D_Y^{(M+i-1)} = \{y_1, y_2, \dots, y_M, y_{M+1}^{(*)}, y_{M+2}^{(*)}, \dots, y_{M+i-1}^{(*)}\}$ denote the input and output points of the augmented training data $D_{train}^{(M+i-1)}$ respectively. Then, the forecast value $y_i^{(*)}$ at the test point $x_i^{(*)}$ will be generated using GP with the updated training data $D_{train}^{(M+i-1)}$ as follows.

$$y_i^{(*)} | D_X^{(M+i-1)}, D_Y^{(M+i-1)}, x_i^{(*)} \sim GP(0, K_N(.,. | \theta_{i-1})) \text{ where}$$

$$y_i^{(*)} = K(x_i^{(*)}, D_X^{(M+i-1)} | \theta_{i-1}) [K(D_X^{(M+i-1)}, D_X^{(M+i-1)} | \theta_{i-1}) + I_i]^{-1} D_Y^{(M+i-1)} \text{ where}$$

$$I_i = \begin{bmatrix} \sigma_{noise} & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ 0 & \sigma_{noise} & \dots & 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & 0 & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \sigma_{noise} & 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & \sigma_1^{(*)} & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & \sigma_2^{(*)} & \dots & 0 \\ \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & \sigma_{i-1}^{(*)} \end{bmatrix} \quad (A.1.5)$$

Some of the works in the literature have suggested a pre-designed prior (Corani et al., 2021) for the GP to work. However, the pre-designed priors are not data-driven, and there will be an issue of generalisation for forecasting with GP using such pre-chosen priors without considering the data at hand. In this work, I have modelled the prior using a data driven approach. This data driven approach models the information redundancy present in the training data using which a guidance is obtained on the output value at the test input point at which the forecast is sought. As currently there is no, mapping from the space of output values to the space of values of the hyperparameters, therefore, the prior mean is experimentally chosen having the guidance on the output value at each step of the re-learning to generate forecast at each test point.

A.2 Information redundancy modelling for time series data

The idea behind the modelling of information redundancy is to use a numerical model to explore the training data to get an idea of the expected value of the output at a given test point for forecast. As this is a time series data, so, my aim is to go back to the past in the time series and find the neighbouring time point that is adjacent to the test time point along with the output value at that adjacent neighbouring time point. Then I will find the output values similar to the output value at that adjacent neighbouring time point in the past, while the similarity is manually chosen. For these similar output values, I then find what is the input time point at which those output values are realised. Now, for those input time points, I find what is the immediate next time point along the future time and the output value along with the slope, i.e. I try to capture the trend (discussed in detail in subsequent paragraphs). Then a weighted averaging, which is called affinity weight, is done in order to get a tentative output value at the test point and a slope where the weighted averaging is done in a way that the recent trend has more weightage than far away trend. That means, I am trying to get a tentative trend at a test point. I will now present the modelling of information redundancy with the context of the oil price data that I have considered in this thesis.

Let my training data contain $\{T_1, T_2, \dots, T_n\}$ input time points with corresponding output oil prices $\{P_1, P_2, \dots, P_n\}$. Let, my test time point be T_{n+1} for which I want to forecast the oil price. For the training data, I have computed the slope of each point as follows. Let say I want to compute the slope of i -th training point.

$$Slope_i = \frac{P_i - P_{i-1}}{T_i - T_{i-1}}$$

Now, I will take the adjacent neighbouring time point of the test point T_{n+1} in the past from the training data, which is T_n with corresponding output oil price P_n . Then I will find the neighbouring points of the P_n using a manually set cut off c . Then the neighbouring set will contain the k number of prices that fall within the range of $(P_n - c, P_n + c)$, let this set be $\{P_n^{(i)}\}_{i=1}^k$ and the corresponding set of time points is $\{T_n^{(i)}\}_{i=1}^k$.

Now, for each time point $T_n^{(i)}$ from the set $\{T_n^{(i)}\}_{i=1}^k$, I will find the next adjacent time point along the future time which is denoted by $T_n^{(i+1)}$, the output oil price $P_n^{(i+1)}$ and slope $S_n^{(i+1)}$ at $T_n^{(i+1)}$ respectively. Let the set $\{P_n^{(i+1)}\}_{i=1}^k$ and the set $\{S_n^{(i+1)}\}_{i=1}^k$ denote the oil price and slope of the next training point along the future time for each neighbouring point. Using these sets $\{P_n^{(i+1)}\}_{i=1}^k$, $\{S_n^{(i+1)}\}_{i=1}^k$, now the affinity weighted price and slope will be computed so that a tentative trend can be estimated while one moves from the time point T_n to the test time point T_{n+1} . As mentioned before, the affinity weighted price and slope will be numerically computed in such a way, so the recent trends among the considered similar

trends to be analysed from the past are given more weights than the far away trends. The numerical model for computing affinity weighted slope AW_{slope} and affinity weighted price AW_{price} are as follows.

$$AW_{slope} = \frac{\sum_{i=1}^k \frac{S_n^{(i+1)}}{|T_n^{(i)} - T_n|}}{\sum_{i=1}^k \frac{1}{|T_n^{(i)} - T_n|}} \quad (A.2.1)$$

$$AW_{price} = \frac{\sum_{i=1}^k \frac{P_n^{(i+1)}}{|T_n^{(i)} - T_n|}}{\sum_{i=1}^k \frac{1}{|T_n^{(i)} - T_n|}} \quad (A.2.2)$$

For, example, let us say, I got three neighbours whose prices are P_k, P_l, P_m and time points are T_k, T_l, T_m . Now I will find the next training point for each neighbour along with their slopes whose prices are $P_{k+1}, P_{l+1}, P_{m+1}$ and slopes are $S_{k+1}, S_{l+1}, S_{m+1}$.

So, the affinity weighted slope AW_{slope} and affinity weighted price AW_{price} are computed as following way.

$$AW_{slope} = \frac{\frac{S_{k+1}}{|T_k - T_n|} + \frac{S_{l+1}}{|T_l - T_n|} + \frac{S_{m+1}}{|T_m - T_n|}}{\frac{1}{|T_k - T_n|} + \frac{1}{|T_l - T_n|} + \frac{1}{|T_m - T_n|}}$$

$$AW_{price} = \frac{\frac{P_{k+1}}{|T_k - T_n|} + \frac{P_{l+1}}{|T_l - T_n|} + \frac{P_{m+1}}{|T_m - T_n|}}{\frac{1}{|T_k - T_n|} + \frac{1}{|T_l - T_n|} + \frac{1}{|T_m - T_n|}}$$

A.3 Results on forecasting with new strategy

Currently I will include the forecast results for the six test points $T_0, T_1, T_2, T_3, T_4, T_5$ that cover the time period 07th January 2022 to 25th January 2022 which is a challenging period to forecast due to the pandemic situation and geopolitical tension that I will elaborate upon later in this section while discussing the forecasting results. The date pertaining to each test point is 3-6 days ahead of the corresponding date of the previous test point. In the running days in total, the test set is, $\{1463, 1466, 1469, 1473, 1476, 1481\}$, where the first test point i.e. 1463 denotes the date 07/01/2022. It is to be recalled that the training data has been updated with the test input and predicted value at that test input pair to be used as training data for the forecast at the first test point.

Non-parametric model using the SQE-looking kernel (given in Chapter 4) is used in order to perform the forecasting at the test points given the updated training data following the novel strategy presented in the Section A.1.2, while the priors on the hyperparameters are guided by the strategy depicted in the Section A.2. The MCMC chain is run for 30000 iterations while the computation of the forecast has been started performing from the 20000-

th iteration till the end of the iteration. For convergence diagnostics, the values from the trace of the hyperparameter are sampled during the MCMC chain, over three distinct, equally-wide (covering 3000 consecutive iterations, at iterations 21000, 24000, 27000), non-overlapping iteration windows. The three histograms are coloured in blue, red and black, and overplotted.

Forecasting results at T_0 : Information redundancy analysis gives affinity price and slope for first test point using $c = 1.5$, as 80.4801 and 1.1225 respectively. Normal prior mean and seed are chosen experimentally based on the affinity price as there is no automatic mapping from the space of output value to the space of the values of the hyperparameters. The values of the parameters for running the MCMC chain is presented in the Table A.1. Figure A.2 presents the traces of re-learning of the hyperparameters of the non-parametric model that uses SQE-looking kernel for generating forecast at the first test point T_0 . the mean of 95% HPD credible region on the learnt length scale hyperparameter ℓ (which is modelled as an unknown function of the iteration index) is learnt to be ≈ 14.35 . The mean ℓ value predicted at the end of the second block of post-burnin part the chain is ≈ 14.35 again while mean δ is learnt as ≈ 0.59 . Figure A.3 presents the forecasting at T_0 along with predictions made at the previous five test points for comparing the quality of prediction and the forecast. First row of the Table A.2 depicts the results of this forecast along with the date corresponding to this test point, true value of oil price at that test point and the values obtained from information redundancy analysis for this point.

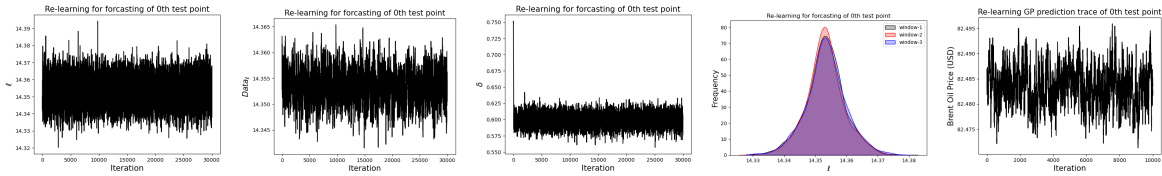


Fig. A.2 Results of re-learning hyperparameters with non-parametric model using SQE-looking kernel to perform forecast at test point T_0 given the updated Brent crude oil price data. The panels from left: the trace of ℓ that is updated in the first block, predicted ℓ using the inner GP at the end of second block, δ learnt in the second block, convergence diagnostics pertaining to the ℓ . The trace of the forecast at T_0 .

Forecasting results at T_1 : Information redundancy analysis gives affinity price and slope for this test point using $c = 1.5$, as 80.9907 and -0.5068 respectively. Normal prior mean and seed are chosen experimentally based on the affinity price as there is no automatic mapping from the space of output value to the space of the values of the hyperparameters. The values of the parameters for running the MCMC chain is presented in the Table A.1.

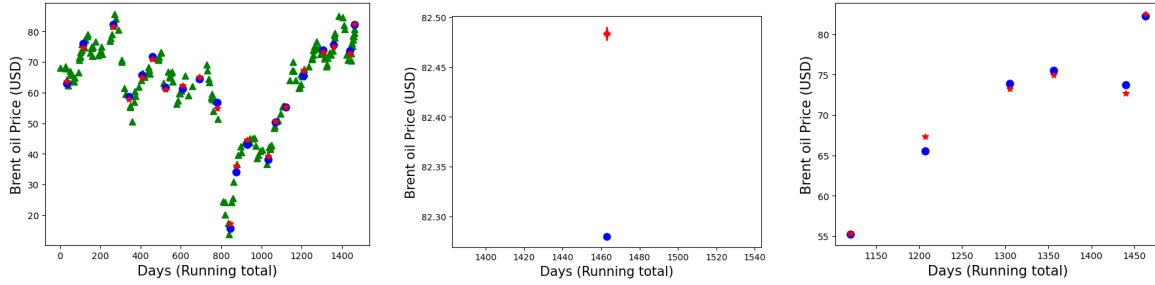


Fig. A.3 Forecasting at T_0 . From left: the prediction and forecasting at the test points plotted in a single plot showing the training data as well; the forecasting generated at T_0 ; forecasting made at T_0 along with predictions made at the previous five test points. True values are displayed with blue solid circle, training points are displayed in green triangle, forecasts are presented with red plus, while the predictions are depicted with red stars.

Figure A.4 presents the traces of re-learning of the hyperparameters of the non-parametric model that uses SQE-looking kernel for generating forecast at the this test point T_1 . the mean of 95% HPD credible region on the learnt length scale hyperparameter ℓ (which is modelled as an unknown function of the iteration index) is learnt to be ≈ 3.47 . The mean ℓ value predicted at the end of the second block of post-burnin part the chain is ≈ 3.47 again while mean δ is learnt as ≈ 0.598 . Figure A.5 presents the forecasting at T_1 along with predictions made at the previous five test points for comparing the quality of prediction and the forecast. Second row of the Table A.2 depicts the results of this forecast along with the date corresponding to this test point, true value of oil price at that test point and the values obtained from information redundancy analysis for this point.

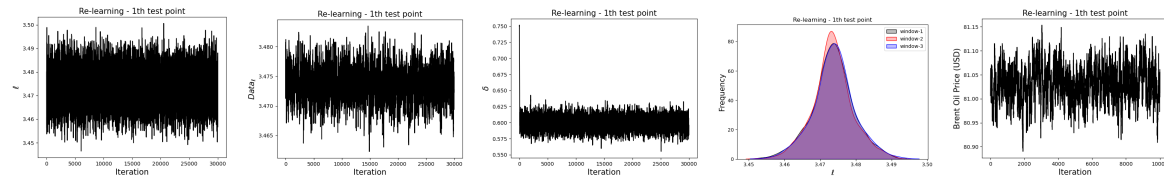


Fig. A.4 Results of re-learning hyperparameters with non-parametric model using SQE-looking kernel to perform forecast at test point T_1 given the updated Brent crude oil price data. The panels from left: the trace of ℓ that is updated in the first block, predicted ℓ using the inner GP at the end of second block, δ learnt in the second block, convergence diagnostics pertaining to the ℓ . The trace of the forecast at T_1 .

Forecasting results at T_2 : Information redundancy analysis gives affinity price and slope for this test point using $c = 1.5$, as 81.2357 and -0.0024 respectively. Normal prior mean

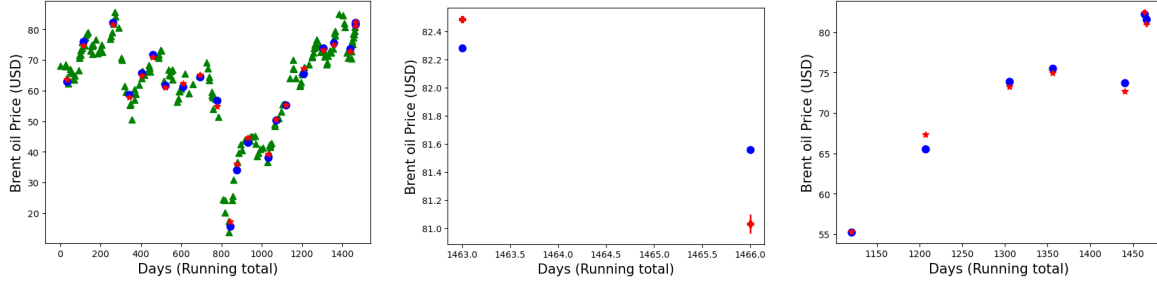


Fig. A.5 Forecasting at T_1 . From left: the prediction and forecasting at the test points plotted in a single plot showing the training data as well; the forecasting generated at T_0 and T_1 ; forecasting made at T_0 and T_1 along with predictions made at the last five test points. True values are displayed with blue solid circle, training points are displayed in green triangle, forecasts are presented with red plus, while the predictions are depicted with red stars.

and seed are chosen experimentally based on the affinity price as there is no automatic mapping from the space of output value to the space of the values of the hyperparameters. The values of the parameters for running the MCMC chain is presented in the Table A.1. Figure A.6 presents the traces of re-learning of the hyperparameters of the non-parametric model that uses SQE-looking kernel for generating forecast at the this test point T_2 . the mean of 95% HPD credible region on the learnt length scale hyperparameter ℓ (which is modelled as an unknown function of the iteration index) is learnt to be ≈ 14.45 . The mean ℓ value predicted at the end of the second block of post-burnin part the chain is ≈ 14.45 while mean δ is learnt as ≈ 0.599 . Figure A.7 presents the forecasting at T_2 along with predictions made at the previous five test points for comparing the quality of prediction and the forecast. Third row of the Table A.2 depicts the results of this forecast along with the date corresponding to this test point, true value of oil price at that test point and the values obtained from information redundancy analysis for this point.

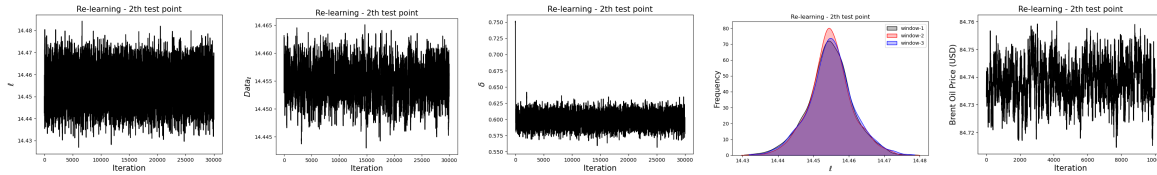


Fig. A.6 Results of re-learning hyperparameters with non-parametric model using SQE-looking kernel to perform forecast at test point T_2 given the updated Brent crude oil price data. The panels from left: the trace of ℓ that is updated in the first block, predicted ℓ using the inner GP at the end of second block, δ learnt in the second block, convergence diagnostics pertaining to the ℓ . The trace of the forecast at T_2 .

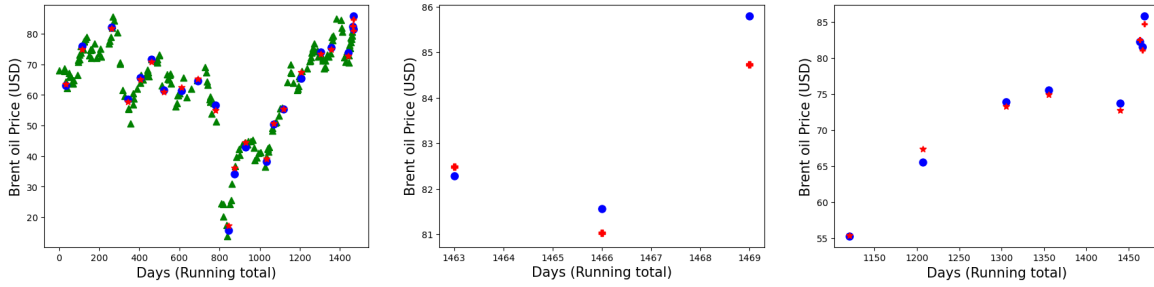


Fig. A.7 Forecasting at T_2 . From left: the prediction and forecasting at the test points plotted in a single plot showing the training data as well; the forecasting generated at T_0 , T_1 and T_2 ; forecasting made at T_0 , T_1 and T_2 along with predictions made at the last five test points. True values are displayed with blue solid circle, training points are displayed in green triangle, forecasts are presented with red plus, while the predictions are depicted with red stars.

Forecasting results at T_3 : Information redundancy analysis gives affinity price and slope for this test point using $c = 1.5$, as 83.0086 and -0.2549 respectively. Normal prior mean and seed are chosen experimentally based on the affinity price as there is no automatic mapping from the space of output value to the space of the values of the hyperparameters. The values of the parameters for running the MCMC chain is presented in the Table A.1. Figure A.8 presents the traces of re-learning of the hyperparameters of the non-parametric model that uses SQE-looking kernel for generating forecast at the this test point T_3 . the mean of 95% HPD credible region on the learnt length scale hyperparameter ℓ (which is modelled as an unknown function of the iteration index) is learnt to be ≈ 3.64 . The mean ℓ value predicted at the end of the second block of post-burnin part the chain is ≈ 3.64 again while mean δ is learnt as ≈ 0.599 . Figure A.9 presents the forecasting at T_3 along with predictions made at the previous five test points for comparing the quality of prediction and the forecast. Fourth row of the Table A.2 depicts the results of this forecast along with the date corresponding to this test point, true value of oil price at that test point and the values obtained from information redundancy analysis for this point.

Forecasting results at T_4 : Information redundancy analysis gives affinity price and slope for this test point using $c = 1.5$, as 84.22 and -0.2349 respectively. Normal prior mean and seed are chosen experimentally based on the affinity price as there is no automatic mapping from the space of output value to the space of the values of the hyperparameters.

The values of the parameters for running the MCMC chain is presented in the Table A.1. Figure A.10 presents the traces of re-learning of the hyperparameters of the non-parametric model that uses SQE-looking kernel for generating forecast at the this test point T_4 . the

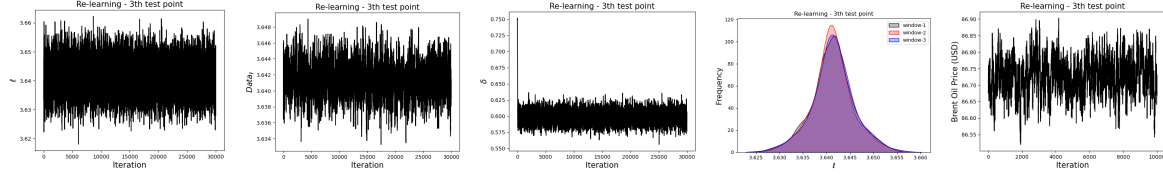


Fig. A.8 Results of re-learning hyperparameters with non-parametric model using SQE-looking kernel to perform forecast at test point T_3 given the updated Brent crude oil price data. The panels from left: the trace of ℓ that is updated in the first block, predicted ℓ using the inner GP at the end of second block, δ learnt in the second block, convergence diagnostics pertaining to the ℓ . The trace of the forecast at T_3 .

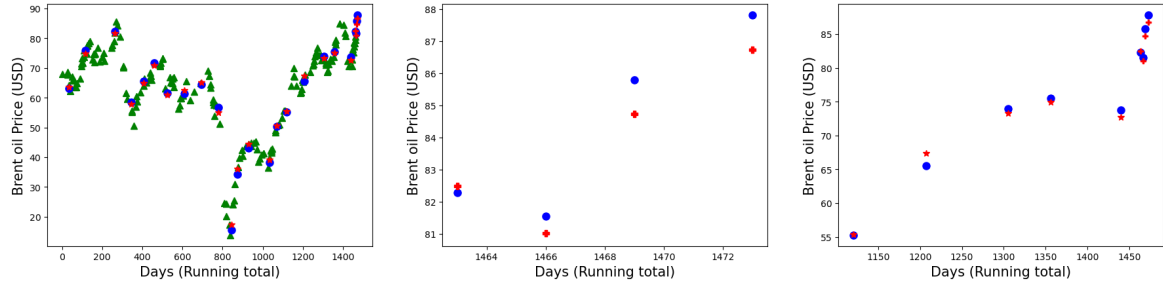


Fig. A.9 Forecasting at T_3 . From left: the prediction and forecasting at the test points plotted in a single plot showing the training data as well; the forecasting generated at T_0 , T_1 , T_2 and T_3 ; forecasting made at T_0 , T_1 , T_2 and T_3 along with predictions made at the last five test points. True values are displayed with blue solid circle, training points are displayed in green triangle, forecasts are presented with red plus, while the predictions are depicted with red stars.

mean of 95% HPD credible region on the learnt length scale hyperparameter ℓ (which is modelled as an unknown function of the iteration index) is learnt to be ≈ 10.77 . The mean ℓ value predicted at the end of the second block of post-burnin part the chain is ≈ 10.77 again while mean δ is learnt as ≈ 0.599 . Figure A.11 presents the forecasting at T_4 along with predictions made at the previous five test points for comparing the quality of prediction and the forecast. Fourth row of the Table A.2 depicts the results of this forecast along with the date corresponding to this test point, true value of oil price at that test point and the values obtained from information redundancy analysis for this point.

Forecasting results at T_5 : Information redundancy analysis can not produce affinity price and slope for this test point because the oil price with which the info redundancy is computed, exceeds the maximum value of the oil price data, but I have idea based on previous values

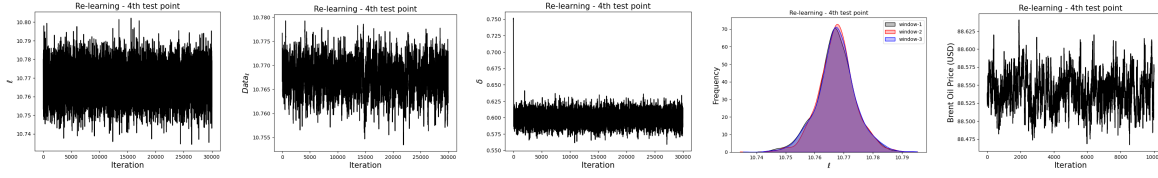


Fig. A.10 Results of re-learning hyperparameters with non-parametric model using SQE-looking kernel to perform forecast at test point T_4 given the updated Brent crude oil price data. The panels from left: the trace of ℓ that is updated in the first block, predicted ℓ using the inner GP at the end of second block, δ learnt in the second block, convergence diagnostics pertaining to the ℓ . The trace of the forecast at T_4 .

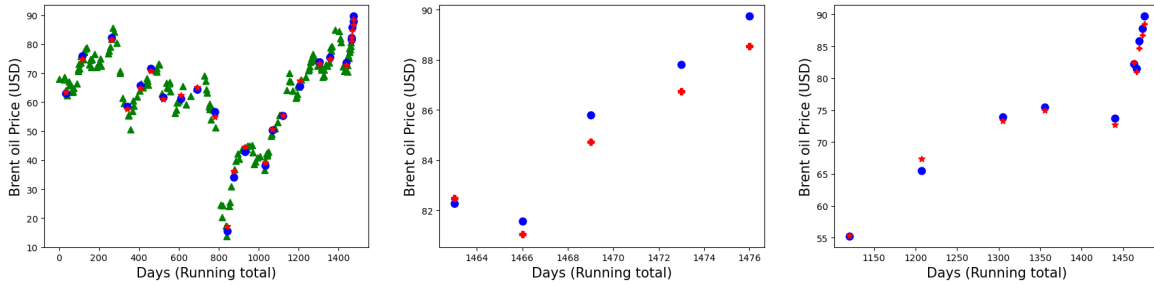


Fig. A.11 Forecasting at T_4 . From left: the prediction and forecasting at the test points plotted in a single plot showing the training data as well; the forecasting generated at T_0, T_1, T_2, T_3 and T_4 ; forecasting made at T_0, T_1, T_2, T_3 and T_4 along with predictions made at the previous five test points. True values are displayed with blue solid circle, training points are displayed in green triangle, forecasts are presented with red plus, while the predictions are depicted with red stars.

that the value might be higher, may be more than 84. Accordingly an uninformative Normal prior mean and seed are chosen experimentally. This poses challenges for future work when information redundancy can not provide any guidance for experimentally choosing priors.

The values of the parameters for running the MCMC chain is presented in the Table A.1. Figure A.12 presents the traces of re-learning of the hyperparameters of the non-parametric model that uses SQE-looking kernel for generating forecast at the test point T_5 . The mean of 95% HPD credible region on the learnt length scale hyperparameter ℓ (which is modelled as an unknown function of the iteration index) is learnt to be ≈ 9.48 . The mean ℓ value predicted at the end of the second block of post-burnin part the chain is ≈ 9.47 while mean δ is learnt as ≈ 0.598 . Figure A.13 presents the forecasting at T_5 along with predictions made at the previous five test points for comparing the quality of prediction and the forecast. Fifth row of the Table A.2 depicts the results of this forecast along with the date corresponding to this

test point, true value of oil price at that test point and the values obtained from information redundancy analysis for this point.

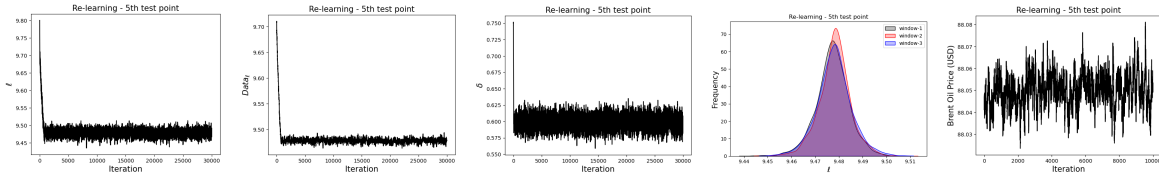


Fig. A.12 Results of re-learning hyperparameters with non-parametric model using SSE-looking kernel to perform forecast at test point T_5 given the updated Brent crude oil price data. The panels from left: the trace of ℓ that is updated in the first block, predicted ℓ using the inner GP at the end of second block, δ learnt in the second block, convergence diagnostics pertaining to the ℓ . The trace of the forecast at T_5 .

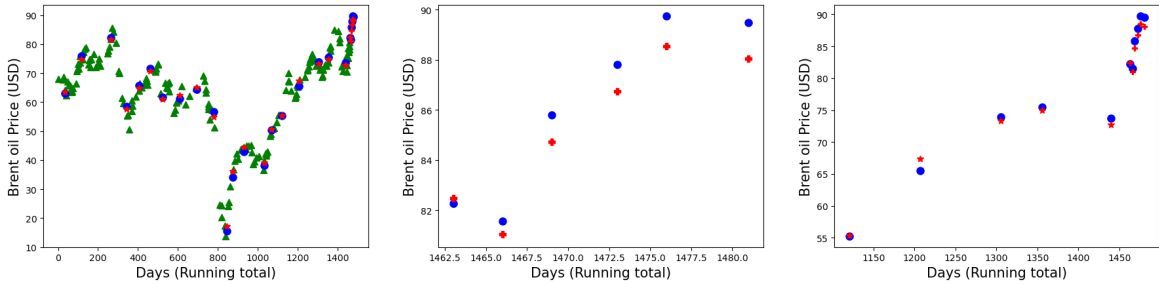


Fig. A.13 Forecasting at T_5 . From left: the prediction and forecasting at the test points plotted in a single plot showing the training data as well; the forecasting generated at T_0 , T_1 , T_2 , T_3 , T_4 and T_5 ; forecasting made at T_0 , T_1 , T_2 , T_3 , T_4 and T_5 along with predictions made at the previous five test points. True values are displayed with blue solid circle, training points are displayed in green triangle, forecasts are presented with red plus, while the predictions are depicted with red stars.

The Oil Market Report (Oil Market Report, 2023) for the forecast period i.e. January 2022 to start of February 2022 (dated 07/01/2022 to 04/02/2022), mentions that the uncertainties related to the COVID (Omicron) surge along with its impact, the rise of geopolitical tension pertaining to the fear of Russian attack on Ukraine heavily impacted the demand-supply in turn effecting the oil price in January-February 2022 after a struggling phase of oil price in December 2021. There was also the threat of central banks raising interest rates which contributed to the demand and production of the crude oils.

Due to the overall effect, the crude oil prices rapidly increased in our forecasting test period and kept on increasing for a longer period where as in the past, the oil price trend shows a frequent up and down pattern within the similar amount of time interval. Therefore,

	T_0		T_1		T_2		T_3		T_4		T_5	
	ℓ	δ	ℓ	δ	ℓ	δ	ℓ	δ	ℓ	δ	ℓ	δ
σ_{prop}	0.01 ²	0.01 ²	0.01 ²	0.01 ²	0.01 ²	0.01 ²	0.008 ²	0.01 ²	0.01 ²	0.01 ²	0.01 ²	0.01 ²
μ_{prior}	14.4	0.6	3.5	0.6	14.5	0.6	3.8	0.6	10.8	0.6	9.5	0.6
σ_{prior}	0.01 ²	0.01 ²	0.01 ²	0.01 ²	0.01 ²	0.01 ²	0.01 ²	0.01 ²	0.01 ²	0.01 ²	0.01 ²	0.01 ²
Seed	14.35	0.75	3.48	0.75	14.45	0.75	3.64	0.75	10.78	0.75	9.8	0.75

Table A.1 Values of the parameters of the MCMC chain run given the oil price data for forecasting. Where σ_{prop} is the variance of the Normal proposal densities for the hyperparameters, μ_{prior} and σ_{prior} are the mean and variance of the Normal priors invoked to the hyperparameters respectively.

Test Days	Date	True price	Forecast mean	Forecast uncertainty	AW_{price} ($c = 1.5$)	AW_{slope} ($c = 1.5$)
T_0	1463 07/01/2022	82.28	82.48	[82.47, 82.49]	80.48	1.12
T_1	1466 10/01/2022	81.56	81.03	[80.96, 81.03]	80.99	-0.50
T_2	1469 13/01/2022	85.8	84.74	[84.73, 84.75]	81.23	-0.0024
T_3	1473 17/01/2022	87.82	86.73	[86.62, 86.83]	83.01	-0.25
T_4	1476 20/01/2022	89.75	88.53	[88.49, 88.58]	84.22	-0.23
T_5	1481 25/01/2022	89.49	88.05	[88.03, 88.06]	-	-

Table A.2 tab forecast all test

for the last test point, the simple heuristic affinity model fails to produce any information redundancy analysis, because of the absence of the information. Although, forecasting will be challenging when there is no information redundancy present in the training data, however, the forecast results presented here are quite close to the true values and better forecasting can be expected had the current geopolitical situation not change drastically and unexpectedly involving a war situation.

Appendix B

Figures for Table 2.2

B.1 Monte Carlo Simulation:

Monte Carlo simulation techniques help us learning the probability models by simulating them i.e. help simulating random processes. We can compute the probabilities, expectations by averaging over the simulations. This is particularly helpful when do not have closed-form expression of the entities of our interest. This situation arises in many areas where we want to describe an accurate model involving probabilistic structure of the system-under-study rather than using any standard, convenient model (Robert and Casella, 2010). The term 'Monte Carlo' was introduced during World War II for stochastic simulations of models of atomic collisions (branching processes). Though, Sometimes this term is used synonymously with stochastic simulation, but sometimes it carries a more specialized meaning (Ripley, 2009). Following paragraph shows how Monte Carlo simulation technique approximate the sought expectation using the Law of Large Numbers (Changye and Robert, 2020b; Geyer, 1998; Robert and Casella, 2010).

Let $X_1, X_2, \dots, X_n$ are a sequence of independent, identically distributed (i. i. d.) simulations of a probability model X . Hence, $X_i \in \{X_1, X_2, \dots, X_n\}$ have the same distribution as X . Now, if we want to compute the expectation of a random variable $g(X)$, i.e. $\mu = E[g(X)]$ and the expectation does not have a closed-form expression, then The Monte Carlo approximation of μ is the sample average over the simulations $X_1, X_2, \dots, X_n$.

$$\hat{\mu}_n = \frac{1}{n} \sum_{i=1}^n g(X_i) \quad (\text{B.1.1})$$

As $\hat{\mu}_n$ is the sample mean of i. i. d. random variables $\{g(X_1), g(X_2), \dots, g(X_n)\}$ which have expectation μ , then using the strong law of large numbers, it is depicted that $\hat{\mu}_n$ converges

almost surely to μ as the number of simulations tends to infinity.

$$\hat{\mu}_n \xrightarrow{a.s.} \mu, \quad n \rightarrow \infty \quad (\text{B.1.2})$$

Although many methods exist for simulating univariate random quantities (Ripley, 2009), however, the ordinary independent-sample Monte Carlo which is able to draw independent samples from the distribution is not so useful for most of the cases of multivariate random quantities. Also, in many practical scenarios, the target distribution of our interest becomes so complex and high-dimensional that it is impossible to directly sample from the target distribution. Hence, we need further techniques which leads to another subset of Monte Carlo methods called Markov Chain Monte Carlo (MCMC) simulation techniques. MCMC simulation techniques are based on Markov Chain as it is able to draw samples where the next sample is dependent on the existing sample.

B.2 Figures for Table 2.2

This appendix includes all the corresponding figures of the MCMC chains that are presented in Table 2.2. The $\star$ mark in the pages because the figures are included as a separate pdf file, hence, its page numbers are not in continuation with that of the main report.

RW

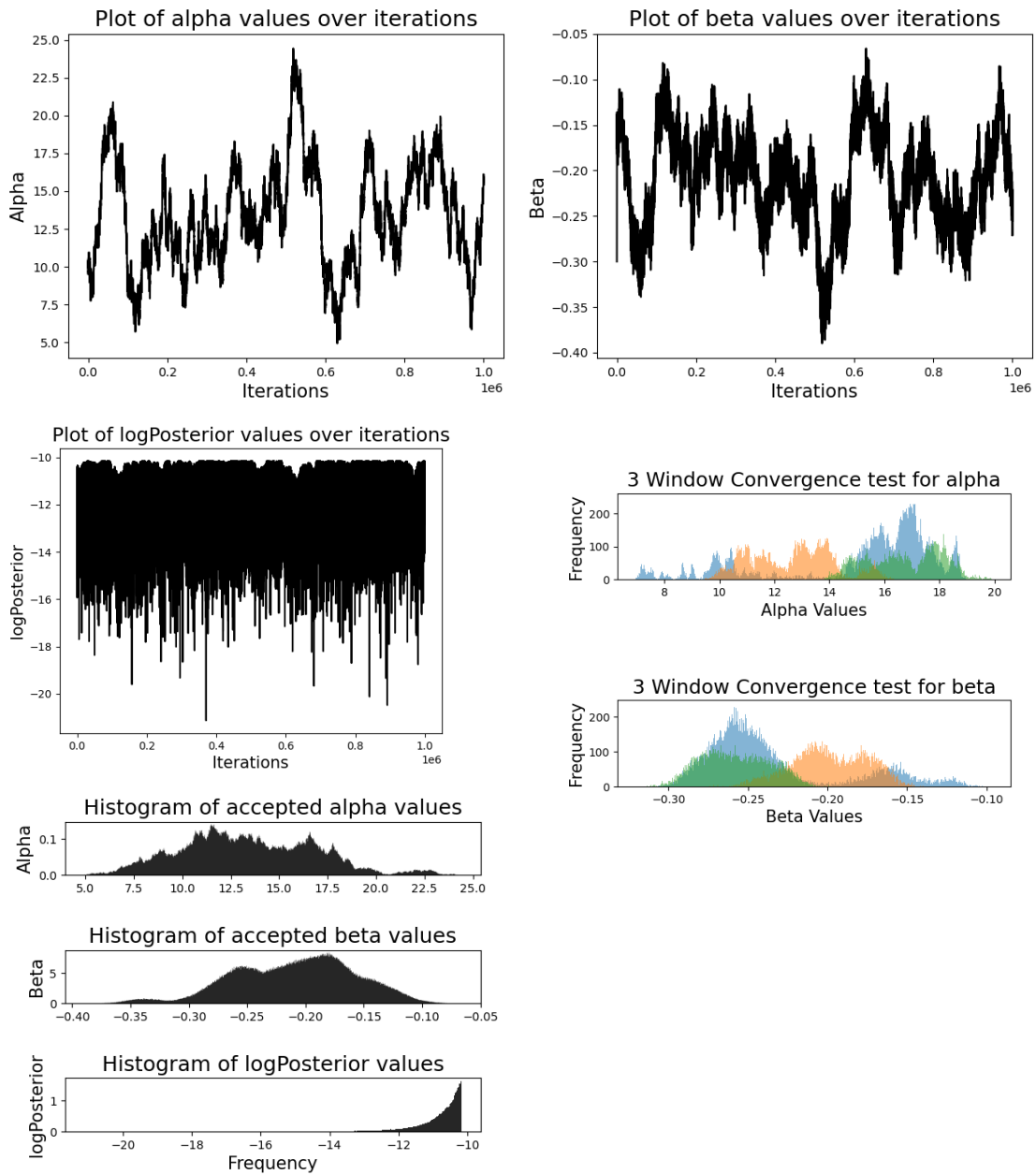
Chain A :

initialAlphaSeed = 10 initialBetaSeed = -0.3, Iteration = 1000000

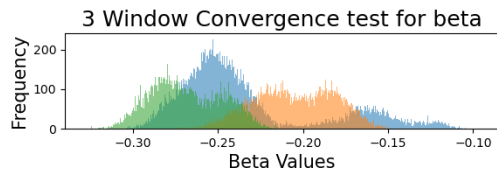
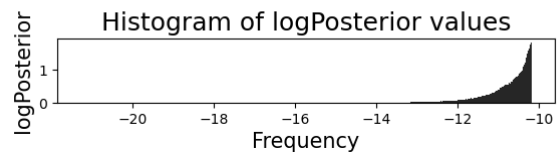
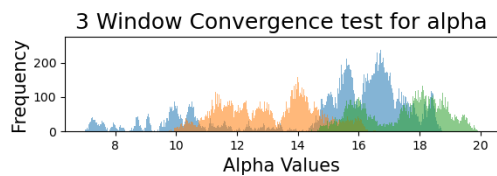
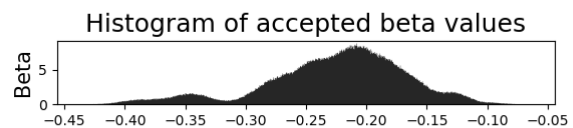
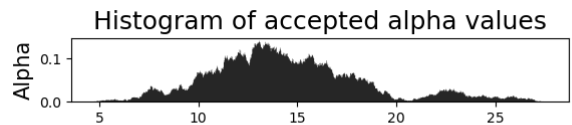
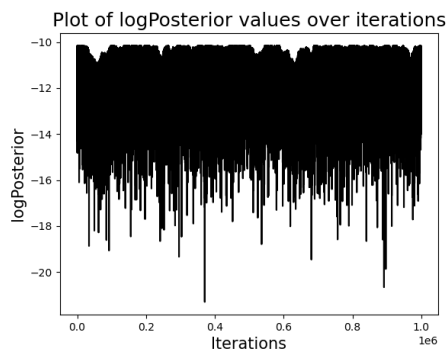
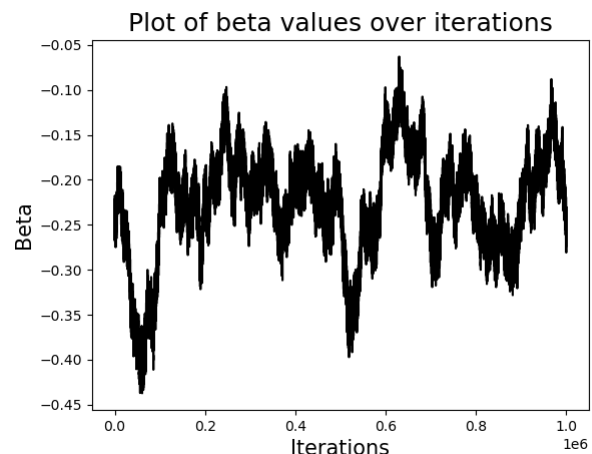
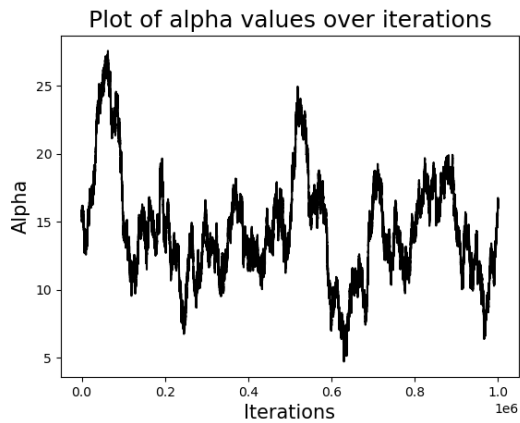
stdDev_alpha_prop = 0.027 stdDev_beta_prop = 0.00405 (0.15*stdDev_alpha_prop)

No Prior (i.e. uniform prior) [seed for random generator=0]

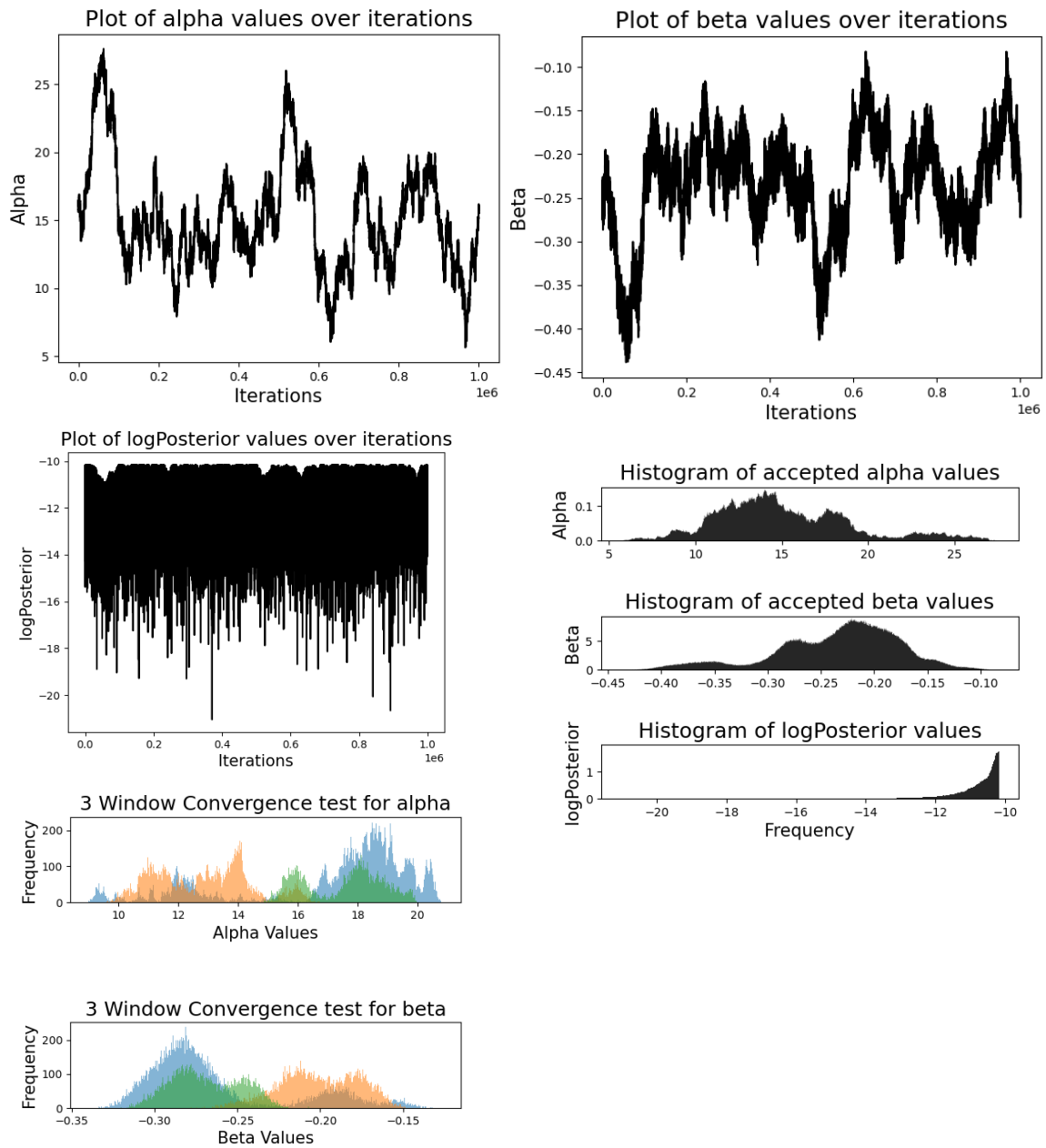
50000 samples at 550000, 750000, 850000 (posterior plot: first 1000 samples truncated)

alpha[end]: 15.42, beta[end]: -0.232, pdf[end]: -10.421607, acc = 84.1%

RW

Chain B: run from t-final of chain-A with same configuration**alpha[end]: 16.067848867697943, beta[end]: -0.24134217484796375**
pdf[end]: -10.428249591133754 , Acc = 84.3559%

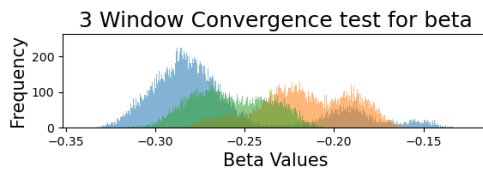
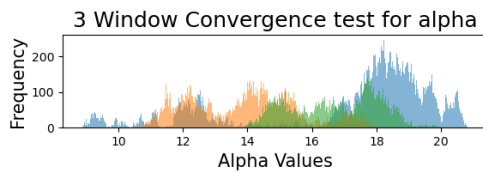
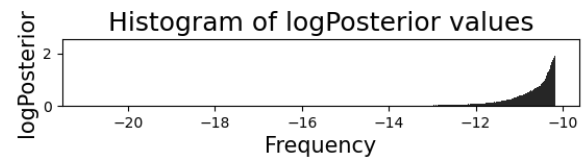
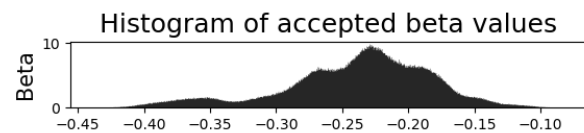
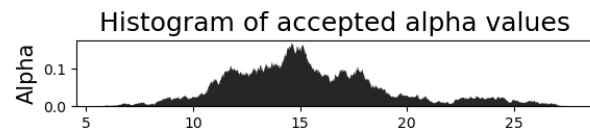
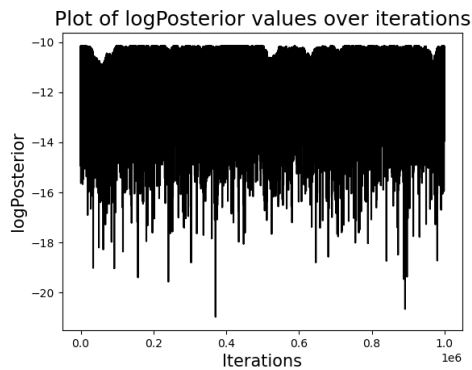
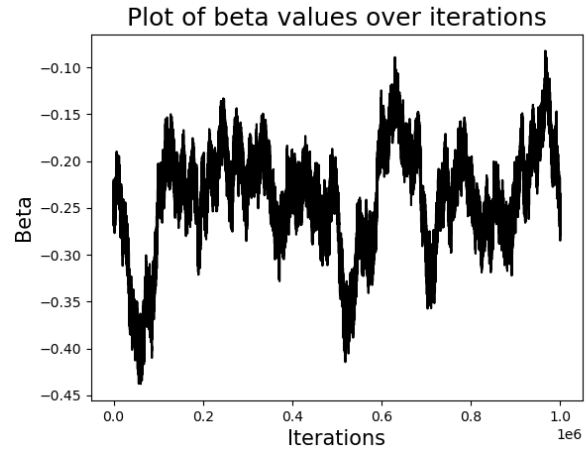
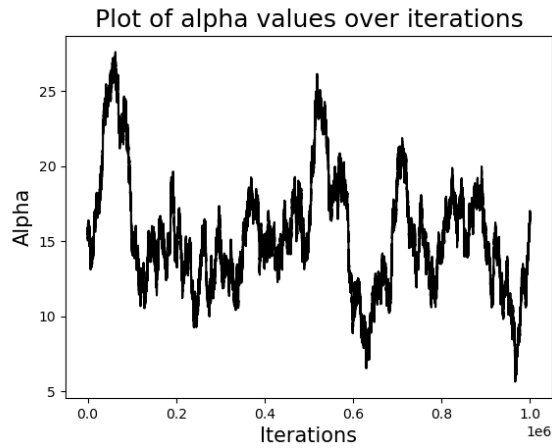
RW

Chain C: run from t-final of chain-B with same configuration**alpha[end]: 15.528052193816638, beta[end]: -0.23341627069004145****pdf[end]: -10.428041542137311, acc = 84.4806%**

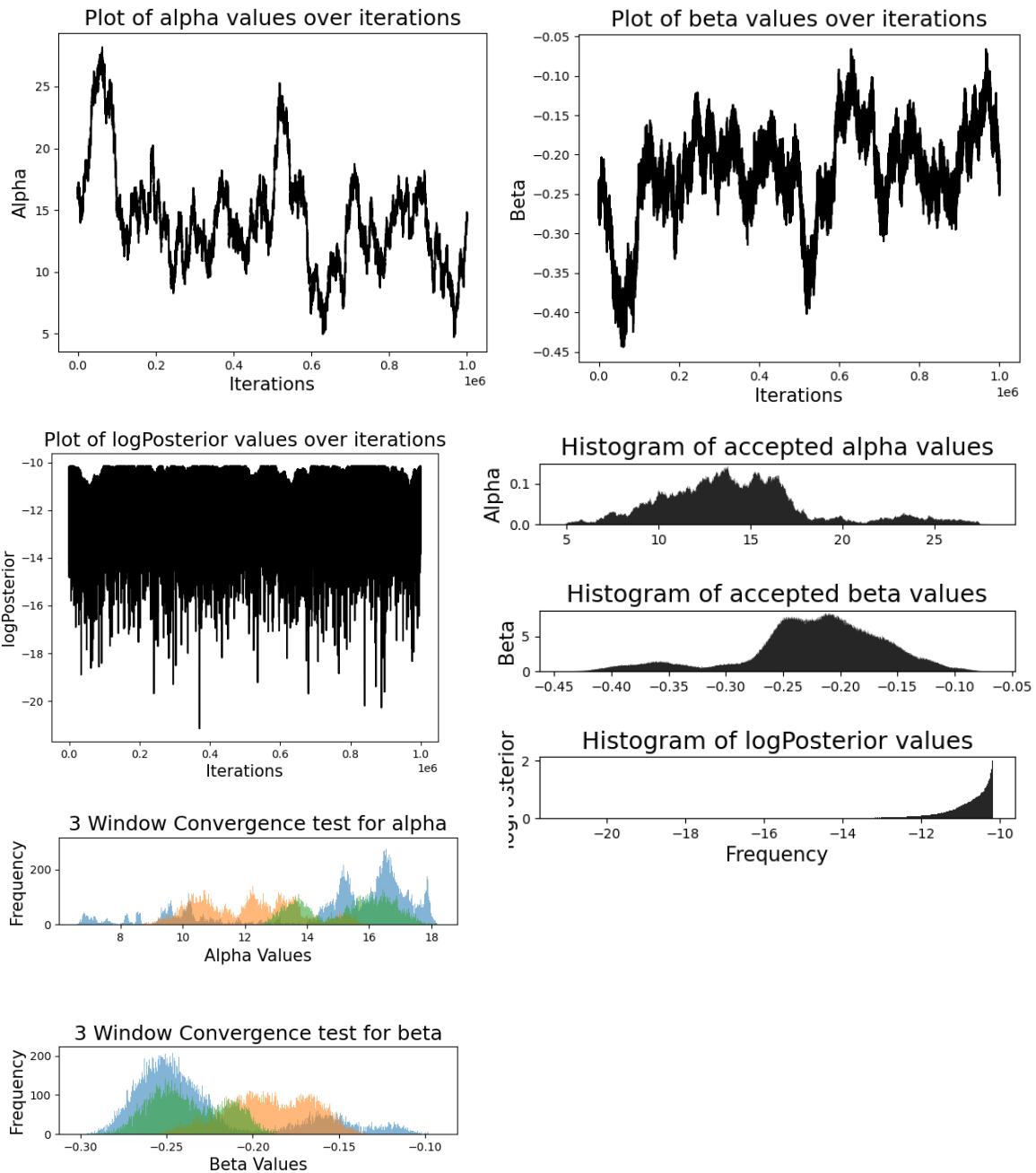
RW

Chain D: run from t-final of chain-C with same configuration**alpha[end]: 16.35729, beta[end]: -0.2454, pdf[end]: -10.446259408059992**

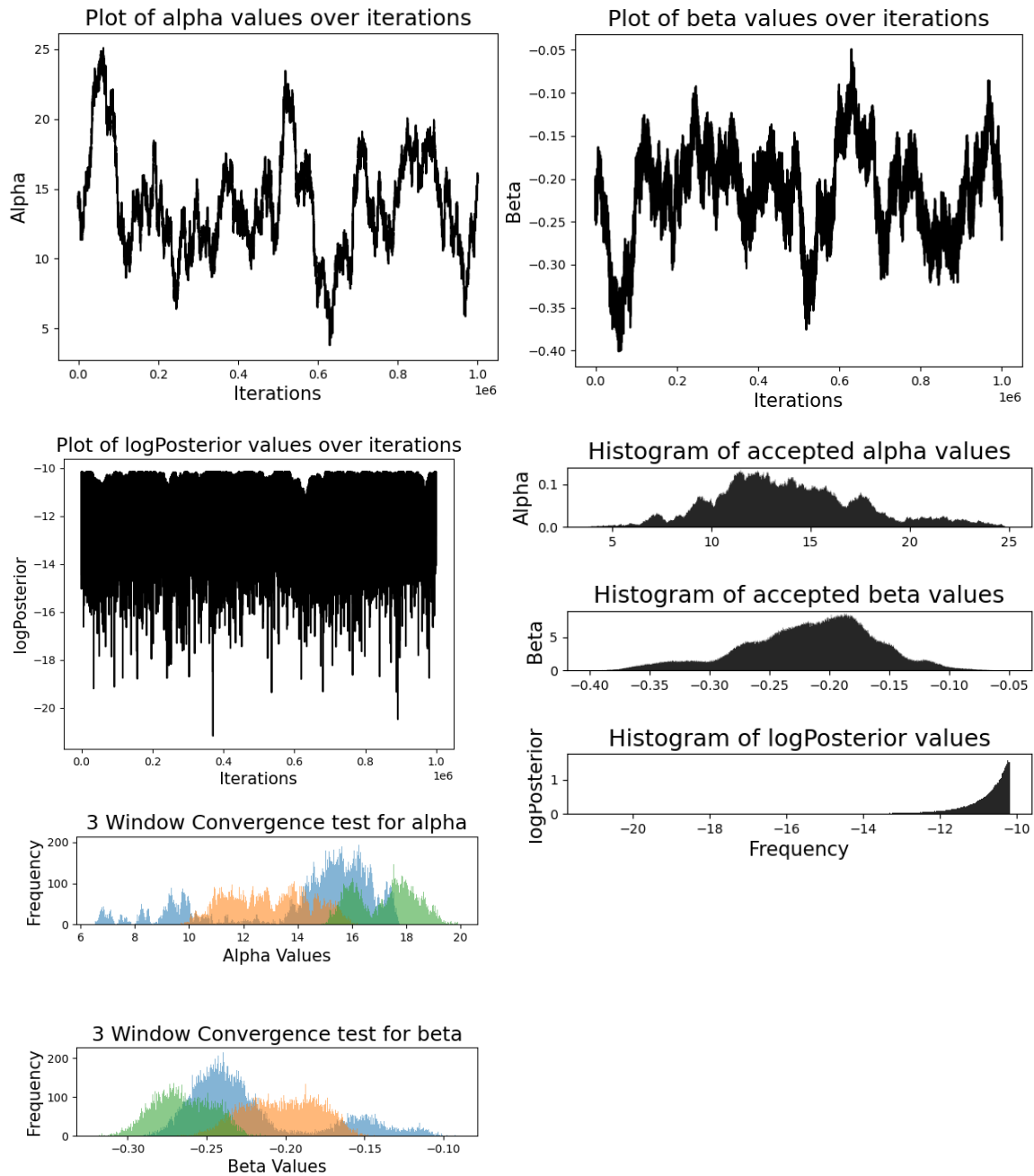
Acc = 84.5549%



RW

Chain E: run from t-final of chain-D with same configuration**alpha[end]: 14.14, beta[end]: -0.2193, pdf[end]: -10.16587, acc = 84.2898%**

RW

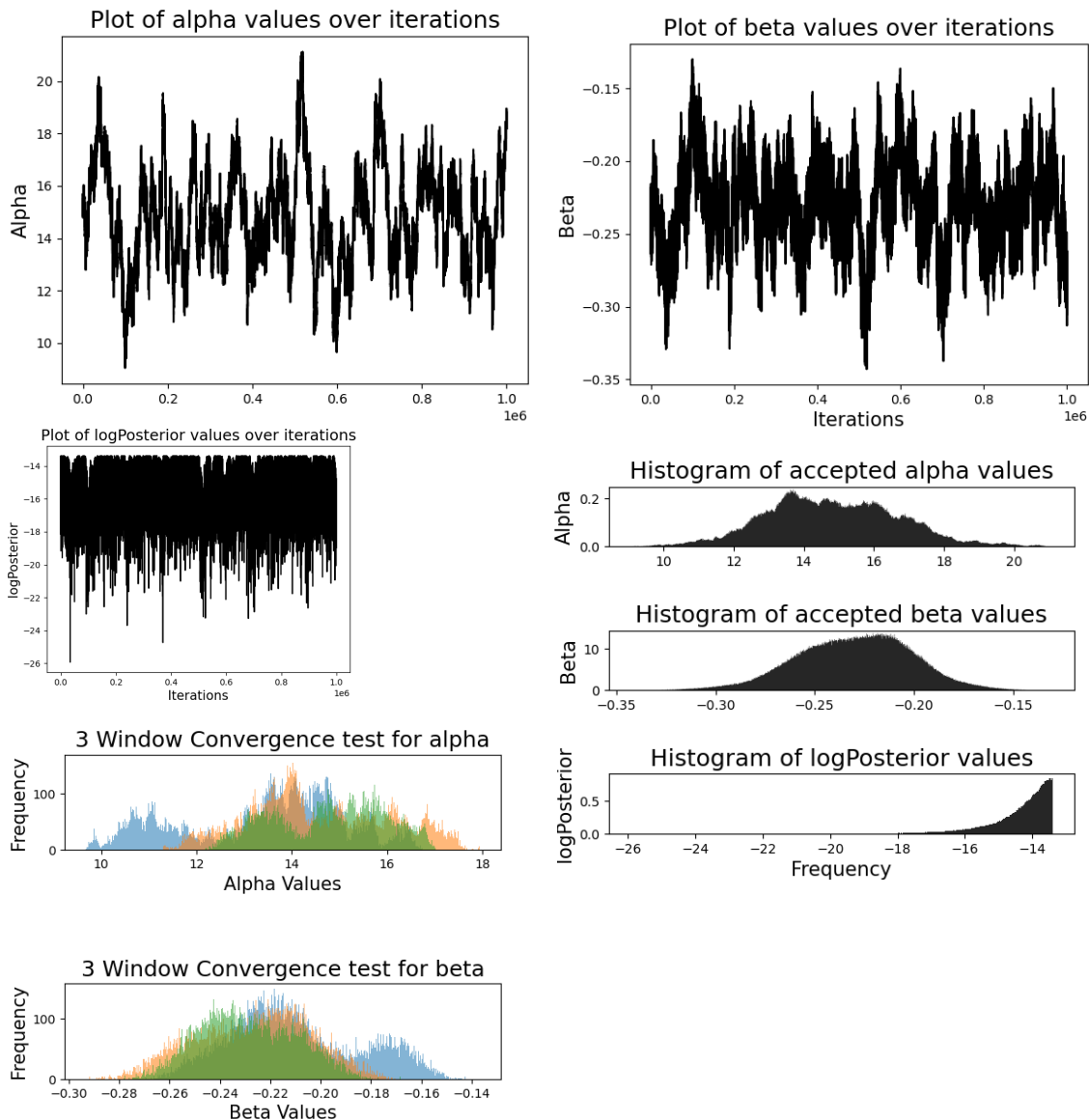
Chain F: run from t-final of chain-E with same configuration**alpha[end]: 15.4294, beta[end]: -0.2320, pdf[end]: -10.4217, acc = 84.2199%**

RW

With 6 pilot chains, we see that alpha value ranges in 14, 15, 16 with mean 15 and beta value ranges in -0.22, -0.23, -0.24 with mean -0.23. So, now we set the prior mean for alpha as 15 and prior mean for beta -0.23 and run the chain with a little flat prior.

Chain - G (approaching convergence, strengthen prior)

initialAlphaSeed = 15.429 initialBetaSeed = -0.2320, Iteration = 1000000
 stdDev_alpha_prop = 0.027 stdDev_beta_prop = 0.00405 (0.15*stdDev_alpha_prop)
 mean_alpha_prior = 15 stdDev_alpha_prior = 2 [seed for random generator=0]
 mean_beta_prior = -0.23. stdDev_beta_prior = 2
 50000 samples at 550000, 750000, 850000 (posterior plot: first 1000 samples truncated)
 alpha[end]: 15.42, beta[end]: -0.232, pdf[end]: -10.421607, acc = 84.4572%



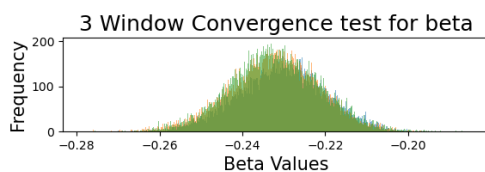
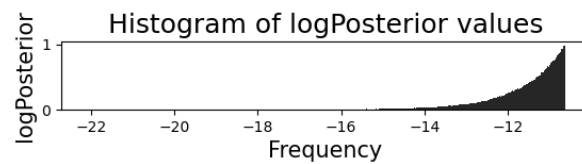
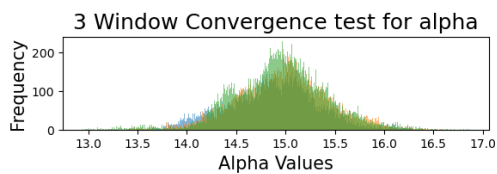
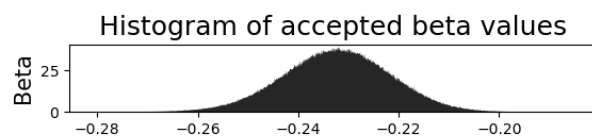
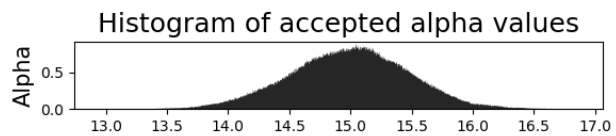
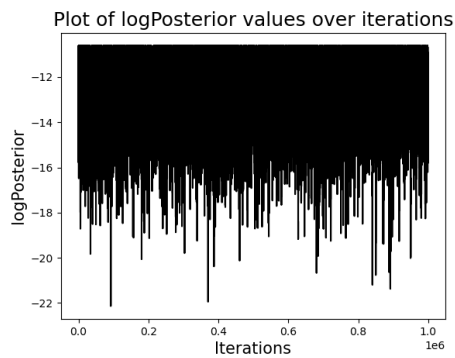
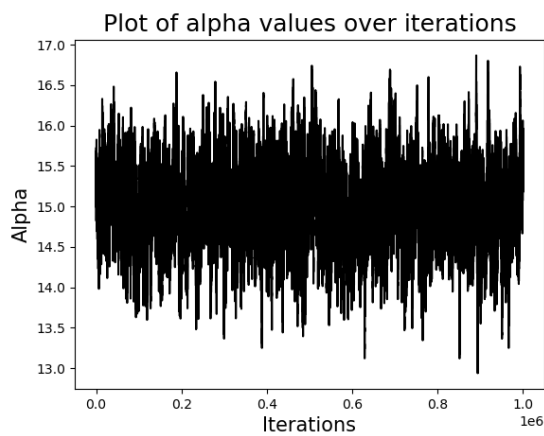
RW

Chain - H (converged but alpha need to be better, increase alpha jump scale)initialAlphaSeed = **15.429** initialBetaSeed = **-0.2320**, Iteration = 1000000

stdDev_alpha_prop = 0.027 stdDev_beta_prop = 0.00405 (0.15*stdDev_alpha_prop)

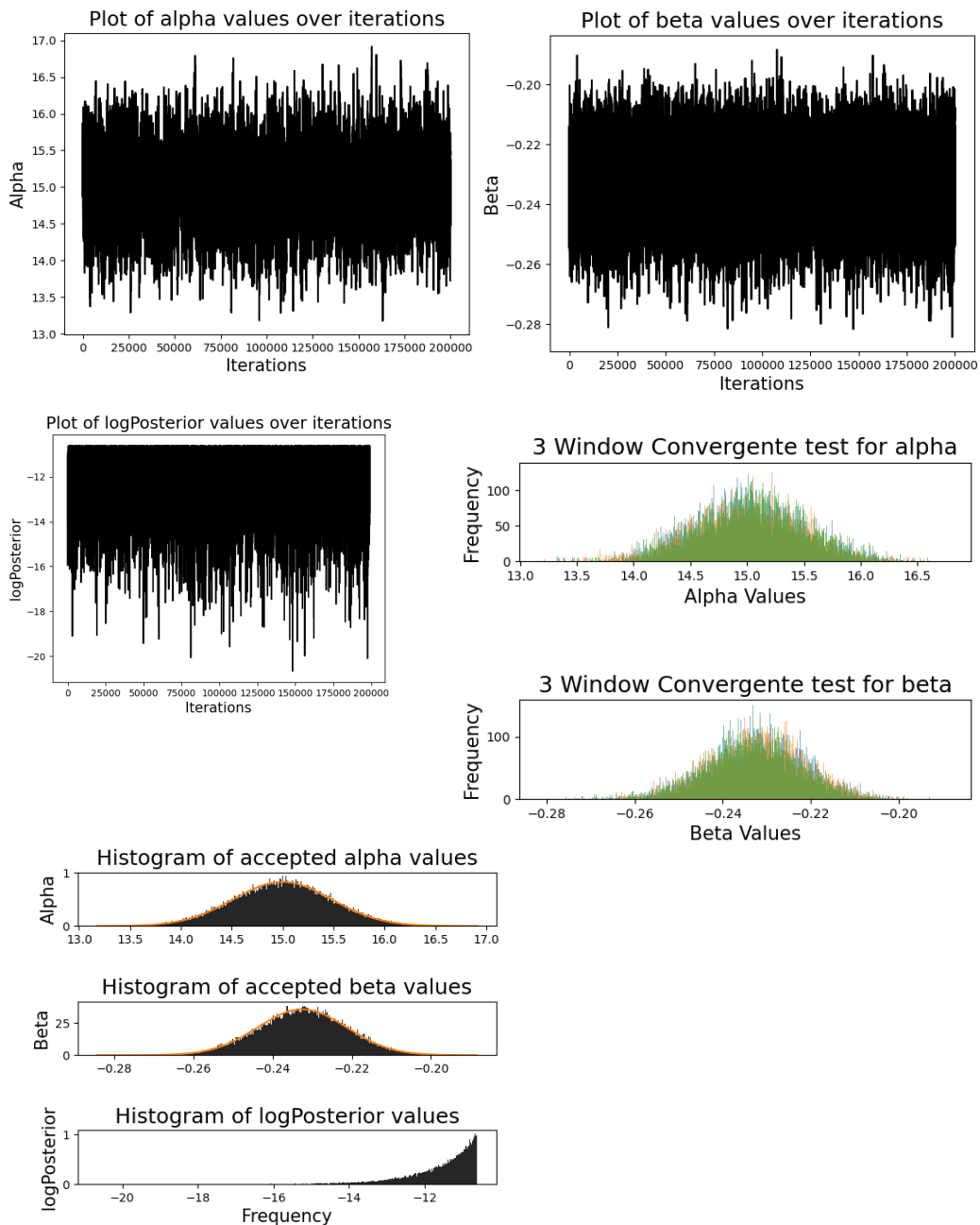
mean_alpha_prior = **15** stdDev_alpha_prior = **0.5** [seed for random generator=0]mean_beta_prior = **-0.23** stdDev_beta_prior = **0.5**

50000 samples at 550000, 750000, 850000 (posterior plot: first 1000 samples truncated)

alpha[end]: 15.184, beta[end]: -0.232, pdf[end]: -10.7055, acc = 84.2712%

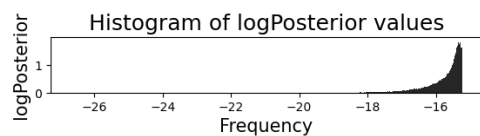
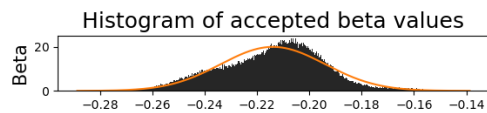
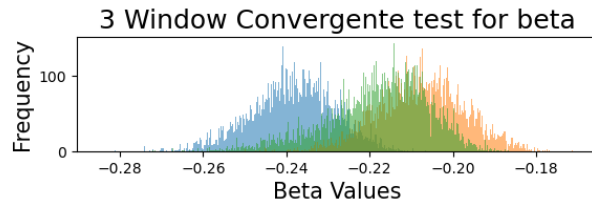
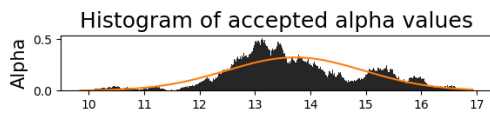
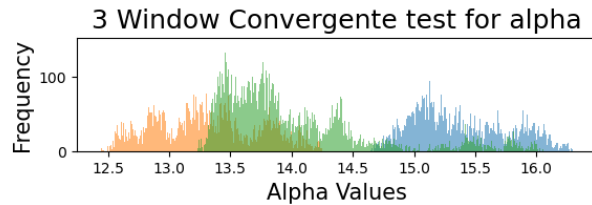
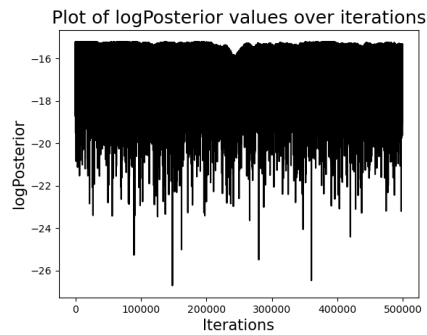
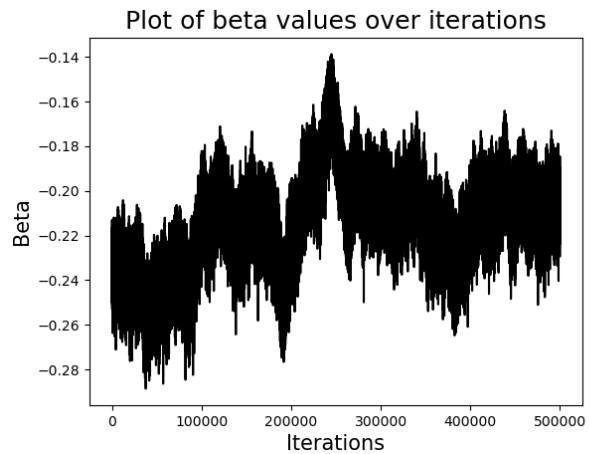
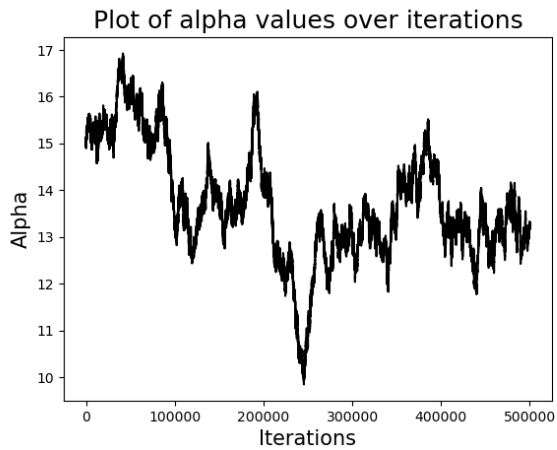
RW

initialAlphaSeed = 15 initialBetaSeed = -0.22, Iteration = 200000
 stdDev_alpha_prop = **0.14** stdDev_beta_prop = **0.0154** ($0.11 \times \text{stdDev_alpha_prop}$)
 mean_alpha_prior = 15 stdDev_alpha_prior = 0.5. [seed for random generator=0]
 mean_beta_prior = -0.235. stdDev_beta_prior = 0.5 [acc rate = 50.1%]
 30000 samples at 100000, 140000, 160000, 95% HPD Alpha: [14.04496711, 15.96026827]
 95% HPD Beta: [-0.25459884, -0.21089]



RW

initialAlphaSeed = 15 initialBetaSeed = -0.23, Iteration = **500000** (8 times more than previous)
 stdDev_alpha_prop = 0.013 stdDev_beta_prop = 0.0156 ($1.2 \times \text{stdDev_alpha_prop}$)
 mean_alpha_prior = 15 stdDev_alpha_prior = 5 [seed for random generator=0]
 mean_beta_prior = -0.235. stdDev_beta_prior = 5, acc = 51.1398%
 30000 samples at 60000, 100000, 160000 (posterior plot: first 1000 samples truncated)

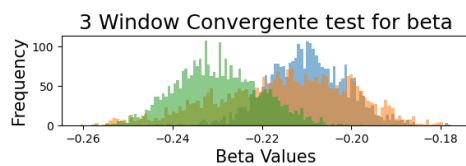
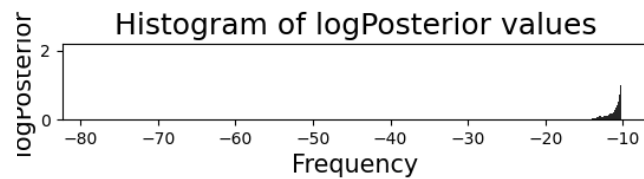
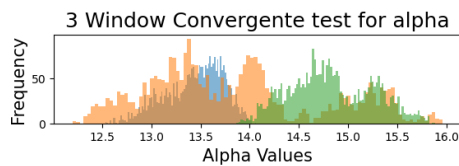
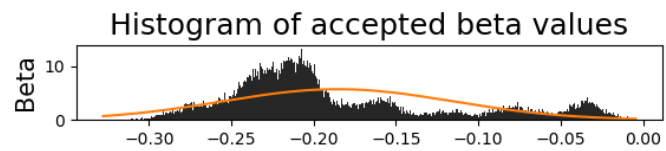
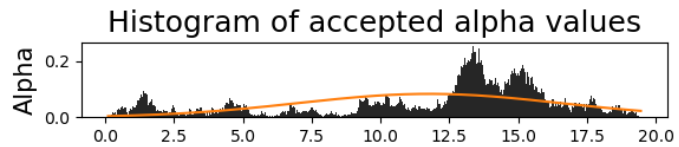
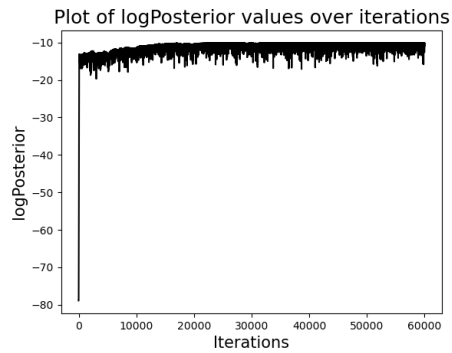
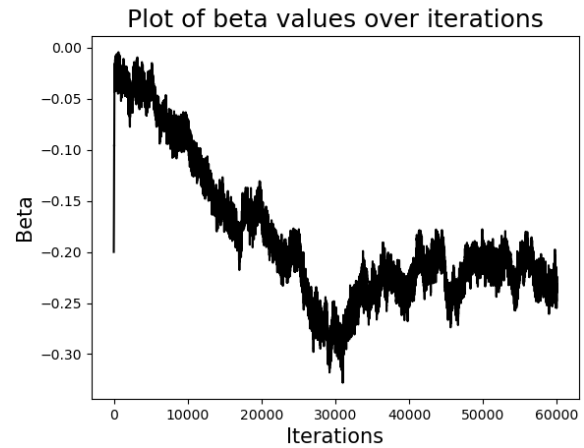
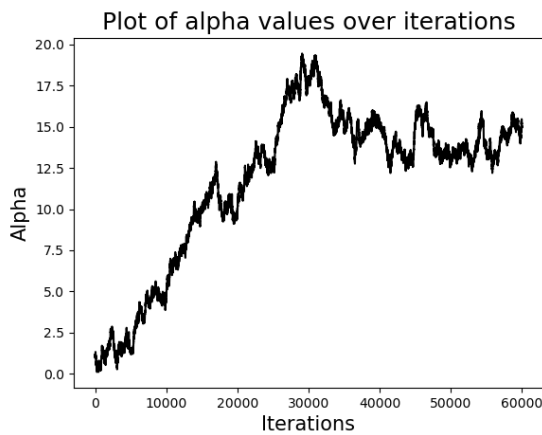


RW

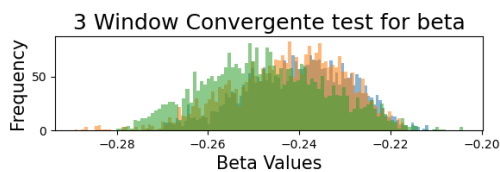
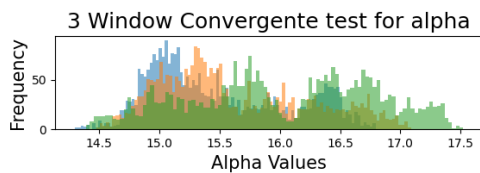
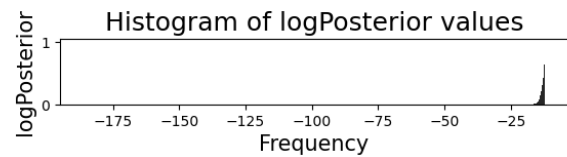
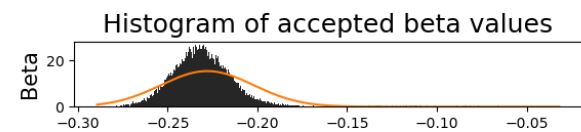
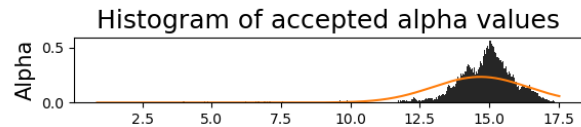
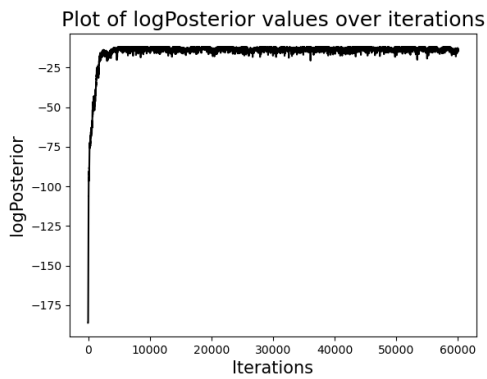
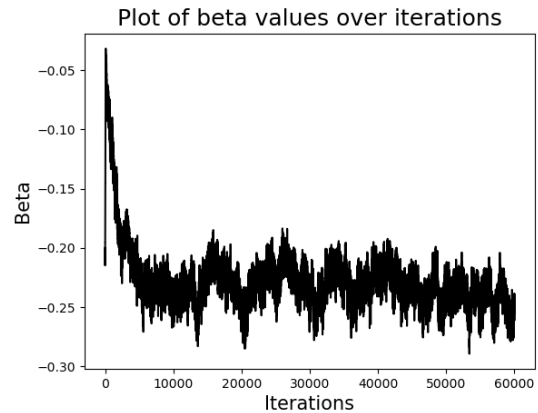
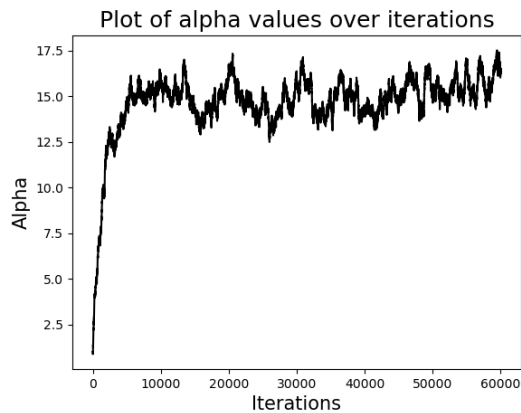
Without priorStd Dev alpha prop = **0.045**; Std Dev beta prop = **0.00495** ($0.11 \times \text{stdDev_alpha_prop}$)initialAlphaSeed = **1**; initialBetaSeed = **-0.2**;

Iteration = 60000, 3000 samples at 49000, 53000, 57000 for convergence

Alpha values range upto 18.5, Acc rate: 80.175%

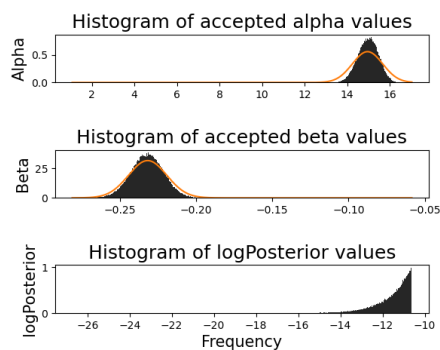
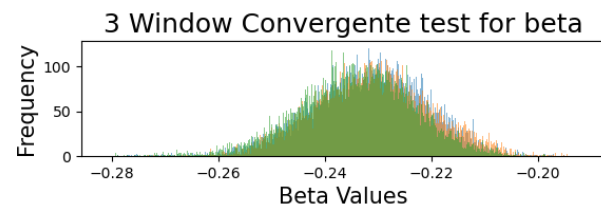
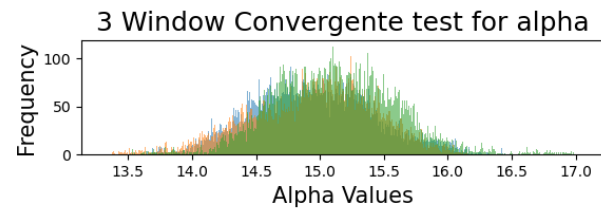
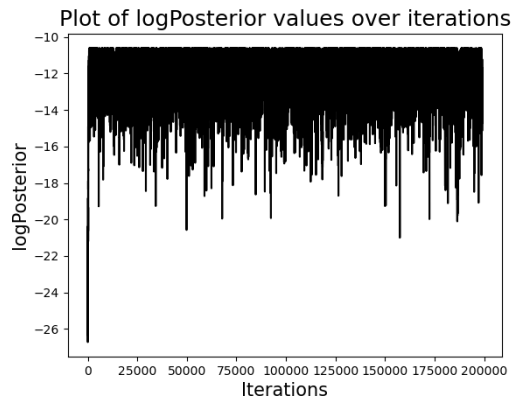
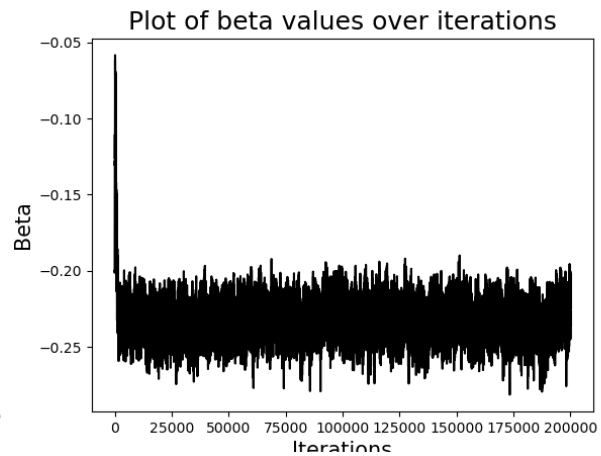
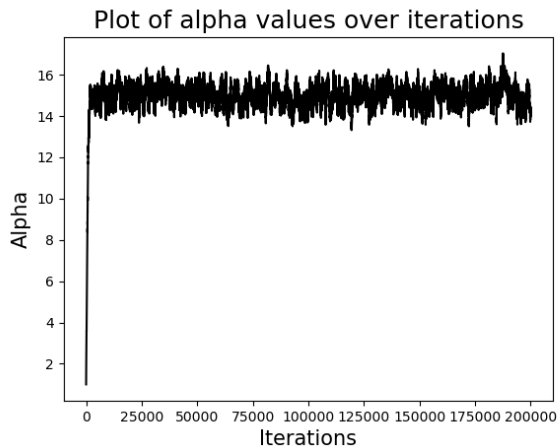


RW

Strict prior(v2 folder in pythonWorkspace) - same configuration as beforeStd Dev α prop = 0.045; Std Dev β prop = 0.00495 ($0.11 \times \text{stdDev_}\alpha\text{_prop}$) $\text{initialAlphaSeed} = 1$; $\text{initialBetaSeed} = -0.2$; $\text{mean_}\alpha\text{_prior} = 15$, $\text{stdDev_}\alpha\text{_prior} = 1$, $\text{mean_}\beta\text{_prior} = -0.235$, $\text{stdDev_}\beta\text{_prior} = 1$ $\text{Iteration} = 60000$, 3000 samples at 49000, 53000, 57000, Acc rate: 80.85%, [Alpha values range upto 17.5]

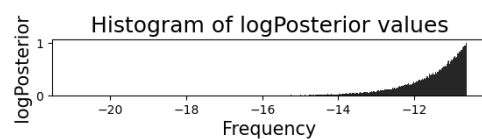
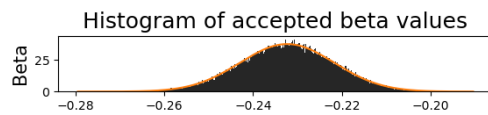
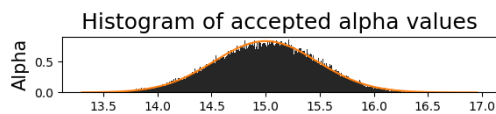
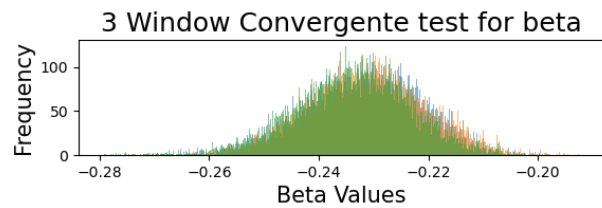
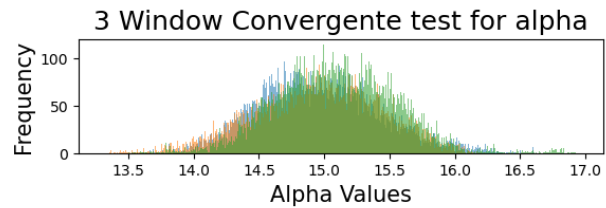
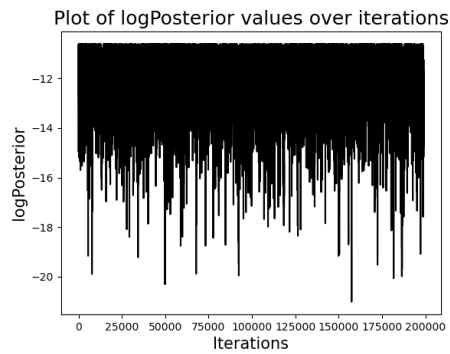
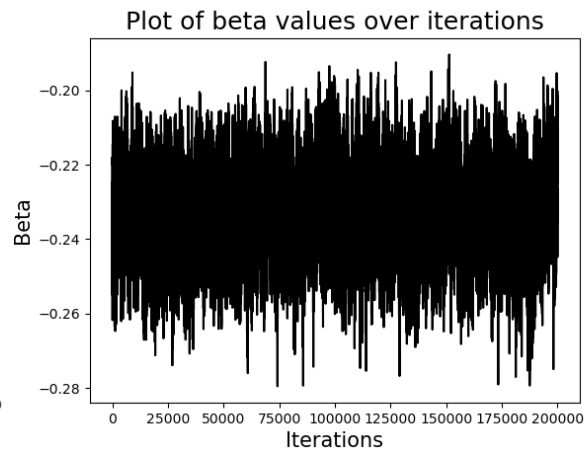
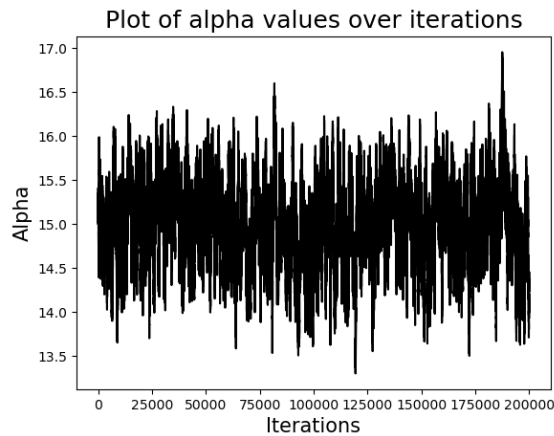
RW

initialAlphaSeed = 1 initialBetaSeed = -0.2, Iteration = 200000
 stdDev_alpha_prop = 0.045 stdDev_beta_prop = 0.00495 (0.11*stdDev_alpha_prop)
 mean_alpha_prior = 15 stdDev_alpha_prior = 0.5. [seed for random generator=0]
 mean_beta_prior = -0.235. stdDev_beta_prior = 0.5
 30000 samples at 140000, 200000, 260000 (posterior plot: first 1000 samples truncated)



RW

initialAlphaSeed = 15 initialBetaSeed = -0.22, Iteration = 200000 (3 times more than previous)
 stdDev_alpha_prop = 0.045 stdDev_beta_prop = 0.00495 (0.11*stdDev_alpha_prop)
 mean_alpha_prior = 15 stdDev_alpha_prior = 0.5. [seed for random generator=0]
 mean_beta_prior = -0.235. stdDev_beta_prior = 0.5 [acc rate ~ 80%]
 30000 samples at 140000, 200000, 260000 (posterior plot: first 1000 samples truncated)
 95% HPD - Alpha: [14.05707892, 15.89744148], Beta: [-0.25327694, -0.21155802]

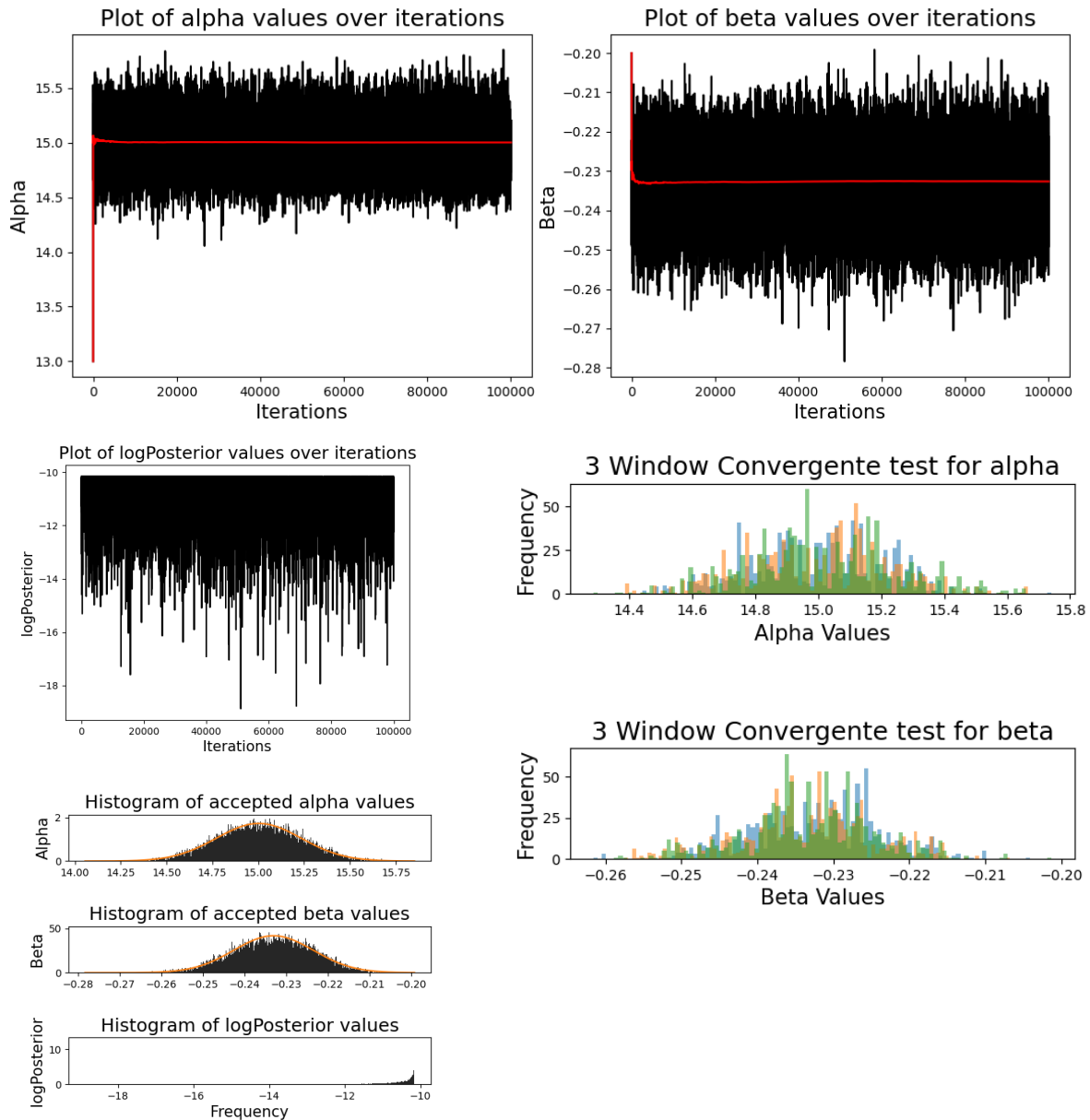


IS

Without prior(v1 folder in pythonWorkspace)mean_alpha_prop=15, Std Dev *alpha* prop = 0.3;mean_beta_prop = -0.235, Std Dev *beta* prop = 0.03*initialAlphaSeed* = 13; *initialBetaSeed* = -0.2;*Iteration* = 30000, 1000 samples at 19000, 23000, 27000 for convergence

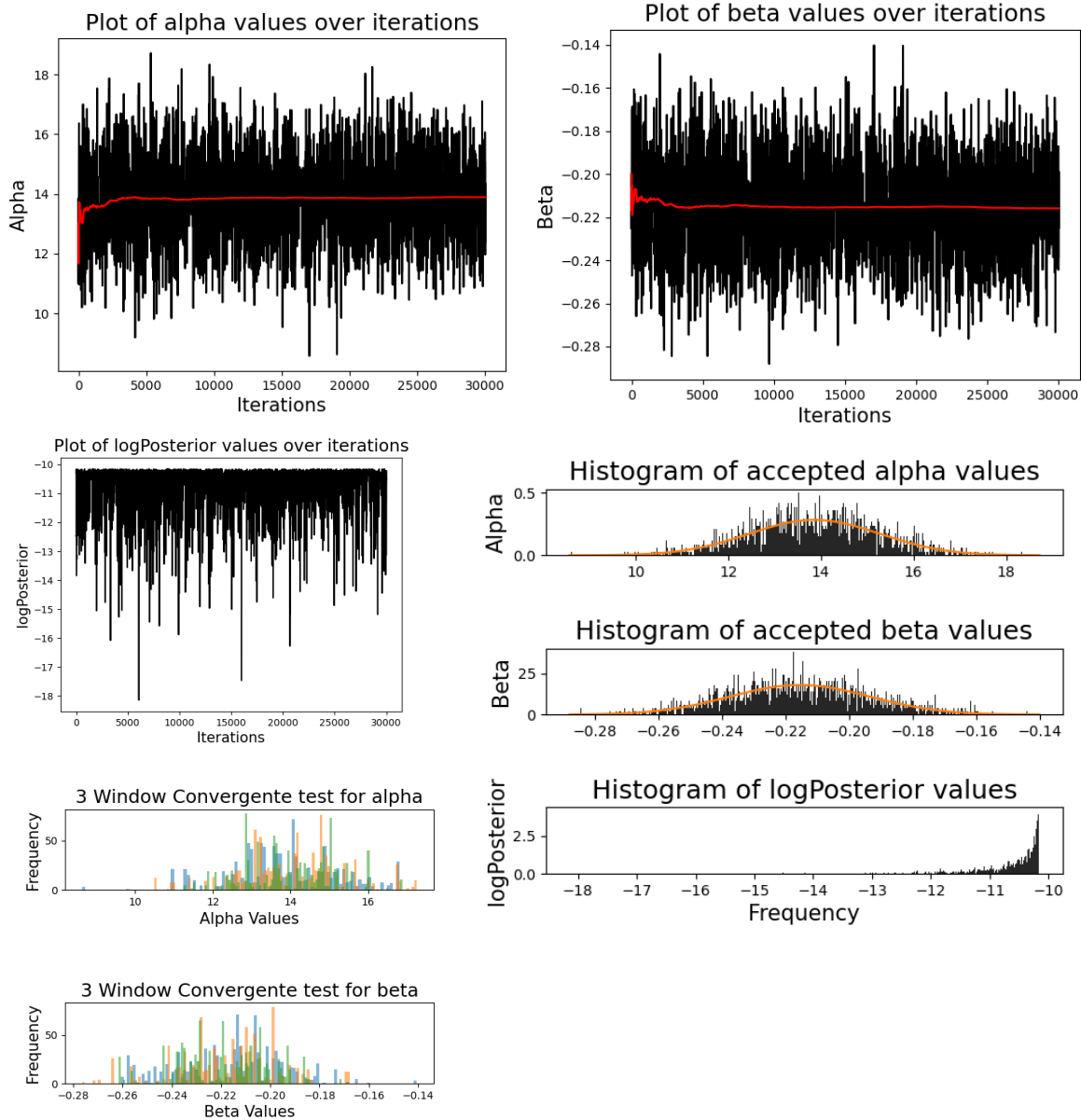
Alpha values range upto <16, Acc rate: 27.956%,

95% HPD, Alpha: [14.54871437, 15.442416], Beta: [-0.25162499, -0.215028]



IS

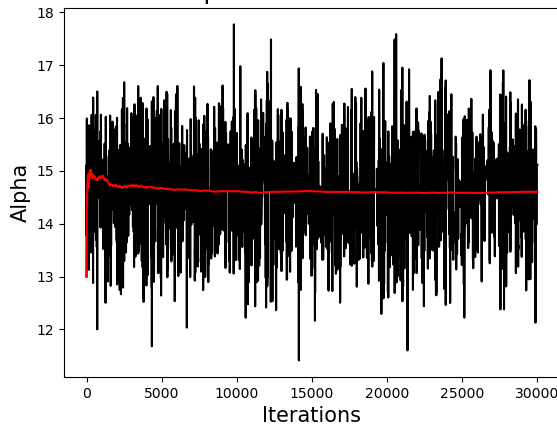
Without prior(v3 folder in pythonWorkspace)
 mean_alpha_prop=**14**, Std Dev alpha prop = **2**;
 mean_beta_prop = **-0.2**, Std Dev beta prop = **0.07**
 initialAlphaSeed = 13; initialBetaSeed = -0.2;
 Iteration = 30000, 1000 samples at 19000, 23000, 27000 for convergence
 Alpha values range upto <18.5, Acc rate: 11.1%



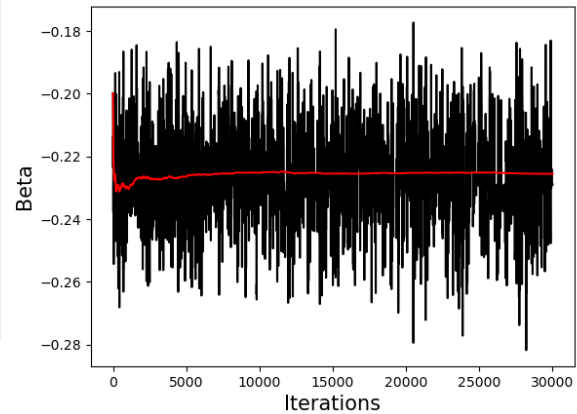
IS

mean_alpha_prop=**14**, Std Dev *alpha prop* = **2**; mean_beta_prop = **-0.2**, Std Dev beta prop = **0.07**
 mean_alpha_prior=15, stdDev_alpha_prior=1
 mean_beta_prior = -0.235, stdDev_beta_prior = 5
initialAlphaSeed = 13; *initialBetaSeed* = -0.2;
Iteration = 30000, 1000 samples at 19000, 23000, 27000 for convergence
 Alpha values range upto <18, Acc rate: 11.85%
 95% HPD Alpha: [11.34360346, 16.6846409], Beta: [-0.25673242, -0.174463]

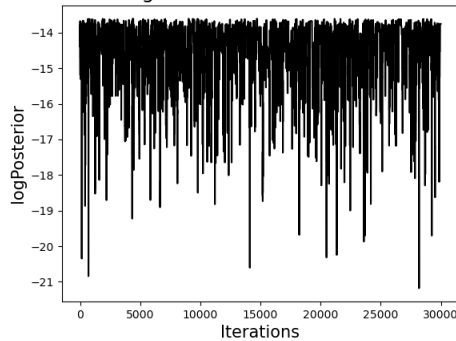
Plot of alpha values over iterations



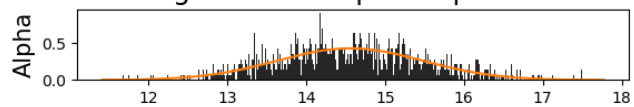
Plot of beta values over iterations



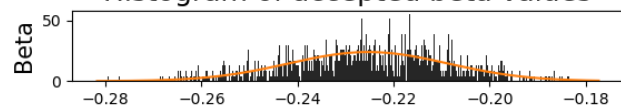
Plot of logPosterior values over iterations



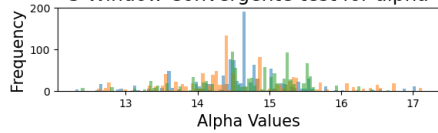
Histogram of accepted alpha values



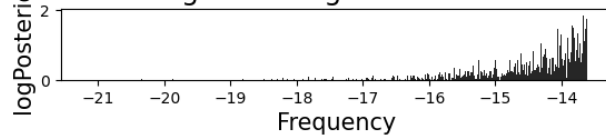
Histogram of accepted beta values



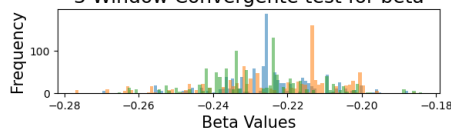
3 Window Convergente test for alpha



Histogram of logPosterior values

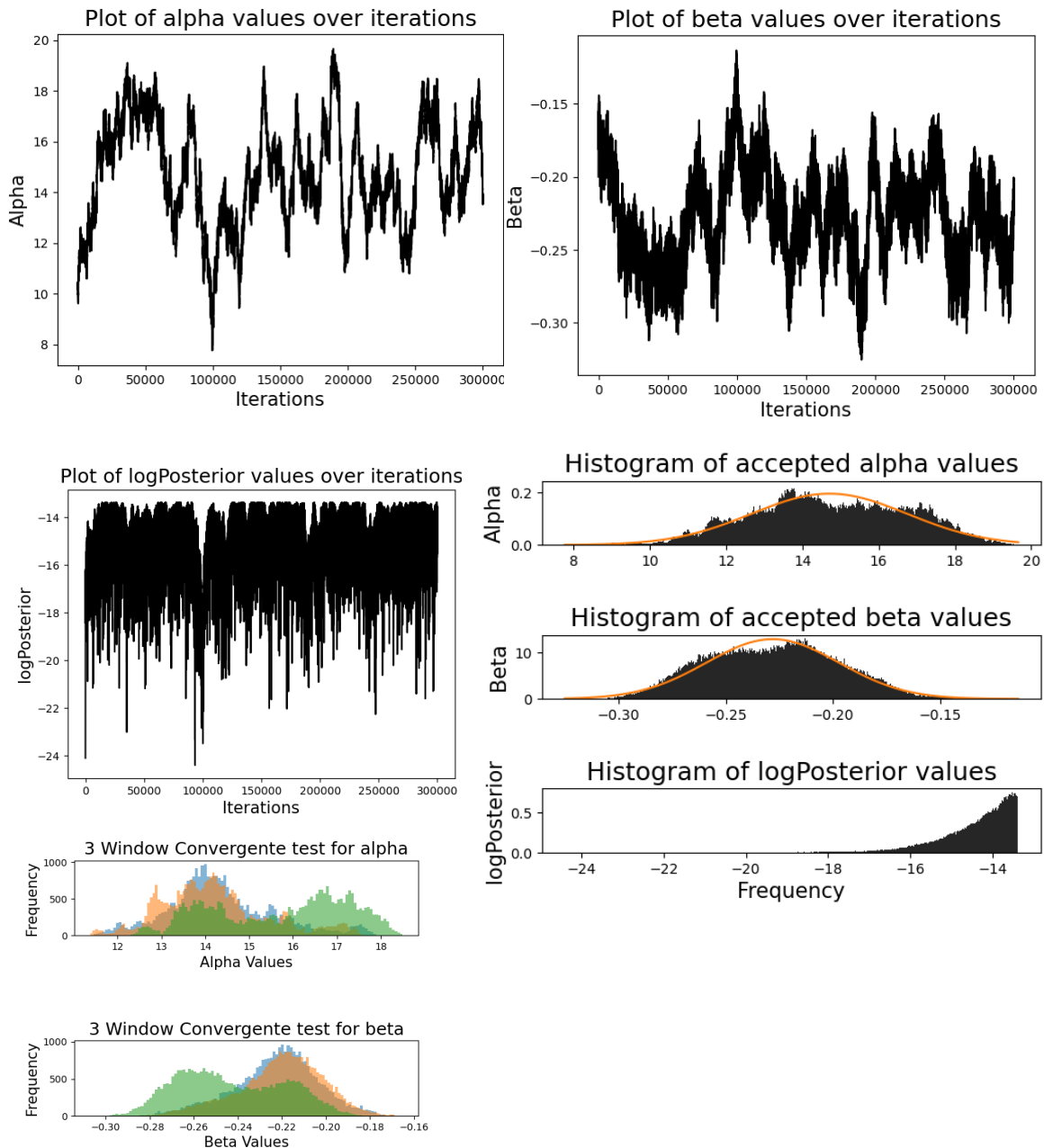


3 Window Convergente test for beta



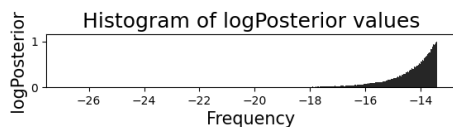
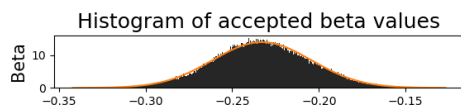
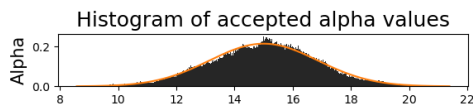
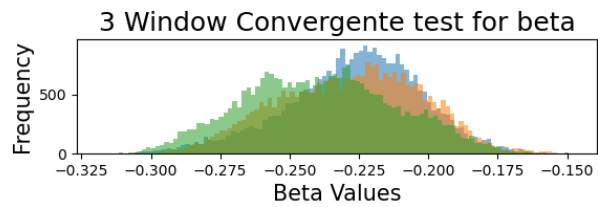
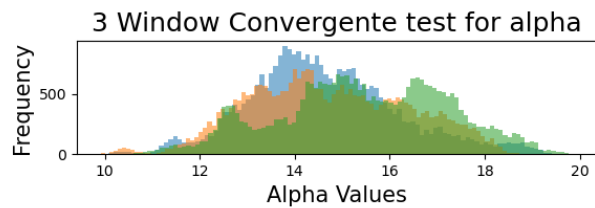
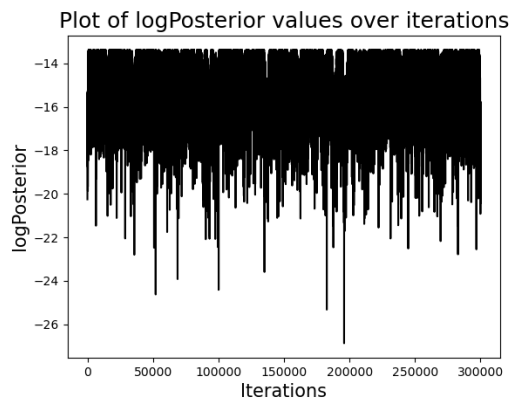
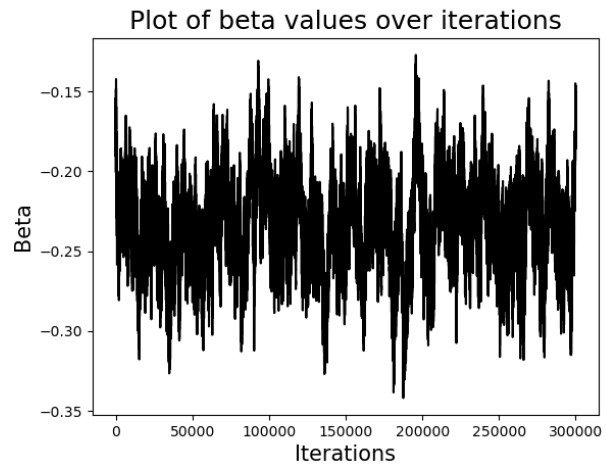
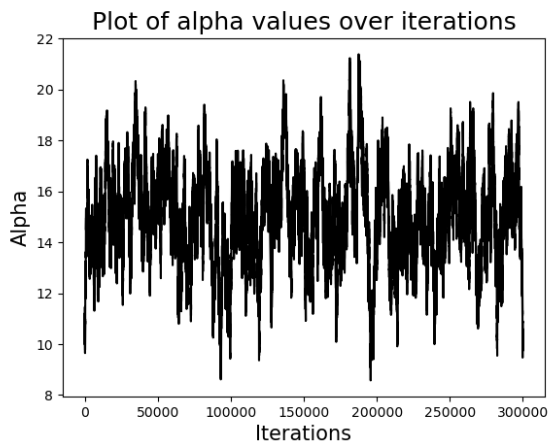
RWG

initialAlphaSeed = 10 initialBetaSeed = -0.2, Iteration = 300000
 stdDev_alpha_prop = 0.032 stdDev_beta_prop = 0.0036
 mean_alpha_prior = 15 stdDev_alpha_prior = 2. [seed for random generator=0]
 mean_beta_prior = -0.23. stdDev_beta_prior = 2
 30000 samples at 150000, 200000, 250000 acc = 98%



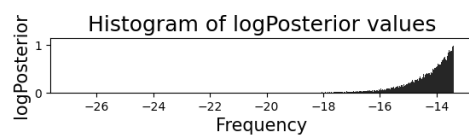
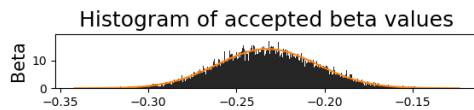
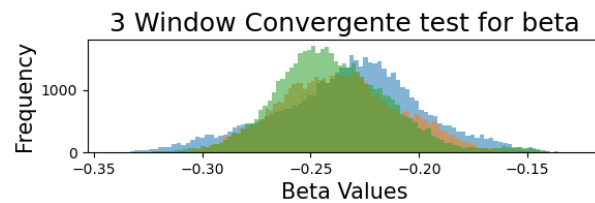
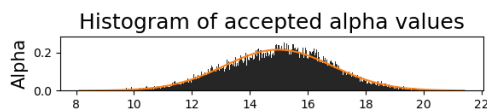
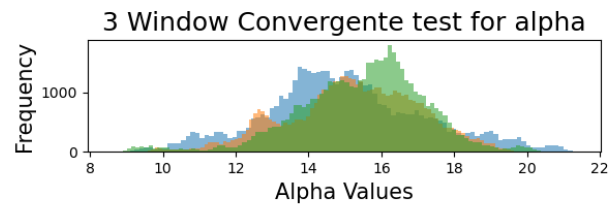
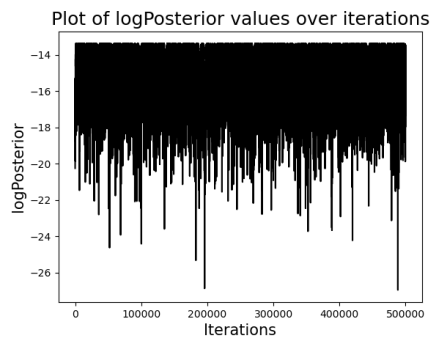
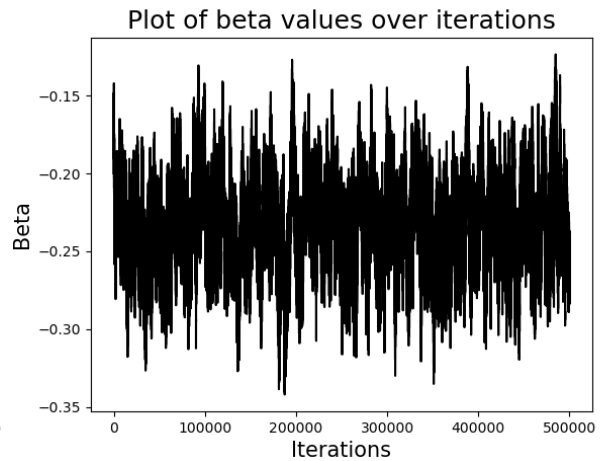
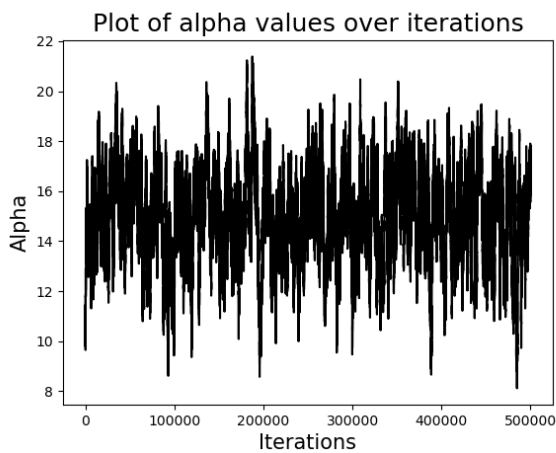
RWG

initialAlphaSeed = 10 initialBetaSeed = -0.2, Iteration = 300000
 stdDev_alpha_prop = **0.08** stdDev_beta_prop = **0.0096** ((0.12*stdDev_alpha_prop))
 mean_alpha_prior = 15 stdDev_alpha_prior = 2. [seed for random generator=0]
 mean_beta_prior = -0.23. stdDev_beta_prior = 2, acc = 95%
 30000 samples at 150000, 200000, 250000



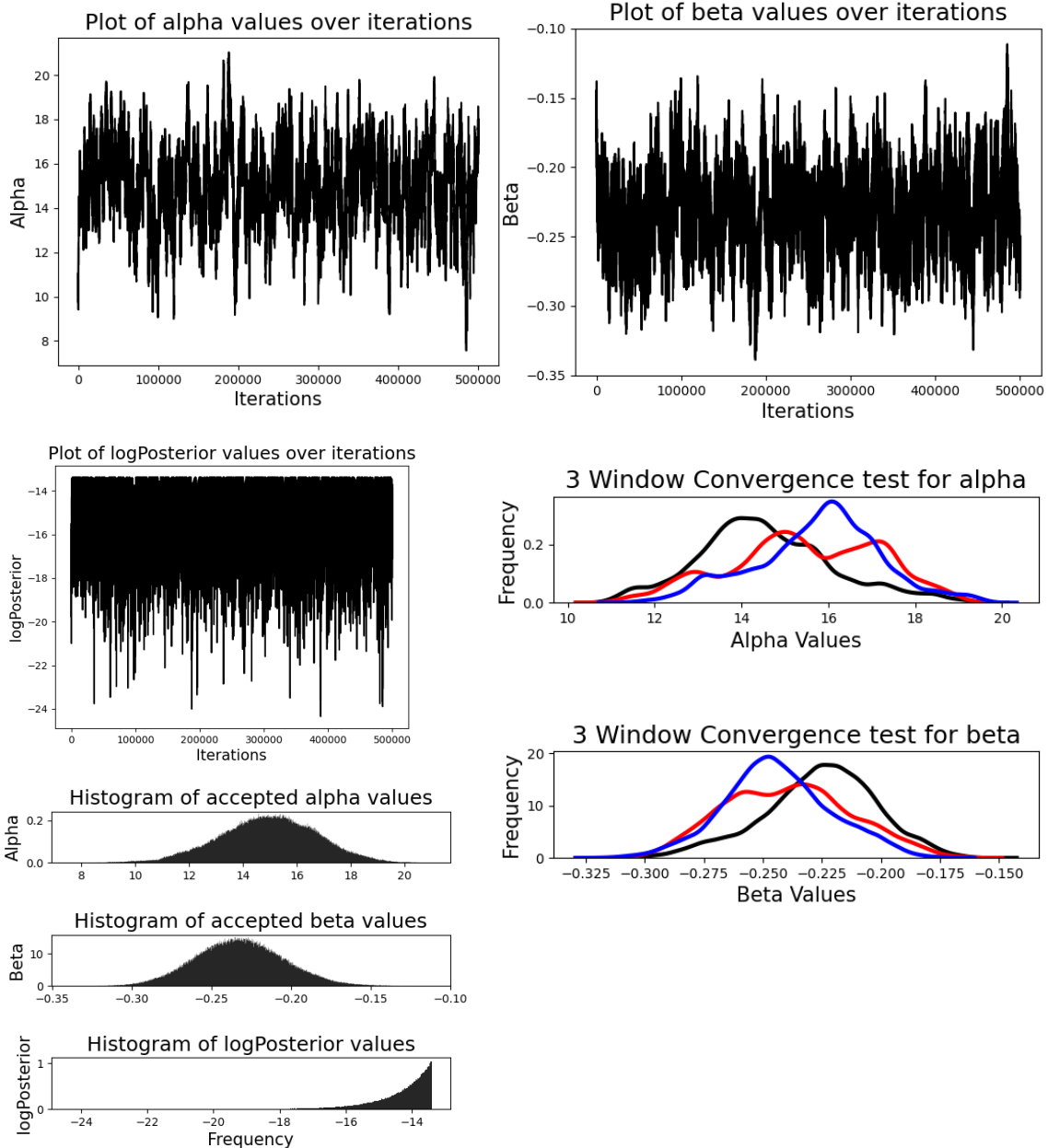
RWG

initialAlphaSeed = 10 initialBetaSeed = -0.2, Iteration = **500000**
 stdDev_alpha_prop = 0.08 stdDev_beta_prop = 0.0096 ((0.12*stdDev_alpha_prop))
 mean_alpha_prior = 15 stdDev_alpha_prior = 2 [seed for random generator=0]
 mean_beta_prior = -0.23. stdDev_beta_prior = 2, acc = 95.2%
50000 samples at **150000, 250000, 350000**



RWG

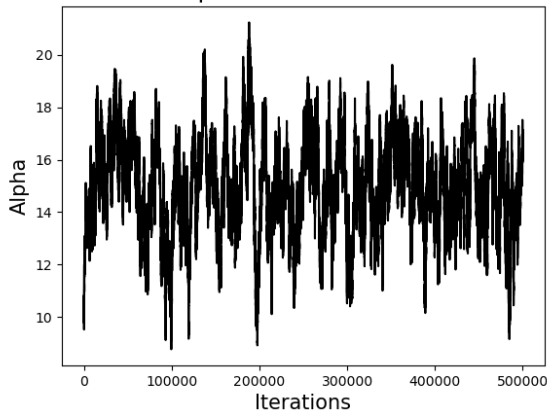
initialAlphaSeed = 10 initialBetaSeed = -0.2, Iteration = 500000
 stdDev_alpha_prop = **0.072** stdDev_beta_prop = **0.0108** ((0.15*stdDev_alpha_prop))
 mean_alpha_prior = 15 stdDev_alpha_prior = 2 [seed for random generator=0]
 mean_beta_prior = -0.23. stdDev_beta_prior = 2, acc = 95.7%
 50000 samples at 150000, 250000, 350000



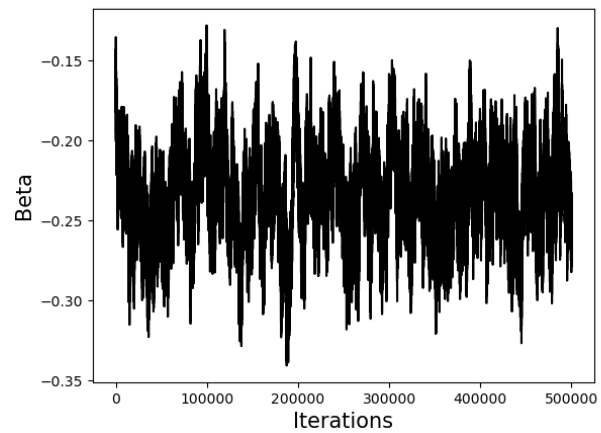
RWG

initialAlphaSeed = 10 initialBetaSeed = -0.2, Iteration = 500000
 stdDev_alpha_prop = **0.055** stdDev_beta_prop = **0.01375** ((0.25*stdDev_alpha_prop))
 mean_alpha_prior = 15 stdDev_alpha_prior = 2 [seed for random generator=0]
 mean_beta_prior = -0.23. stdDev_beta_prior = 2, acc = 96.7%
 50000 samples at 150000, 250000, 350000

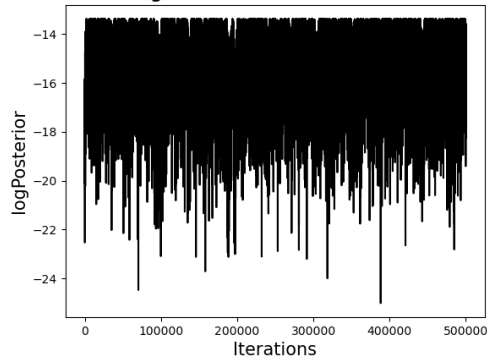
Plot of alpha values over iterations



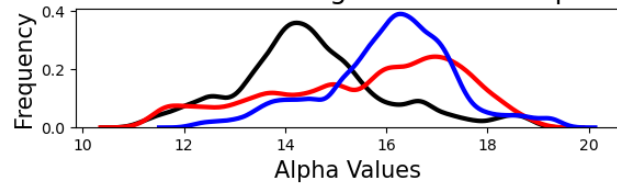
Plot of beta values over iterations



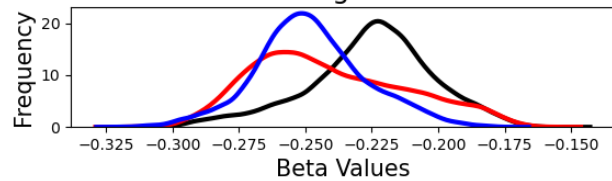
Plot of logPosterior values over iterations



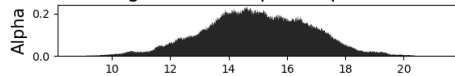
3 Window Convergence test for alpha



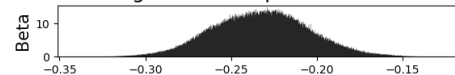
3 Window Convergence test for beta



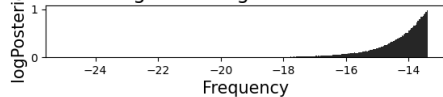
Histogram of accepted alpha values



Histogram of accepted beta values

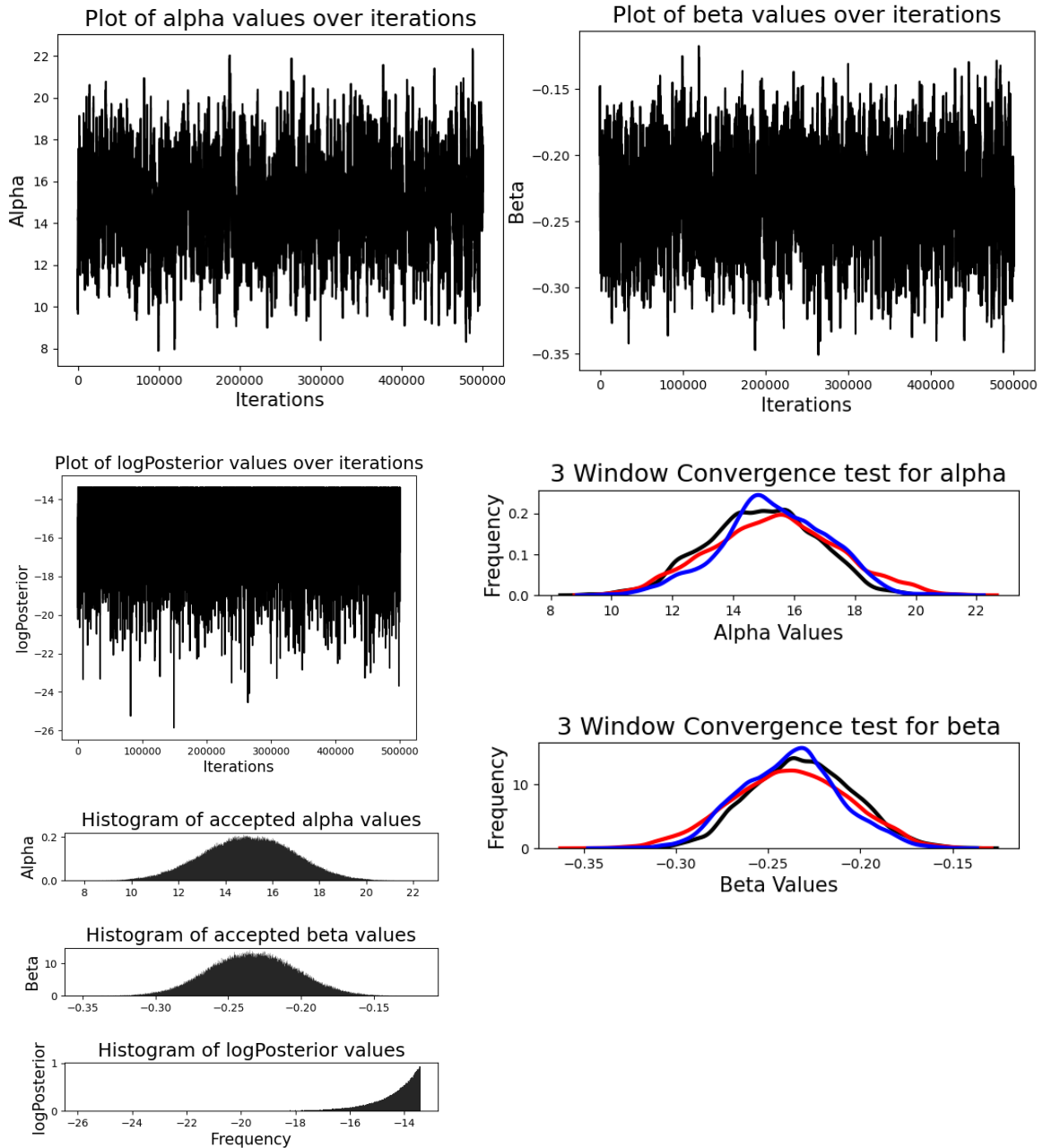


Histogram of logPosterior values



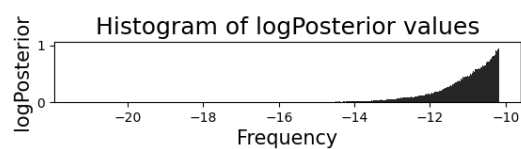
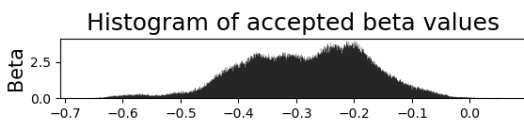
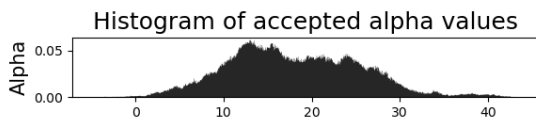
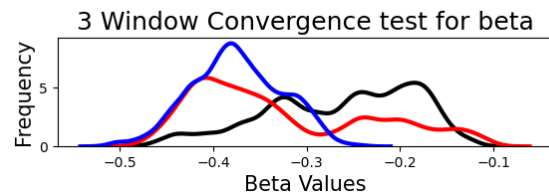
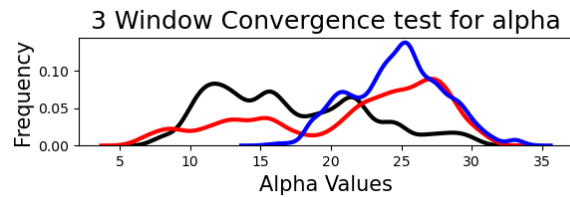
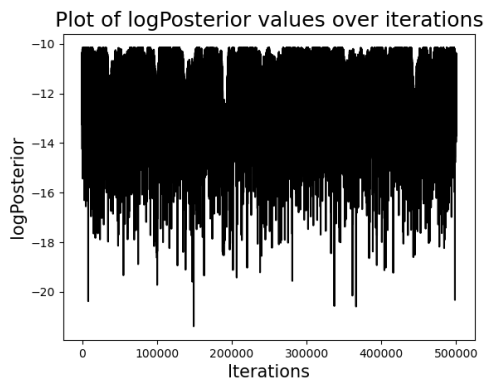
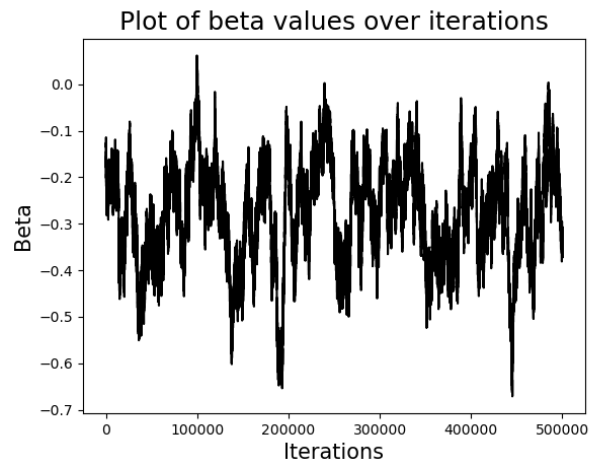
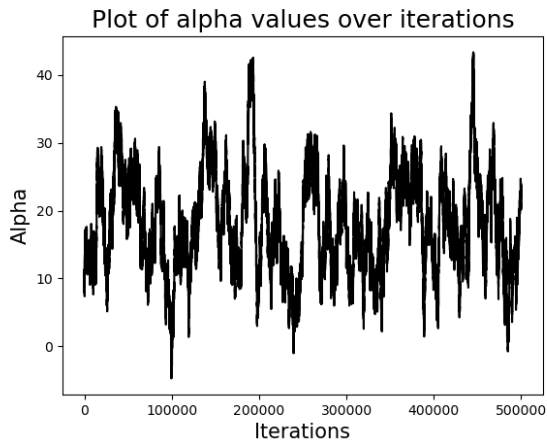
RWG

initialAlphaSeed = 10 initialBetaSeed = -0.2, Iteration = 500000
 stdDev_alpha_prop = **0.155** stdDev_beta_prop = **0.02015** ((0.13*stdDev_alpha_prop))
 mean_alpha_prior = 15 stdDev_alpha_prior = 2 [seed for random generator=0]
 mean_beta_prior = -0.23. stdDev_beta_prior = 2, acc = 90.8%
 50000 samples at 150000, 250000, 350000
 95% HPD Alpha: [11.25775957, 18.76617123], Beta: [-0.29005272 -0.17476262]



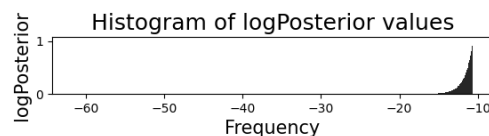
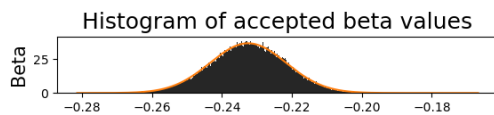
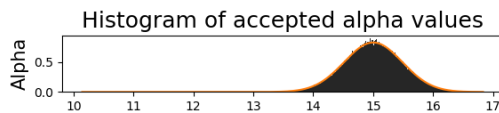
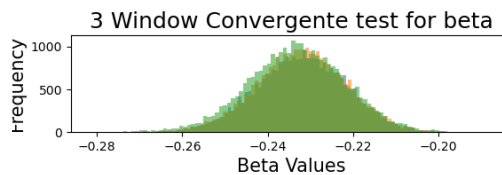
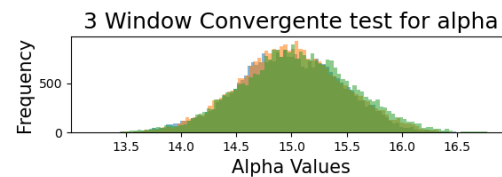
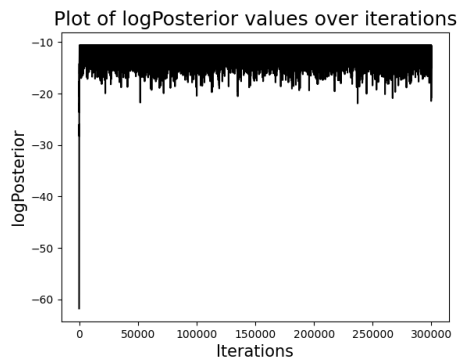
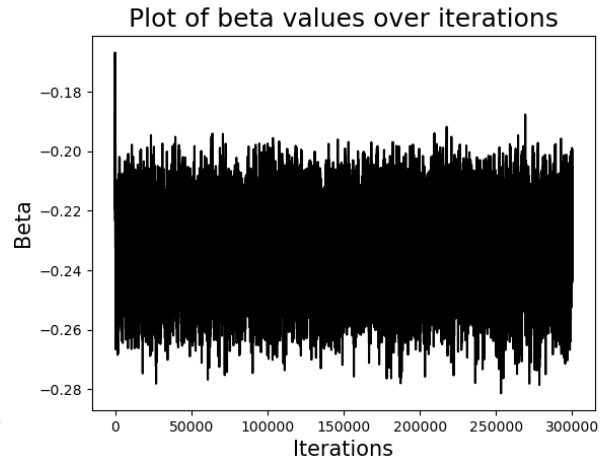
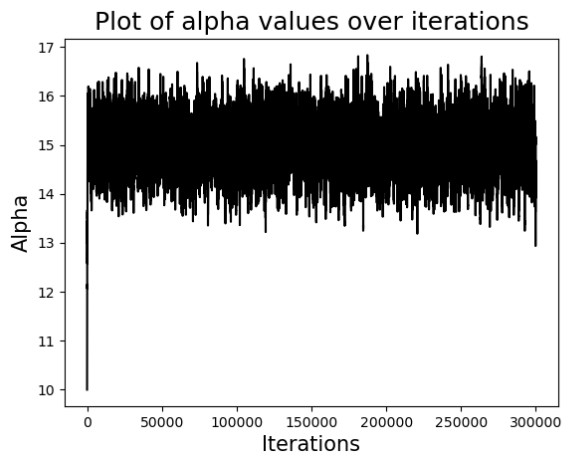
RWG

initialAlphaSeed = 10 initialBetaSeed = -0.2, Iteration = 500000
 stdDev_alpha_prop = **0.155** stdDev_beta_prop = **0.02015** ((0.13*stdDev_alpha_prop))
 acc = 90.8%, 50000 samples at 150000, 250000, 350000 [seed for random generator=0]



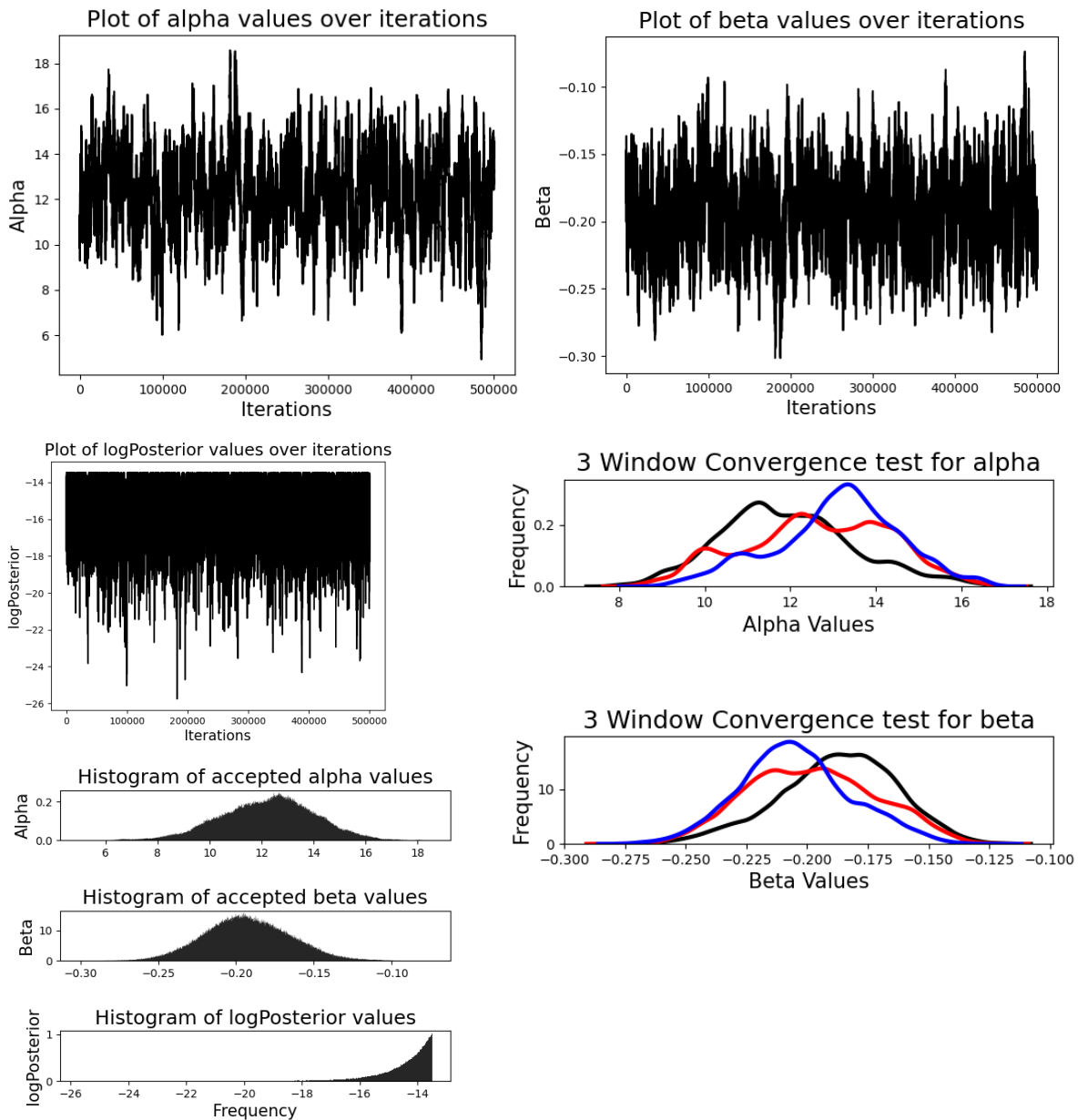
RWG

initialAlphaSeed = 10 initialBetaSeed = -0.2, Iteration = 300000
 stdDev_alpha_prop = 0.08 stdDev_beta_prop = 0.0096 ((0.12*stdDev_alpha_prop))
 mean_alpha_prior = 15 stdDev_alpha_prior = **0.5** [seed for random generator=0]
 mean_beta_prior = -0.23. stdDev_beta_prior = **0.5**
 30000 samples at 150000, 200000, 250000
 95% HPD Alpha: [14.05557124, 15.94575323], Beta: [-0.2540643, -0.21158743]



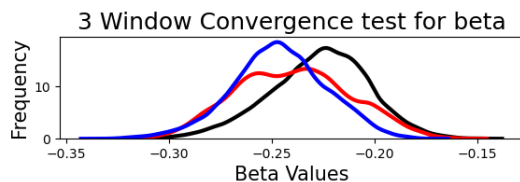
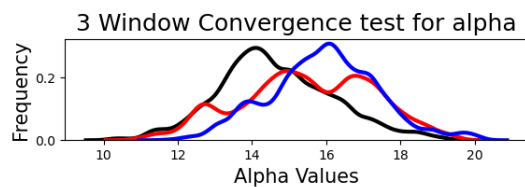
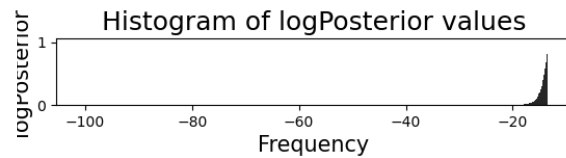
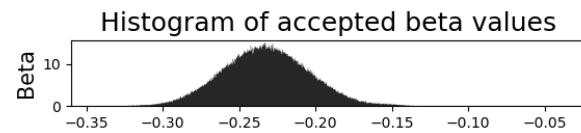
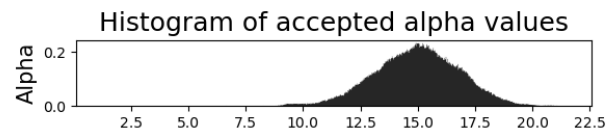
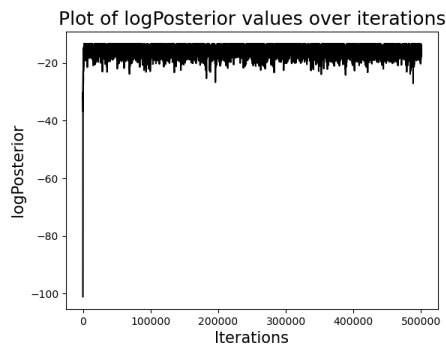
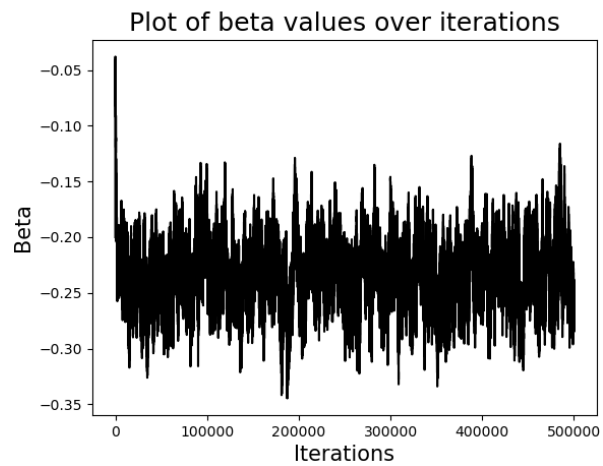
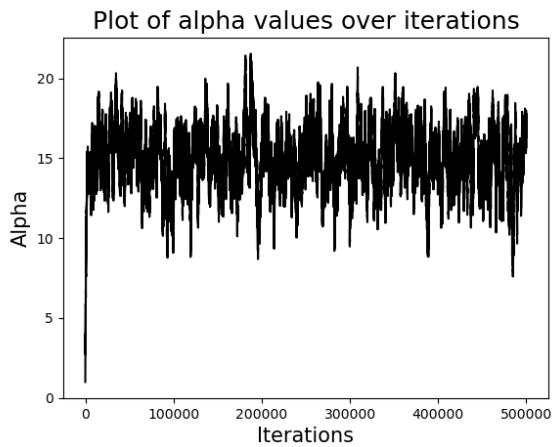
RWG

initialAlphaSeed = 10 initialBetaSeed = -0.2, Iteration = 500000
 stdDev_alpha_prop = 0.08 stdDev_beta_prop = 0.0096 ((0.12*stdDev_alpha_prop))
 mean_alpha_prior = **12** stdDev_alpha_prior = **2** [seed for random generator=0]
 mean_beta_prior = -0.23. stdDev_beta_prior = **2**, acc = 95.08%
 30000 samples at 150000, 250000, 350000



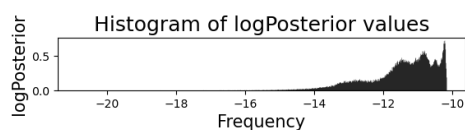
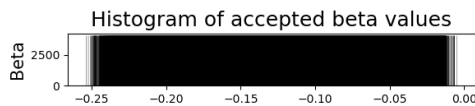
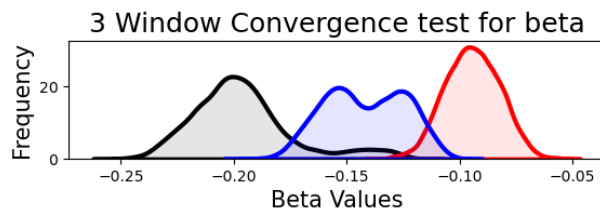
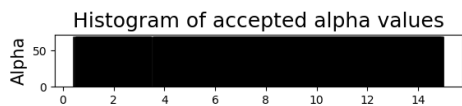
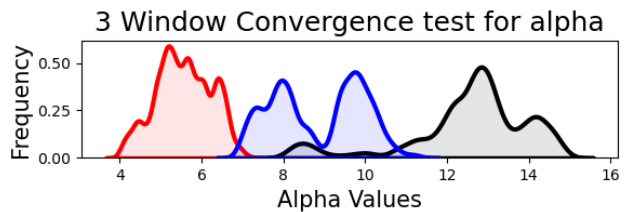
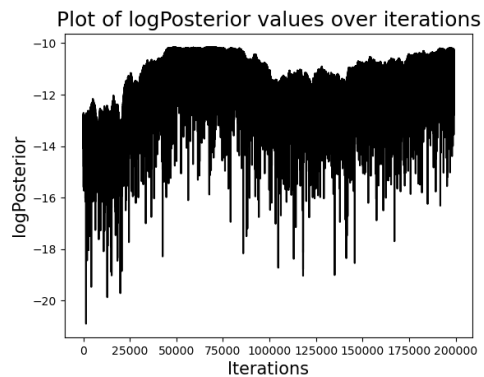
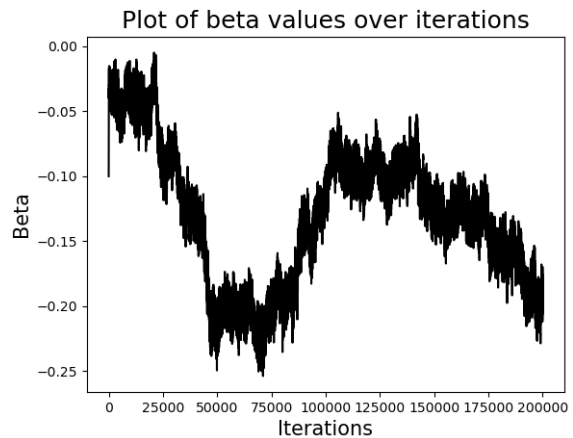
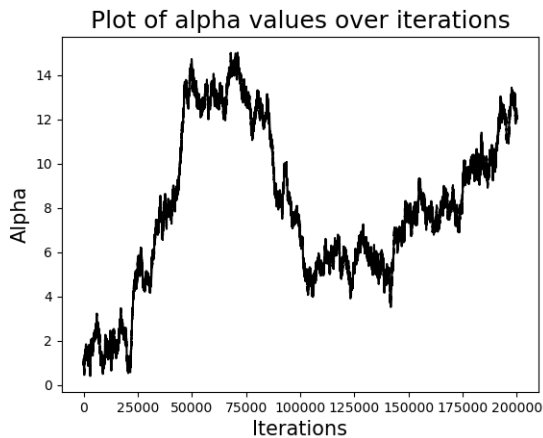
RWG

initialAlphaSeed = 1 initialBetaSeed = -0.2, Iteration = 500000
 stdDev_alpha_prop = 0.08 stdDev_beta_prop = 0.0096 ((0.12*stdDev_alpha_prop))
 mean_alpha_prior = 15 stdDev_alpha_prior = 2 [seed for random generator=0]
 mean_beta_prior = -0.23. stdDev_beta_prior = 2, acc = 95.08%
 30000 samples at 150000, 250000, 350000
 95% HPD Alpha: [11.30940603 18.6207483], Beta: [-0.28870294 -0.17681632]



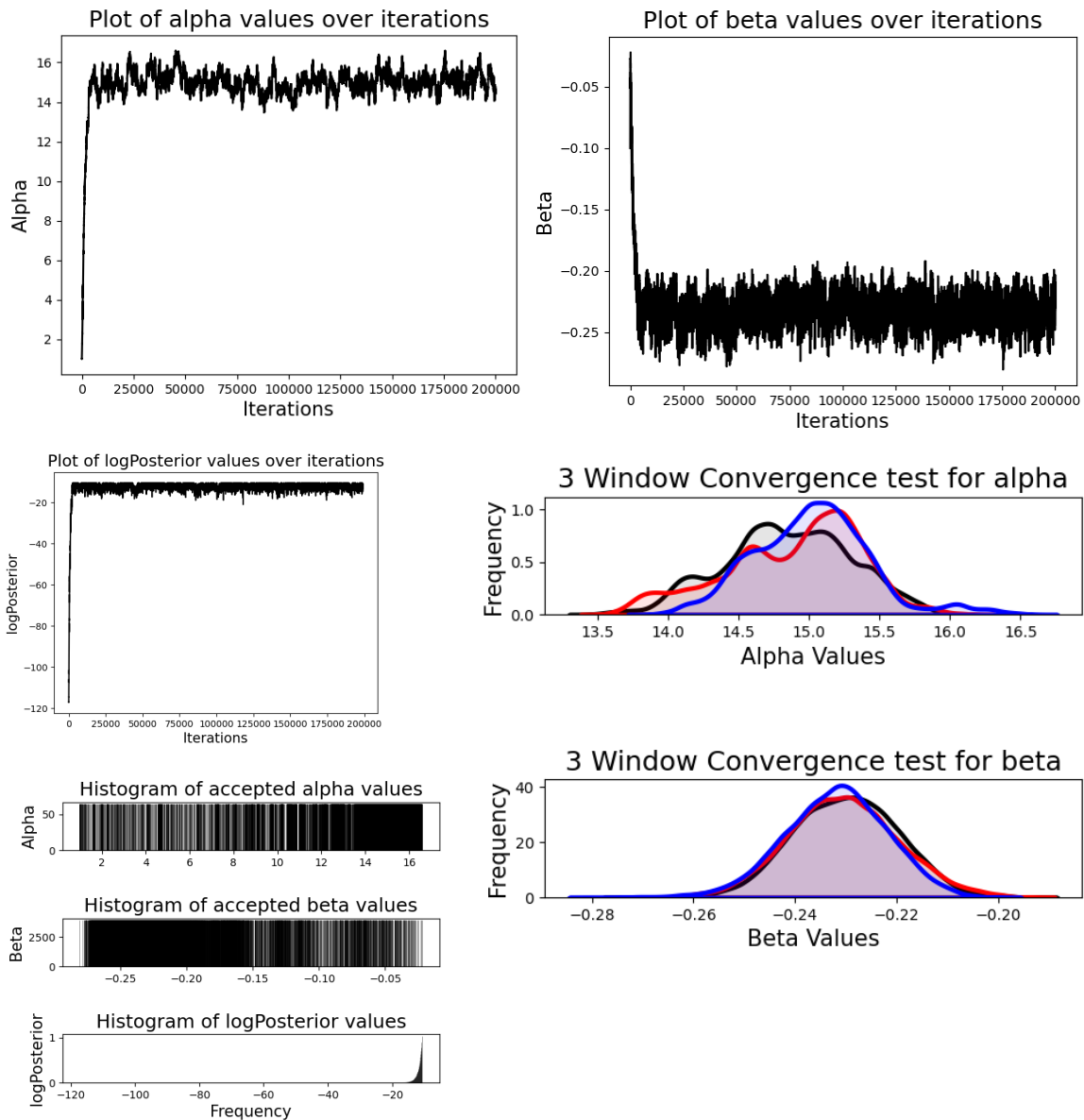
VMH

initialAlphaSeed = 1 initialBetaSeed = -0.1, Iteration = 200000
 stdDev_alpha_prop = 0.025 stdDev_beta_prop = 0.000625 ((0.13*stdDev_alpha_prop))
No prior [seed for random generator=0]
 30000 samples at 60000, 100000, 160000, acc rate = 86.08%



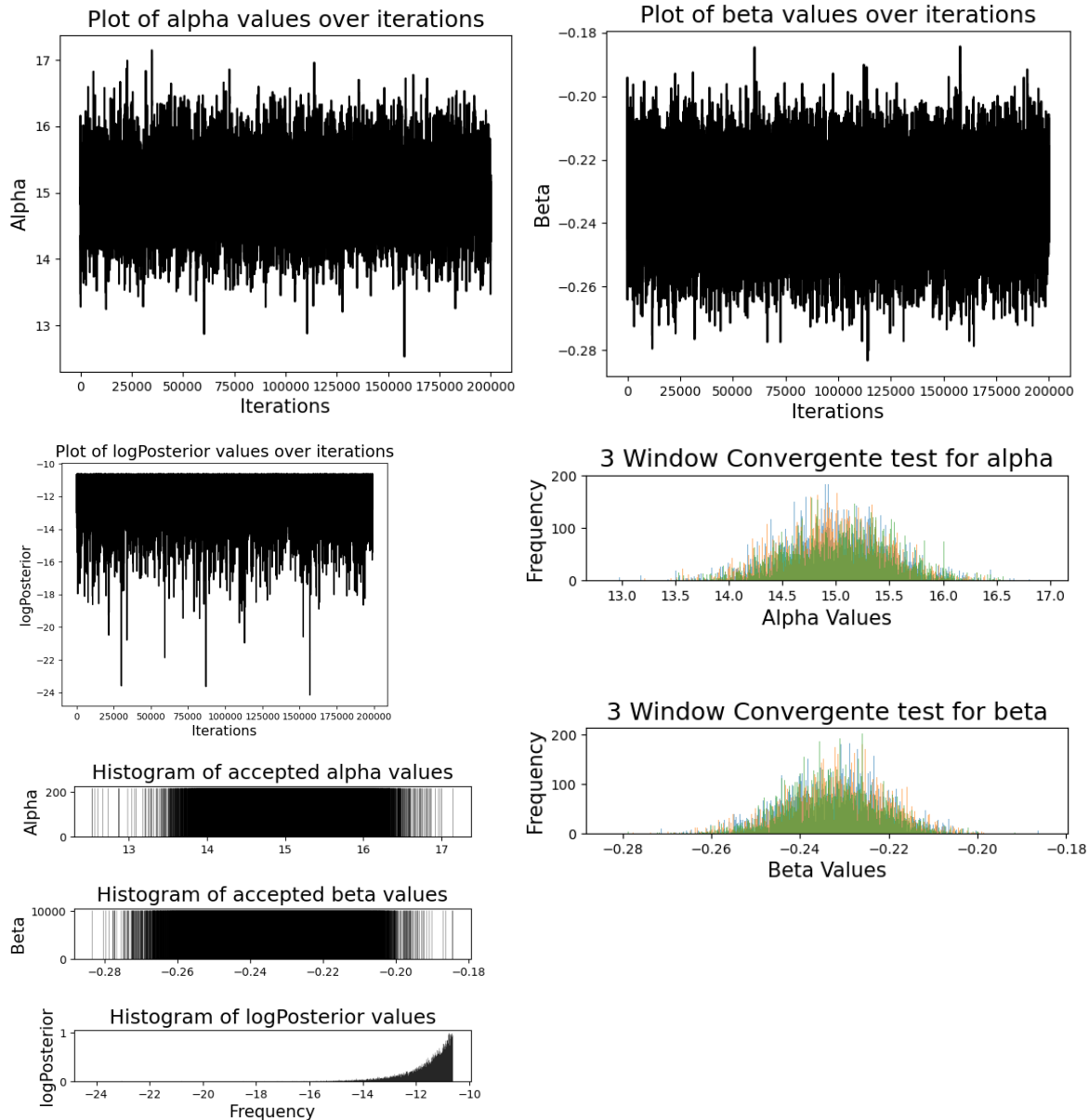
VMH

initialAlphaSeed = 1 initialBetaSeed = -0.1, Iteration = 200000
 stdDev_alpha_prop = 0.025 stdDev_beta_prop = 0.000625 ((0.13*stdDev_alpha_prop))
 mean_alpha_prior = **15** stdDev_alpha_prior = **0.5** [seed for random generator=0]
 mean_beta_prior = **-0.23** stdDev_beta_prior = **0.5**
 30000 samples at 60000, 100000, 160000, acc rate = 87.09%
 95% HPD Alpha: [14.07952157, 15.89743659], Beta: [-0.25392067, -0.07958005]



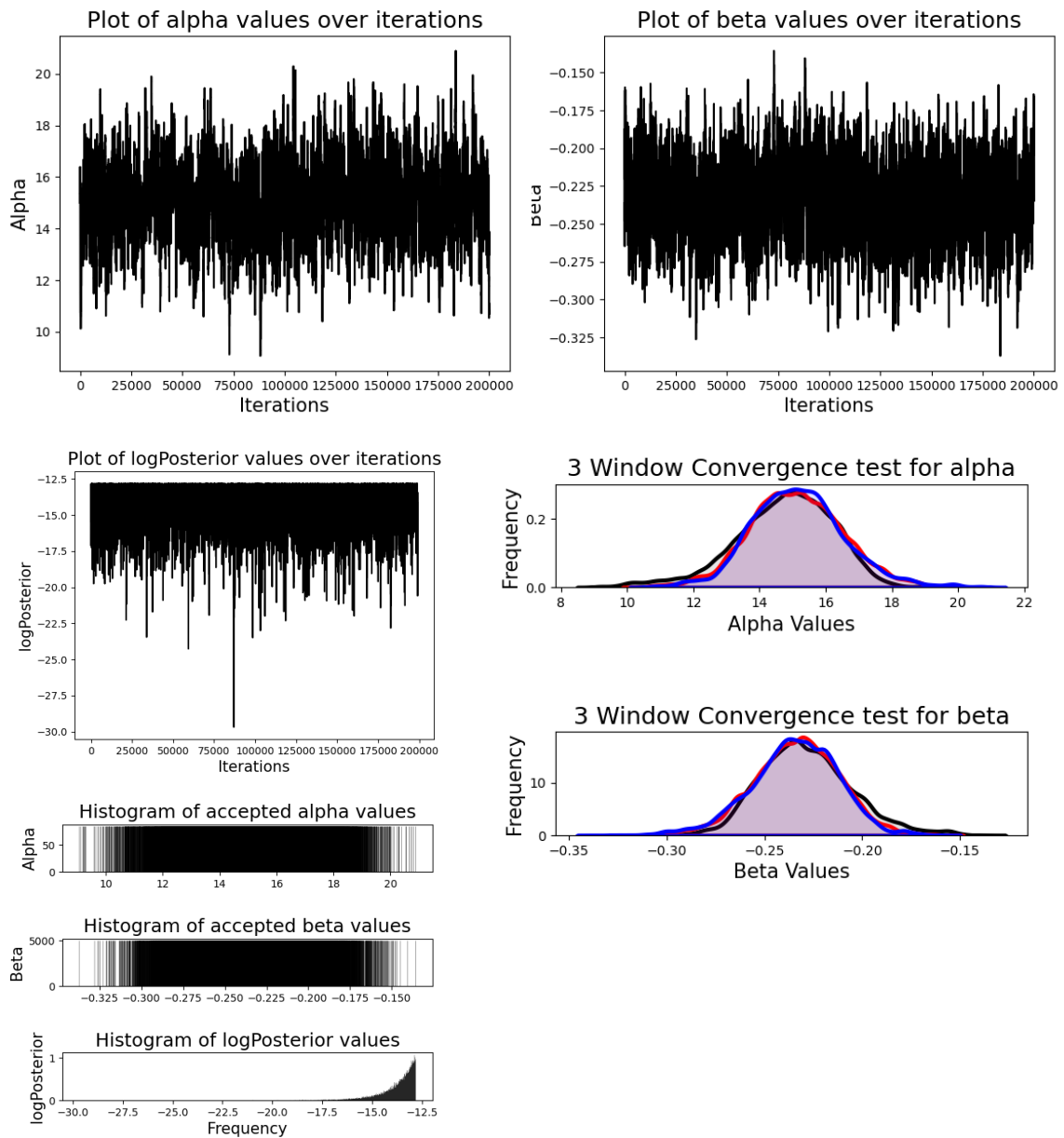
VMH

initialAlphaSeed = 15 initialBetaSeed = -0.22, Iteration = 200000
 stdDev_alpha_prop = 0.35 stdDev_beta_prop = 0.0455 ((0.13*stdDev_alpha_prop))
 mean_alpha_prior = 15 stdDev_alpha_prior = 0.5 [seed for random generator=0]
 mean_beta_prior = -0.23. stdDev_beta_prior = 0.5
 30000 samples at 60000, 100000, 160000, acc rate = ~20%
 95% HPD Alpha: [13.627615, 15.762757], Beta: [-0.2563959, -0.20444]



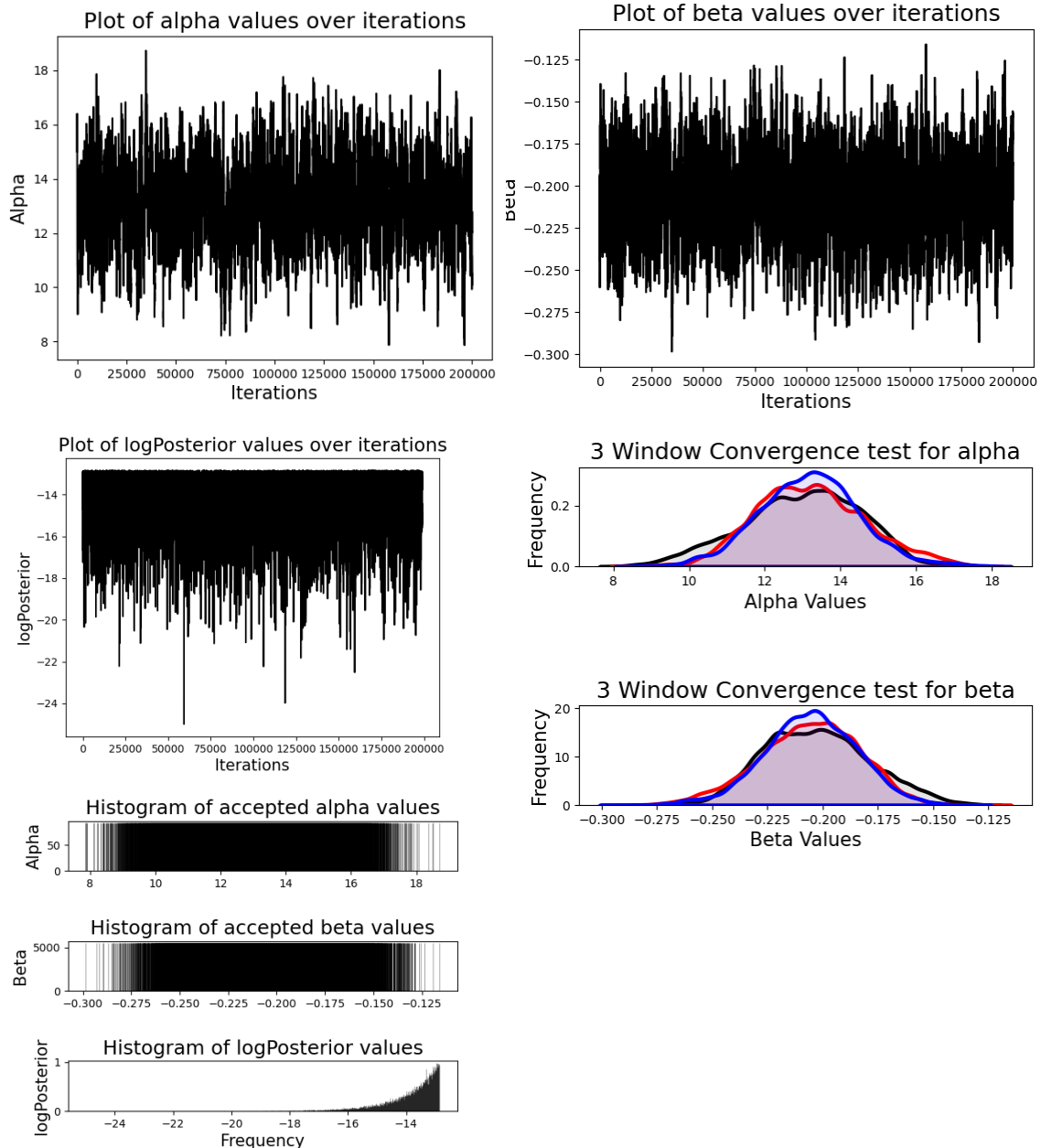
VMH

initialAlphaSeed = 15 initialBetaSeed = -0.22, Iteration = 200000
 stdDev_alpha_prop = 0.35 stdDev_beta_prop = 0.0455 ((0.13*stdDev_alpha_prop))
 mean_alpha_prior = **15** stdDev_alpha_prior = **1.5** [seed for random generator=0]
 mean_beta_prior = **-0.23** stdDev_beta_prior = **1.5**
 30000 samples at 60000, 100000, 160000, acc rate = 21.5%
 95% HPD Alpha: [11.95177, 19.405813], Beta: [-0.308437, -0.1980502]



VMH)

initialAlphaSeed = 15 initialBetaSeed = -0.22, Iteration = 200000
 stdDev_alpha_prop = 0.35 stdDev_beta_prop = 0.0455 ((0.13*stdDev_alpha_prop))
 mean_alpha_prior = **13** stdDev_alpha_prior = **1.5** [seed for random generator=0]
 mean_beta_prior = **-0.23** stdDev_beta_prior = **1.5**
 30000 samples at 60000, 100000, 160000, acc rate = 21.5%



References

- (2018). Cotton yield prediction with markov chain monte carlo-based simulation model integrated with genetic programming algorithm: A new hybrid copula-driven approach. *Agricultural and Forest Meteorology*, 263:428–448.
- Abrahamsen, P. and regnesentral, N. (1997). *A Review of Gaussian Random Fields and Correlation Functions*. Norsk Regnesentral/Norwegian Computing Center.
- Adams, R. P. and Stegle, O. (2008). Gaussian process product models for nonparametric nonstationarity. In *Proceedings of the 25th international conference on Machine learning*, pages 1–8.
- Ademujimi, T. T., Brundage, M. P., and Prabhu, V. V. (2017). A review of current machine learning techniques used in manufacturing diagnosis. In *Advances in Production Management Systems. The Path to Intelligent, Collaborative and Sustainable Manufacturing: IFIP WG 5.7 International Conference, APMS 2017, Hamburg, Germany, September 3-7, 2017, Proceedings, Part I*, pages 407–415. Springer.
- Alanazi, A. (2022). Using machine learning for healthcare challenges and opportunities. *Informatics in Medicine Unlocked*, 30:100924.
- Alie, R., Stephens, D. A., and Schmidt, A. M. (2024). Computational considerations for the linear model of coregionalization. *arXiv preprint arXiv:2402.08877*.
- Almon, S. (1965). The distributed lag between capital appropriations and expenditures. *Econometrica: Journal of the Econometric Society*, pages 178–196.
- Alvarez, M. A. and Lawrence, N. D. (2011). Computationally efficient convolved multiple output gaussian processes. *The Journal of Machine Learning Research*, 12:1459–1500.
- Alvarez, M. A., Rosasco, L., Lawrence, N. D., et al. (2012). Kernels for vector-valued functions: A review. *Foundations and Trends® in Machine Learning*, 4(3):195–266.
- Ba, S. and Joseph, V. R. (2012). Composite gaussian process models for emulating expensive functions. *The Annals of Applied Statistics*, pages 1838–1860.
- Bach, F. R., Lanckriet, G. R., and Jordan, M. I. (2004). Multiple kernel learning, conic duality, and the smo algorithm. In *Proceedings of the twenty-first international conference on Machine learning*, page 6.
- Banerjee, S., Carlin, B. P., and Gelfand, A. E. (2014). Hierarchical modeling and analysis for spatial data.

- Begoli, E., Bhattacharya, T., and Kusnezov, D. (2019). The need for uncertainty quantification in machine-assisted medical decision making. *Nature Machine Intelligence*, 1(1):20–23.
- Bi, X., Tang, X., Yuan, Y., Zhang, Y., and Qu, A. (2021). Tensors in statistics. *Annual review of statistics and its application*, 8(1):345–368.
- Birkhoff, G. D. (1931). Proof of the ergodic theorem. *Proceedings of the National Academy of Sciences*, 17(12):656–660.
- Birkhoff, G. D. (1942). What is the ergodic theorem? *The American Mathematical Monthly*, 49(4):222–226.
- Bowerman, B. L. (1974). *NONSTATIONARY MARKOV DECISION PROCESSES AND RELATED TOPICS IN NONSTATIONARY MARKOV CHAINS*. Iowa State University.
- Brooks, S., Gelman, A., Jones, G., and Meng, X.-L. (2011). *Handbook of markov chain monte carlo*. CRC press.
- Cambanis, S. (1973). On some continuity and differentiability properties of paths of gaussian processes. *Journal of Multivariate Analysis*, 3(4):420–434.
- Carbonell, J. G., Michalski, R. S., and Mitchell, T. M. (1983). An overview of machine learning. *Machine learning*, pages 3–23.
- Chakrabarty, D. (2007). Phase space structure in the solar neighbourhood. *Astronomy & Astrophysics*, 467(1):145–162.
- Chakrabarty, D. (2025). *Supervised Learning: Mathematical Foundations and Real-World Applications*. CRC Press LLC.
- Chakrabarty, D., Wang, K., Roy, G., Bhojgaria, A., Zhang, C., Pavlu, J., and Chakrabartty, J. (2023). Constructing training set using distance between learnt graphical models of time series data on patient physiology, to predict disease scores. *Plos one*, 18(10):e0292404.
- Changey, W. and Robert, C. P. (2020a). Markov chain monte carlo algorithms for bayesian computation, a survey and some generalisation. In *Case Studies in Applied Bayesian Data Science*, pages 89–119. Springer.
- Changey, W. and Robert, C. P. (2020b). *Markov Chain Monte Carlo Algorithms for Bayesian Computation, a Survey and Some Generalisation*, pages 89–119. Springer International Publishing, Cham.
- Cheng, C.-H. and Wei, L.-Y. (2010). One step-ahead anfis time series model for forecasting electricity loads. *Optimization and Engineering*, 11:303–317.
- Ching, W.-K. and Ng, M. K. (2006). Markov chains. *Models, algorithms and applications*.
- Chinnathambi, R. A., Plathottam, S. J., Hossen, T., Nair, A. S., and Ranganathan, P. (2018). Deep neural networks (dnn) for day-ahead electricity price markets. In *2018 IEEE electrical power and energy conference (EPEC)*, pages 1–6. IEEE.
- Cohen, E. G. D. (1996). Boltzmann and statistical mechanics.

- Corani, G., Benavoli, A., and Zaffalon, M. (2021). Time series forecasting with gaussian processes needs priors. In *Machine Learning and Knowledge Discovery in Databases. Applied Data Science Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part IV 21*, pages 103–117. Springer.
- Cortes, C., Mohri, M., and Rostamizadeh, A. (2009). Learning non-linear combinations of kernels. *Advances in neural information processing systems*, 22.
- Cortes, C., Mohri, M., and Rostamizadeh, A. (2010). Two-stage learning kernel algorithms. In *ICML*, pages 239–246.
- Cotter, S. L., Dashti, M., and Stuart, A. M. (2010). Approximation of bayesian inverse problems for pdes. *SIAM journal on numerical analysis*, 48(1):322–345.
- Cremer, C. Z. (2021). Deep limitations? examining expert disagreement over deep learning. *Progress in Artificial Intelligence*, 10:449–464.
- Dalal, S. R., Fowlkes, E. B., and Hoadley, B. (1989). Risk analysis of the space shuttle: Pre-challenger prediction of failure. *Journal of the American Statistical Association*, 84(408):945–957.
- Das, S. and Nason, G. P. (2016). Measuring the degree of non-stationarity of a time series. *Stat*, 5(1):295–305.
- Dasgupta, D., Akhtar, Z., and Sen, S. (2022). Machine learning in cybersecurity: a comprehensive survey. *The Journal of Defense Modeling and Simulation*, 19(1):57–106.
- Dashti, M. and Stuart, A. M. (2013). The bayesian approach to inverse problems. *arXiv preprint arXiv:1302.6989*.
- Davis, C. B., Hans, C. M., and Santner, T. J. (2021). Prediction of non-stationary response functions using a bayesian composite gaussian process. *Computational Statistics & Data Analysis*, 154:107083.
- Der Kiureghian, A. and Ditlevsen, O. (2009). Aleatory or epistemic? does it matter? *Structural safety*, 31(2):105–112.
- Dixon, M. F., Halperin, I., and Bilokon, P. (2020). *Machine learning in finance*, volume 1170. Springer.
- Dua, D. and Graff, C. (2017). UCI machine learning repository.
- Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. (1987). Hybrid monte carlo. *Physics letters B*, 195(2):216–222.
- Dunlop, M. M., Iglesias, M. A., and Stuart, A. M. (2017). Hierarchical bayesian level set inversion. *Statistics and Computing*, 27:1555–1584.
- Duvenaud, D. (2014). *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge.

- Duvenaud, D., Lloyd, J., Grosse, R., Tenenbaum, J., and Zoubin, G. (2013). Structure discovery in nonparametric regression through compositional kernel search. In *International Conference on Machine Learning*, pages 1166–1174. PMLR.
- Feurer, M. and Hutter, F. (2019). Hyperparameter optimization. *Automated machine learning: Methods, systems, challenges*, pages 3–33.
- Gelfand, A. E. (2012). Hierarchical modeling for spatial data problems. *Spatial statistics*, 1:30–39.
- Gelfand, A. E. and Banerjee, S. (2017). Bayesian modeling and analysis of geostatistical data. *Annual review of statistics and its application*, 4(1):245–266.
- Gelfand, A. E. and Smith, A. F. (1990). Sampling-based approaches to calculating marginal densities. *Journal of the American statistical association*, 85(410):398–409.
- Gelman, A. (1993). Iterative and non-iterative simulation algorithms. *Computing science and statistics*, pages 433–433.
- Geman, S. and Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6):721–741.
- Genton, M. G. (2001). Classes of kernels for machine learning: a statistics perspective. *Journal of machine learning research*, 2(Dec):299–312.
- Geweke, J. and Tanizaki, H. (1999). On markov chain monte carlo methods for nonlinear and non-gaussian state-space models. *Communications in Statistics-Simulation and Computation*, 28(4):867–894.
- Geweke, J. and Tanizaki, H. (2001). Bayesian estimation of state-space models using the metropolis–hastings algorithm within gibbs sampling. *Computational statistics & data analysis*, 37(2):151–170.
- Geyer, C. J. (1998). Markov chain monte carlo lecture notes. *Course notes, Spring Quarter*, 80.
- Geyer, C. J. (2011). Introduction to markov chain monte carlo. *Handbook of markov chain monte carlo*, 20116022(45):22.
- Ghahramani, Z. (2015). Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452–459.
- Ghanizadeh, A. N., Ghiasi-Shirazi, K., Monsefi, R., and Qaraei, M. (2024). Neural generalization of multiple kernel learning. *Neural Processing Letters*, 56(1):12.
- Gibbs, M. N. (1998). *Bayesian Gaussian processes for regression and classification*. PhD thesis, Citeseer.
- Giordani, P. and Kohn, R. (2010). Adaptive independent metropolis–hastings by fast estimation of mixtures of normals. *Journal of Computational and Graphical Statistics*, 19(2):243–259.

- Giordano, M. and Nickl, R. (2020). Consistency of bayesian inference with gaussian process priors in an elliptic inverse problem. *Inverse problems*, 36(8):085001.
- Girard, A., Rasmussen, C., Candela, J. Q., and Murray-Smith, R. (2002). Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting. *Advances in neural information processing systems*, 15.
- Girolami, M. and Calderhead, B. (2011). Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214.
- Girolami, M. and Rogers, S. (2005). Hierarchic bayesian models for kernel learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 241–248.
- Gönen, M. (2012). A bayesian multiple kernel learning framework for single and multiple output regression. In *ECAI 2012*, pages 354–359. IOS Press.
- Gönen, M. and Alpaydın, E. (2011). Multiple kernel learning algorithms. *The Journal of Machine Learning Research*, 12:2211–2268.
- Gönen, M. and Alpaydın, E. (2013). Localized algorithms for multiple kernel learning. *Pattern Recognition*, 46(3):795–807.
- Goulard, M. and Voltz, M. (1992). Linear coregionalization model: tools for estimation and choice of cross-variogram matrix. *Mathematical Geology*, 24:269–286.
- Granero-Belinchon, C., Roux, S. G., and Garnier, N. B. (2021). Quantifying non-stationarity with information theory. *Entropy*, 23(12):1609.
- Gray, B. E. F. L. F. (1997). *A Modern Approach to Probability Theory*. Birkhäuser Basel.
- Grenander, U. and Miller, M. I. (1994). Representations of knowledge in complex systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(4):549–581.
- Guhaniyogi, R., Qamar, S., and Dunson, D. B. (2017). Bayesian tensor regression. *Journal of Machine Learning Research*, 18(79):1–31.
- Guo, W., Kotsia, I., and Patras, I. (2011). Tensor learning for regression. *IEEE Transactions on Image Processing*, 21(2):816–827.
- Haario, H., Laine, M., Mira, A., and Saksman, E. (2006). Dram: efficient adaptive mcmc. *Statistics and computing*, 16(4):339–354.
- Haario, H., Saksman, E., and Tamminen, J. (2001). An adaptive metropolis algorithm. *Bernoulli*, pages 223–242.
- Hansen, L. P. and Sargent, T. J. (2001). Acknowledging misspecification in macroeconomic theory. *Review of Economic Dynamics*, 4(3):519–535.
- Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications.

- Heinonen, M., Mannerström, H., Rousu, J., Kaski, S., and Lähdesmäki, H. (2016). Non-stationary gaussian process regression with hamiltonian monte carlo. In *Artificial Intelligence and Statistics*, pages 732–740. PMLR.
- Higdon, D. M., Swall, J., and Kern, J. (1999). Non-stationary spatial modeling. In Bernardo, J. M., Berger, J. O., Dawid, A. P., and Smith, A. F. M., editors, *Bayesian Statistics 6. Proceedings of the Sixth Valencia International Meeting*, pages 761–768. Oxford University Press.
- Hilbert, S., Coors, S., Kraus, E., Bischl, B., Lindl, A., Frei, M., Wild, J., Krauss, S., Goretzko, D., and Stachl, C. (2021). Machine learning for the educational sciences. *Review of Education*, 9(3):e3310.
- Hino, M., Benami, E., and Brooks, N. (2018). Machine learning for environmental monitoring. *Nature Sustainability*, 1(10):583–588.
- Hoff, P. D. (2011). Separable covariance arrays via the tucker product, with applications to multivariate relational data. *Bayesian Analysis*, 6(2):179–196.
- Hutchinson, M., Terenin, A., Borovitskiy, V., Takao, S., Teh, Y. W., and Deisenroth, M. P. (2021). Vector-valued gaussian processes on riemannian manifolds via gauge equivariant projected kernels. In *Advances in Neural Information Processing Systems, 2021*, volume 35. NeurIPS Proceedings.
- Jaynes, E. T. (2003). *Probability theory: The logic of science*. Cambridge university press.
- Journel, A. G. and Huijbregts, C. J. (1976). Mining geostatistics.
- Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285.
- Knapp, A. W. (2007). *Basic real analysis*. Springer Science & Business Media.
- Kolda, T. G. and Bader, B. W. (2009). Tensor decompositions and applications. *SIAM review*, 51(3):455–500.
- Kondor, R. and Jebara, T. (2003). A kernel between sets of vectors. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 361–368.
- Kossaifi, J., Lipton, Z. C., Kolbeinsson, A., Khanna, A., Furlanello, T., and Anandkumar, A. (2020). Tensor regression networks. *Journal of Machine Learning Research*, 21(123):1–21.
- Koutsoukas, A., Monaghan, K. J., Li, X., and Huan, J. (2017). Deep-learning: investigating deep neural networks hyper-parameters and comparison of performance to shallow methods for modeling bioactivity data. *Journal of cheminformatics*, 9:1–13.
- Kruschke, J. K. (2015). Doing bayesian data analysis (second edition). In *Academic Press*.
- Kuhn, M., Johnson, K., et al. (2013). *Applied predictive modeling*, volume 26. Springer.
- Kuss, M. and Rasmussen, C. (2003). Gaussian processes in reinforcement learning. In Thrun, S., Saul, L., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems*, volume 16. MIT Press.

- Lavine, M. (1991). Problems in extrapolation illustrated with space shuttle o-ring data. *Journal of the American Statistical Association*, 86(416):919–921.
- Lee, C.-H., Chang, K., Chen, Y.-M., Tsai, J.-T., Chen, Y. J., and Ho, W.-H. (2021). Epidemic prediction of dengue fever based on vector compartment model and markov chain monte carlo method. *BMC bioinformatics*, 22(5):1–11.
- Lewis, D. P., Jebara, T., and Noble, W. S. (2006). Nonstationary kernel combination. In *Proceedings of the 23rd international conference on Machine learning*, pages 553–560.
- Liu, Y., Liu, J., Long, Z., and Zhu, C. (2022a). *Tensor computation for data analysis*. Springer.
- Liu, Y., Liu, J., Long, Z., and Zhu, C. (2022b). *Tensor Regression*, pages 163–198. Springer International Publishing, Cham.
- Lu, H., Kim, E., and Gutierrez, M. (2022). A probabilistic q-system using the markov chain to predict rock mass quality in tunneling. *Computers and Geotechnics*, 145:104689.
- Luengo, D. and Martino, L. (2013). Fully adaptive gaussian mixture metropolis-hastings algorithm. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6148–6152. IEEE.
- Luengo, D., Martino, L., Bugallo, M., Elvira, V., and Särkkä, S. (2020). A survey of monte carlo methods for parameter estimation. *EURASIP Journal on Advances in Signal Processing*, 2020:1–62.
- Luo, G. (2016). A review of automatic selection methods for machine learning algorithms and hyper-parameter values. *Network Modeling Analysis in Health Informatics and Bioinformatics*, 5:1–16.
- M Ghari, P. and Shen, Y. (2022). Personalized online federated learning with multiple kernels. *Advances in Neural Information Processing Systems*, 35:33316–33329.
- Manceur, A. M. and Dutilleul, P. (2013). Maximum likelihood estimation for the tensor normal distribution: Algorithm, minimum sample size, and empirical bias and dispersion. *Journal of Computational and Applied Mathematics*, 239:37–49.
- Marcotte, D. (2012). *Revisiting the Linear Model of Coregionalization*, pages 67–78. Springer Netherlands, Dordrecht.
- Mbalawata, I. S., Särkkä, S., Vihola, M., and Haario, H. (2015). Adaptive metropolis algorithm using variational bayesian adaptive kalman filter. *Computational Statistics & Data Analysis*, 83:101–115.
- McCullagh, P. (2018). *Tensor methods in statistics: Monographs on statistics and applied probability*. Chapman and Hall/CRC.
- Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092.

- Mika, S., Ratsch, G., Weston, J., Scholkopf, B., and Mullers, K.-R. (1999). Fisher discriminant analysis with kernels. In *Neural networks for signal processing IX: Proceedings of the 1999 IEEE signal processing society workshop (cat. no. 98th8468)*, pages 41–48. Ieee.
- Monard, F., Nickl, R., and Paternain, G. P. (2021). Consistent inversion of noisy non-abelian x-ray transforms. *Communications on Pure and Applied Mathematics*, 74(5):1045–1099.
- Montavon, G., Orr, G., and Müller, K.-R. (2012). *Neural networks: tricks of the trade*, volume 7700. springer.
- Muhammad, I. and Yan, Z. (2015). Supervised machine learning approaches: A survey. *ICTACT Journal on Soft Computing*, 5(3).
- Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.
- Neal, R. M. (1993). *Probabilistic inference using Markov chain Monte Carlo methods*. Department of Computer Science, University of Toronto Toronto, ON, Canada.
- Neal, R. M. (1997). Monte carlo implementation of gaussian process models for bayesian regression and classification. *arXiv preprint physics/9701026*.
- Neal, R. M. (1998). Regression and classification using gaussian process priors (with discussion). In J. M. Bernardo et. al, editor, *Bayesian Statistics 6*, pages 475–501. Oxford University Press.
- Nelder, J. A. and Wedderburn, R. W. (1972). Generalized linear models. *Journal of the Royal Statistical Society Series A: Statistics in Society*, 135(3):370–384.
- Neumann, J. v. (1932). Proof of the quasi-ergodic hypothesis. *Proceedings of the National Academy of Sciences*, 18(1):70–82.
- Noack, M. M., Luo, H., and Risser, M. D. (2024a). A unifying perspective on non-stationary kernels for deeper gaussian processes. *APL Machine Learning*, 2(1).
- Noack, M. M., Luo, H., and Risser, M. D. (2024b). A unifying perspective on non-stationary kernels for deeper gaussian processes. *APL Machine Learning*, 2(1).
- O’Hagan, A. (1978). Curve fitting and optimal design for prediction. *Journal of the Royal Statistical Society: Series B (Methodological)*, 40(1):1–24.
- Oil Market Report (2023). Data, forecasts and analysis on the global oil market. <https://www.iea.org/topics/oil-market-report>.
- Oseledets, I. V. (2011). Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317.
- Paciorek, C. and Schervish, M. (2003). Nonstationary covariance functions for gaussian process regression. *Advances in neural information processing systems*, 16.
- Paciorek, C. J. (2003). *Nonstationary Gaussian processes for regression and spatial modelling*. PhD thesis, Carnegie Mellon University.

- Papacharalampous, G., Tyrallis, H., and Koutsoyiannis, D. (2018). One-step ahead forecasting of geophysical processes within a purely statistical framework. *Geoscience Letters*, 5(1):1–19.
- Park, S. and Choi, S. (2010). Hierarchical gaussian process regression. In *Proceedings of 2nd Asian conference on machine learning*, pages 95–110. JMLR Workshop and Conference Proceedings.
- Peebles Jr, P. Z. (2001). *Probability, random variables, and random signal principles*. McGraw-Hill.
- Peters, O. and Gell-Mann, M. (2014). Evaluating gambles using dynamics. *Chaos*, 26 2:023103.
- Qian, S. S., Stow, C. A., and Borsuk, M. E. (2003). On monte carlo methods for bayesian inference. *Ecological modelling*, 159(2-3):269–277.
- Rabusseau, G. and Kadri, H. (2016). Low-rank regression with tensor responses. *Advances in Neural Information Processing Systems*, 29.
- Rasmussen, C. and Williams, C. (2006). *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, USA.
- Rasmussen, C. E. (1997). *Evaluation of Gaussian processes and other methods for non-linear regression*. PhD thesis, University of Toronto Toronto, Canada.
- Rastogi, R., Ghosh, S., Ghosh, A. K., Vaze, K. K., and Singh, P. K. (2017). Fatigue crack growth prediction in nuclear piping using markov chain monte carlo simulation. *Fatigue & Fracture of Engineering Materials & Structures*, 40(1):145–156.
- Remes, S., Heinonen, M., and Kaski, S. (2017). Non-stationary spectral kernels. *Advances in neural information processing systems*, 30.
- Ripley, B. D. (2009). *Stochastic simulation*, volume 316. John Wiley & Sons.
- Risser, M. D. and Calder, C. A. (2015). Regression-based covariance functions for nonstationary spatial modeling. *Environmetrics*, 26(4):284–297.
- Risser, M. D. and Turek, D. (2020). Bayesian inference for high-dimensional nonstationary gaussian processes. *Journal of Statistical Computation and Simulation*, 90(16):2902–2928.
- Robert, C. P. and Casella, G. (2004). Monte carlo statistical methods. In *Springer Texts in Statistics*.
- Robert, C. P. and Casella, G. (2005). *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer-Verlag, Berlin, Heidelberg.
- Robert, C. P. and Casella, G. (2010). *Monte Carlo Statistical Methods*. Springer Publishing Company, Incorporated.
- Robert, C. P. and Changye, W. (2020). Markov chain monte carlo methods, a survey with some frequent misunderstandings. *arXiv preprint arXiv:2001.06249*.

- Roberts, G. O. and Rosenthal, J. S. (1998). Markov-chain monte carlo: some practical implications of theoretical results. *The Canadian Journal of Statistics/La Revue Canadienne de Statistique*, pages 5–20.
- Roberts, G. O. and Stramer, O. (2002). Langevin diffusions and metropolis-hastings algorithms. *Methodology and computing in applied probability*, 4(4):337–357.
- Rossky, P. J., Doll, J., and Friedman, H. (1978). Brownian dynamics as smart monte carlo simulation. *The Journal of Chemical Physics*, 69(10):4628–4633.
- Roy, G. and Chakrabarty, D. (2023). Efficient uncertainty quantification for under-constraint prediction following learning using mcmc. In *Neural Information Processing*, pages 275–287, Singapore. Springer Nature Singapore.
- Roy, G. and Chakrabarty, D. (2024). A new reliable & parsimonious learning strategy comprising two layers of gaussian processes, to address inhomogeneous empirical correlation structures.
- Roy, G., Warrior, K., and Chakrabarty, D. (2023a). Covariance kernel learning schemes for gaussian process based prediction using markov chain monte carlo. In *EPIA Conference on Artificial Intelligence*, pages 184–195. Springer.
- Roy, G., Warrior, K., and Chakrabarty, D. (2023b). Covariance kernel learning schemes for gaussian process based prediction using markov chain monte carlo. In Moniz, N., Vale, Z., Cascalho, J., Silva, C., and Sebastião, R., editors, *Progress in Artificial Intelligence*, pages 184–195, Cham. Springer Nature Switzerland.
- Rue, H., Martino, S., and Chopin, N. (2009). Approximate bayesian inference for latent gaussian models by using integrated nested laplace approximations. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 71(2):319–392.
- Salam, A. and El Hibaoui, A. (2018). Comparison of machine learning algorithms for the power consumption prediction:-case study of tetouan city-. In *2018 6th International Renewable and Sustainable Energy Conference (IRSEC)*, pages 1–5. IEEE.
- Shen, X., Zhang, Y., Zhang, J., and Wu, X. (2022). An interval analysis scheme based on empirical error and mcmc to quantify uncertainty of wind speed. *IEEE Transactions on Industry Applications*, 58(6):7754–7763.
- Signoretto, M., De Lathauwer, L., and Suykens, J. A. (2010). Kernel-based learning from infinite dimensional 2-way tensors. In *International Conference on Artificial Neural Networks*, pages 59–69. Springer.
- Signoretto, M., Tran Dinh, Q., De Lathauwer, L., and Suykens, J. A. (2014). Learning with tensors: a framework based on convex optimization and spectral regularization. *Machine Learning*, 94:303–351.
- Simpson, F., Lalchand, V., and Rasmussen, C. E. (2021). Marginalised gaussian processes with nested sampling. *Advances in neural information processing systems*, 34:13613–13625.

- Smith, H. J., Dinev, T., and Xu, H. (2011). Information privacy research: an interdisciplinary review. *MIS quarterly*, pages 989–1015.
- Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25.
- Sonnenburg, S., Rätsch, G., Schäfer, C., and Schölkopf, B. (2006). Large scale multiple kernel learning. *The Journal of Machine Learning Research*, 7:1531–1565.
- Stone, P. and Veloso, M. (2000). Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8:345–383.
- Stuart, A. M. (2010). Inverse problems: a bayesian perspective. *Acta numerica*, 19:451–559.
- Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.
- Sze, V., Chen, Y.-H., Yang, T.-J., and Emer, J. S. (2017). Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329.
- Tensorflow (2022). Basic regression: Predict fuel efficiency. <https://www.tensorflow.org/tutorials/keras/regression>.
- Thisted, R. A. (1988). *Elements of statistical computing: numerical computation*. Chapman & Hall, Ltd., GBR.
- Tierney, L. (1994). Markov chains for exploring posterior distributions. *the Annals of Statistics*, pages 1701–1728.
- Tobar, F., Bui, T. D., and Turner, R. E. (2015). Learning stationary time series using gaussian processes with nonparametric kernels. *Advances in neural information processing systems*, 28.
- van de Schoot, R., Depaoli, S., King, R., Kramer, B., Märtens, K., Tadesse, M. G., Vannucci, M., Gelman, A., Veen, D., Willemsen, J., et al. (2021). Bayesian statistics and modelling. *Nature Reviews Methods Primers*, 1(1):1.
- van der Vaart, A. and van Zanten, J. (2009). Adaptive bayesian estimation using a gaussian random field with inverse gamma bandwidth. *The Annals of Statistics*, 37(5B):2655–2675.
- van Lith, J. (2001). Ergodic theory, interpretations of probability and the foundations of statistical mechanics. *Studies in History and Philosophy of Science Part B: Studies in History and Philosophy of Modern Physics*, 32(4):581–594.
- Von Plato, J. (1988). *Ergodic Theory and the Foundations of Probability*, pages 257–277. Springer Netherlands, Dordrecht.
- Waldrop, M. M. (2019). What are the limits of deep learning? *Proceedings of the National Academy of Sciences*, 116(4):1074–1077.
- Wang, K. and Chakrabarty, D. (2018). Deep bayesian supervised learning given hypercuboidally-shaped, discontinuous data, using compound tensor-variate & scalar-variate gaussian processes.

- Wang, Y., Zhou, C., and Chen, Z. (2023). An enhanced approach for load behavior and battery residual capacity prediction using markov chain and monte carlo method. *IEEE Journal of Emerging and Selected Topics in Industrial Electronics*, 4(1):159–167.
- Wilson, A. and Adams, R. (2013). Gaussian process kernels for pattern discovery and extrapolation. In *International conference on machine learning*, pages 1067–1075. PMLR.
- Wilson, A. G. and Ghahramani, Z. (2011). Generalised wishart processes. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, pages 736–744.
- Wilson, J., Borovitskiy, V., Terenin, A., Mostowsky, P., and Deisenroth, M. (2020). Efficiently sampling functions from gaussian process posteriors. In *International Conference on Machine Learning*, pages 10292–10302. PMLR.
- Wu, P., Yin, Q., and Guo, P. (2014). Method of evolving non-stationary multiple kernel learning. In *International Conference on Neural Information Processing*, pages 17–25. Springer.
- Xu, Z., Guo, Y., and Saleh, J. H. (2021). Accurate remaining useful life prediction with uncertainty quantification: a deep learning and nonstationary gaussian process approach. *IEEE Transactions on Reliability*, 71(1):443–456.
- Xuan, J., Lu, J., and Zhang, G. (2019). A survey on bayesian nonparametric learning. *ACM Computing Surveys (CSUR)*, 52(1):1–36.
- Yan, B., Ren, F., Cai, M., and Qiao, C. (2020). Bayesian model based on markov chain monte carlo for identifying mine water sources in submarine gold mining. *Journal of Cleaner Production*, 253:120008.
- Yang, L. and Shami, A. (2020). On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316.
- Yang, Y. and Tokdar, S. T. (2015). Minimax-optimal nonparametric regression in high dimensions. *Annals of Statistics*, 43(2):652–674.
- Young, D. S. (2018). *Handbook of regression methods*. Chapman and Hall/CRC.
- Yu, R. and Liu, Y. (2016). Learning from multiway data: Simple and efficient tensor regression. In *International Conference on Machine Learning*, pages 373–381. PMLR.
- Zhao, Q., Zhou, G., Zhang, L., and Cichocki, A. (2014). Tensor-variate gaussian processes regression and its application to video surveillance. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1265–1269. IEEE.
- Zhou, H., Li, L., and Zhu, H. (2013). Tensor regression with applications in neuroimaging data analysis. *Journal of the American Statistical Association*, 108(502):540–552.
- Zhou, Y., Wong, R. K., and He, K. (2024). Broadcasted nonparametric tensor regression. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, page qkae027.
- Zhu, L. and Laptev, N. (2017). Deep and confident prediction for time series at uber. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 103–110. IEEE.

- Zhuang, J., Tsang, I. W., and Hoi, S. C. (2011). Two-layer multiple kernel learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 909–917. JMLR Workshop and Conference Proceedings.

