Heteroscedastic Gaussian Processes for Uncertain and Incomplete Data

Ibrahim Almosallam

Somerville College

Supervisors:
Stephen J. Roberts and Matt J. Jarvis

A thesis submitted for the degree of

Doctor of Philosophy

Hilary 2017

This thesis is dedicated to my children

Ahmad and Abdulmalik.

Acknowledgements

This is easily the hardest section I had to write in my thesis, not because it is difficult to acknowledge the people who are responsible for being where I am today, but because I believe that the gratitude I feel towards them cannot be expressed in a human language. I wish I can use a precise language like mathematics to express my feelings in the same way I used it to make my arguments in this thesis. Parts of what attracted me to science, and especially computer science, in the first place are the languages created by design to be as unambiguous and as precise as possible; but as powerful as they are, their expressive power are still limited. That being said, the following is my best attempt.

To my supervisors, Stephen Roberts and Matt Jarvis, thank you for your patience and giving me the freedom to explore while at the same time keeping me on track and ensuring that I do not get lost in my own thoughts. Matt, the opportunities you created and the platforms you have provided for me to work on projects which could change our understanding of the Universe have been constant positive motivators and speaks to your faith in me, an idea I am still trying to live up to. Steve, thank you for your guidance, mentorship and mostly your friendship. I came out from this experience with more than just additional knowledge in machine learning, but more importantly, with a deeper understanding of the field and how to better consume and conduct research. Working under your supervision for the past two and a half years is the best chapter of my academic career.

I would also like to acknowledge my sponsor, King Abdulaziz City for Science and Technology (KACST), not only for their financial support for me and my family throughout my studies, but also for the opportunities I have been provided and the experience that built the foundation of my research career. The most influential person being Dr. Mohamed Alkanhal, who has given me unconditional support and opened doors for me that otherwise would have still been closed, based only on his strong belief in me and despite very little assurances. Nevertheless, has gone out of his way to help me take the path I drew for myself. I can only hope to be as influential to other people's careers as you have been to mine.

Many thanks to every member of the machine learning research group for creating an intellectually stimulating environment. Part of this thesis is inspired from the many insightful discussions, challenging questions and constructive criticism I received from the members of the group.

Last but not least, thanks to my father who taught me how to program at the age of 14.

Publications

Papers

Some portions of the work presented in Chapters 3, 4 and 6 of this thesis have been published in the following peer-reviewed articles:

- I. A. Almosallam, M. J. Jarvis, and S. J. Roberts. GPz: Non-stationary Sparse Gaussian Processes for Heteroscedastic Uncertainty Estimation in Photometric Redshifts. *MNRAS*, 462:726–739, October 2016a. doi: 10.1093/mnras/stw1618

  The non-stationarity (Chapter 4), cost-sensitive learning and prior mean-function (Chapter 3) concepts were first published and applied in this publication. The first author conceived the ideas, implemented them, conducted all the experiments and the machine learning literature review. Sam Lindsay was in charge of data retrieval, selection and formatting. Matt Jarvis advised on the experimental and evaluation design, conducted the astrophysics literature review and participated in editing the paper. Stephen Roberts directed the paper, advised on the mathematical formulation of the proposed methods, advised on the testing methodology and edited the paper. The experiments in this publication have been repeated in Chapter 6 with the updated model and to include additional models.

- I. A. Almosallam, S. N. Lindsay, M. J. Jarvis, and S. J. Roberts. A Sparse Gaussian Process Framework for Photometric Redshift Estimation. *MNRAS*, 455:2387–2401, January 2016b. doi: 10.1093/mnras/stv2425

  The basis function model view of sparse GPs with automatic relevance determination and heteroscedastic noise estimation (Chapter 3) and the optimisation procedure (Chapter 4), for the noiseless input case, were first published and applied in this publication. The first author conceived the ideas, implemented them, retrieved the data, conducted all the experiments and the machine learning literature review. Matt Jarvis advised on the experimental and evaluation design, conducted the astrophysics literature review and participated in editing the paper. Stephen Roberts directed the paper, advised on the mathematical formulation of the proposed methods, advised on the testing methodology and edited the paper. The experiments and results in this publication are also reported in Chapter 6.

Software

Python and Matlab implementations of some of the work presented in this thesis are published in the Machine Learning Research Group (MLRG) GitHub repository:

- **OxfordML/GPz**: Non-stationary sparse Gaussian processes for heteroscedastic uncertainty estimation.
  `https://github.com/OxfordML/GPz`

Abstract

In probabilistic inference, many implicit and explicit assumptions are taken about the nature of input noise and the function fit to either simplify the mathematics, improve the time complexity or optimise for space. It is often assumed that the inputs are noiseless or that the noise is drawn from the same distribution for all inputs, that all the variables used during training will be present during prediction and with the same degrees of uncertainties, and that the confidence about the prediction is uniform across the input space. This thesis presents a more generalised sparse Gaussian process model that relaxes these assumptions to inputs with variable degrees of uncertainty, or completeness in the input, and produces variable uncertainty estimation over the output. The capabilities of sparse Gaussian processes are further enhanced to allow for non-stationarity which minimises the number of required basis functions, a prior mean function for better extrapolation performance and cost-sensitive learning for non-uniform weighting of samples. The results are demonstrated on an astrophysical problem of estimating galactic redshifts from their photometry. This problem, by its nature, can capitalise on the features of the proposed model as the noise on the photometry can vary across different galaxies or catalogues, not all photometry might be available during prediction or shared amongst different surveys, and the input-dependent uncertainty estimation gives astrophysicists the ability to trade off completeness for

accuracy to answer a range of different questions related to astronomy and cosmology.

Contents

Nomenclature **1**

1 Introduction **5**

2 Gaussian Processes for Regression **11**

2.1 Full Gaussian Processes 11

2.2 Numerical Approximation 14

2.2.1 Low Rank Approximation 14

2.2.2 The Conjugate Gradient Method 16

2.3 Sparse Gaussian Processes 18

2.3.1 Subset of Regressors 18

2.3.2 Fully Independent Training Conditional 20

2.4 Basis Function Models 23

2.5 Relations to other Methods 27

2.5.1 Relations to Artificial Neural Networks 27

2.5.2 Relations to Stacked Auto-encoders 28

2.6 Summary 29

3 Extending Sparse Gaussian Processes **31**

3.1 Automatic Relevance Determination 31

3.2 Heteroscedastic Noise 33

3.3 Strictly Positive Outputs 37

3.4 Cost-sensitive Learning 39
3.5 Prior Mean Function 41
3.6 Summary 44

4 Optimisation **47**
4.1 Gradient-based Optimisation 47
4.2 The General Case 49
4.3 The Sigmoidal Function 51
4.3.1 Gradients 52
4.3.2 Illustrative Examples 53
4.4 Radial Basis Functions 56
4.4.1 Gradients 57
4.4.2 Illustrative Examples 58
4.5 Summary 59

5 Noisy or Missing Variables **63**
5.1 Prediction 64
5.1.1 Predicting with Noisy Variables 64
5.1.2 Predicting with Missing Variables 67
5.1.3 Predicting with Missing and Noisy Variables 78
5.2 Training 84
5.2.1 Training with Noisy Variables 84
5.2.2 Training with Missing and Noisy Variables 88
5.3 Summary 94

6 Photometric Redshift Estimation **97**
6.1 Background 97
6.2 Results on Simulated Data 104
6.2.1 Models and Performance Metrics 104

6.2.2 The Dataset . . . 106
6.2.3 Modelling Performance . . . 107
6.2.4 Prior Mean Function . . . 108
6.2.5 Size of the Basis Set . . . 110
6.2.6 Cost-sensitive Learning . . . 113
6.3 Results on Real Data . . . 116
6.3.1 The Dataset . . . 116
6.3.2 Models and Metrics . . . 119
6.3.3 Model Complexity . . . 121
6.3.4 Performance Analysis . . . 121
6.3.5 Selection Performance . . . 122
6.3.6 Uncertainty Analysis . . . 125
6.4 Missing Photometry . . . 128
6.4.1 Predicting with Missing Photometry . . . 129
6.4.2 Training with Missing Photometry . . . 131
6.5 Summary . . . 132

7 Summary and Future Work **135**

A Probability and Matrix Identities **139**
A.1 Probability Identities . . . 139
A.1.1 The Chain Rule of Probability . . . 139
A.1.2 Bayes' Theorem . . . 139
A.1.3 The Marginal Distribution . . . 139
A.1.4 Expectation and Variance . . . 140
A.2 Gaussian Identities . . . 140
A.2.1 The Marginal Distribution . . . 140
A.2.2 The Conditional Distribution . . . 141

A.2.3 The Product of Two Gaussians 141
A.2.4 Gaussian Raised to a Power 142
A.3 Matrix Identities 142
A.3.1 Matrix Inversion Lemma 142
A.3.2 Inverse of a Partitioned Matrix 142
A.3.3 Determinant of a Partitioned Matrix 143
A.3.4 Bounds of a Determinant 143
A.3.5 The Determinant of the Sum of Positive Definite Matrices . . 144

Bibliography **145**

List of Figures

2.1 The effect of changing the hyper-parameters λ, from (a) to (c), and σ, from (1) to (3), of an RBF kernel on a GP model. The plots show the mean function (red), draws from the function distribution (Equation (2.10)) and the 95% confidence range (grey), i.e. plus or minus two standard deviations from the mean. The value of h was set to the optimal value achieved after optimising the log marginal likelihood (shown above each plot). 15

2.2 The effect of changing the number of pseudo-points (m), from (a) to (f) in multiples of 2, on FITC using an RBF kernel. The plots show the mean function (red), draws from the function distribution (Equation (2.41)), the 95% confidence range (grey), i.e. plus or minus two standard deviations from the mean, and the locations of the pseudo-points (black). The log marginal likelihood values are shown above each plot. 22

3.1 A comparison between (a) a basis function model with an isotropic precision matrix $\mathbf{A} = \alpha\mathbf{I}_m$, (b) a basis function model with a diagonal precision matrix $\mathbf{A} = \text{diag}\,[\boldsymbol{\alpha}]$, (c) The fully independent training conditional (FITC) sparse GP and (d) Relevance Vector Machines (RVM) all using the RBF kernel. 100 basis functions were used in (a) BFM, (b) BFM with ARD and (c) FITC; whereas in (d) RVM, the entire dataset of 1000 points were used as basis functions. The top row shows the learned distributions for each along with the locations of the basis functions, the log marginal likelihoods are shown above each plot in the top row. The bottom row shows the corresponding weights for each basis function as a stacked bar sorted in descending order in case of overlapping pseudo-points. The numbers of relevant vectors for each methods are shown above each plot in the bottom row. 34

3.2 A comparison between (a) a full GP with a constant noise variance, (b) FITC using 100 pseudo-points and (c) a BFM using 100 basis functions with variable uncertainty prediction, all using the RBF kernel. The plots show the mean of the function (red), the 95% confidence range (grey), i.e. plus or minus two standard deviations from the mean, and samples from the distribution $p\,(\mathbf{f}_*|\mathbf{X}_*, \mathcal{D}, \boldsymbol{\theta})$. The log marginal likelihood is shown above each plot. 38

3.3 The mean (red), median (green) and 95% area (grey), i.e. plus or minus two standard deviations from the mean, of the predictive distribution of a basis function model with 200 radial basis functions for strictly positive outputs with non-Gaussian noise. 40

3.4 A comparison between two basis function models, the first (a) using a constant prior mean of 0 and the second (b) using a prior linear mean function showing the mean of the function (red), the 95% confidence range (grey) and the log marginal likelihood. Both models were trained using 100 RBFs. 44

4.1 A 2D toy example described in Equation (4.35) 54

4.2 The results of training a hyperbolic tangent based model with different numbers of basis functions (m), from (a) to (f) in multiples of 2. The lines plotted on the xy-plane represent the learned parameters of the basis functions, where the degree of transparency is proportional to the relevance of the basis as determined by Equation (4.37). The log marginal likelihood is shown above each plot. 55

4.3 The results of training FITC, GPVL, GPVD and GPVC with different numbers of basis functions (m) on the same data collected using Equation (4.35). The ellipses represent the learned covariances of the RBFs, where the degree of transparency is proportional to the relevance (Equation (4.37)). The log marginal likelihoods are shown above each plot. 60

5.1 The result of using the expected value of a missing variable on the 2D artificial example from Figure 4.1 compared to the reference model that is trained only on the observed variable. 69

5.2 The result of treating the missing variable as a noisy input with probability $\mathcal{N}\left(\mathbf{u}|\mathbb{E}\left[\mathbf{u}|\mathbf{o}\right], \mathbb{V}\left[\mathbf{u}|\mathbf{o}\right]\right)$ on the 2D artificial example from Figure 4.1 compared to the reference model trained only on the observed variable. 71

5.3 The result of (a) treating the missing variable y as a noisy input, distributed as in Equation (5.23), on the 2D artificial example from Figure 4.1 compared to (b) the reference model trained only on the observed variable, (c) a Gaussian mixture model optimised using Expectation Maximisation and (d) a random forest model. 76

5.4 The result of (a) treating the missing variable x as a noisy input, distributed as in Equation (5.23), on the 2D artificial example from Figure 4.1 compared to (b) the reference model trained only on the observed variable, (c) a Gaussian mixture model optimised using Expectation Maximisation and (d) a random forest model. 77

5.5 A comparison between a sampling method (top) and the proposed approximation in this section (bottom) in predicting with a missing y variable and a noisy x variable with probability $\mathcal{N}(x, \psi)$ on the example from Figure 4.1. The variance ψ is varied from (a) 1 to (d) 8 (in multiples of 2). 83

5.6 Comparisons between six different methods for training with input noise and homoscedastic output noise, (a) a deterministic model, (b) a sampling method where we draw 100 samples from each input distribution, (c) treating the noise as input, (d) training with cost-sensitive learning such that $\omega_i = 1/\sqrt{\psi_i}$, (e) using the ERBF method and (f) using NIGP, all using 100 RBFs. 89

5.7 Comparisons between five different methods for training with input noise and heteroscedastic output noise, (a) a deterministic model, (b) a sampling method where we draw 100 samples from each input distribution, (c) treating the noise as input, (d) training with cost-sensitive learning such that $\omega_i = 1/\sqrt{\psi_i}$ and (e) with ERBF, all using 100 RBFs. 90

5.8 A comparison between the proposed approach and a random forest implementation, using 100 basis functions and 100 trees respectively, on the example from Figure 4.1 trained on a varying percentage of observed data. The y-axis reports the RMSE on the test set which has no missing variables. 94

6.1 A demonstration of how the Doppler effect changes the frequency, and consequently the position of the absorption line, of the light emitted from a galaxy moving away from Earth (bottom) and a galaxy moving towards Earth (middle) compared to the reference frequency at rest (top). 98

6.2 A comparison between the absorption lines of a supercluster of distant galaxies (bottom) and that of the Sun (top) in the visible spectrum (Wikimedia Commons, 2005). 99

6.3 A gravitational lensing illustration. A distant galaxy (red) behind a foreground source (blue) where its light is bent due to gravity creating the effect of observing two distinct sources (magenta) in the recorded image in the left panel (NASA/JPL-Caltech, 2010). 101

6.4 An example of (a) an Einstein's ring (LRG 3-757; ESA/Hubble & NASA, 2011) and (b) an Einstein's cross (HE0435-1223; ESA/Hubble & NASA, 2017). The gravity of the luminous red galaxy in (a) curved the light of the background blue galaxy, creating the illusion of of ring-shaped galaxy. The foreground galaxy in (b) creates four copies of a distant quasar. 102

6.5 Histograms and gamma distribution fits of the spectroscopic sample using (a) the full data set, (b) sources with RIZ magnitude<23 and (c) sources with RIZ magnitude<22 from the simulated data. 107

6.6 Density scatter plots of the true spectroscopic redshift z vs the predicted photometric redshift $\acute{z}$ for (a) ANNz, (b) stableGP, (c) SPGP, (d) GPVL, (e) GPVD and (f) GPVC ($m = 10$). The plots show the performance on the same test set, the colours however are scaled differently according to the density range in each plot to avoid colour saturation. 109

6.7 Density scatter plots of the true z versus the predicted $\acute{z}$ after training the GPVC model on samples with $RIZ < 23$ (top) and $RIZ < 22$ (bottom) using $m = 10$ basis functions with (a) a zero-mean prior, (b) linear regression prior and (c) a joint prior optimisation and (d) ANNz. The plots show the performance on the same test set, the colours however are scaled differently according to the density range in each plot to avoid colour saturation. 111

6.8 Δz as a function of m for all the methods. The y-axis is shown on a log scale to enhance the visualisation. 113

6.9 The density scatter plot for the final (a) ANNz, (b) stableGP, (c) SPGP, (d) GPVL, (e) GPVD and (f) GPVC models on the simulated survey. ANNz is based on a committee of 5 networks with 8:16:16:1 architectures, whereas the rest of the models are based on $m = 200$. . 114

6.10 The density scatter plot of z vs $\acute{z}$ and a box plot of the residual errors on the test set, showing median (bar), inter-quartile range (box) and range (whiskers) for a GPVC model with $m = 100$ using (a) "Normal", (b) "Normalised" and "Balanced" cost-sensitive learning weights. . . 117

6.11 The performance metrics of each method on the test set using different values of m. 122

6.12 The scatter plots of the spectroscopic redshift z versus the predicted photometric redshift $\acute{z}$ on the test set for (a) ANNz2, (b) TPZ, (c) SPGP, (d) GPVL, (e) GPVD and (f) GPVC using $m = 100$. The predictive variance is colour coded, on a log scale, by the value of the predictive variance of each method to avoid colour saturation and enhance visualisation. 123

6.13 The $\mathcal{F}_\epsilon$ for different values of ϵ on the test set for each method with $m = 100$. 124

6.14 The performance metrics as functions of the percentage of data selected based on the predictive variance generated by each method using $m = 100$ on the test set. 126

6.15 The percentage of difference between GPVC and the other methods, computed as $100 \times (\text{Method} - \text{GPVC}) / |\text{Method}|$ as a function of the percentage of data selected based on the predictive variance generated by each method using $m = 100$ on the test set. The values are plotted on a log-scaled y-axis to enhance visibility. 127

6.16 Box plots showing the square root of (a) the model uncertainty, $\nu(x)$, and (b) the noise uncertainty, $\sigma(x)$, as functions of the spectroscopic redshift showing median (bar), inter-quartile range (box) and range (whiskers) on the test set using a GPVC model with 100 basis functions. 128

6.17 The performance metrics of the GPVC and random forests models, using $m = 200$, when trained with the full data and tested on different numbers of missing inputs from 1 to 4. 129

6.18 The scatter plots of the GPVC and random forests models, using $m = 200$, when trained with the full data and tested on different numbers of missing inputs from (a) 1 to (d) 4. The plots are colour-coded by the predictive variance, on a log scale, with different scales to avoid colour saturation. 130

6.19 The performance metrics of the GPVC and random forests models, using $m = 200$, on predicting with the full data as the percentage of missing data from the training set is increased. 132

List of Tables

4.1 Different valid activation functions. 52

4.2 The time complexity of each approach. 59

5.1 The mean z-scores, plus or minus one standard deviation, for $\mathbb{E}[f(x)]$, $\mathbb{E}[f(x)^2]$, $\mathbb{E}[\nu^2(x)]$ and $\mathbb{E}[\sigma^2(x)]$ based on the analytical method described in this section for noisy input prediction and their estimated values based on a sampling method. 67

5.2 The mean log likelihood, plus or minus one standard deviation, of the predictive distribution for each method in handling missing values compared to the reference model on the held-out test set. 78

5.3 The mean z-scores, plus or minus one standard deviation, for $\mathbb{E}[f(x,y)]$, $\mathbb{E}[f(x,y)^2]$, $\mathbb{E}[\nu^2(x,y)]$ and $\mathbb{E}[\sigma^2(x,y)]$ based on the analytical method described in this section for noisy and missing input and their estimated values based on a sampling method. 84

5.4 The RMSE between the tested models and the true underlying function using both homoscedastic and heteroscedastic variance estimations. The results are reported on 1000 test samples that are linearly spaced across the domain of the input. 88

6.1 Performance metrics used to evaluate the models, where n is the number of samples, z_i is the spectroscopic redshift of sample i and $\acute{z}_i$ is its predicted photometric redshift. 105

6.2 Performance metrics for each method (with $m = 10$). The best-performing method is highlighted in bold font. 110

6.3 The Δz score for the GPVC model when trained using $m = 10$ basis functions with different prior mean functions and RIZ splits. The results for ANNz are shown for comparison. 112

6.4 Performance measures for the final ANNz model using a committee of 5 networks with 8:16:16:1 architectures and the final models using $m = 200$ basis functions, with a jointly optimised linear function for GPVL, GPVD and GPVC, on the simulated survey. 115

6.5 Performance measures of training a GPVC model with $m = 100$ and different weighting schemes trained on the simulated survey. 118

6.6 Performance metrics used to evaluate the models, where n is the number of samples, z_i is the spectroscopic redshift of sample i, $\acute{z}_i$ is the predicted photometric redshift and $\acute{\sigma}_i^2$ is the predicted variance. . . . 120

6.7 Performance measures for each algorithm trained using $m = 100$ for all the models on the held-out test set. The best-performing method is highlighted in bold font. 124

6.8 The average relative improvement of GPVC over other tested methods on all metrics on the test set using $m = 100$. 125

Nomenclature

Notations

$\mathbf{1}_n$	A column vector of length n with all ones.
$\mathbf{I}_n$	The identity matrix of size $n \times n$.
x	A scalar.
$\mathbf{x}$	A column vector.
$\mathbf{X}$	A matrix.
X	A random variable.

Indexing

x_i	The i-th element of vector $\mathbf{x}$, i might be a set of indices.
$\mathbf{x}[i]$	The i-th element of vector $\mathbf{x}$, i might be a set of indices.
$\mathbf{x}[i:j]$	Elements i to j of vector $\mathbf{x}$.
$\mathbf{x}_i$	The i-th column of matrix $\mathbf{X}$, i might be a set of indices.
$\mathbf{X}_i$	The i-th $\mathbf{X}$ matrix.
$\mathbf{X}[i,:]$	The i-th row of matrix $\mathbf{X}$, i might be a set of indices.
$\mathbf{X}[:,j]$	The j-th column of matrix $\mathbf{X}$, j might be a set of indices.

$\mathbf{X}[i:j,:]$	The sub-matrix of $\mathbf{X}$ containing the rows from i to j.
$\mathbf{X}[:,i:j]$	The sub-matrix of $\mathbf{X}$ containing the columns from i to j.
$\mathbf{X}[i,j]$	The sub-matrix of $\mathbf{X}$ indexed by i and j, i and j might be sets of indices.
$\mathbf{X}[i,j]^{-1}$	The inverse of the sub-matrix of $\mathbf{X}$ indexed by i and j, i and j might be sets of indices.
$\mathbf{X}^{-1}[i,j]$	The sub-matrix of $\mathbf{X}^{-1}$ indexed by i and j, i and j might be sets of indices.

Operations

$\mathbf{x}^p$	Each element of $\mathbf{x}$ raised to the power p, i.e. $\mathbf{x}^p = \{x_i^p\}_{i=1}^n$ for any vector $\mathbf{x}$ of length n.
$\|\|\mathbf{x}\|\|$	The L-2 norm of vector $\mathbf{x}$, i.e. $\|\|\mathbf{x}\|\| = (\sum_{i=1}^m x_i^2)^{\frac{1}{2}}$.
$\|\mathbf{X}\|$	The determinant of matrix $\mathbf{X}$.
$\mathrm{tr}\,(\mathbf{X})$	The trace of matrix $\mathbf{X}$, i.e. $\mathrm{tr}\,(\mathbf{X}) = \sum_{i=1}^n \mathbf{X}[i,i]$ for any matrix $\mathbf{X}$ of size $n \times n$.
$\mathrm{diag}\,[\mathbf{x}]$	A diagonal matrix where the elements in the diagonal are the corresponding elements in vector $\mathbf{x}$.
$\mathrm{diag}\,[\mathbf{X}]$	A matrix containing only the diagonal elements of $\mathbf{X}$ and zeros elsewhere.
$\mathbf{A} \oplus \mathbf{b}$	Broadcast addition, i.e. adding the vector $\mathbf{b}$ to each column in $\mathbf{A}$, and similarly for $\mathbf{b} \oplus \mathbf{A}$. If $\mathbf{b}$ is a scalar, the operation is applied to all elements of the matrix $\mathbf{A}$.

$\mathbf{A} \ominus \mathbf{b}$	Broadcast subtraction, i.e. subtracting the vector $\mathbf{b}$ from each column in $\mathbf{A}$, and vice versa for $\mathbf{b} \ominus \mathbf{A}$. If $\mathbf{b}$ is a scalar, the operation is applied to all elements of the matrix $\mathbf{A}$.
$\mathbf{A} \oslash \mathbf{b}$	Broadcast division, i.e. an element-wise division of each column in matrix $\mathbf{A}$ by the vector $\mathbf{b}$, and vice versa for $\mathbf{b} \oslash \mathbf{A}$. If $\mathbf{b}$ is a scalar, the operation is applied to all elements of the matrix $\mathbf{A}$.
$\mathbf{A} \odot \mathbf{b}$	Broadcast multiplication, i.e. an element-wise multiplication between the vector $\mathbf{b}$ and each column in $\mathbf{A}$, and similarly for $\mathbf{b} \odot \mathbf{A}$. If $\mathbf{b}$ is a scalar, the operation is applied to all elements of the matrix $\mathbf{A}$.
$\mathbf{A} \odot \mathbf{B}$	Element-wise multiplication between each element in matrix $\mathbf{A}$ and its corresponding location in matrix $\mathbf{B}$.

Number Sets

$\mathbb{Z}$	The set of integers.
$\mathbb{N}$	The set of natural numbers.
$\mathbb{R}$	The set of real numbers.

Probability and Statistics

$\mathbb{E}[X]$	The expected value of the random variable X.
$\mathbb{V}[X]$	The variance of the random variable X.
$\mathbb{C}[X, Y]$	The covariance between the two random variables X and Y.
$\mathcal{N}(\mathbf{x}\|\boldsymbol{\mu}, \boldsymbol{\Sigma})$	A multivariate Gaussian probability density function over $\mathbf{x}$, with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$.

$\text{Gamma}(X\|\alpha,\beta)$	A gamma probability density function over X, with shape parameter α and rate parameter β.

Chapter 1

Introduction

Gaussian Processes (Rasmussen and Williams, 2006) are powerful probabilistic models for regression and classification that are easy to implement, but the squared storage cost and the cubic computational complexity to train them makes them impractical for many applications where scalability is a major concern. This complexity thus created a demand for more efficient approximations. One approach to approximate a GP is to reduce the computational cost required to invert the $n \times n$ covariance matrix which gives GPs their computational complexity, where n is the number of samples in the training set. One can take advantage of the structure of the covariance matrix, if the recordings are evenly spread in a time series problem for example, then the covariance matrix will have a Toeplitz structure which can be inverted much faster (Zhang et al., 2005). Another approach is to decompose the covariance matrix as a sum of Kronecker products to simplify the computation of the inverse (Tsiligkaridis and Hero, 2013). These properties do not always hold; however, the covariance matrix will always be positive semi-definite matrix which one can exploit to compute a good approximation of the inverse by treating the problem as a system of linear equations and using the Conjugate Gradient (CG) method to solve it (Gibbs and MacKay, 1997). However, the inverse needs to be computed several times during the optimisation process, providing an approximate inverse to the optimiser will cause it to be unstable.

A second class of approaches avoids the complexity of inverting a large matrix by reducing the size of the covariance matrix by means of sparse approximations, or replacing the n samples with a smaller set of m data, and using the $m \ll n$ samples to construct the covariance matrix. The samples can be pre-selected either randomly or in an unsupervised fashion such as in Foster et al. (2009) where the active set is selected to increase the stability of the computation. Alternatively, one may search for "inducing" points, not necessarily present in the training set, and not necessarily even lying within the data range, to use as the active set such that the probability of the original data being generated from the model is maximised (Snelson and Ghahramani, 2006). A comprehensive overview of sparse approximation methods is detailed in Candela and Rasmussen (2005) and in Chapter 2 we show the relationship between them as well as discussing the linkage between GPs, artificial neural networks and sparse auto-encoders.

In addition to the computational and storage concerns, GPs typically lack some desired modelling capabilities such as input-dependent noise prediction and handling noisy or incomplete data. Generating an input-dependent predictive variance, which models heteroscedastic noise, is vital for applications where there is a bound on the acceptable error or in order to identify where the data needs to be improved in terms of quality or quantity during the data acquisition phase. Although classical GPs provide a posterior variance estimate as a function of the input, it is important to note that this variance reflects our uncertainty about the *mean* function, not the output; which is sensitive to variable data density and input-dependent intrinsic noise in the data. We note that, the variance at a specific input location will always decrease with the number of samples observed during training at some location, regardless of the actual noise present in the data. The intrinsic noise variance is typically assumed to be a constant Normally distributed process in GP models. Except for the work detailed in Snelson and Ghahramani (2006), none of the previously discussed methods account

for variable noise, This is achieved in Snelson and Ghahramani (2006) by use of richer covariance that is a function of the input. However, the covariance function takes a fixed form, similar to the form of the posterior variance estimate, which in effect forces the model to increase the uncertainty about the function in noisy regions of the input space even if that region is dense. This has the undesirable effect that, the uncertainty about the function might be high even in densely populated regions of the space. We regard this property as counter intuitive, our certainty about the mean function should always increase as we observe more data. Ideally, we would like to model these two sources of uncertainties separately to identify regions where data is sparse versus areas where more precise measurements, or more features, are needed. One can fit a GP model, then regress on the log of the its errors; however, this approach approximates the expected value of the log variance of the residuals, which is problematic. As we will show in Section 3.3, the exponential of the expected log variance is the median not the mean variance. To obtain a more accurate estimate of the mean variance one needs to account for the uncertainty about the log variance; thus, this approach itself requires a heteroscedastic variance estimation. One can separate the posterior mean and variance and consider a dual estimation, using for example, an augmented output from a GP, or pair of GPs. This is achieved by first holding the noise fixed and optimising with respect to the mean, then holding the mean fixed and optimising with respect to the noise and repeating until convergence (Kersting et al., 2007), this can be viewed as a group-coordinate ascent optimisation. We provide in Chapter 3 a more formal analysis, specifically in Sections 3.2 and 3.3. We also introduce extra features for custom sample weighting and prior mean function co-optimisation in Sections 3.4 and 3.5 respectively.

In addition to the numerical aspects of GPs, other modelling decisions can also affect the computational cost and the accuracy of the model. For example, one kernel type might require a greater number of inducing points to reach the same accuracy

that can be achieved by a different kernel function, some kernels have more expensive gradients to compute and not all kernels guarantee universality, i.e. the ability to approximate any arbitrary continuous function. We explore these modelling decisions in Chapter 4 and provide an efficient way to optimise much richer kernels to use the minimal number of inducing points while maximising the accuracy of the model.

In real world problems, we would also expect to be uncertain about the input as well as the output. There is always some degree of uncertainty associated with any measurement. Moreover, this degree of uncertainty might vary if the measurement process during deployment is different from that used during training, e.g. different sensors. If inputs are associated with uncertainty, or precision, one might exclude very uncertain inputs during training and use the associated variance to generate random samples and compute an expected value of the function during testing. However, excluding uncertain data during training might result in an insufficient sample size for training for small datasets or in a biased sample if the noise is correlated with the input, e.g. removing all night photos from the data set. One can use the associated uncertainty to sample from a Normal distribution and generate an average prediction during training, this however comes with an extra computational cost. Training with input noise can be shown to be equivalent to Tikhonov regularisation (Bishop, 1995), hence learning a smoother version of the function compared to training without input noise. Furthermore, it can be shown that the expected value of the output given a noisy input is not equal to the expected value of the output given the *hidden* noiseless input, and that the expected variance will have an added term that depends on the input uncertainty (Wright, 1999). In the context of neural networks, by treating the unobserved noiseless input as hidden variables and using Markov Chain Monte Carlo (MCMC) to sample jointly the network weights, one can obtain a more accurate inference from the noiseless input (Wright, 1999). In the context of GPs, the approach proposed in Mchutchon and Rasmussen (2011) uses Taylor series expansion to express

the model as a function of the noisy input plus a function of its gradient and the input variance; hence, learning heteroscedastic noise. The intuition is that, in areas of the input space where the output function is relatively flat, i.e. the gradient is zero, the input noise has less effect on the mapping as compared to areas where the input changes rapidly or has a high gradient.

A more difficult scenario that must be considered, is when a variable is completely missing. This is also a common problem in real world applications especially during deployment, as some data is not available during testing. Many machine learning models do not address this problem in a principled way, but rather use techniques to approximate the missing variables or build separate models targeting reduced subsets of the variables to reduce the chance of observing a sample with a missing input. An example of such a method is the random forest model (Breiman, 2001), which trains many decision trees on random subsets of the features using random samples from the data with replacement. Decision trees sub-divide the data based on a series of decisions, learned during training, then reports the average of the samples satisfying these constraints as the predictive mean and its standard deviation as the predictive variance. During prediction, a missing variable is considered a decision branch of the tree. An example of a model that fits data with missing values in a probabilistic way, but prone to overfitting, is Expectation Maximisation (EM) for Gaussian Mixture Models (GMM) (Murphy, 2012, p. 372). We address these challenges more formally in Chapter 5, where we propose an analytical approach to produce inferences from missing and uncertain inputs in the context of GPs with radial basis functions (RBF) and provide comparisons with other methods and techniques.

Finally, we apply the proposed methods to a problem in astrophysics that utilises these features, namely to predict galactic redshifts (distances) from their photometry (light intensities at different ranges) in order to study the geometry and evolution of the Universe. Importantly, the goals of current and future cosmology projects' suc-

cess rely on having accurate photometric redshift estimation models. The datasets involved are noisy and non-random, e.g. faint sources (which tend to be distant galaxies) tend to have more intrinsic uncertainty; thus, removing noisy input will exclude certain types of galaxies from being modelled. The proposed approach outperforms the state of the art methods in photometric redshift estimation, including artificial neural networks and random forests. Results on simulated as well as real datasets are presented, discussed and analysed in Chapter 6.

Chapter 2

Gaussian Processes for Regression

In this chapter, we introduce the basis function methodology by starting with a description of full GPs in Section 2.1 followed by a discussion of reducing the computational complexity through numerical approximation using a subset of the training data as an "active set" in Section 2.2 or an "inducing set" through sparse approximations in Section 2.3. In Section 2.4 we show how linear combinations of basis functions are equivalent to a sparse GP but with a different prior on the latent variable. We conclude in Section 2.5 by highlighting the relations between basis function models, artificial neural networks and sparse auto-encoders. The content of this chapter is a review of related work, except for the last subsection which is a new contribution.

2.1 Full Gaussian Processes

A GP is a supervised non-linear function modelling lying within the class of Bayesian non-parametric models due to the few explicit parametric assumptions that it makes about the nature of the function fit. We consider a dataset $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$ consisting of input $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n \in \mathbb{R}^{d \times n}$ and target outputs $\mathbf{y} = \{y_i\}_{i=1}^n \in \mathbb{R}^n$, where n is the number of samples in the data set and d is the dimensionality of the input. The underlying assumption of regression is that the observed target y_i is generated by a function of the input $\mathbf{x}_i$ plus additive noise ϵ_i:

$$y_i = f(\mathbf{x}_i) + \epsilon_i, \tag{2.1}$$

as in the norm, we take $\epsilon_i \sim \mathcal{N}(0, \sigma^2) \forall i \in \{1, \ldots, n\}$, then we have a Gaussian likelihood. It is assumed that $\mathbf{y}$ has a zero mean (this can readily be achieved without loss of generality) and is univariate, although the derivation can be readily extended to the multivariable case. The likelihood, the probability of observing the targets, given the function, is hence distributed as follows:

$$p\left(\mathbf{y}|\mathbf{f}_\mathbf{x}, \sigma^2\right) = \mathcal{N}\left(\mathbf{y}|\mathbf{f}_\mathbf{x}, \sigma^2\mathbf{I}_n\right), \tag{2.2}$$

where $\mathbf{f}_\mathbf{x} = \left[f\left(\mathbf{x}_1\right), \ldots, f\left(\mathbf{x}_n\right)\right]^T$. A GP then proceeds by applying Bayes' theorem to infer the distribution of the function $\mathbf{f}_\mathbf{x}$ given the observations:

$$p\left(\mathbf{f}_\mathbf{x}|\mathbf{y}, \mathbf{X}, \sigma^2\right) = \frac{p\left(\mathbf{y}|\mathbf{f}_\mathbf{x}, \sigma^2\right) p\left(\mathbf{f}_\mathbf{x}|\mathbf{X}\right)}{p\left(\mathbf{y}|\mathbf{X}, \sigma^2\right)}. \tag{2.3}$$

This requires us to define a prior, $p\left(\mathbf{f}_\mathbf{x}|\mathbf{X}\right)$, over our space of functions. Most widely used priors assume local similarity in the data, i.e. closeby inputs are mapped to similar outputs. More formally, we assume a normally distributed prior with a mean of zero, to match the mean of the normalised target $\mathbf{y}$, with a covariance function $\mathbf{K}$ to capture our prior belief of data locality, i.e. $p\left(\mathbf{f}_\mathbf{x}|\mathbf{X}\right) = \mathcal{N}\left(\mathbf{0}, \mathbf{K}\right)$. The covariance $\mathbf{K}$ is modelled as a function of the input, $\mathbf{K} = \kappa\left(\mathbf{X}, \mathbf{X}\right)$. Each element at the i-th row and the j-th column of $\mathbf{K}$ is set equal to $\kappa\left(\mathbf{x}_i, \mathbf{x}_j\right)$, where κ is the covariance (kernel) function. The function κ cannot be any arbitrary mapping, as it has to guarantee that $\mathbf{K}$ is a valid covariance matrix, i.e. symmetric and positive semi-definite. A class of functions referred to as Mercer kernels guarantees these structural constraints (Mercer, 1909). The following are examples of valid Mercer kernels:

$$\text{Radial Basis Function} : \kappa\left(\mathbf{x}_i, \mathbf{x}_j\right) = h^2 \exp\left(-\frac{1}{2\lambda^2}\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2\right), \tag{2.4}$$

$$\text{Polynomial} : \kappa\left(\mathbf{x}_i, \mathbf{x}_j\right) = \left(\mathbf{x}_i^T\mathbf{x}_j + b\right)^p, \tag{2.5}$$

$$\text{Sigmoidal} : \kappa\left(\mathbf{x}_i, \mathbf{x}_j\right) = \left(1 + \exp\left(-\lambda\mathbf{x}_i^T\mathbf{x}_j\right)\right)^{-1}. \tag{2.6}$$

The kernel choice is largely a domain-dependant modelling decision that captures the concept of *similarity* between inputs. The reader is referred to Rasmussen and

Williams (2006) or Roberts et al. (2013) for in-depth discussion of covariance functions and kernels. With a likelihood $p(\mathbf{y}|\mathbf{f}_\mathbf{x}, \boldsymbol{\theta})$ and a prior $p(\mathbf{f}_\mathbf{x}|\mathbf{X}, \boldsymbol{\theta})$, the marginal likelihood $p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})$ can be computed from Equations (A.6) and (A.14) as follows:

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \int p(\mathbf{y}|\mathbf{f}_\mathbf{x}, \boldsymbol{\theta})\, p(\mathbf{f}_\mathbf{x}|\mathbf{X}, \boldsymbol{\theta})\, \mathrm{d}\mathbf{f}_\mathbf{x}, \tag{2.7}$$

$$= \mathcal{N}\left(\mathbf{y}|0, \mathbf{K} + \sigma^2 \mathbf{I}_n\right). \tag{2.8}$$

The parameters of the kernel, $\boldsymbol{\varphi}$ and the noise variance σ^2, are collectively referred to as the hyper-parameters of the model, which we denote here by $\boldsymbol{\theta} = \{\boldsymbol{\varphi}, \sigma^2\}$. For example, for the RBF kernel $\boldsymbol{\varphi} = \{h, \lambda\}$.

The probability distribution of the function $\mathbf{f}_*$ for future test cases $\mathbf{X}_*$ given the training set, can be inferred from the joint distribution of $\mathbf{f}_*$ and the observed targets $\mathbf{y}$. If we assume that the joint distribution is a multivariate Gaussian, then the joint probability is distributed as follows:

$$p(\mathbf{y}, \mathbf{f}_*|\mathbf{X}, \mathbf{X}_*, \boldsymbol{\theta}) = \mathcal{N}\left(\begin{bmatrix}\mathbf{y} \\ \mathbf{f}_*\end{bmatrix} \middle| \mathbf{0}, \begin{bmatrix}\mathbf{K_{xx}} + \sigma^2 \mathbf{I}_n & \mathbf{K_{x*}} \\ \mathbf{K_{*x}} & \mathbf{K_{**}}\end{bmatrix}\right), \tag{2.9}$$

where $\mathbf{K_{xx}} = \kappa(\mathbf{X}, \mathbf{X})$, $\mathbf{K_{x*}} = \kappa(\mathbf{X}, \mathbf{X}_*)$, $\mathbf{K_{*x}} = \kappa(\mathbf{X}_*, \mathbf{X})$ and $\mathbf{K_{**}} = \kappa(\mathbf{X}_*, \mathbf{X}_*)$. Therefore, the conditional probability $p(\mathbf{f}_*|\mathbf{X}_*, \mathcal{D}, \boldsymbol{\theta})$, from Equation (A.11), is distributed as follows:

$$p(\mathbf{f}_*|\mathbf{X}_*, \mathcal{D}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{f}_*|\mathbb{E}[\mathbf{f}_*], \mathbb{V}[\mathbf{f}_*]), \tag{2.10}$$

$$\mathbb{E}[\mathbf{f}_*] = \mathbf{K_{*x}}\left(\mathbf{K_{xx}} + \sigma^2 \mathbf{I}_n\right)^{-1} \mathbf{y}, \tag{2.11}$$

$$\mathbb{V}[\mathbf{f}_*] = \mathbf{K_{**}} - \mathbf{K_{*x}}\left(\mathbf{K_{xx}} + \sigma^2 \mathbf{I}_n\right)^{-1} \mathbf{K_{x*}}. \tag{2.12}$$

Finally, the predictive distribution of future observations $\mathbf{y}_*$ given the dataset can be derived from Equations (A.6) and (A.14) as follows:

$$p(\mathbf{y}_*|\mathbf{X}_*, \mathcal{D}, \boldsymbol{\theta}) = \int p(\mathbf{y}_*|\mathbf{f}_*, \boldsymbol{\theta})\, p(\mathbf{f}_*|\mathbf{X}_*, \mathcal{D}, \boldsymbol{\theta})\, \mathrm{d}\mathbf{f}_*, \tag{2.13}$$

$$= \mathcal{N}(\mathbf{y}_*|\mathbb{E}[\mathbf{y}_*], \mathbb{V}[\mathbf{y}_*]), \tag{2.14}$$

$$\mathbb{E}[\mathbf{y}_*] = \mathbb{E}[\mathbf{f}_*], \tag{2.15}$$

$$\mathbb{V}[\mathbf{y}_*] = \mathbb{V}[\mathbf{f}_*] + \sigma^2 \mathbf{I}_{n_*}, \tag{2.16}$$

where n_* is the number of test cases. The hyper-parameters of the model are then optimised by maximising the log of the marginal likelihood in Equation (2.8):

$$\ln p\left(\mathbf{y}|\mathbf{X},\boldsymbol{\theta}\right) = -\frac{1}{2}\mathbf{y}^T\left(\mathbf{K}+\sigma^2\mathbf{I}_n\right)^{-1}\mathbf{y} - \frac{1}{2}\ln\left|\mathbf{K}+\sigma^2\mathbf{I}_n\right| - \frac{n}{2}\ln\left(2\pi\right). \tag{2.17}$$

Figure 2.1 illustrates the effect of changing the hyper-parameters on the function and the marginal likelihood using an RBF kernel on a synthetic univariate example. The input training set consists of 1000 samples that are linearly spaced from -10 to 10 and $y_i \sim \mathcal{N}\left(\mathrm{sinc}\left(x_i\right), 0.01\right)$. Note that the set of parameters which visually fits the data best has the maximum marginal likelihood value. We used the GPML toolbox implementation (Rasmussen and Nickisch, 2010) to generate the plots in Figure 2.1 and the optimisation procedure is discussed in more length in Chapter 4.

2.2 Numerical Approximation

2.2.1 Low Rank Approximation

The time complexity cost for training a full GP is $O\left(n^3\right)$ which comes from inverting the $n \times n$ covariance matrix. Therefore, one must be selective about the quality of the samples in the training set not just the quantity. Even though the quality of the fit will improve as we observe more data, but do the benefits outweigh the costs of training? For example, increasing the amount of data by 10 folds might only improve the accuracy of the model by a small percentage but will increase the cost of training by 1000. Furthermore, in case of repeated identical examples, the covariance matrix will be rank deficient and therefore not invertible under any covariance function mapping. Therefore, we would like to be selective about the training samples contributing to the covariance matrix without discarding the remaining data points. This can be achieved through numerical approximation of the covariance matrix using Nyström's method (Kumar et al., 2012). Consider that we have a representative set of $m \ll n$ points $\mathbf{P} = \{\mathbf{p}_j\}_{j=1}^m \in \mathbb{R}^{d\times m}$, where $\mathbf{p}_j = \mathbf{x}_i$ for some $i \in \{1,\ldots,n\}$ and $\mathbf{p}_j \neq \mathbf{p}_k \forall j \neq k$,

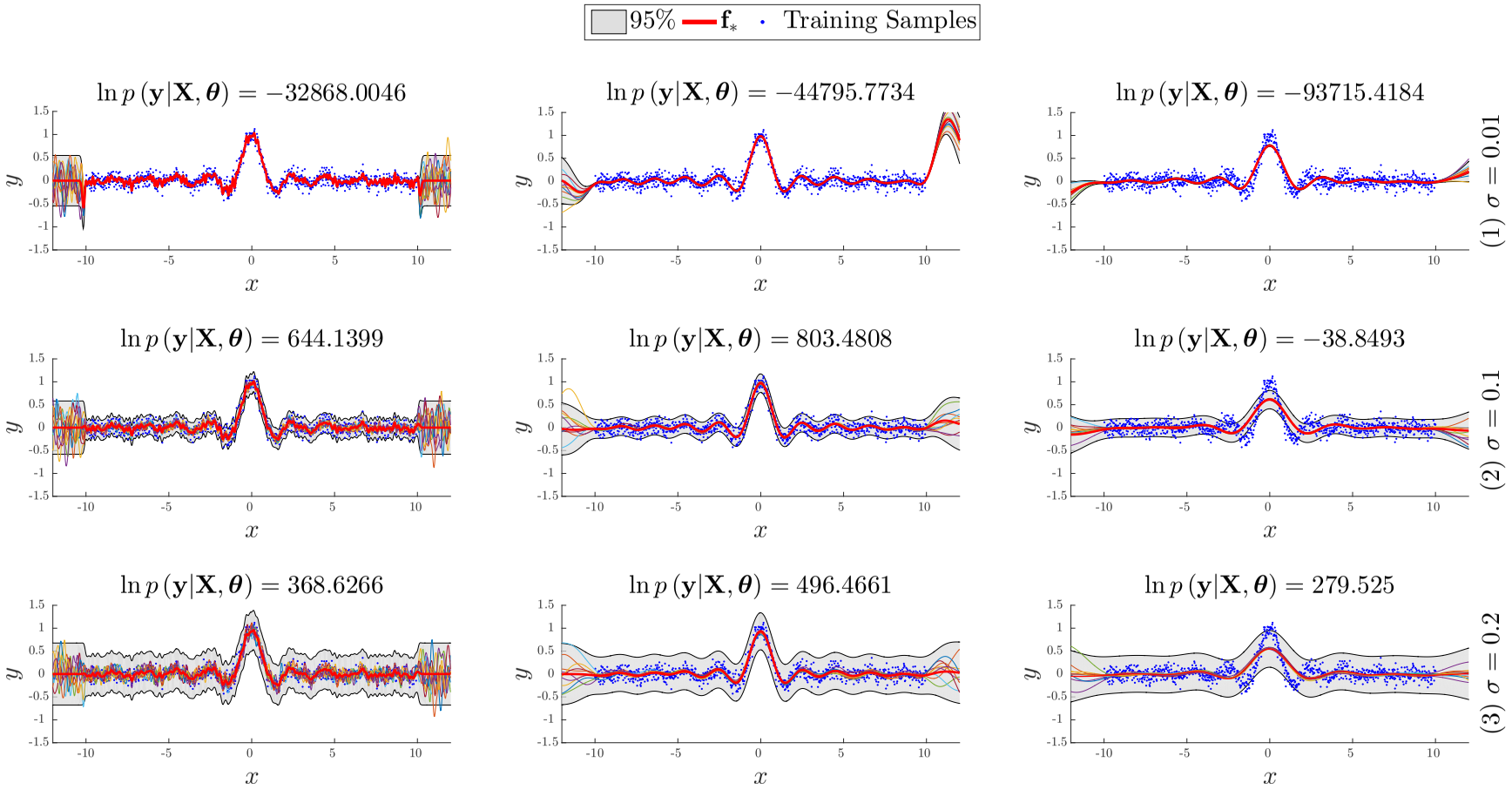


Figure 2.1: The effect of changing the hyper-parameters λ, from (a) to (c), and σ, from (1) to (3), of an RBF kernel on a GP model. The plots show the mean function (red), draws from the function distribution (Equation (2.10)) and the 95% confidence range (grey), i.e. plus or minus two standard deviations from the mean. The value of h was set to the optimal value achieved after optimising the log marginal likelihood (shown above each plot).

then the covariance matrix can be approximated as follows:

$$\mathbf{K_{xx}} \approx \tilde{\mathbf{K}}_{\mathbf{xx}} = \mathbf{K_{xp}}\mathbf{K_{pp}^{-1}}\mathbf{K_{px}}, \tag{2.18}$$

where $\mathbf{K_{xp}} = \mathbf{K_{px}^T}$ is a $n \times m$ matrix such that $\mathbf{K_{xp}}[i,j] = \kappa(\mathbf{x}_i, \mathbf{p}_j)$ and $\mathbf{K_{pp}}$ is a $m \times m$ matrix such that $\mathbf{K_{pp}}[i,j] = \kappa(\mathbf{p}_i, \mathbf{p}_j)$. More generally, let $\tilde{\mathbf{K}}_{\mathbf{ab}} = \mathbf{K_{ap}}\mathbf{K_{pp}^{-1}}\mathbf{K_{pb}}$, then the approximate covariance matrix can now be used to approximate the mean and variance functions in Equations (2.11) and (2.12), respectively, as follows:

$$\mathbb{E}[\mathbf{f}_*] \approx \tilde{\mathbb{E}}[\mathbf{f}_*] = \tilde{\mathbf{K}}_{*\mathbf{x}}\left(\tilde{\mathbf{K}}_{\mathbf{xx}} + \sigma^2\mathbf{I}_n\right)^{-1}\mathbf{y}, \tag{2.19}$$

$$\mathbb{V}[\mathbf{f}_*] \approx \tilde{\mathbb{V}}[\mathbf{f}_*] = \tilde{\mathbf{K}}_{**} - \tilde{\mathbf{K}}_{*\mathbf{x}}\left(\tilde{\mathbf{K}}_{\mathbf{xx}} + \sigma^2\mathbf{I}_n\right)^{-1}\tilde{\mathbf{K}}_{\mathbf{x}*}. \tag{2.20}$$

We can compute Equations (2.19) and (2.20) more efficiently by using the matrix inversion lemma (Equation (A.21)) as follows:

$$\tilde{\mathbb{E}}[\mathbf{f}_*] = \sigma^{-1}\mathbf{K}_{*\mathbf{p}}\mathbf{\Sigma}_{\mathbf{p}}^{-1}\mathbf{K_{px}}\mathbf{y}, \tag{2.21}$$

$$\tilde{\mathbb{V}}[\mathbf{f}_*] = \mathbf{K}_{*\mathbf{p}}\mathbf{\Sigma}_{\mathbf{p}}^{-1}\mathbf{K}_{\mathbf{p}*}, \tag{2.22}$$

$$\mathbf{\Sigma_p} = \sigma^{-2}\mathbf{K_{px}}\mathbf{K_{xp}} + \mathbf{K_{pp}}. \tag{2.23}$$

Note that since the computational cost of inversion is now $O(m^3)$, the overall computational cost has now been reduced to $O(nm^2)$ since $m < n$, the approximation performance however depends on the selection of the representative points in $\mathbf{P}$, also known as the active set. The active set can be chosen either randomly for speed, best approximation (Seeger et al., 2003; Smola and Bartlett, 2000) or numerical stability (Foster et al., 2009).

2.2.2 The Conjugate Gradient Method

Another numerical approximation method approximates the solution of the linear system $(\mathbf{K_{xx}} + \sigma^2\mathbf{I}_n)\mathbf{w} = \mathbf{y}$, making the approximate expected value $\mathbb{E}[\mathbf{f}_*] = \mathbf{K}_{*\mathbf{x}}\mathbf{w}$. Since the matrix $\mathbf{K_{xx}} + \sigma^2\mathbf{I}_n$ is positive-semidefinite, one can use the iterative conjugate gradient (CG) method to solve for $\mathbf{w}$ (Gibbs and MacKay, 1997). The conjugate

gradient method is an iterative algorithm to solve for $\mathbf{x}$ in a system of linear equations $\mathbf{Ax} = \mathbf{b}$ and has a time complexity of $O(n^2 t)$, where t is the number of iterations. If $\mathbf{A}$ is positive-semidefinite, then it is guaranteed that $t \leq n$. Since $(\mathbf{K_{xx}} + \sigma^2 \mathbf{I})$ satisfies this condition, the maximum time complexity for the conjugate gradient method is no worse than that of a full GP; however, we can always terminate the algorithm early when a satisfactory approximation is achieved. The relationship between the number of iterations and the approximation performance, the convergence rate, highly depends on the structure of the matrix $\mathbf{A}$, specifically its condition number. The condition number of a matrix, formally the ratio between its largest and smallest magnitude eigenvalues, indicates the rate of change in the approximation as we change the solution. For example, if the condition number of matrix $\mathbf{A}$ is high, then a very small change in $\mathbf{x}$ will result in a very large change in the matrix-vector multiplication $\mathbf{Ax}$. In the context of CG, the higher the condition number, the higher the minimum number of iterations t required to reach a specific accuracy, say ϵ. The condition of the matrix will increase, for example, if the training set contains data points that are very close to each other or, in the worst case, will be infinite if two or more data points are identical; because, in theory, the matrix will have a minimum magnitude eigenvalue of zero. We can achieve a better convergence rate by *pre-conditioning* the matrix, i.e. instead of solving the original system $\mathbf{Ax} = \mathbf{b}$, we solve for $\mathbf{x}$ in $\mathbf{M}^{-1}\mathbf{Ax} = \mathbf{M}^{-1}\mathbf{b}$ such that the condition of $\mathbf{M}^{-1}\mathbf{A}$ is less than that of $\mathbf{A}$. The matrix $\mathbf{M}$ should approximate $\mathbf{A}$ but be easier to invert. A simple choice is the Jacobian pre-conditioner $\mathbf{M} = \text{diag}[\mathbf{A}]$; and if $\mathbf{A}$ is sparse, then the incomplete Cholesky factorisation $\mathbf{M} = \mathbf{L}\mathbf{L}^T$ is suitable. There is a trade-off between enhancing the rate of convergence of CG and the extra computational cost for computing $\mathbf{M}$, an extreme case for example is $\mathbf{M} = \mathbf{A}$. For large values of n, pre-conditioning is expected to be applied in most applications and, in the context of GPs, CG can be combined with low-rank approximation to reduce the time complexity further to

$O(nmt)$ (Davies, 2014).

Even though numerical approximations might reduce the time complexity of GPs, these approximations most often are not suitable to be used in conjunction with hyper-parameter optimisation, as this makes the optimisation procedure unstable. Furthermore, the choice of the "active set" is a combinatorial problem and is restricted to the samples in the training set. In the next section, we will see how sparse approximation GPs address these issues by replacing the active set with an "inducing set", which treats the issue of finding the samples as a search problem rather than a subset selection problem.

2.3 Sparse Gaussian Processes

2.3.1 Subset of Regressors

Sparse GP approximation methods introduce $m \ll n$ latent variables, $\mathbf{f_p}$, referred to as the *inducing* variables, which are the values of a Gaussian Process corresponding to a set of inducing points $\mathbf{P} = [\mathbf{p}_1, \ldots, \mathbf{p}_m] \in \mathbb{R}^{d \times m}$, where d is the dimensionality of the input. Due to the probabilistic treatment of GPs, the joint probability distribution $p(\mathbf{f_x}, \mathbf{f}_*)$ can be written as:

$$p(\mathbf{f_x}, \mathbf{f}_*) = \int p(\mathbf{f_x}, \mathbf{f}_*, \mathbf{f_p}) \, \mathrm{d}\mathbf{f_p} = \int p(\mathbf{f_x}, \mathbf{f}_* | \mathbf{f_p}) \, p(\mathbf{f_p}) \, \mathrm{d}\mathbf{f_p}, \tag{2.24}$$

where we set the distribution of the prior over the latent variable $\mathbf{f_p}$ as per the distribution of the prior over $\mathbf{f_x}$, or $p(\mathbf{f_p}) = \mathcal{N}(\mathbf{f_p}|0, \mathbf{K_{pp}})$. The expression in Equation (2.24) is exact, the approximation that gives rise to a group of sparse GP methods assumes that the conditional distributions of $\mathbf{f_x}$ and $\mathbf{f}_*$ are independent given $\mathbf{f_p}$. The marginal distribution in Equation (2.24) can then be approximated as follows:

$$p(\mathbf{f_x}, \mathbf{f}_*) \approx q(\mathbf{f_x}, \mathbf{f}_*) = \int q(\mathbf{f_x}|\mathbf{f_p}) \, q(\mathbf{f}_*|\mathbf{f_p}) \, p(\mathbf{f_p}) \, \mathrm{d}\mathbf{f_p}. \tag{2.25}$$

The exact expressions for the *training conditional* $p(\mathbf{f_x}|\mathbf{f_p})$ and the *test conditional* $p(\mathbf{f_*}|\mathbf{f_p})$ can be derived by treating the problem as a noise free Gaussian Process on the noiseless latent observations $\mathbf{f_p}$.

$$p(\mathbf{f_x}|\mathbf{f_p}) = \mathcal{N}\left(\mathbf{K_{xp}}\mathbf{K_{pp}^{-1}}\mathbf{f_p}, \mathbf{K_{xx}} - \mathbf{K_{xp}}\mathbf{K_{pp}^{-1}}\mathbf{K_{px}}\right), \tag{2.26}$$

$$p(\mathbf{f_*}|\mathbf{f_p}) = \mathcal{N}\left(\mathbf{K_{*p}}\mathbf{K_{pp}^{-1}}\mathbf{f_p}, \mathbf{K_{**}} - \mathbf{K_{*p}}\mathbf{K_{pp}^{-1}}\mathbf{K_{p*}}\right). \tag{2.27}$$

Different assumptions about the training and test conditionals produce different approximate distributions $q(\mathbf{f_x}|\mathbf{f_p}) \approx p(\mathbf{f_x}|\mathbf{f_p})$ and $q(\mathbf{f_*}|\mathbf{f_p}) \approx p(\mathbf{f_*}|\mathbf{f_p})$ that give rise to a group of different sparse GPs used in the literature (Schwaighofer and Tresp, 2002; Seeger et al., 2003; Smola and Bartlett, 2000). For an in-depth analysis of different sparse approximations for Gaussian Processes the reader is referred to Candela and Rasmussen (2005). Under the assumption that both the training and the test conditionals are deterministic

$$q(\mathbf{f_x}|\mathbf{f_p}) = \mathcal{N}\left(\mathbf{K_{xp}}\mathbf{K_{pp}^{-1}}\mathbf{f_p}, 0\right), \tag{2.28}$$

$$q(\mathbf{f_*}|\mathbf{f_p}) = \mathcal{N}\left(\mathbf{K_{*p}}\mathbf{K_{pp}^{-1}}\mathbf{f_p}, 0\right). \tag{2.29}$$

The effective prior $q(\mathbf{f_x}, \mathbf{f_*})$ in Equation (2.25) is distributed as follows:

$$q(\mathbf{f_x}, \mathbf{f_*}) = \mathcal{N}\left(\begin{bmatrix}\mathbf{f_x}\\ \mathbf{f_*}\end{bmatrix}\middle|\mathbf{0}, \begin{bmatrix}\mathbf{Q_{xx}} & \mathbf{Q_{x*}}\\ \mathbf{Q_{*x}} & \mathbf{Q_{**}}\end{bmatrix}\right), \tag{2.30}$$

where $\mathbf{Q_{ab}} = \mathbf{K_{ap}}\mathbf{K_{pp}^{-1}}\mathbf{K_{pb}}$. The problem now transforms to a full GP but with the equivalent covariance function $\mathbf{Q_{ab}}$ replacing the original covariance function $\mathbf{K_{ab}}$. However, since $\mathbf{Q_{xx}} = \mathbf{K_{xp}}\mathbf{K_{pp}^{-1}}\mathbf{K_{px}}$, one can exploit the matrix inversion lemma (Equation (A.21)) to bring the computational cost from $O(n^3)$ to $O(m^3)$.

$$\left(\mathbf{Q_{xx}} + \sigma^2\mathbf{I}_n\right)^{-1} = \left(\mathbf{K_{xp}}\mathbf{K_{pp}^{-1}}\mathbf{K_{px}} + \sigma^2\mathbf{I}_n\right)^{-1}, \tag{2.31}$$

$$= \sigma^{-2}\mathbf{I}_n - \sigma^{-2}\mathbf{K_{xp}}\mathbf{\Sigma_p^{-1}}\mathbf{K_{px}}\sigma^{-2}, \tag{2.32}$$

$$\mathbf{\Sigma_p} = \sigma^{-2}\mathbf{K_{px}}\mathbf{K_{xp}} + \mathbf{K_{pp}}. \tag{2.33}$$

The predictive mean and variance can be expressed more efficiently in terms of $\mathbf{K}$ as follows:

$$\mathbb{E}\left[\mathbf{f}_*\right] = \sigma^{-2}\mathbf{K}_{*\mathbf{p}}\boldsymbol{\Sigma}_{\mathbf{p}}^{-1}\mathbf{K}_{\mathbf{px}}\mathbf{y}, \tag{2.34}$$

$$\mathbb{V}\left[\mathbf{f}_*\right] = \mathbf{K}_{*\mathbf{p}}\boldsymbol{\Sigma}_{\mathbf{p}}^{-1}\mathbf{K}_{\mathbf{p}*}. \tag{2.35}$$

This is referred to as the subset of regressors (SoR) sparse Gaussian process (Candela and Rasmussen, 2005), which is identical to the low rank approximation method except that the set of inducing points are now optimised as part of the hyper-parameter set. Thus, the low rank approximation method is a sparse GP where both the training and test conditionals are deterministic and the set of inducing points are held fixed.

2.3.2 Fully Independent Training Conditional

The state of the art sparse approximation GP is the *Fully Independent Training Conditional* (FITC) (Snelson and Ghahramani, 2006), also known as *Sparse Pseudo-input Gaussian Processes* (SPGP), assumes the following training and test conditionals:

$$q\left(\mathbf{f}_\mathbf{x}|\mathbf{f}_\mathbf{p}\right) = \mathcal{N}\left(\mathbf{K}_{\mathbf{xp}}\mathbf{K}_{\mathbf{pp}}^{-1}\mathbf{f}_\mathbf{p}, \operatorname{diag}\left[\mathbf{K}_{\mathbf{xx}} - \mathbf{K}_{\mathbf{xp}}\mathbf{K}_{\mathbf{pp}}^{-1}\mathbf{K}_{\mathbf{px}}\right]\right), \tag{2.36}$$

$$q\left(\mathbf{f}_*|\mathbf{f}_\mathbf{p}\right) = p\left(\mathbf{f}_*|\mathbf{f}_\mathbf{p}\right). \tag{2.37}$$

Moreover, an additional assumption is made about the likelihood to give it a richer covariance and to allow for input-dependent noise prediction:

$$p\left(\mathbf{y}|\mathbf{f}_\mathbf{x}\right) \approx q\left(\mathbf{y}|\mathbf{f}_\mathbf{x}\right) = \mathcal{N}\left(\mathbf{f}_\mathbf{x}, \boldsymbol{\Lambda}\right), \tag{2.38}$$

where $\boldsymbol{\Lambda} = \operatorname{diag}\left[\mathbf{K}_{\mathbf{xx}} - \mathbf{Q}_{\mathbf{xx}}\right] + \sigma^2\mathbf{I}_n$. The effective prior in this case is distributed as follows:

$$q\left(\mathbf{f}_\mathbf{x}, \mathbf{f}_*\right) = \mathcal{N}\left(\begin{bmatrix}\mathbf{f}_\mathbf{x}\\ \mathbf{f}_*\end{bmatrix}\middle|0, \begin{bmatrix}\mathbf{Q}_{\mathbf{xx}} + \operatorname{diag}\left[\mathbf{K}_{\mathbf{xx}} - \mathbf{Q}_{\mathbf{xx}}\right] & \mathbf{Q}_{\mathbf{x}*}\\ \mathbf{Q}_{*\mathbf{x}} & \mathbf{K}_{**}\end{bmatrix}\right), \tag{2.39}$$

Using the new effective prior, the approximate marginal likelihood $q(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})$ is therefore

$$\ln q(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^T \left(\mathbf{Q_{xx}} + \boldsymbol{\Lambda}\right)^{-1} \mathbf{y} - \frac{1}{2}\ln\left|\mathbf{Q_{xx}} + \boldsymbol{\Lambda}\right| - \frac{n}{2}\ln(2\pi). \tag{2.40}$$

In addition to optimising with respect to the kernel and noise parameters, the inducing points' positions $\mathbf{P}$ can be treated as part of the hyper-parameters $\boldsymbol{\theta}$ when maximising the log marginal likelihood. Once the parameters have been optimised, the predictive distribution of the function for future test cases is distributed as follows:

$$q(\mathbf{f}_*|\mathbf{X}_*, \mathcal{D}, \boldsymbol{\theta}) = \mathcal{N}\left(\mathbb{E}[\mathbf{f}_*], \mathbb{V}[\mathbf{f}_*]\right), \tag{2.41}$$

$$\mathbb{E}[\mathbf{f}_*] = \mathbf{Q_{*x}}\left(\mathbf{Q_{xx}} + \boldsymbol{\Lambda}\right)^{-1}\mathbf{y}, \tag{2.42}$$

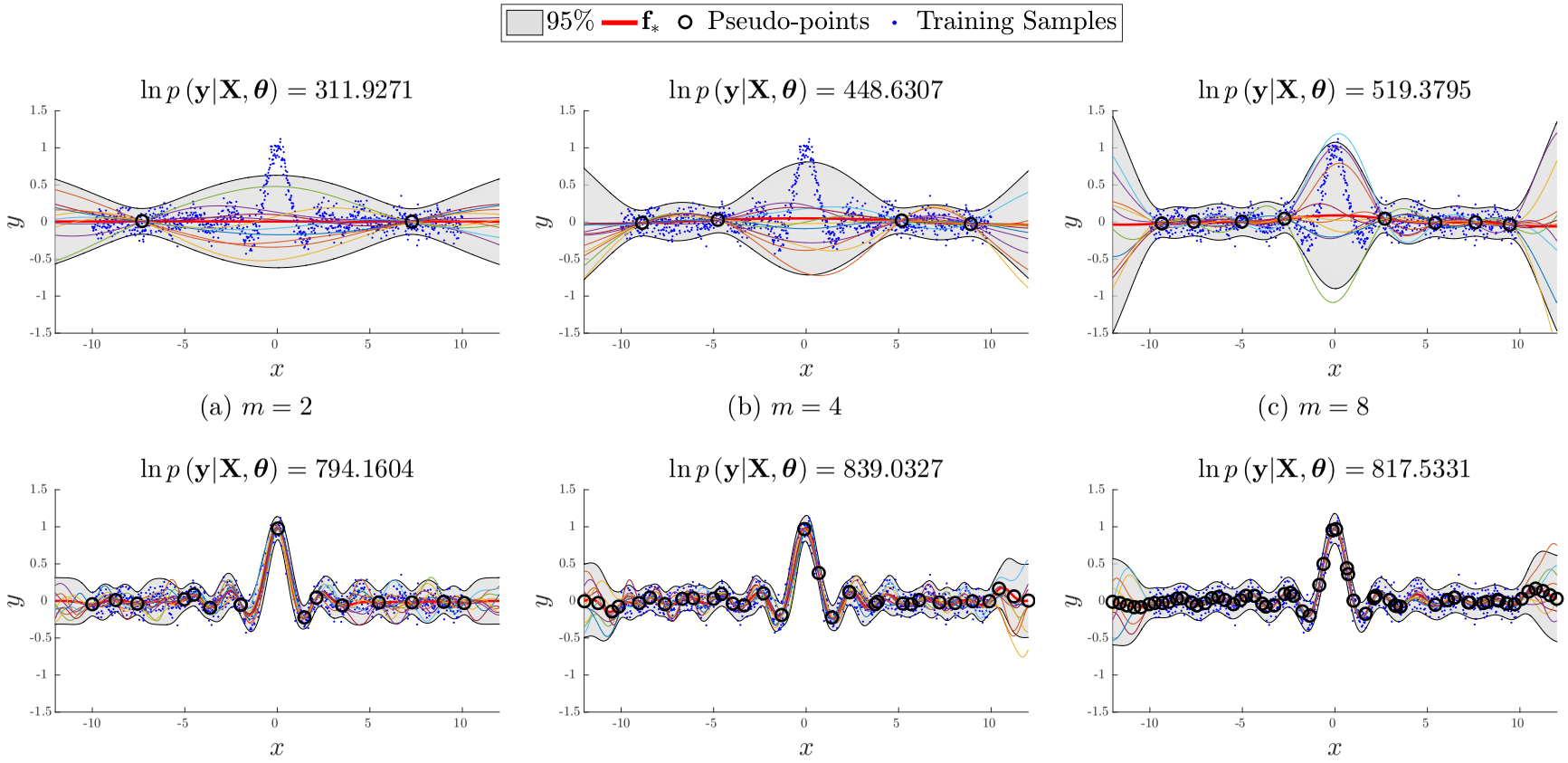
$$\mathbb{V}[\mathbf{f}_*] = \mathbf{K_{**}} - \mathbf{Q_{*x}}\left(\mathbf{Q_{xx}} + \boldsymbol{\Lambda}\right)^{-1}\mathbf{Q_{x*}}, \tag{2.43}$$

and more efficiently in terms of $\mathbf{K}$ as:

$$\mathbb{E}[\mathbf{f}_*] = \mathbf{K_{*p}}\boldsymbol{\Sigma}_\mathbf{p}^{-1}\mathbf{K_{px}}\boldsymbol{\Lambda}^{-1}\mathbf{y}, \tag{2.44}$$

$$\mathbb{V}[\mathbf{f}_*] = \mathbf{K_{**}} - \mathbf{Q_{**}} + \mathbf{K_{*p}}\boldsymbol{\Sigma}_\mathbf{p}^{-1}\mathbf{K_{p*}}. \tag{2.45}$$

Figure 2.2 shows the effects of using different numbers of pseudo-points on the distribution of the function and on the marginal likelihood using the same synthetic example from Figure 2.1. Note that the marginal likelihood does not necessary increase as the number of pseudo-points increases and that they tend to spread unevenly. This is an undesired effect, ideally, we would like the pseudo-points to be as spread out as possible to avoid any redundant information. We will see latter how this can be achieved by using automatic relevance determination, which we will discuss later in Section 3.1. Even though the computational cost is reduced significantly, sparse GP approximations are still restricted to a single kernel function which assumes a global relation between input and output across the input space. Moreover, the covariance function has to be a valid kernel function which further restricts our modelling capabilities. The work presented in Walder et al. (2008) extended FITC to allow for



(a) $m = 2$ (b) $m = 4$ (c) $m = 8$

(d) $m = 16$ (e) $m = 32$ (f) $m = 64$

Figure 2.2: The effect of changing the number of pseudo-points (m), from (a) to (f) in multiples of 2, on FITC using an RBF kernel. The plots show the mean function (red), draws from the function distribution (Equation (2.41)), the 95% confidence range (grey), i.e. plus or minus two standard deviations from the mean, and the locations of the pseudo-points (black). The log marginal likelihood values are shown above each plot.

non-stationarity, the kernels however are restricted to be Gaussian kernels with diagonal covariances. In the next section, we discuss how basis function models can be used to address these limitations.

2.4 Basis Function Models

In this section, we describe sparse Gaussian processes as basis function models (BFM), whose semi-parametric form is defined via a set of parameters. The underlying assumption in a BFM is that, given the inputs $\mathbf{X}$ and the target outputs $\mathbf{y}$, that the observed target y_i is generated by a linear combination of m non-linear functions $\boldsymbol{\phi}(\mathbf{x}_i) = [\phi_1(\mathbf{x}_i), \ldots, \phi_m(\mathbf{x}_i)]^T \in \mathbb{R}^m$ of the input plus additive noise $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$:

$$y_i = \boldsymbol{\phi}(\mathbf{x}_i)^T \mathbf{w} + \epsilon_i, \tag{2.46}$$

where $\mathbf{w}$ is a vector of length m of real-valued coefficients, or weights of the basis functions. Note that the mean of the predictive distribution derived from a GP is a linear combination of n kernel functions. The BFM approach is to assume the form of the function to be a linear combination of $m \ll n$ basis functions and integrate out its parameters. We assume, for now, that our observations are noisy with a constant noise variance σ^2 and a mean of zero. Note that these assumptions are made to simplify our illustration and we relax these later in the thesis. The likelihood is hence defined as follows:

$$\begin{aligned} p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}, \mathbf{w}) &= \int p(\mathbf{y}|\mathbf{f}_\mathbf{x}, \boldsymbol{\theta})\, p(\mathbf{f}_\mathbf{x}|\mathbf{X}, \boldsymbol{\theta}, \mathbf{w})\, \mathrm{d}\mathbf{f}_\mathbf{x}, && (2.47)\\ &= \int \mathcal{N}\left(\mathbf{y}|\mathbf{f}_\mathbf{x}, \sigma^2\mathbf{I}_n\right) \mathcal{N}\left(\mathbf{f}_\mathbf{x}|{\boldsymbol{\Phi}_\mathbf{x}}^T\mathbf{w}, 0\right) \mathrm{d}\mathbf{f}_\mathbf{x}, && (2.48)\\ &= \mathcal{N}\left(\mathbf{y}|{\boldsymbol{\Phi}_\mathbf{x}}^T\mathbf{w}, \sigma^2\mathbf{I}_n\right), && (2.49) \end{aligned}$$

where

$$\boldsymbol{\Phi}_\mathbf{x} = \begin{bmatrix} \phi_1(\mathbf{x}_1) & \ldots & \phi_1(\mathbf{x}_n) \\ \vdots & \ddots & \vdots \\ \phi_m(\mathbf{x}_1) & \ldots & \phi_m(\mathbf{x}_n) \end{bmatrix}. \tag{2.50}$$

We now need to define a prior on $\mathbf{w}$ in order to proceed. We use a prior that promotes a smooth function, hence preferring the simplest explanation that fits the data. The smoothness assumption also transforms the objective from an ill-posed problem to a well-posed one, as there are an infinite number of functions that would fit the data. Thus, in addition to the function fit, we also require the function to be as simple as possible, or more precisely have small derivatives. This can be achieved by requiring the norm of the weights in $\mathbf{w}$ to be as small as possible, so that a small change in the input space does not correspond to a large change in the output space, which can be formulated probabilistically by taking $p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|0, \mathbf{A^{-1}})$, where $\mathbf{A} = \alpha\mathbf{I}_m$ is the prior precision of the parameters $\mathbf{w}$ and is included as part of the parameter set $\boldsymbol{\theta} = \{\boldsymbol{\varphi}, \sigma^2, \alpha\}$. Using such prior in effect will apply a Tikhonov regularisation on the weights in the log space of $p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}, \mathbf{w})\, p(\mathbf{w}|\alpha)$ that favours functions with smaller derivatives, i.e. smooth, and best fit the data (Murphy, 2012, p. 124). We can now derive the effective prior of the function by integrating out $\mathbf{w}$

$$p(\mathbf{f_x}, \mathbf{f}_*|\mathbf{X}, \boldsymbol{\theta}) = \int p(\mathbf{f_x}, \mathbf{f}_*|\mathbf{X}, \boldsymbol{\theta}, \mathbf{w})\, p(\mathbf{w}|\alpha)\, \mathrm{d}\mathbf{w}, \tag{2.51}$$

$$= \int \mathcal{N}\left(\begin{bmatrix}\mathbf{f_x}\\ \mathbf{f}_*\end{bmatrix} \middle| \left[\boldsymbol{\Phi_x}, \boldsymbol{\Phi}_*\right]^T \mathbf{w}, 0\right) \mathcal{N}\left(\mathbf{w}|0, \mathbf{A^{-1}}\right) \mathrm{d}\mathbf{w}. \tag{2.52}$$

From Equations (A.6) and (A.17) in the appendix, this can be shown to have the following normal distribution:

$$p(\mathbf{f_x}, \mathbf{f}_*|\mathbf{X}, \boldsymbol{\theta}) = \mathcal{N}\left(\begin{bmatrix}\mathbf{f_x}\\ \mathbf{f}_*\end{bmatrix} \middle| 0, \left[\boldsymbol{\Phi_x}, \boldsymbol{\Phi}_*\right]^T \mathbf{A}^{-1} \left[\boldsymbol{\Phi_x}, \boldsymbol{\Phi}_*\right]\right), \tag{2.53}$$

$$= \mathcal{N}\left(\begin{bmatrix}\mathbf{f_x}\\ \mathbf{f}_*\end{bmatrix} \middle| 0, \begin{bmatrix}\boldsymbol{\Phi_x}^T\mathbf{A}^{-1}\boldsymbol{\Phi_x} & \boldsymbol{\Phi_x}^T\mathbf{A}^{-1}\boldsymbol{\Phi}_*\\ \boldsymbol{\Phi}_*^T\mathbf{A}^{-1}\boldsymbol{\Phi_x} & \boldsymbol{\Phi}_*^T\mathbf{A}^{-1}\boldsymbol{\Phi}_*\end{bmatrix}\right). \tag{2.54}$$

We have thus turned the problem again into a full GP with an equivalent covariance function $\mathbf{Q_{ab}} = \boldsymbol{\Phi_a}^T\mathbf{A}^{-1}\boldsymbol{\Phi_b}$. The marginal likelihood, can be derived by integrating out $\mathbf{f_x}$:

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \int p(\mathbf{y}|\mathbf{f_x}, \boldsymbol{\theta})\, p(\mathbf{f_x}|\mathbf{X}, \boldsymbol{\theta})\, \mathrm{d}\mathbf{f_x}, \tag{2.55}$$

$$= \mathcal{N}\left(\mathbf{y}|0, \boldsymbol{\Phi_x}^T\mathbf{A}^{-1}\boldsymbol{\Phi_x} + \sigma^2\mathbf{I}_n\right). \tag{2.56}$$

The posterior distribution of the parameters $\mathbf{w}$ can be derived using Bayes' theorem:

$$p\left(\mathbf{w}|\mathbf{y},\mathbf{X},\boldsymbol{\theta}\right)=\frac{p\left(\mathbf{y}|\mathbf{X},\boldsymbol{\theta},\mathbf{w}\right)p\left(\mathbf{w}|\alpha\right)}{p\left(\mathbf{y}|\mathbf{X},\boldsymbol{\theta}\right)}. \tag{2.57}$$

Substituting Equation (2.56) into Equation (2.57) and using Equation (A.17) again on the numerator, the posterior probability can be shown to have the following distribution:

$$p\left(\mathbf{w}|\mathcal{D},\boldsymbol{\theta}\right)=\mathcal{N}\left(\mathbf{w}|\bar{\mathbf{w}},\boldsymbol{\Sigma}_{\mathbf{w}}^{-1}\right), \tag{2.58}$$

$$\bar{\mathbf{w}}=\beta\boldsymbol{\Sigma}_{\mathbf{w}}^{-1}\boldsymbol{\Phi}_{\mathbf{x}}\mathbf{y}, \tag{2.59}$$

$$\boldsymbol{\Sigma}_{\mathbf{w}}=\beta\boldsymbol{\Phi}_{\mathbf{x}}{\boldsymbol{\Phi}_{\mathbf{x}}}^{T}+\mathbf{A}, \tag{2.60}$$

where $\beta=\sigma^{-2}$. The marginal likelihood can also be expressed as:

$$p\left(\mathbf{y}|\mathbf{X},\boldsymbol{\theta}\right)=\frac{p\left(\mathbf{y}|\mathbf{X},\boldsymbol{\theta},\mathbf{w}\right)p\left(\mathbf{w}|\alpha\right)}{p\left(\mathbf{w}|\mathcal{D},\boldsymbol{\theta}\right)}, \tag{2.61}$$

$$=\frac{\mathcal{N}\left(\mathbf{y}|{\boldsymbol{\Phi}_{\mathbf{x}}}^{T}\mathbf{w},\beta^{-1}\mathbf{I}_{n}\right)\mathcal{N}\left(\mathbf{w}|0,\mathbf{A}^{-1}\right)}{\mathcal{N}\left(\mathbf{w}|\bar{\mathbf{w}},\boldsymbol{\Sigma}_{\mathbf{w}}^{-1}\right)}, \tag{2.62}$$

substituting $\mathbf{w}=\bar{\mathbf{w}}$ we get,

$$p\left(\mathbf{y}|\mathbf{X},\boldsymbol{\theta}\right)=\mathcal{N}\left(\mathbf{y}|{\boldsymbol{\Phi}_{\mathbf{x}}}^{T}\bar{\mathbf{w}},\beta^{-1}\mathbf{I}\right)\mathcal{N}\left(\bar{\mathbf{w}}|0,\mathbf{A}^{-1}\right)\left(2\pi\right)^{\frac{m}{2}}\left|\boldsymbol{\Sigma}_{\mathbf{w}}\right|^{-\frac{1}{2}}, \tag{2.63}$$

thus, the log marginal likelihood can be expressed more efficiently in terms of the mean $\bar{\mathbf{w}}$ and the covariance $\boldsymbol{\Sigma}_{\mathbf{w}}$ of the posterior distribution:

$$\ln p\left(\mathbf{y}|\mathbf{X},\boldsymbol{\theta}\right)=-\frac{\beta}{2}\left\|\boldsymbol{\Phi}_{\mathbf{x}}\bar{\mathbf{w}}-\mathbf{y}\right\|^{2}+\frac{n}{2}\ln\beta-\frac{n}{2}\ln\left(2\pi\right)-\frac{\alpha}{2}\bar{\mathbf{w}}^{T}\bar{\mathbf{w}}+\frac{m}{2}\ln\alpha-\frac{1}{2}\ln\left|\boldsymbol{\Sigma}_{\mathbf{w}}\right|. \tag{2.64}$$

The hyper-parameters of the basis functions, the precision β and the weight precision α can now be optimised with respect to the log marginal likelihood defined in Equation (2.64). Once the parameters have been inferred, the predictive distribution of $\mathbf{f}_{*}$ can be found using the same approach used in the full GP case and is distributed as

follows:

$$p\left(\mathbf{f}_*|\mathbf{X}_*, \mathcal{D}, \boldsymbol{\theta}\right) = \mathcal{N}\left(\mathbb{E}\left[\mathbf{f}_*\right], \mathbb{V}\left[\mathbf{f}_*\right]\right), \tag{2.65}$$

$$\mathbb{E}\left[\mathbf{f}_*\right] = \mathbf{Q}_{*\mathbf{x}}\left(\mathbf{Q}_{\mathbf{xx}} + \sigma^2\mathbf{I}_n\right)^{-1}, \tag{2.66}$$

$$\mathbb{V}\left[\mathbf{f}_*\right] = \mathbf{Q}_{**} - \mathbf{Q}_{*\mathbf{x}}\left(\mathbf{Q}_{\mathbf{xx}} + \sigma^2\mathbf{I}_n\right)^{-1}\mathbf{Q}_{\mathbf{x}*}. \tag{2.67}$$

The same tools used in the sparse GP case can be used to re-formulate the predictive distribution more efficiently in terms of $\boldsymbol{\Phi}$ as:

$$\mathbb{E}\left[\mathbf{f}_*\right] = \boldsymbol{\Phi}_*\bar{\mathbf{w}}, \tag{2.68}$$

$$\mathbb{V}\left[\mathbf{f}_*\right] = {\boldsymbol{\Phi}_*}^T\boldsymbol{\Sigma}_{\mathbf{w}}^{-1}\boldsymbol{\Phi}_*, \tag{2.69}$$

and for the target output $\mathbf{y}_*$

$$p\left(\mathbf{y}_*|\mathbf{X}_*, \mathcal{D}, \boldsymbol{\theta}\right) = \mathcal{N}\left(\mathbf{y}_*|\mathbb{E}\left[\mathbf{y}_*\right], \mathbb{V}\left[\mathbf{y}_*\right]\right), \tag{2.70}$$

$$\mathbb{E}\left[\mathbf{y}_*\right] = \mathbb{E}\left[\mathbf{f}_*\right], \tag{2.71}$$

$$\mathbb{V}\left[\mathbf{y}_*\right] = \mathbb{V}\left[\mathbf{f}_*\right] + \sigma^2\mathbf{I}_{n_*}. \tag{2.72}$$

Basis function models are equivalent to the subset-of-regressors sparse GP, that is deterministic training and test conditionals, but with a different prior on the latent variables $\mathbf{f}_\mathbf{p}$

$$q\left(\mathbf{f}_\mathbf{x}|\mathbf{f}_\mathbf{p}, \mathbf{X}, \boldsymbol{\theta}\right) = \mathcal{N}\left(\mathbf{f}_\mathbf{x}|\boldsymbol{\Phi}_\mathbf{x}\mathbf{A}^{-1}\mathbf{f}_\mathbf{p}, 0\right), \tag{2.73}$$

$$q\left(\mathbf{f}_*|\mathbf{f}_\mathbf{p}, \mathbf{X}_*, \boldsymbol{\theta}\right) = \mathcal{N}\left(\mathbf{f}_*|\boldsymbol{\Phi}_*\mathbf{A}^{-1}\mathbf{f}_\mathbf{p}, 0\right), \tag{2.74}$$

$$q\left(\mathbf{f}_\mathbf{p}|\boldsymbol{\theta}\right) = \mathcal{N}\left(\mathbf{f}_\mathbf{p}|0, \mathbf{A}\right). \tag{2.75}$$

Similarly, the SoR sparse GP can be viewed as a basis function model with a prior on the weights $p(\mathbf{w}|\boldsymbol{\theta}) = \mathcal{N}\left(\mathbf{w}|0, \mathbf{K}_{\mathbf{pp}}^{-1}\right)$. Note that we are no longer restricted to Mercer kernels or a single basis function definition. The basis functions can therefore be modelled using variable length-scales and variable covariances, to capture different kinds of patterns that can arise in different regions of the input space.

2.5 Relations to other Methods

2.5.1 Relations to Artificial Neural Networks

If we define $\phi_j(\mathbf{x}_i) = \text{sigmoid}\left(\mathbf{h}_j^T\mathbf{x}_i + b_j\right)$, then $\mathbf{\Phi}_\mathbf{x}$ can be thought of as the activations of the m hidden units in a single-layer ANN, where $\mathbf{h}_j$ plays the role of the weights or connections of hidden unit j. In an ANN regressor, the output weights connecting the hidden layer to the output layer are equivalent to the parameter $\mathbf{w}$. To incorporate the biases of the output layer, the basis functions can be augmented with an additional function with a constant output of 1, such that the additional corresponding weight will act as a bias term. Thus, a single layer ANN with m hidden units is a BFM with m basis functions, defined as the sigmoid function, and an additional basis function with a constant output of 1. However, a main distinction between ANNs and BFMs, is that the weights in the output layer in ANNs are treated as parameters of the models to be optimised and are not integrated out. Moreover, the objective to be optimised is different. In ANNs the objective is to minimise the regularised sum of squares:

$$\mathcal{L}(\mathcal{D}|\boldsymbol{\theta}) = \frac{1}{2}\|\mathbf{y} - \mathbf{\Phi}_\mathbf{x}\mathbf{w}\|^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} + \frac{\lambda}{2}\sum_{j=1}^{m}\mathbf{h}_j^T\mathbf{h}_j, \tag{2.76}$$

where $\boldsymbol{\theta} = \{\mathbf{w}, \mathbf{h}_1, \ldots \mathbf{h}_m, b_1, \ldots, b_m\}$ is the set of free parameters to be optimised. We recognise the first two terms as the negative of the first two terms in the log marginal likelihood defined in Equation (3.1) divided by β; thus, $\lambda = \alpha/\beta$. However, λ is typically not treated as a hyper-parameter of the model but rather as a tuning-parameter optimised using cross validation methods. Another distinction is that ANNs also minimise the norm of the weights in the hidden layer as well as the output layer with no penalty on the bias terms. Moreover, the $\ln|\mathbf{\Sigma}_\mathbf{w}|$ term is missing from the regularised sum of squares. This term is crucial as it drives the model towards reducing the uncertainty on the parameter $\mathbf{w}$, therefore producing more confident models with more accurate variance prediction.

2.5.2 Relations to Stacked Auto-encoders

Artificial Neural Networks with many hidden layers, deep neural networks, have been around since the invention of ANNs. Training them however proved to be more difficult because the gradient of the objective function with respect to the hyper-parameters diminishes quickly in back-propagation. This is a consequence of the objective function being highly non-convex with many local minima. Moreover, there is no need to go beyond a single layer to give ANN any additional approximation power, as stated in the Universal Approximation Theorem (Cybenko, 1989). One advantage of deep architectures is the ability to post analyse the learned features and learn interesting patterns. The problem with training a deep ANN has been solved through the use of greedy layer-wise unsupervised training followed by supervised fine-tuning (Bengio et al., 2006; Hinton and Salakhutdinov, 2006; Ranzato et al., 2006). An additional *sparsity* cost was added to the objective, inspired from research in neuroscience (Olshausen and Field, 1996), to favour models that produce a sparse feature representation by penalising models with high average activation per neurone, or in other words an input to the network should activate the minimum number of neurones possible. The objective in Equation (2.76) is modified as follows:

$$\mathcal{L}\left(\mathcal{D}|\boldsymbol{\theta}\right) = \sum_{k=1}^{d}\left(\left\|\mathbf{X}[k,:] - \boldsymbol{\Phi}_{\mathbf{x}}\mathbf{w}_k\right\|^2 + \lambda\mathbf{w}_k^T\mathbf{w}_k\right) + \sum_{j=1}^{m}\left(\lambda\mathbf{h}_j^T\mathbf{h}_j + \beta\mathrm{KL}\left(\epsilon\|\mu_j\right)\right), \quad (2.77)$$

$$\mu_j = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{\Phi}_{\mathbf{x}}[i,j], \quad (2.78)$$

$$\mathrm{KL}\left(\epsilon\|\mu_j\right) = \epsilon\ln\frac{\epsilon}{\mu_j} + (1-\epsilon)\ln\frac{1-\epsilon}{1-\mu_j} \quad (2.79)$$

where β is the sparsity cost, ϵ is the desired average activation, both are inputs to the model and are tuned using cross validation, and $\mathrm{KL}\left(\epsilon\|\mu_j\right)$ is the Kullback-Leibler divergence. Note that the desired output is the input itself; hence, the name sparse auto-encoders. Once the parameters of a layer are learned, the same processes is repeated but using the activations of the previous layer as the new input. After

the parameters of all layers have been learned, they are used as an initialisation for a supervised multilayer ANN, also known as the fine-tuning phase. This approach has proven very effective in many areas of machine learning such as computer vision (Krizhevsky and Hinton, 2011), speech recognition (Hinton et al., 2012) and natural language processing (Socher, 2014). The reasons for its success have been studied in Erhan et al. (2010), in which it suggests that unsupervised pre-training has the characteristics of a regulariser, or acts as a better prior. In this review, we note, from Equation (A.28) in the appendix, that the sparsity constraint actually mimics the role of minimising the determinant of the covariance of $\mathbf{w}$, or the $\ln|\mathbf{\Sigma}_{\mathbf{w}}|$ term in Equation (2.64):

$$\begin{aligned}\ln|\mathbf{\Sigma}_{\mathbf{w}}| &\leq \sum_{j=1}^{m} \ln\left(\mathbf{\Sigma}_{\mathbf{w}}[j,j]\right), \\ &\leq \sum_{j=1}^{m} \ln\left(\beta \sum_{i=1}^{n} \mathbf{\Phi}_{\mathbf{x}}[j,i]^2 + \alpha\right). \end{aligned} \tag{2.80}$$

Another way to understand this part of the objective, is to minimise the sum of *squared* activations per hidden unit. Thus, the sparsity constraint can be viewed as filling the role of the missing term in the marginal likelihood objective, although with a different penalty function.

2.6 Summary

We have shown in this chapter that basis function models are sparse Gaussian processes of type SoR, but with a different prior on the latent variables. Basis function models are related to artificial neural networks and stacked auto-encoders, i.e. ANNs are BFMs with basis functions defined as the sigmoid activation function and the weights are not integrated out, but rather included to the set of parameters to be optimised. A main distinction between the way ANNs are traditionally optimised and how BFMs are trained, is that the former performs a regularised maximum likelihood

estimation, or maximum a posterior (MAP), while the latter maximises the marginal likelihood. Maximising the marginal likelihood is less prone to overfitting, due to the additional cost term that minimises the variance on the weight vector $\mathbf{w}$, which we have shown is related to the sparsity constraint in sparse auto-encoders. In the next chapter, we show how we can use this view of sparse GPs to extend their modelling capabilities.

Chapter 3

Extending Sparse Gaussian Processes

In the previous chapter, we focused mainly on reducing the computational cost of GPs. We now turn our attention to altering the modelling capabilities of GPs. Particularly we minimise the number of basis functions through automatic relevance determination, introduce input-dependent variance estimation (heteroscedastic noise), model strictly positive outputs, develop cost-sensitive learning and introduce prior mean function co-optimisation for better extrapolation performance. In Chapter 4 we will also introduce non-stationarity to the model and the ability to handle noisy and missing data in Chapter 5. Although some of these issues have been previously addressed in the context of GPs, each previous model has some, but not all these capabilities. We will show how all these features can be unified in a single generalised framework without sacrificing performance. Automatic relevance determination (ARD) was first addressed in the context of basis function models in Tipping (2001), but used the entire training set as fixed basis. The remaining sections, unifying ARD with the other features as a single sparse GP framework, are new contributions by this thesis.

3.1 Automatic Relevance Determination

In addition to achieving accurate predictions, we also wish to minimise the number of basis functions to produce a sparse model representation. Instead of adding an

additional prior over the number of basis functions, we can achieve this goal by incorporating a sparsity-inducing prior on $\mathbf{w}$. We use a diagonal precision matrix $\mathbf{A} = \text{diag}[\boldsymbol{\alpha}]$ for the prior, where $\boldsymbol{\alpha} = \{\alpha_i\}_{i=1}^m$, or a precision parameter per weight. The log marginal likelihood is simply extended as follows:

$$\begin{aligned}\ln p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = &-\frac{\beta}{2}\left\|\mathbf{y} - \boldsymbol{\Phi}_{\mathbf{x}}\bar{\mathbf{w}}\right\|^2 + \frac{n}{2}\ln\beta - \frac{n}{2}\ln(2\pi) \\ &- \frac{1}{2}\bar{\mathbf{w}}^T\mathbf{A}\bar{\mathbf{w}} + \frac{1}{2}\ln|\mathbf{A}| - \frac{1}{2}\ln|\boldsymbol{\Sigma}_{\mathbf{w}}|. \end{aligned} \tag{3.1}$$

By modelling each weight with its associated precision, we enable a natural shrinkage (or regularisation). Take, for example, a specific precision α_i; note that maximising $\frac{1}{2}\ln\alpha_i$ will minimise $-\frac{1}{2}\bar{w}_i^2\alpha_i$, unless $\bar{w}_i = 0$. The optimisation routine will therefore drive as many of the weights as possible to zero; thus, maintaining the least number of basis functions relevant to model the data. A similar approach was proposed by Tipping (2001), coined as the relevance vector machine (RVM), where the set of basis function locations $\mathbf{P}$ was set equal to the locations of the training samples $\mathbf{X}$ and held fixed. Only the precision parameter β and the $\boldsymbol{\alpha}$ values are optimised to determine the relevant set of vectors from the training set. This approach is still computationally expensive and the work discussed in Tipping (2001) proposed an iterative workaround to add and remove vectors incrementally.

To illustrate the effect of automatic relevance determination, consider the plots in Figure 3.1 comparing basis function models (BFMs), BFMs with ARD, the fully independent training conditional (FITC) and RVMs all using 100 RBF kernels on the same example from Figure 2.1. For BFMs and FITC, the optimal length-scale of the RBF is found by optimising the marginal likelihood using a gradient descent approach whereas in RVM it is found using a line search. Note that without ARD, basis function models learned a group of overlapping pseudo-points and their respective weights are almost distributed evenly amongst each group. On the other hand, BFMs with ARD resulted in a dominant basis in each group of overlapping pseudo-points with the

most share of the total weight. The worst performing method, in terms of sparseness, is FITC, where the basis vectors were spread almost evenly across the range of the data. To determine the number of relevant basis, we choose the minimum k such that

$$\frac{\sum_{j=1}^{k} |\bar{w}_j|}{\sum_{i=1}^{m} |\bar{w}_i|} \geq 0.99, \tag{3.2}$$

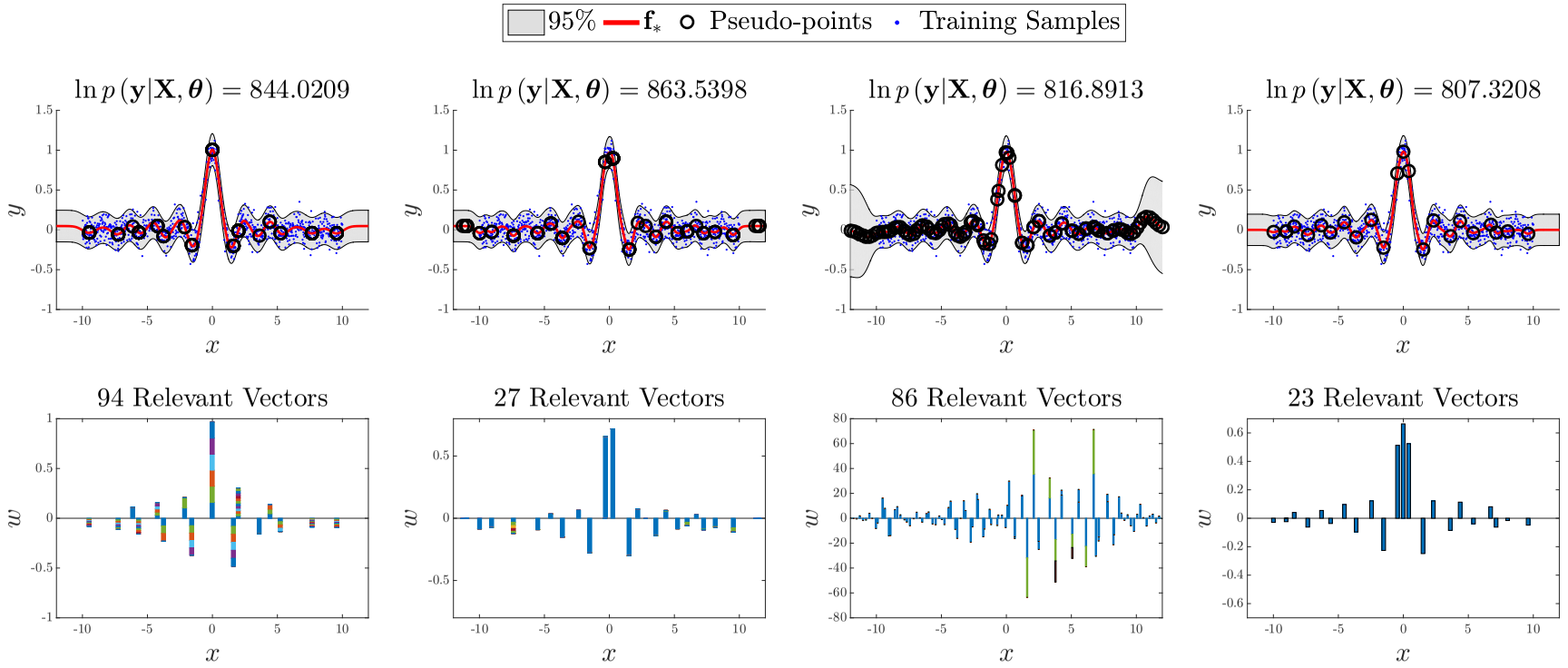
where $|\bar{\mathbf{w}}|$ is sorted in descendant order. Using this metric, the number of relevant vectors for BFMs without ARD, BFMs with ARD, FITC and RVM are 94, 27, 86 and 23 respectively. BFMs with ARD also achieved the maximum marginal likelihood.

3.2 Heteroscedastic Noise

The predictive variance in Equation (2.72) has two components; the first term, $\mathbb{V}[\mathbf{f}_*]$, is the model variance and the second term, σ^2, is the noise uncertainty. The model variance thus depends on the data density of the training sample at $\mathbf{x}_*$. Theoretically, this component of the model variance will go to zero as the size of the data set increases. This term hence models our underlying uncertainty about the mean function. The model becomes very confident about the posterior mean when presented with a large number of samples at $\mathbf{x}_*$, at which point the predictive variance reduces to the intrinsic noise variance. The latter, at this point, is assumed to be white Gaussian noise with a fixed precision β.

In this section, we enhance the model's predictive variance estimation by modelling the noise variance as a function of input to account for variable and input-dependent noise, i.e. heteroscedastic noise. We model the function as a linear combination of basis functions via $\beta(\mathbf{x}) = \exp\left(\boldsymbol{\phi}(\mathbf{x})^T \mathbf{v} + b\right)$, where we choose the exponential form to ensure positivity of $\beta(\mathbf{x})$. Note that if $\mathbf{v} = \mathbf{0}$ and $b = \ln\beta$, the model reduces to the original assumption of a fixed precision β. We thus redefine the likelihood as follows:

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}, \mathbf{w}) = \mathcal{N}\left(\mathbf{y}|\boldsymbol{\Phi}_{\mathbf{x}}\mathbf{w}, \mathbf{B}^{-1}\right), \tag{3.3}$$



(a) BFM without ARD (b) BFM with ARD (c) FITC (d) RVM

Figure 3.1: A comparison between (a) a basis function model with an isotropic precision matrix $\mathbf{A} = \alpha\mathbf{I}_m$, (b) a basis function model with a diagonal precision matrix $\mathbf{A} = \text{diag}[\boldsymbol{\alpha}]$, (c) The fully independent training conditional (FITC) sparse GP and (d) Relevance Vector Machines (RVM) all using the RBF kernel. 100 basis functions were used in (a) BFM, (b) BFM with ARD and (c) FITC; whereas in (d) RVM, the entire dataset of 1000 points were used as basis functions. The top row shows the learned distributions for each along with the locations of the basis functions, the log marginal likelihoods are shown above each plot in the top row. The bottom row shows the corresponding weights for each basis function as a stacked bar sorted in descending order in case of overlapping pseudo-points. The numbers of relevant vectors for each methods are shown above each plot in the bottom row.

where $\mathbf{B}$ is a $n \times n$ diagonal matrix where each element across the diagonal $\mathbf{B}[i,i] = \beta(\mathbf{x}_i)$ and the hyper-parameter set is now expanded to $\boldsymbol{\theta} = \{\boldsymbol{\varphi}, \boldsymbol{\alpha}, \mathbf{v}, b\}$. Following the same procedure, the posterior distribution is expressed as follows:

$$p(\mathbf{w}|\mathcal{D}, \boldsymbol{\theta}) = \mathcal{N}\left(\mathbf{w}|\bar{\mathbf{w}}, \boldsymbol{\Sigma}_{\mathbf{w}}^{-1}\right), \tag{3.4}$$

$$\bar{\mathbf{w}} = \boldsymbol{\Sigma}_{\mathbf{w}}^{-1}\boldsymbol{\Phi}_{\mathbf{x}}\mathbf{B}\mathbf{y}, \tag{3.5}$$

$$\boldsymbol{\Sigma}_{\mathbf{w}} = \boldsymbol{\Phi}_{\mathbf{x}}\mathbf{B}\boldsymbol{\Phi}_{\mathbf{x}}^T + \mathbf{A}, \tag{3.6}$$

and the updated log marginal likelihood becomes:

$$\begin{aligned}\ln p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = &-\frac{1}{2}\boldsymbol{\delta}^T\mathbf{B}\boldsymbol{\delta} + \frac{1}{2}\ln|\mathbf{B}| - \frac{n}{2}\ln(2\pi) \\ &- \frac{1}{2}\bar{\mathbf{w}}^T\mathbf{A}\bar{\mathbf{w}} + \frac{1}{2}\ln|\mathbf{A}| - \frac{1}{2}\ln|\boldsymbol{\Sigma}_{\mathbf{w}}|,\end{aligned} \tag{3.7}$$

where $\boldsymbol{\delta} = \mathbf{y} - \boldsymbol{\Phi}_{\mathbf{x}}^T\bar{\mathbf{w}}$. In addition, we also add a prior on the set of weights $\mathbf{v}$ to favour the simplest precision function, namely that $\mathbf{v}$ is normally distributed with a mean of 0 and a diagonal precision matrix $\mathbf{T} = \text{diag}[\boldsymbol{\tau}]$, or $p(\mathbf{v}|\boldsymbol{\tau}) = \mathcal{N}(\mathbf{v}|0, \mathbf{T}^{-1})$, where $\boldsymbol{\tau} = \{\tau_i\}_{i=1}^m$ and is added to the parameter set $\boldsymbol{\theta}$. The final objective function to be optimised is thus the log marginal likelihood plus the log of the prior on $\mathbf{v}$, which we refer to here as $\mathcal{L}(\mathcal{D}, \boldsymbol{\theta})$,

$$\mathcal{L}(\mathcal{D}, \boldsymbol{\theta}) = \ln\left(p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})\, p(\mathbf{v}|\boldsymbol{\tau})\right), \tag{3.8}$$

$$\begin{aligned}&= -\frac{1}{2}\boldsymbol{\delta}^T\mathbf{B}\boldsymbol{\delta} + \frac{1}{2}\ln|\mathbf{B}| - \frac{n}{2}\ln(2\pi) - \frac{1}{2}\bar{\mathbf{w}}^T\mathbf{A}\bar{\mathbf{w}} + \frac{1}{2}\ln|\mathbf{A}| - \frac{1}{2}\ln|\boldsymbol{\Sigma}_{\mathbf{w}}| \\ &\quad - \frac{1}{2}\mathbf{v}^T\mathbf{T}\mathbf{v} + \frac{1}{2}\ln|\mathbf{T}| - \frac{m}{2}\ln(2\pi).\end{aligned} \tag{3.9}$$

The parameter $\boldsymbol{\tau}$ hence acts as an automatic relevance determination cost for the noise process, allowing the objective to dynamically select different sets of relevant basis functions for both the posterior mean and variance estimation. For unseen test

cases, the predictive distribution is normally distributed as follows:

$$p\left(y_*|\mathbf{x}_*, \mathcal{D}, \boldsymbol{\theta}\right) = \mathcal{N}\left(y_*|\mathbb{E}\left[y_*\right], \mathbb{V}\left[y_*\right]\right), \tag{3.10}$$

$$\mathbb{E}\left[y_*\right] = \boldsymbol{\phi}\left(\mathbf{x}_*\right)^T \bar{\mathbf{w}}, \tag{3.11}$$

$$\mathbb{V}\left[y_*\right] = \boldsymbol{\phi}\left(\mathbf{x}_*\right)^T \boldsymbol{\Sigma}_{\mathbf{w}}^{-1} \boldsymbol{\phi}\left(\mathbf{x}_*\right) + \sigma^2\left(\mathbf{x}_*\right), \tag{3.12}$$

where $\sigma^2\left(\mathbf{x}_*\right) = \beta\left(\mathbf{x}_*\right)^{-1}$ is the input-dependent noise uncertainty. Note that even though we use the same set of m basis functions for the mean and variance functions, the weight parameters for each ($\mathbf{w}$ and $\mathbf{v}$) can be optimised such that the relevant basis functions for each are assigned non-zero values independently. For example, each can use different subsets of the m basis functions and/or share a subset of the basis functions with different weights. It is worth mentioning that in the parameter space, $\mathbf{w}$, the problem is convex; and thus, can be modelled using a single Gaussian distribution. In the hyper-parameter space, however, the problem can be highly non-convex with many local minima. This adds an extra source of uncertainty about the model due to training that is dependent on the initial condition and the optimisation procedure. This can be addressed using a committee of models, where each model is initialised differently, to fit a mixture of Gaussian distributions instead of a single one to better fit the true model distribution (Penny and Roberts, 1997; Roberts et al., 1996).

Figure 3.2 shows the effect of modelling the noise variance as a function of the input $\sigma^2\left(\mathbf{x}\right)$ compared to the full GP and FITC, modelled and optimised using GPML, all using an RBF kernel. The training set consists of 4000 samples drawn from the distribution $p\left(y|x\right) = \mathcal{N}\left(\mathrm{sinc}\left(x\right), \sigma^2\left(x\right)\right)$, where $\sigma\left(\mathbf{x}\right) = 0.01 + \frac{0.2(1+\sin(2x))}{1+\exp(-0.2x)}$. Note that the BFM with heteroscedastic noise prediction fits the data best with the largest log marginal likelihood. Moreover, it enables us to distinguish between areas where the uncertainty is high due to the lack of data versus areas where the overwhelming contribution of the uncertainty is due to the lack of features. For example, adding

more data in areas where the intrinsic noise is high will not improve the model, e.g. $7 < x < 8$, this area needs more precise, or extra, features. On the other hand, adding more features will not benefit the region where $-5 < x < -4$, as this region requires more data to reduce the uncertainty about the function. Note that in FITC the variance about the function is high even in regions of high data density. Note that in the top panel of Figure 3.2 that the predictive variance over estimates in some regions and under estimates in others. In other words, taking two standard deviations from the mean in the homooscedastic model does not reflect the 95% range, of the observed outputs given the input, as we would expect if the errors are Gaussians. This is due to the assumption of a fixed noise variance, the other model's predictive distributions on the other hand better fit the data.

3.3 Strictly Positive Outputs

Assume that our target values $y_i \in \mathbb{R}_+ \forall i \in \{1, \ldots, n\}$, an appropriate likelihood in this case is the log-normal probability density function:

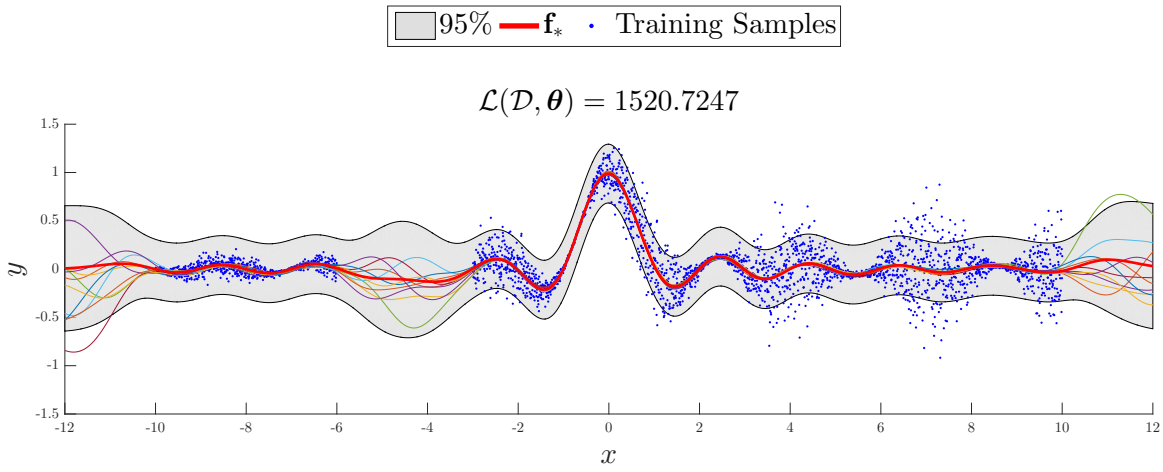
$$\begin{aligned} p\left(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}, \mathbf{w}\right) &= \prod_{i=1}^{n} \ln \mathcal{N}\left(y_i|\boldsymbol{\phi}\left(\mathbf{x}_i\right)^T \mathbf{w}, \sigma^2\left(\mathbf{x}_i\right)\right), \\ &= \prod_{i=1}^{n} \frac{1}{y_i} \mathcal{N}\left(\ln y_i|\boldsymbol{\phi}\left(\mathbf{x}_i\right)^T \mathbf{w}, \sigma^2\left(\mathbf{x}_i\right)\right), \\ &= \left(\prod_{i=1}^{n} \frac{1}{y_i}\right) \mathcal{N}\left(\ln \mathbf{y}|\boldsymbol{\Phi}_{\mathbf{x}}^T \mathbf{w}, \mathbf{B}^{-1}\right)., \end{aligned} \tag{3.13}$$

since the first term on the RHS of Equation (3.13) is independent of the parameters $\boldsymbol{\theta}$ or $\mathbf{w}$, we can repeat the same procedure with $\ln \mathbf{y}$ as the target outputs and arrive at the following predictive distribution for the test input $\mathbf{x}_*$:

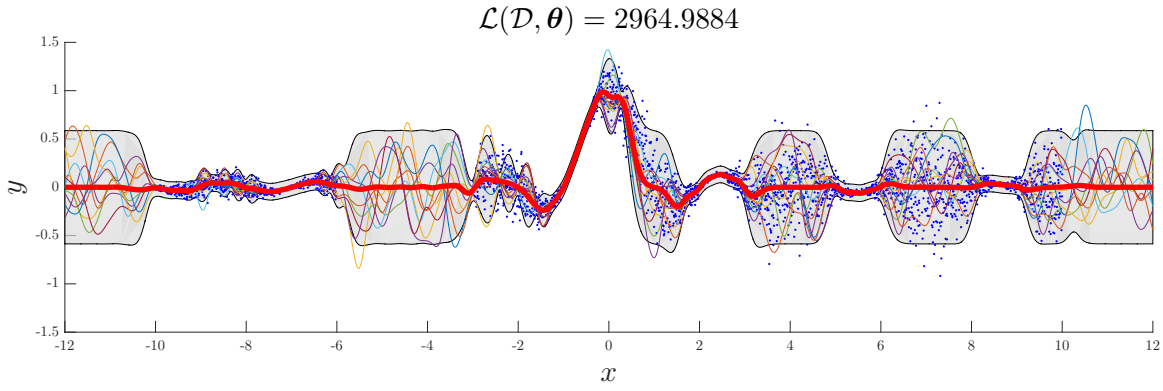
$$p\left(\ln y_*|\mathbf{x}_*, \mathcal{D}, \boldsymbol{\theta}\right) = \mathcal{N}\left(\ln y_*|\mathbb{E}\left[\ln y_*\right], \mathbb{V}\left[\ln y_*\right]\right), \tag{3.14}$$

$$\mathbb{E}\left[\ln y_*\right] = \boldsymbol{\phi}\left(\mathbf{x}_*\right)^T \bar{\mathbf{w}}, \tag{3.15}$$

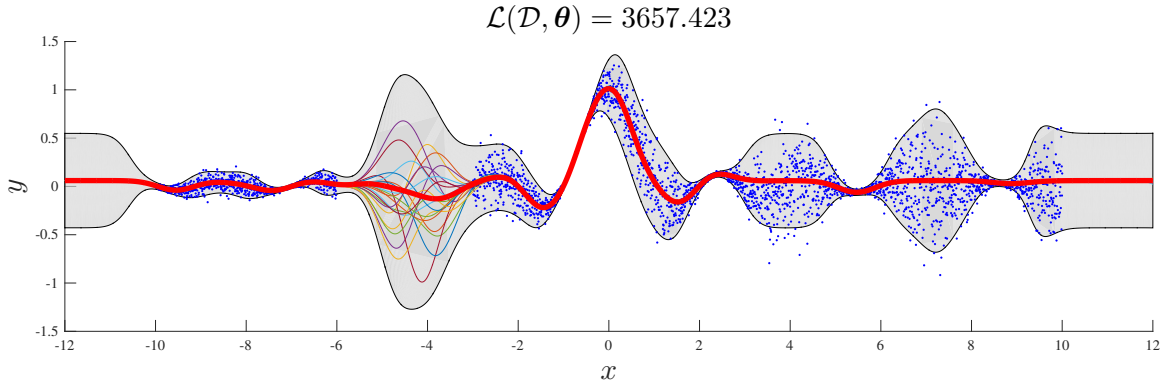
$$\mathbb{V}\left[\ln y_*\right] = \boldsymbol{\phi}\left(\mathbf{x}_*\right)^T \boldsymbol{\Sigma}_{\mathbf{w}}^{-1} \boldsymbol{\phi}\left(\mathbf{x}_*\right) + \sigma^2\left(\mathbf{x}_*\right), \tag{3.16}$$



(a) Full GP



(b) FITC



(c) BFM

Figure 3.2: A comparison between (a) a full GP with a constant noise variance, (b) FITC using 100 pseudo-points and (c) a BFM using 100 basis functions with variable uncertainty prediction, all using the RBF kernel. The plots show the mean of the function (red), the 95% confidence range (grey), i.e. plus or minus two standard deviations from the mean, and samples from the distribution $p(\mathbf{f}_*|\mathbf{X}_*, \mathcal{D}, \boldsymbol{\theta})$. The log marginal likelihood is shown above each plot.

However, the expected value of the output $\mathbb{E}[y_*] \neq \exp(\mathbb{E}[\ln y_*])$, but rather it is distributed log-normally as follows:

$$p(y_*|\mathbf{x}_*, \mathcal{D}, \boldsymbol{\theta}) = \ln \mathcal{N}(y_*|\mathbb{E}[\ln y_*], \mathbb{V}[\ln y_*]), \tag{3.17}$$

$$\mathbb{E}[y_*] = \exp\left(\mathbb{E}[\ln y_*] + \frac{1}{2}\mathbb{V}[\ln y_*]\right), \tag{3.18}$$

$$\mathbb{V}[y_*] = (\exp(\mathbb{V}[\ln y_*]) - 1)\,\mathbb{E}[y_*]^2. \tag{3.19}$$

Note that $\mathbb{E}[y_*] = \exp(\mathbb{E}[\ln y_*])$ if and only if the variance in the log space is zero, which is most likely not the case. This can be viewed as a special case of the Warped Gaussian Processes (WGP) (Snelson et al., 2004), where the link function $f(y) = \ln(y)$, and it can be generalised to any monotonic function of y. As is evident from Equations (3.18) and (3.19), input-dependent noise prediction is now necessary to model in the context of WGPs as the expected value of the output is dependent on it. This also demonstrates an additional disadvantage of modelling the uncertainty prediction as a separate function mapping the input to the log of the error of the mean function. Because, as demonstrated, the expected value of the error is not the exponential of the expected value of the function in the log space, it is the median of the distribution in Equation (3.17). Figure 3.3 is an example of fitting a model for strictly positive outputs with non-Gaussian noise showing the mean, median and the 95% area. Note how the median deviates from the mean as the variance increases. The root-mean-square error (RMSE) score on the test set is 2.73 when the median was used as a predictor, whereas using the mean as a predictor results in an RMSE score of 2.62.

3.4 Cost-sensitive Learning

Thus far, our objective has been to find the simplest function that maximises the probability of observing the target output given the input. However, this is intrinsically biased by uneven distributions of training data, sacrificing accuracy in less

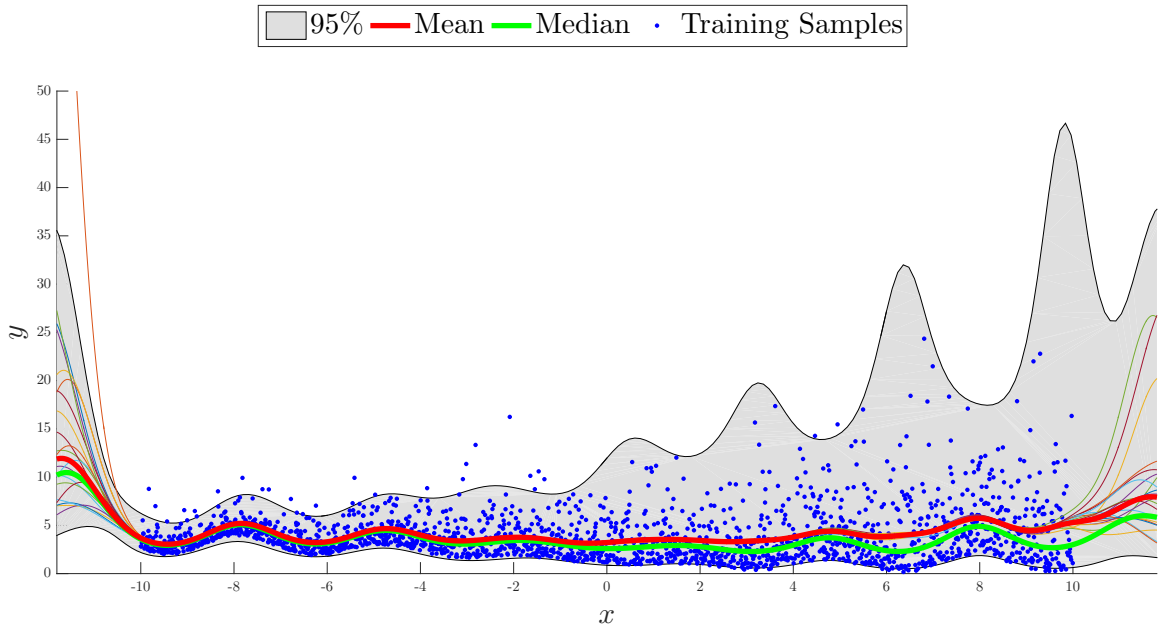


Figure 3.3: The mean (red), median (green) and 95% area (grey), i.e. plus or minus two standard deviations from the mean, of the predictive distribution of a basis function model with 200 radial basis functions for strictly positive outputs with non-Gaussian noise.

represented regions of the space in order to achieve a simpler model. Ideally we would like to train a model with a balanced data distribution to avoid such bias. This however, is a luxury that we often do not have. A common technique is to either over-sample or under-sample the data to achieve balance (Weiss et al., 2007). In under-sampling, samples are removed from highly represented regions to achieve balance, over-sampling on the other hand duplicates under represented samples. Both approaches come with a cost; in the former good data are wasted and in the latter more computation is introduced due to the data size increase. The same effect can be achieved by modifying the likelihood as follows:

$$p\left(\mathbf{y}|\mathbf{f}_{\mathbf{x}}, \boldsymbol{\omega}, \boldsymbol{\theta}\right) = \prod_{i=1}^{n} \mathcal{N}\left(y_i|f\left(\mathbf{x}_i\right), \beta\left(\mathbf{x}_i\right)^{-1}\right)^{\omega_i}, \tag{3.20}$$

where $\boldsymbol{\omega} = \{\omega_i\}_{i=1}^{n}$ is a per sample weight used to mimic over-sampling or under-sampling, now included as part of the dataset $\mathcal{D} = \{\mathbf{y}, \mathbf{X}, \boldsymbol{\omega}\}$. For example, instead of duplicating the sample i, ω_i can be set to 2 to achieve the same effect. From

Equation (A.20), Equation (3.20) can be rewritten as:

$$\begin{aligned} p\left(\mathbf{y}|\mathbf{f}_{\mathbf{x}}, \boldsymbol{\omega}, \boldsymbol{\theta}\right) &= \prod_{i=1}^{n} \mathcal{N}\left(y_i|f\left(\mathbf{x}_i\right), \omega_i^{-1}\beta\left(\mathbf{x}_i\right)^{-1}\right)\left(2\pi\beta\left(\mathbf{x}_i\right)^{-1}\right)^{\frac{1-\omega_i}{2}}\omega_i^{-\frac{1}{2}}, \\ &= \mathcal{N}\left(\mathbf{y}|\mathbf{f}_{\mathbf{x}}, \mathbf{B}^{-1}\boldsymbol{\Omega}^{-1}\right)|\mathcal{C}_{\omega}|, \end{aligned}$$

where $\boldsymbol{\Omega}$ is a diagonal matrix with elements $\boldsymbol{\Omega}[i,i] = \omega_i$ and $\mathcal{C}_{\omega}$ is a diagonal matrix with $\left(2\pi\beta\left(\mathbf{x}_i\right)^{-1}\right)^{(1-\omega_i)/2}\omega_i^{-1/2}$ along its diagonal. This in effect multiplies the likelihood with the constant $|\mathcal{C}_{\omega}|$ and changes the precision matrix to $\boldsymbol{\Omega}\mathbf{B}$. We can now repeat the procedure with the updated likelihood to get the following updated posterior mean and posterior covariance of $\mathbf{w}$:

$$p\left(\mathbf{w}|\mathcal{D}, \boldsymbol{\theta}\right) = \mathcal{N}\left(\mathbf{w}|\bar{\mathbf{w}}, \boldsymbol{\Sigma}_{\mathbf{w}}^{-1}\right), \tag{3.21}$$

$$\bar{\mathbf{w}} = \boldsymbol{\Sigma}_{\mathbf{w}}^{-1}\boldsymbol{\Phi}_{\mathbf{x}}\boldsymbol{\Omega}\mathbf{B}\mathbf{y}, \tag{3.22}$$

$$\boldsymbol{\Sigma}_{\mathbf{w}} = \boldsymbol{\Phi}_{\mathbf{x}}\boldsymbol{\Omega}\mathbf{B}\boldsymbol{\Phi}_{\mathbf{x}}^{T} + \mathbf{A}, \tag{3.23}$$

and the updated marginal likelihood:

$$p\left(\mathbf{y}|\mathbf{X}, \boldsymbol{\omega}, \boldsymbol{\theta}\right) = |\mathcal{C}_{\omega}|\mathcal{N}\left(\mathbf{y}|\boldsymbol{\Phi}_{\mathbf{x}}^{T}\bar{\mathbf{w}}, \mathbf{B}^{-1}\boldsymbol{\Omega}^{-1}\right)\mathcal{N}\left(\bar{\mathbf{w}}|0, \mathbf{A}^{-1}\right)(2\pi)^{\frac{m}{2}}|\boldsymbol{\Sigma}_{\mathbf{w}}|^{-\frac{1}{2}}, \tag{3.24}$$

$$= \prod_{i=1}^{n}\mathcal{N}\left(y_i|\boldsymbol{\phi}\left(\mathbf{x}_i\right)^{T}\bar{\mathbf{w}}, \beta\left(\mathbf{x}_i\right)^{-1}\right)^{\omega_i}\mathcal{N}\left(\bar{\mathbf{w}}|0, \mathbf{A}^{-1}\right)(2\pi)^{\frac{m}{2}}|\boldsymbol{\Sigma}_{\mathbf{w}}|^{-\frac{1}{2}}. \tag{3.25}$$

The final objective, the log marginal likelihood plus the log of the prior on $\mathbf{v}$, can be expressed as follows:

$$\begin{aligned} \mathcal{L}(\mathcal{D}, \boldsymbol{\theta}) = &-\frac{1}{2}\boldsymbol{\delta}^{T}\boldsymbol{\Omega}\mathbf{B}\boldsymbol{\delta} + \frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Omega}\odot\ln\mathbf{B}\right) - \frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Omega}\ln\left(2\pi\right)\right) \\ &- \frac{1}{2}\bar{\mathbf{w}}^{T}\mathbf{A}\bar{\mathbf{w}} + \frac{1}{2}\ln|\mathbf{A}| - \frac{1}{2}\ln|\boldsymbol{\Sigma}_{\mathbf{w}}| \\ &- \frac{1}{2}\mathbf{v}^{T}\mathbf{T}\mathbf{v} + \frac{1}{2}\ln|\mathbf{T}| - \frac{m}{2}\ln\left(2\pi\right). \end{aligned} \tag{3.26}$$

3.5 Prior Mean Function

In the absence of observations, all Bayesian models, GPs included, rely on their priors to provide function estimation. For the case of BFMs, the effective function prior in

Equation (2.54) is a constant zero. Instead, we may consider a mean *function* that is itself a simple linear regression from the independent to the dependent variable so that in the absence of data, e.g. in extrapolative regions, the GP will fall back to the linear regression prediction (Roberts et al., 2013) instead of a simple constant. The effective prior is thus distributed as follows:

$$p(\mathbf{f}_\mathbf{x}|\mathbf{X},\boldsymbol{\theta}) = \mathcal{N}\left(\mathbf{f}_\mathbf{x}\middle|\begin{bmatrix}\mathbf{X}\\ \mathbf{1}_n^T\end{bmatrix}^T\mathbf{a}, \boldsymbol{\Phi}_\mathbf{x}^T\mathbf{A}^{-1}\boldsymbol{\Phi}_\mathbf{x}\right), \tag{3.27}$$

where $\mathbf{a} = \{a_i\}_{i=1}^{d+1}$ are the coefficients of a linear regression model. By introducing the following design matrices and vectors:

$$\hat{\boldsymbol{\Phi}}_\mathbf{x} = \begin{bmatrix}\boldsymbol{\Phi}_\mathbf{x}\\ \mathbf{X}\\ \mathbf{1}_n^T\end{bmatrix}, \hat{\boldsymbol{\alpha}} = \{\alpha_i\}_{i=1}^{m+d+1}, \hat{\mathbf{A}} = \operatorname{diag}[\hat{\boldsymbol{\alpha}}], \hat{\mathbf{a}} = \{a_i\}_{i=1}^{m+d+1}, \hat{\boldsymbol{\theta}} = \{\boldsymbol{\varphi}, \hat{\boldsymbol{\alpha}}, \hat{\mathbf{a}}, \boldsymbol{\tau}, \mathbf{v}, b\}, \tag{3.28}$$

if we set $\{a_i\}_{i=1}^{m} = 0$, then Equation (3.27) can be expressed as:

$$p(\mathbf{f}_\mathbf{x}|\mathbf{X},\hat{\boldsymbol{\theta}}) = \mathcal{N}\left(\mathbf{f}_\mathbf{x}|\hat{\boldsymbol{\Phi}}_\mathbf{x}^T\hat{\mathbf{a}}, \hat{\boldsymbol{\Phi}}_\mathbf{x}^T\hat{\mathbf{A}}^{-1}\hat{\boldsymbol{\Phi}}_\mathbf{x}\right). \tag{3.29}$$

This has the effect of introducing a new parameter vector $\hat{\mathbf{w}} \in \mathbb{R}^{m+d+1}$ that has the following prior distribution:

$$p(\hat{\mathbf{w}}|\hat{\boldsymbol{\alpha}}) = \mathcal{N}\left(\hat{\mathbf{w}}|\hat{\mathbf{a}}, \hat{\mathbf{A}}^{-1}\right). \tag{3.30}$$

This can be shown by integrating out $\hat{\mathbf{w}}$ from the likelihood using the new prior in Equation (3.30):

$$\begin{aligned} p(\mathbf{f}_\mathbf{x}|\mathbf{X},\hat{\boldsymbol{\theta}}) &= \int p(\mathbf{f}_\mathbf{x}|\mathbf{X},\hat{\boldsymbol{\theta}},\hat{\mathbf{w}})p(\hat{\mathbf{w}}|\hat{\boldsymbol{\alpha}})\mathrm{d}\hat{\mathbf{w}},\\ &= \int \mathcal{N}(\mathbf{f}_\mathbf{x}|\hat{\boldsymbol{\Phi}}_\mathbf{x}^T\hat{\mathbf{w}}, 0)\mathcal{N}\left(\hat{\mathbf{w}}|\hat{\mathbf{a}}, \hat{\mathbf{A}}^{-1}\right)\mathrm{d}\hat{\mathbf{w}}, \end{aligned} \tag{3.31}$$

using Equation (A.17) of the appendix, we can show that Equations (3.29) and (3.31) are equivalent. The posterior distribution of $\hat{\mathbf{w}}$ has to be updated as follows:

$$p\left(\hat{\mathbf{w}}|\mathcal{D},\hat{\boldsymbol{\theta}}\right) = \mathcal{N}\left(\hat{\mathbf{w}}|\hat{\bar{\mathbf{w}}}, \hat{\boldsymbol{\Sigma}}_\mathbf{w}^{-1}\right), \tag{3.32}$$

$$\hat{\bar{\mathbf{w}}} = \hat{\boldsymbol{\Sigma}}_\mathbf{w}^{-1}\left(\hat{\mathbf{A}}\hat{\mathbf{a}} + \hat{\boldsymbol{\Phi}}_\mathbf{x}\boldsymbol{\Omega}\mathbf{B}\mathbf{y}\right), \tag{3.33}$$

$$\hat{\boldsymbol{\Sigma}}_\mathbf{w} = \hat{\boldsymbol{\Phi}}_\mathbf{x}\boldsymbol{\Omega}\mathbf{B}\hat{\boldsymbol{\Phi}}_\mathbf{x}^T + \hat{\mathbf{A}}. \tag{3.34}$$

Using the new prior and posterior, the same procedure can be repeated to arrive at the following general marginal likelihood:

$$p\left(\mathbf{y}|\mathbf{X},\boldsymbol{\omega},\hat{\boldsymbol{\theta}}\right) = |\mathcal{C}_\omega|\mathcal{N}\left(\mathbf{y}|\hat{\boldsymbol{\Phi}}_{\mathbf{x}}^T\hat{\mathbf{w}},\mathbf{B}^{-1}\boldsymbol{\Omega}^{-1}\right)\mathcal{N}\left(\hat{\mathbf{w}}|\hat{\mathbf{a}},\hat{\mathbf{A}}^{-1}\right)(2\pi)^{\frac{m}{2}}|\hat{\boldsymbol{\Sigma}}_{\mathbf{w}}|^{-\frac{1}{2}}, \tag{3.35}$$

and the final objective as:

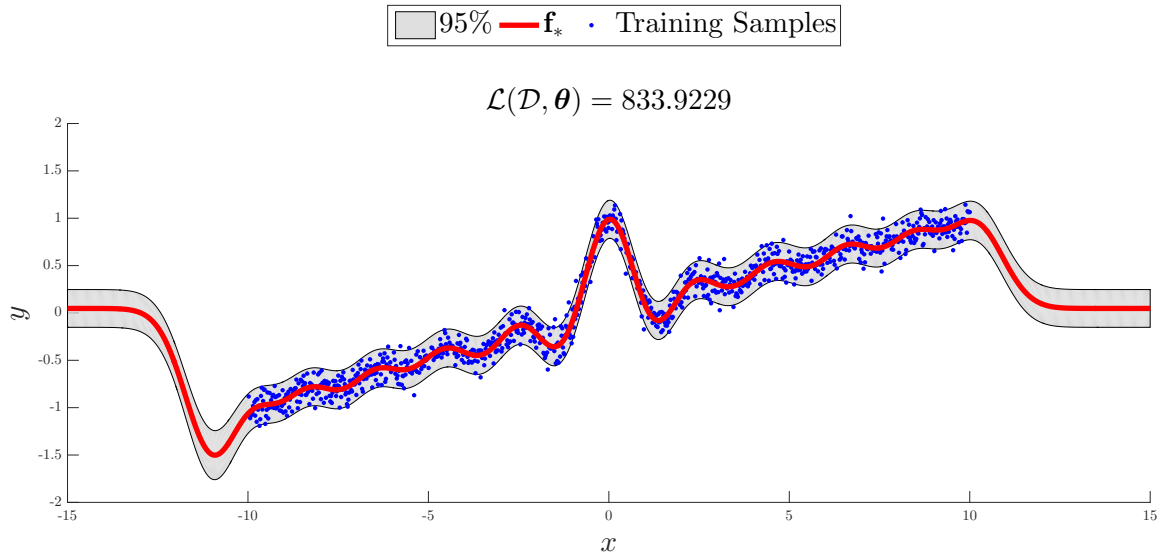
$$\begin{aligned}\mathcal{L}(\mathcal{D},\hat{\boldsymbol{\theta}}) = &-\frac{1}{2}\hat{\boldsymbol{\delta}}^T\boldsymbol{\Omega}\mathbf{B}\hat{\boldsymbol{\delta}} + \frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Omega}\odot\ln\mathbf{B}\right) - \frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Omega}\ln\left(2\pi\right)\right)\\ &-\frac{1}{2}\left(\hat{\mathbf{w}}-\hat{\mathbf{a}}\right)^T\hat{\mathbf{A}}\left(\hat{\mathbf{w}}-\hat{\mathbf{a}}\right) + \frac{1}{2}\ln|\hat{\mathbf{A}}| - \frac{1}{2}\ln|\hat{\boldsymbol{\Sigma}}_{\mathbf{w}}|\\ &-\frac{1}{2}\mathbf{v}^T\mathbf{T}\mathbf{v} + \frac{1}{2}\ln|\mathbf{T}| - \frac{m}{2}\ln\left(2\pi\right),\end{aligned} \tag{3.36}$$

where $\hat{\boldsymbol{\delta}} = \mathbf{y} - \hat{\boldsymbol{\Phi}}_{\mathbf{x}}^T\hat{\mathbf{w}}$. An appropriate prior mean $\mathbf{a}$, of the linear projection part of $\hat{\mathbf{a}}$, can be set to the parameters of a linear regression fit, or they can be optimised as part of the set $\hat{\boldsymbol{\theta}}$. In practice however, we would like to use the minimum amount of information from the linear function as well, so we set the prior mean $\hat{\mathbf{a}} = 0$ after we decorrelate the input $\mathbf{X}$ and output $\mathbf{y}$. The final objective function to be maximised in Equation (3.36) can then be computed more efficiently as follows:

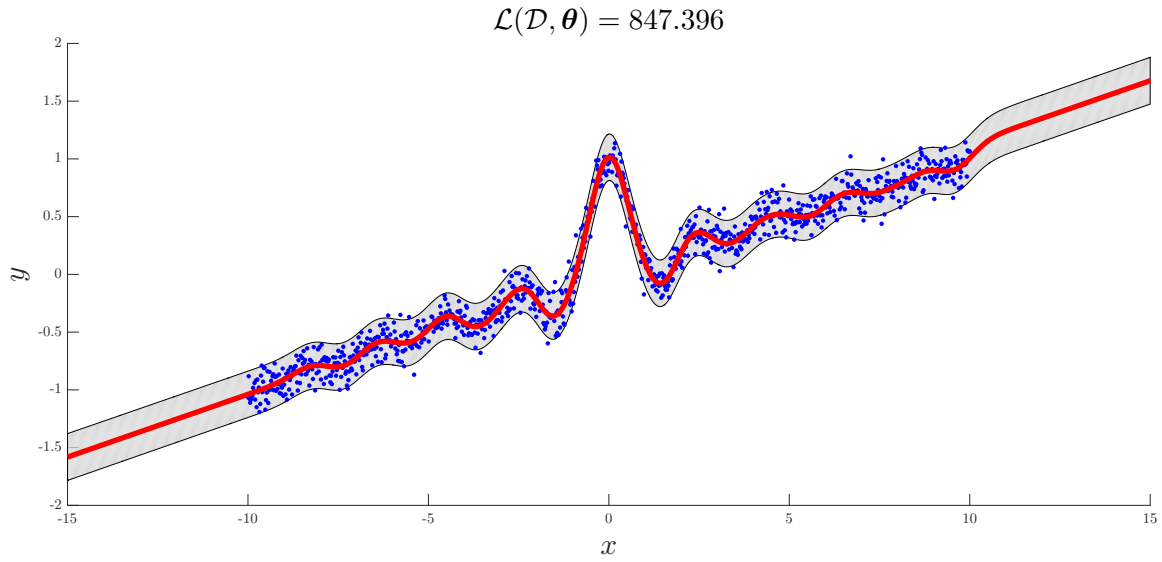
$$\begin{aligned}\mathcal{L}(\mathcal{D},\hat{\boldsymbol{\theta}}) = &-\frac{1}{2}\left(\boldsymbol{\beta}\odot\boldsymbol{\omega}\right)^T\hat{\boldsymbol{\delta}}^2 + \frac{1}{2}\boldsymbol{\omega}^T\ln\boldsymbol{\beta} - \frac{1}{2}\ln\left(2\pi\right)\mathbf{1}_n^T\boldsymbol{\omega}\\ &-\frac{1}{2}\hat{\boldsymbol{\alpha}}^T\left(\hat{\mathbf{w}}-\hat{\mathbf{a}}\right)^2 + \frac{1}{2}\mathbf{1}_n^T\ln\hat{\boldsymbol{\alpha}} - \frac{1}{2}\ln|\hat{\boldsymbol{\Sigma}}_{\mathbf{w}}|\\ &-\frac{1}{2}\boldsymbol{\tau}^T\mathbf{v}^2 + \frac{1}{2}\mathbf{1}_m^T\ln\boldsymbol{\tau} - \frac{m}{2}\ln\left(2\pi\right),\end{aligned} \tag{3.37}$$

$$\boldsymbol{\beta} = \left\{\beta\left(\mathbf{x}_i\right)\right\}_{i=1}^n. \tag{3.38}$$

We illustrate the advantage of this approach in Figure 3.4, where we compare two basis function models with the same number of basis functions of the same form but with two different mean functions. Both performed equally well within the bounds of the dataset; however, the approach with a prior mean function had a better extrapolation performance as it continued the general linear trend of the data instead of falling back to a constant value.



(a) Zero mean function



(b) Linear mean function

Figure 3.4: A comparison between two basis function models, the first (a) using a constant prior mean of 0 and the second (b) using a prior linear mean function showing the mean of the function (red), the 95% confidence range (grey) and the log marginal likelihood. Both models were trained using 100 RBFs.

3.6 Summary

We showed in this chapter how the most relevant basis functions can be discovered by modelling each weight with a different precision scale. We allow for heteroscedastic

noise estimation by modelling the variance of the likelihood distribution as a function of the input. We improve the modelling performance in the log-space of the, strictly positive, output using heteroscedasticity, and show how the per-sample weights can be incorporated into the objective as added variances to the likelihood. We detail how a prior linear mean-function can be integrated into the formulation and how it changes the effective prior on the function. In the next chapter we will describe how we optimise the objective function using gradient-based search.

Chapter 4

Optimisation

In this chapter, we discuss the basis functions we use and detail the optimisation procedure, generically and, for two specific basis functions; the broad class of radial basis functions and sigmoid-like activation functions. The first section of the chapter offers an overview of the quasi-Newton optimisation method and details the gradient calculations.

4.1 Gradient-based Optimisation

Gradient-based optimisation is a local search method, in that it makes moves from a location in the parameter space to a local optimum. Consider a function $f(\mathbf{x})$ to be optimised and consider starting from a random location $\mathbf{x}_*$. We consider a move $\Delta\mathbf{x}$ and expend using the multivariate Taylor series expansion of $f(\mathbf{x}_* + \Delta\mathbf{x})$ as follows:

$$f(\mathbf{x}_* + \Delta\mathbf{x}) = f(\mathbf{x}_*) + \Delta\mathbf{x}^T\mathbf{g} + \frac{1}{2}\Delta\mathbf{x}^T\mathbf{H}\Delta\mathbf{x} + \ldots, \tag{4.1}$$

where $\mathbf{g}$ is the gradient vector, i.e. $\mathbf{g}[i] = \partial f(\mathbf{x}_* + \Delta\mathbf{x})/\partial\Delta\mathbf{x}_i$, and $\mathbf{H}$ is the hessian matrix, i.e. $\mathbf{H}[i,j] = \partial f(\mathbf{x}_* + \Delta\mathbf{x})/\partial\Delta\mathbf{x}_i\partial\Delta\mathbf{x}_j$. The first three terms in Equation (4.1) describe the best quadratic approximation of $f(\mathbf{x}_* + \Delta\mathbf{x})$ about $\mathbf{x}_*$, namely

$$f(\mathbf{x}_* + \Delta\mathbf{x}) \approx f(\mathbf{x}_*) + \Delta\mathbf{x}^T\mathbf{g} + \frac{1}{2}\Delta\mathbf{x}^T\mathbf{H}\Delta\mathbf{x}. \tag{4.2}$$

To find $\Delta\mathbf{x}$ such that $f(\mathbf{x}_* + \Delta\mathbf{x})$ is maximised/minimised, we take the derivative of Equation (4.2) with respect to $\Delta\mathbf{x}$ and set it equal to zero, solving for $\Delta\mathbf{x}$ yields

$$\Delta\mathbf{x} = -\mathbf{H}^{-1}\mathbf{g}. \tag{4.3}$$

Note that if the function is truly quadratic, then the solution in Equation (4.3) is exact. However, typically the curvature of the function around $\mathbf{x}_*$ is not quadratic, thus an iterative procedure is used where we start from an initial location $\mathbf{x}_0$ then update using

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta\mathbf{H}_k^{-1}\mathbf{g}_k, \tag{4.4}$$

where $\mathbf{x}_k$, $\mathbf{g}_k$, $\mathbf{H}_k$ are the location, the gradient and the hessian at iteration k, respectively, and η is a positive scalar referred to as the step size, that is typically found using a line-search method. This is known as Newton's method and has a time complexity of $O(n^3)$ at each iteration, where n is the dimensionality of $\mathbf{x}$. This complexity is not an issue in univariate problems, but in the multivariate case an approximation method is often used to ease computation. One approximation extreme is to set $\mathbf{H} = \mathbf{I}_n$, which also can be arrived at by taking the first two terms in the Taylor expansion, i.e. the best linear approximation around $\mathbf{x}_*$. This is known as the gradient descent method. Newton's method has a quadratic convergence rate but is more costly to compute at each step. To resolve this, instead of requiring the inverse of the hessian at each step, we approximate it using a quasi-update function at each step from the change in position and gradient. Such class of methods are referred to as Quasi-Newton methods. The most widely used member of this family of algorithms is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm (Broyden, 1970; Fletcher, 1970; Goldfarb, 1970; Shanno, 1970). The BFGS method updates the *inverse* of the hessian at each iteration. The updated hessian inverse is required to be as close as possible to the previous matrix, to be symmetric and satisfy the secant condition $\mathbf{H}_{k+1}^{-1}(\mathbf{g}_{k+1} - \mathbf{g}_k) \approx \mathbf{x}_{k+1} - \mathbf{x}_k$, note that if the function is quadratic

then this is no longer an approximation. This formulation yields the following unique update equation for BFGS

$$\mathbf{H}_{k+1}^{-1} = \left(\mathbf{I}_n - \frac{\mathbf{y}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}\right) \mathbf{H}_k^{-1} \left(\mathbf{I}_n - \frac{\mathbf{s}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}\right) + \frac{\mathbf{s}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}, \tag{4.5}$$

where $\mathbf{s}_k = \mathbf{x}_k - \mathbf{x}_{k-1}$ and $\mathbf{y}_k = \mathbf{g}_k - \mathbf{g}_{k-1}$. The update equation in Equation (4.5) only requires the gradients at each iteration, has a $O(n^2)$ computational cost at each step and a super-linear convergence rate. The updated hessian inverse is guaranteed to be positive semi-definite as long as the hessian in the previous iteration is also positive semi-definite. If one starts with an initial $\mathbf{H}_0 = \mathbf{I}_n$, making the first step equivalent to a gradient descent direction, then the subsequent updated hessians will be positive semi-definite and improve over time. We use L-BFGS (Nocedal, 1980), which is a variant of BFGS that uses the last ℓ gradient and position differences to improve the storage performance of BFGS. In the following sections, we provide the gradient calculations in general for any basis function as well as for the special cases of the radial basis functions and sigmoid-like activation functions.

4.2 The General Case

The gradient of the objective function in Equation (3.36) with respect to each hyper-parameter $\hat{\theta}_i$, using the properties of matrix derivatives (Petersen and Pedersen, 2012),

can be computed by calculating the following in order:

$$\frac{\partial \hat{\boldsymbol{\Sigma}}_{\mathbf{w}}}{\partial \hat{\theta}_i} = \hat{\boldsymbol{\Phi}}_{\mathbf{x}} \boldsymbol{\Omega} \left(\frac{\partial \mathbf{B}}{\partial \hat{\theta}_i} \hat{\boldsymbol{\Phi}}_{\mathbf{x}}^T + 2\mathbf{B} \frac{\partial \hat{\boldsymbol{\Phi}}_{\mathbf{x}}^T}{\partial \hat{\theta}_i} \right) + \frac{\partial \hat{\mathbf{A}}}{\partial \hat{\theta}_i}, \tag{4.6}$$

$$\frac{\partial \hat{\bar{\mathbf{w}}}}{\partial \hat{\theta}_i} = \hat{\boldsymbol{\Sigma}}_{\mathbf{w}}^{-1} \left(\frac{\partial \hat{\mathbf{A}}}{\partial \hat{\theta}_i} \hat{\mathbf{a}} + \hat{\mathbf{A}} \frac{\partial \hat{\mathbf{a}}}{\partial \hat{\theta}_i} + \hat{\boldsymbol{\Phi}}_{\mathbf{x}} \boldsymbol{\Omega} \frac{\partial \mathbf{B}}{\partial \hat{\theta}_i} \mathbf{y} + \frac{\partial \hat{\boldsymbol{\Phi}}_{\mathbf{x}}}{\partial \hat{\theta}_i} \boldsymbol{\Omega} \mathbf{B} \mathbf{y} - \frac{\partial \hat{\boldsymbol{\Sigma}}_{\mathbf{w}}}{\partial \hat{\theta}_i} \hat{\bar{\mathbf{w}}} \right), \tag{4.7}$$

$$\frac{\partial \hat{\boldsymbol{\delta}}}{\partial \hat{\theta}_i} = -\frac{\partial \hat{\boldsymbol{\Phi}}_{\mathbf{x}}^T}{\partial \hat{\theta}_i} \hat{\bar{\mathbf{w}}} - \hat{\boldsymbol{\Phi}}_{\mathbf{x}}^T \frac{\partial \hat{\bar{\mathbf{w}}}}{\partial \hat{\theta}_i}, \tag{4.8}$$

$$\begin{aligned} \frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \hat{\theta}_i} = & -\frac{1}{2} \hat{\boldsymbol{\delta}}^T \boldsymbol{\Omega} \left(\frac{\partial \mathbf{B}}{\partial \hat{\theta}_i} \hat{\boldsymbol{\delta}} + 2\mathbf{B} \frac{\partial \hat{\boldsymbol{\delta}}}{\partial \hat{\theta}_i} \right) + \frac{1}{2} \mathrm{tr} \left(\boldsymbol{\Omega} \mathbf{B}^{-1} \frac{\partial \mathbf{B}}{\partial \hat{\theta}_i} \right) \\ & - \frac{1}{2} \left(\hat{\bar{\mathbf{w}}} - \hat{\mathbf{a}} \right)^T \left(\frac{\partial \hat{\mathbf{A}}}{\partial \hat{\theta}_i} \left(\hat{\bar{\mathbf{w}}} - \hat{\mathbf{a}} \right) + 2\hat{\mathbf{A}} \left(\frac{\partial \hat{\bar{\mathbf{w}}}}{\partial \hat{\theta}_i} - \frac{\partial \hat{\mathbf{a}}}{\partial \hat{\theta}_i} \right) \right) + \frac{1}{2} \mathrm{tr} \left(\hat{\mathbf{A}}^{-1} \frac{\partial \hat{\mathbf{A}}}{\partial \hat{\theta}_i} \right) \\ & - \frac{1}{2} \mathrm{tr} \left(\hat{\boldsymbol{\Sigma}}_{\mathbf{w}}^{-1} \frac{\partial \hat{\boldsymbol{\Sigma}}_{\mathbf{w}}}{\partial \hat{\theta}_i} \right) - \frac{1}{2} \mathbf{v}^T \left(\frac{\partial \mathbf{T}}{\partial \hat{\theta}_i} \mathbf{v} + 2\mathbf{T} \frac{\partial \mathbf{v}}{\partial \hat{\theta}_i} \right) + \frac{1}{2} \mathrm{tr} \left(\mathbf{T}^{-1} \frac{\partial \mathbf{T}}{\partial \hat{\theta}_i} \right), \end{aligned} \tag{4.9}$$

where recall that

$$\hat{\boldsymbol{\Sigma}}_{\mathbf{w}} = \hat{\boldsymbol{\Phi}}_{\mathbf{x}} \boldsymbol{\Omega} \mathbf{B} \hat{\boldsymbol{\Phi}}_{\mathbf{x}}^T + \hat{\mathbf{A}}, \tag{4.10}$$

$$\hat{\bar{\mathbf{w}}} = \hat{\boldsymbol{\Sigma}}_{\mathbf{w}}^{-1} \left(\hat{\mathbf{A}} \hat{\mathbf{a}} + \hat{\boldsymbol{\Phi}}_{\mathbf{x}} \boldsymbol{\Omega} \mathbf{B} \mathbf{y} \right), \tag{4.11}$$

$$\hat{\boldsymbol{\delta}} = \mathbf{y} - \hat{\boldsymbol{\Phi}}_{\mathbf{x}}^T \hat{\bar{\mathbf{w}}}. \tag{4.12}$$

More compactly, the gradients with respect to the different subsets of $\hat{\boldsymbol{\theta}}$, i.e. $\ln \hat{\boldsymbol{\alpha}}$, $\hat{\mathbf{a}}$, $\ln \hat{\boldsymbol{\tau}}$, $\mathbf{v}$ and b are:

$$\begin{aligned} \frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \ln \hat{\boldsymbol{\alpha}}} = & -\left(\hat{\boldsymbol{\Phi}}_{\mathbf{x}} \left(\boldsymbol{\omega} \odot \boldsymbol{\beta} \odot \hat{\boldsymbol{\delta}} \right) - \hat{\boldsymbol{\alpha}} \odot \left(\hat{\bar{\mathbf{w}}} - \hat{\mathbf{a}} \right) \right) \odot \left(\hat{\boldsymbol{\Sigma}}_{\mathbf{w}}^{-1} \left(\hat{\boldsymbol{\alpha}} \odot \left(\hat{\bar{\mathbf{w}}} - \hat{\mathbf{a}} \right) \right) \right) \\ & - \frac{1}{2} \hat{\boldsymbol{\alpha}} \odot \left(\hat{\bar{\mathbf{w}}} - \hat{\mathbf{a}} \right)^2 - \frac{1}{2} \mathrm{diag} \left[\hat{\boldsymbol{\Sigma}}_{\mathbf{w}}^{-1} \right] \hat{\boldsymbol{\alpha}} + \frac{1}{2}, \end{aligned} \tag{4.13}$$

$$\frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \ln \hat{\boldsymbol{\tau}}} = -\frac{1}{2} \hat{\boldsymbol{\tau}} \odot \mathbf{v}^2 + \frac{1}{2}, \tag{4.14}$$

$$\frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \hat{\mathbf{a}}} = \left(\hat{\boldsymbol{\Phi}}_{\mathbf{x}} \left(\boldsymbol{\omega} \odot \boldsymbol{\beta} \odot \hat{\boldsymbol{\delta}} \right) - \hat{\boldsymbol{\alpha}} \odot \left(\hat{\bar{\mathbf{w}}} - \hat{\mathbf{a}} \right) \right) \odot \left(\hat{\boldsymbol{\Sigma}}_{\mathbf{w}}^{-1} \hat{\boldsymbol{\alpha}} \right) + \hat{\boldsymbol{\alpha}} \odot \left(\hat{\bar{\mathbf{w}}} - \hat{\mathbf{a}} \right), \tag{4.15}$$

$$\frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \mathbf{v}} = \boldsymbol{\Phi}_{\mathbf{x}} \frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \ln \boldsymbol{\beta}} - \boldsymbol{\tau} \odot \mathbf{v}, \tag{4.16}$$

$$\frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial b} = \mathbf{1}_n^T \frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \ln \boldsymbol{\beta}}, \tag{4.17}$$

where

$$\frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \ln \boldsymbol{\beta}} = -\frac{1}{2} \boldsymbol{\omega} \odot \boldsymbol{\beta} \odot \left(\hat{\boldsymbol{\delta}}^2 + \left(\left(\hat{\boldsymbol{\Phi}}_{\mathbf{x}}^T \hat{\boldsymbol{\Sigma}}_{\mathbf{w}}^{-1} \right) \odot \hat{\boldsymbol{\Phi}}_{\mathbf{x}}^T \right) \mathbf{1}_m \right) + \frac{1}{2} \boldsymbol{\omega}. \tag{4.18}$$

Note that we optimise with respect to the logarithm of the precision parameters $\boldsymbol{\tau}$ and $\boldsymbol{\beta}$ in order to guarantee their positivity. The computation of the gradient with respect to the parameters $\boldsymbol{\varphi}$ of any basis function can be computed using the chain rule as follows:

$$\frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \boldsymbol{\varphi}} = \frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \boldsymbol{\Phi}_{\mathbf{x}}} \frac{\partial \boldsymbol{\Phi}_{\mathbf{x}}}{\partial \boldsymbol{\varphi}}, \tag{4.19}$$

the first term in the right-hand-side of Equation (4.19) can be computed irrespective of the basis function definition as follows:

$$\frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \boldsymbol{\Phi}_{\mathbf{x}}} = \mathbf{v} \left(\frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \ln \boldsymbol{\beta}} \right)^T + \left(\hat{\bar{\mathbf{w}}}[1:m] \hat{\boldsymbol{\delta}}^T - \hat{\boldsymbol{\Sigma}}_{\mathbf{w}}^{-1}[1:m,:] \, \hat{\boldsymbol{\Phi}}_{\mathbf{x}} \right) \odot \boldsymbol{\omega} \odot \boldsymbol{\beta}. \tag{4.20}$$

The above calculations are shared for any choice of basis function, the only remaining term to be computed is the partial derivative of the basis function with respect to its parameters $\frac{\partial \boldsymbol{\Phi}_{\mathbf{x}}}{\partial \boldsymbol{\varphi}}$. In the next sections, we will provide the partial derivatives with respect to the parameters of two popular basis functions and show a comparison between them.

4.3 The Sigmoidal Function

We are now left with basis function selection. We require the basis function to hold certain properties. For example, it is well defined over its input domain, continuously differentiable and that there exists a linear combination of the basis functions that can model the desired function to within some error. Since in most cases we have no prior knowledge about the function space, we also require our model to be malleable enough that it can fit any possible function to within some error. This concept is referred to as the *universality* of the function which for certain types of basis functions, under some mild conditions, can be guaranteed. One such function is the sigmoid activation function:

$$\phi_j(\mathbf{x}_i) = \frac{1}{1 + e^{-\left(\mathbf{h}_j^T \mathbf{x}_i + b_j\right)}}, \tag{4.21}$$

Table 4.1: Different valid activation functions.

Function	Equation	Derivative with respect to v	Range
Sigmoid	$\phi(v) = \frac{1}{1+e^{-v}}$	$\phi'(v) = (1-\phi(v))\,\phi(v)$	$(0,1)$
Hyperbolic Tangent	$\phi(v) = \frac{e^{v}-e^{-v}}{e^{v}+e^{-v}}$	$\phi'(v) = 1-\phi(v)^2$	$(-1,1)$
Inverse Tangent	$\phi(v) = \tan^{-1}(v)$	$\phi'(v) = \frac{1}{1+v^2}$	$\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$
Soft-sign	$\phi(v) = \frac{v}{1+\lvert v\rvert}$	$\phi'(v) = \frac{1}{(1+\lvert v\rvert)^2}$	$(-1,1)$

It is proven in Cybenko (1989), that a function of the form

$$F(\mathbf{x}) = \sum_{j=1}^{m} a_j \kappa(\mathbf{h}_j^T \mathbf{x} + b_j), \tag{4.22}$$

is a universal approximator. More formally, let $\kappa(.)$ be a non-constant, bounded and continuous function, $\mathbb{H}^d$ be a compact subspace of $\mathbb{R}^d$ and $\mathcal{S}\left(\mathbb{H}^d\right)$ be the space of continuous functions on $\mathbb{H}^d$. Then for any $f \in \mathcal{S}\left(\mathbb{H}^d\right)$ and $\epsilon > 0$, there exist $m \in \mathbb{Z}$, $\mathbf{h}_j \in \mathbb{R}^d$ and $a_j, b_j \in \mathbb{R}$, where $j \in \{1, \ldots, m\}$, such that:

$$|F(\mathbf{x}) - f(\mathbf{x})| < \epsilon, \forall \mathbf{x} \in \mathbb{H}^d, \tag{4.23}$$

this is formally known as the Universal Approximation Theorem (Cybenko, 1989). This basis function choice guarantees the existence of a solution; however, there is no guarantee regarding the number of basis functions required. The theorem also shows universality for any non-constant, monotonic, continuous and bounded functions such as the ones shown in Table 4.1.

4.3.1 Gradients

Let $\boldsymbol{\varphi} = \{\mathbf{H}, \mathbf{b}\}$, where $\mathbf{H} = [\mathbf{h}_1, \ldots, \mathbf{h}_m]$, $\mathbf{b} = [b_1, \ldots, b_m]^T$ and $\mathbf{V} = \mathbf{H}^T\mathbf{X} \oplus \mathbf{b}$, so that $\mathbf{V}[j,i] = \mathbf{h}_j^T \mathbf{x}_i + b_j$, then the partial derivative of $\mathcal{L}(\mathcal{D}, \boldsymbol{\theta})$ with respect to these parameters is:

$$\frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \boldsymbol{\varphi}} = \frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \boldsymbol{\Phi}_{\mathbf{x}}} \frac{\partial \boldsymbol{\Phi}_{\mathbf{x}}}{\partial \mathbf{V}} \frac{\partial \mathbf{V}}{\partial \boldsymbol{\varphi}}, \tag{4.24}$$

where

$$\frac{\partial \mathbf{V}}{\partial \boldsymbol{\varphi}} = \mathbf{X} \quad \text{if } \boldsymbol{\varphi} = \mathbf{H}, \tag{4.25}$$

$$\frac{\partial \mathbf{V}}{\partial \boldsymbol{\varphi}} = \mathbf{1}_m \quad \text{if } \boldsymbol{\varphi} = \mathbf{b}, \tag{4.26}$$

and $\frac{\partial \boldsymbol{\Phi}_\mathbf{x}}{\partial \mathbf{V}}$ is computed as follows for each type of activation function:

$$\frac{\partial \boldsymbol{\Phi}_\mathbf{x}}{\partial \mathbf{V}} = (\mathbf{1}_m \ominus \boldsymbol{\Phi}_\mathbf{x}) \odot \boldsymbol{\Phi}_\mathbf{x}, \quad \text{for the sigmoid,} \tag{4.27}$$

$$= \mathbf{1}_m \ominus (\boldsymbol{\Phi}_\mathbf{x} \odot \boldsymbol{\Phi}_\mathbf{x}), \quad \text{for the hyperbolic tangent,} \tag{4.28}$$

$$= \mathbf{1}_m \oslash (\mathbf{1}_m \oplus \mathbf{V} \odot \mathbf{V}), \quad \text{for the inverse tangent,} \tag{4.29}$$

$$= \mathbf{1}_m \oslash (\mathbf{1}_m \oplus |\mathbf{V}|) \odot (\mathbf{1}_m \oplus |\mathbf{V}|), \quad \text{for the soft-sign,} \tag{4.30}$$

where $|\mathbf{V}|$ here denotes the absolute value of each element in $\mathbf{V}$, not its determinant. Finally, from Equations (4.20) and (4.25) to (4.30), we obtain

$$\frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \mathbf{V}} = \frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \boldsymbol{\Phi}_\mathbf{x}} \odot \frac{\partial \boldsymbol{\Phi}_\mathbf{x}}{\partial \mathbf{V}}, \tag{4.31}$$

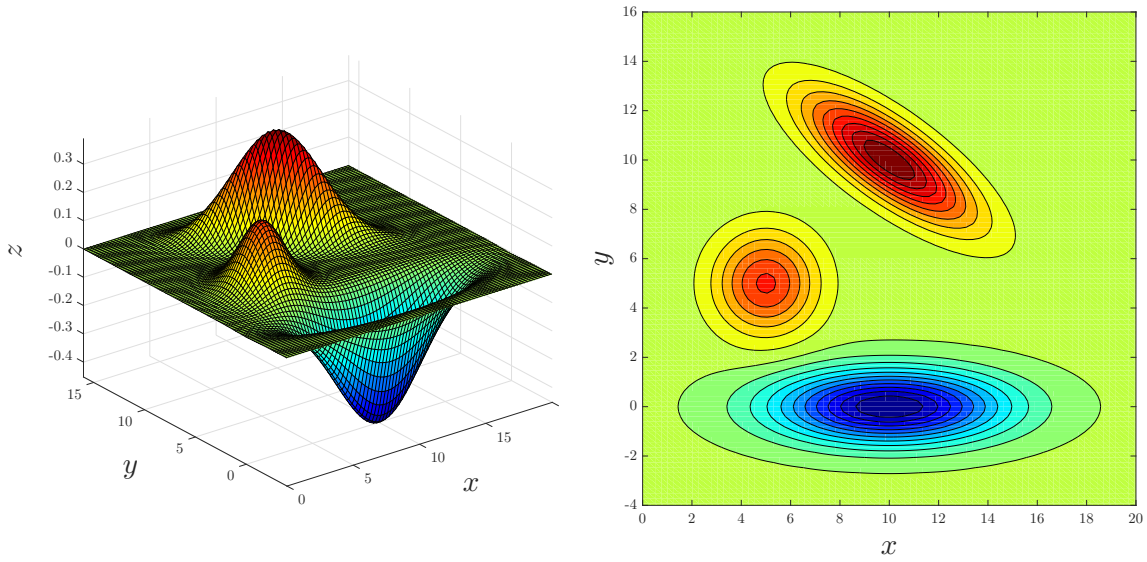
$$\frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \mathbf{H}} = \mathbf{X} \frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \mathbf{V}}^T, \tag{4.32}$$

$$\frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \mathbf{b}} = \frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \mathbf{V}} \mathbf{1}_n. \tag{4.33}$$

4.3.2 Illustrative Examples

The parameters of the basis functions describe a set of hyperplanes $\in \mathbb{R}^d$. These hyperplanes are *soft* decision boundaries which collectively define areas in the input space that map to certain values in the output space. Thus, the more boundaries available, the more the approximation of the original function can be granular. Let $\mathbf{v}_j^T = \frac{\begin{bmatrix}\mathbf{h}_j^T & b_j\end{bmatrix}}{\left\|\begin{bmatrix}\mathbf{h}_j^T & b_j\end{bmatrix}\right\|}$ and $\lambda_j = \left\|\begin{bmatrix}\mathbf{h}_j^T & b_j\end{bmatrix}\right\|$, so that $\mathbf{v}_j$ uniquely determines the direction of the hyperplane and λ_j describes the rate of change, then the hyperbolic tangent activation function for example can be rewritten as:

$$\phi_j(\mathbf{x}_i) = \tanh\left(\lambda_j \mathbf{v}_j^T \begin{bmatrix}\mathbf{x}_i \\ 1\end{bmatrix}\right). \tag{4.34}$$



(a) Target function (b) Contour plot

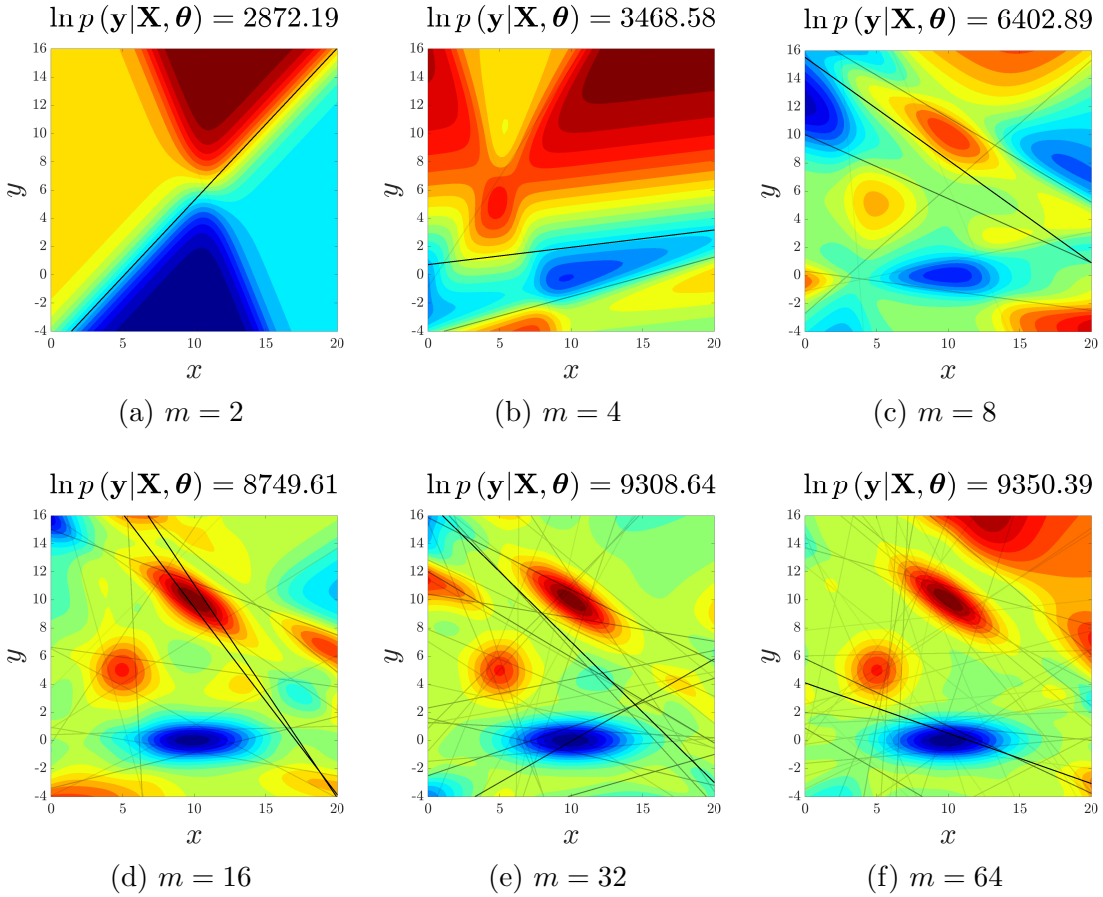
Figure 4.1: A 2D toy example described in Equation (4.35)

We use the two-dimensional artificial dataset in Figure 4.1 to illustrate the effects of sigmoid-like basis functions, specifically the hyperbolic tangent. The target value z is a function of x and y, which is a linear combination of three Gaussian PDFs, one with a full covariance, the second with a diagonal covariance and the third with an isotropic covariance plus Gaussian noise with a standard deviation of $\sigma = 0.01$:

$$\begin{aligned} f(x,y) = &-9\mathcal{N}\left(\begin{bmatrix}x\\y\end{bmatrix}\middle|\begin{bmatrix}10\\0\end{bmatrix},\begin{bmatrix}10 & 0\\0 & 1\end{bmatrix}\right) + 6\mathcal{N}\left(\begin{bmatrix}x\\y\end{bmatrix}\middle|\begin{bmatrix}10\\10\end{bmatrix},\begin{bmatrix}5 & -3\\-3 & 3\end{bmatrix}\right) \\ &+ 3\mathcal{N}\left(\begin{bmatrix}x\\y\end{bmatrix}\middle|\begin{bmatrix}5\\5\end{bmatrix},\begin{bmatrix}2 & 0\\0 & 2\end{bmatrix}\right), \end{aligned} \tag{4.35}$$

$$z_i \sim \mathcal{N}\left(z_i | f(x_i, y_i), \sigma^2\right). \tag{4.36}$$

We sample 1000 training points from each distribution in Equation (4.35), for a total of 3000, then sample the target outputs from Equation (4.36). Figure 4.2 shows the results of training a hyperbolic tangent based BFM on the example in Figure 4.1 using different number of basis functions. The parameters of the basis functions are visualised as lines on the xy-plane where the transparency of each line is proportional



(a) $m = 2$ (b) $m = 4$ (c) $m = 8$

(d) $m = 16$ (e) $m = 32$ (f) $m = 64$

Figure 4.2: The results of training a hyperbolic tangent based model with different numbers of basis functions (m), from (a) to (f) in multiples of 2. The lines plotted on the xy-plane represent the learned parameters of the basis functions, where the degree of transparency is proportional to the relevance of the basis as determined by Equation (4.37). The log marginal likelihood is shown above each plot.

to its relevance score r_j, where

$$r_j = \frac{|\bar{w}_j|}{\sum_{k=1}^{m} |\bar{w}_k|}. \tag{4.37}$$

Note that as the number of basis functions increases, the fitness improves. However, adding too many basis functions creates a complex web of hyperplanes that is difficult to optimise, especially for regression tasks. Since the basis function's decision boundary extends to infinity in both directions, it has a global effect and cannot be adjusted without affecting the other basis functions that will most likely intersect with. There is no ideal basis function that is suitable for every situation, and in this

case the hyperbolic tangent might not be ideal. Without further knowledge about the function, one cannot definitely determine which basis function will result in the least number of them being used. In the next section we will discuss another basis function that can also guarantee universality, can be adjusted with very little effect on other basis functions and, as detalied in Chapter 5, can be utilised to handle noisy and missing data very effectively.

4.4 Radial Basis Functions

As mentioned in Section 4.3, the most important criterion when choosing a basis function is its universality. Another basis function that satisfies this condition is the radial basis function (Park and Sandberg, 1991), defined as follows:

$$\phi_j(\mathbf{x}_i) = \exp\left(-\frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_j)^T \boldsymbol{\Gamma}_j^T \boldsymbol{\Gamma}_j (\mathbf{x}_i - \boldsymbol{\mu}_j)\right), \tag{4.38}$$

i.e. an un-normalised Gaussian PDF. The basis function is parameterised by its centre $\boldsymbol{\mu}_j \in \mathbb{R}^d$ and its precision $\boldsymbol{\Sigma}_j^{-1} = \boldsymbol{\Gamma}_j^T \boldsymbol{\Gamma}_j$, where $\boldsymbol{\Gamma}_j \in \mathbb{R}^{d \times d}$. We refer to the model with such basis functions as sparse Gaussian processes with variable covariances, or GPVC. The framework also allows for other types of covariance structures, such as:

1. GPVC: variable covariances, or a bespoke $\boldsymbol{\Gamma}_j$, for each basis function j.
2. GPGC: a global covariance, or a shared $\boldsymbol{\Gamma}$, for all basis functions.
3. GPVD: variable diagonal covariances, or a bespoke diagonal $\boldsymbol{\Gamma}_j$, for each basis function j.
4. GPGD: a global diagonal covariance, or a shared diagonal $\boldsymbol{\Gamma}$, for all basis functions.
5. GPVL: variable length-scales, or a bespoke isotropic covariance for each basis function j, i.e. $\boldsymbol{\Gamma}_j = \gamma_j \mathbf{I}_d$, where γ_j is a scalar.

6. GPGL: a global length-scale, or a shared isotropic covariance for all basis functions $\boldsymbol{\Gamma} = \gamma \mathbf{I}_d$, where γ is a scalar.

Basis function models with global shared length-scales are broad enough for universal approximation (Park and Sandberg, 1991). Since the function space of GPGL for any given number of basis m is a subspace of the other types, then all of the above-mentioned types are universal approximators. Moreover, for a fixed set of centres, the potential accuracy that can be achieved by GPVC will always be greater than, or equal to, any of the other methods. Since we can always use the parameters of the best fit model, say for GPVL, to initialise GPVC, optimising the covariances while fixing the centres can only improve the objective.

4.4.1 Gradients

We now consider the gradients with respect to the RBF's parameters, i.e. $\boldsymbol{\varphi} = \{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_m, \boldsymbol{\Gamma}_1, \ldots, \boldsymbol{\Gamma}_m\}$. Since the RBF is an exponential function, it is easier to compute the gradient as follows

$$\begin{aligned} \frac{\partial \boldsymbol{\Phi}_{\mathbf{x}}}{\partial \boldsymbol{\varphi}} &= \frac{\partial \boldsymbol{\Phi}_{\mathbf{x}}}{\partial \ln \boldsymbol{\Phi}_{\mathbf{x}}} \frac{\partial \ln \boldsymbol{\Phi}_{\mathbf{x}}}{\partial \boldsymbol{\varphi}}, \\ &= \boldsymbol{\Phi}_{\mathbf{x}} \odot \frac{\partial \ln \boldsymbol{\Phi}_{\mathbf{x}}}{\partial \boldsymbol{\varphi}}, \end{aligned} \tag{4.39}$$

requiring only the partial derivatives with respect to the exponent. Thus, the partial derivatives of the parameters of a specific basis function j for a specific input i with respect to the exponent are as follows:

$$\frac{\partial \ln \boldsymbol{\Phi}_{\mathbf{x}}[j,i]}{\partial \boldsymbol{\mu}_j} = \boldsymbol{\Gamma}_j^T \boldsymbol{\Gamma}_j \left(\mathbf{x}_i - \boldsymbol{\mu}_j\right), \tag{4.40}$$

$$\frac{\partial \ln \boldsymbol{\Phi}_{\mathbf{x}}[j,i]}{\partial \boldsymbol{\Gamma}_j} = -\boldsymbol{\Gamma}_j \left(\mathbf{x}_i - \boldsymbol{\mu}_j\right)\left(\mathbf{x}_i - \boldsymbol{\mu}_j\right)^T. \tag{4.41}$$

For the entire dataset, the gradient of $\mathcal{L}(\mathcal{D}, \boldsymbol{\theta})$ with respect to the parameters can be computed more efficiently by predefining $\boldsymbol{\Delta}_j = \mathbf{X} \ominus \boldsymbol{\mu}_j$ then computing the following:

$$\frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \ln \boldsymbol{\Phi}_{\mathbf{x}}} = \frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \boldsymbol{\Phi}_{\mathbf{x}}} \odot \boldsymbol{\Phi}_{\mathbf{x}}, \tag{4.42}$$

$$\frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \boldsymbol{\mu}_j} = \boldsymbol{\Gamma}_j^T \boldsymbol{\Gamma}_j \boldsymbol{\Delta}_j \frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \ln \boldsymbol{\Phi}_{\mathbf{x}}} \left[j, : \right]^T, \tag{4.43}$$

$$\frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \boldsymbol{\Gamma}_j} = -\boldsymbol{\Gamma}_j \boldsymbol{\Delta}_j \left(\boldsymbol{\Delta}_j^T \odot \frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \ln \boldsymbol{\Phi}_{\mathbf{x}}} \left[j, : \right]^T \right). \tag{4.44}$$

We can now compute the gradient with respect to the special cases of $\boldsymbol{\Gamma}_j$ as follows:

$$\frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \boldsymbol{\Gamma}} = \sum_{j=1}^{m} \frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \boldsymbol{\Gamma}_j}, \tag{4.45}$$

$$\frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \mathrm{diag}\left[\boldsymbol{\Gamma}_j\right]} = \mathrm{diag}\left[\frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \boldsymbol{\Gamma}_j} \right], \tag{4.46}$$

$$\frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \mathrm{diag}\left[\boldsymbol{\Gamma}\right]} = \sum_{j=1}^{m} \mathrm{diag}\left[\frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \boldsymbol{\Gamma}_j} \right], \tag{4.47}$$

$$\frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \gamma_j} = \mathrm{tr}\left(\frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \boldsymbol{\Gamma}_j} \right), \tag{4.48}$$

$$\frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \gamma} = \sum_{j=1}^{m} \mathrm{tr}\left(\frac{\partial \mathcal{L}(\mathcal{D}, \boldsymbol{\theta})}{\partial \boldsymbol{\Gamma}_j} \right), \tag{4.49}$$

4.4.2 Illustrative Examples

We use the same data from Figure 4.1 to demonstrate the effects of using RBFs with different types of covariances. Figure 4.3 shows the results of training such sparse GPs with differing covariances and increasing number of basis functions from 2 to 64; namely the fully independent training conditional (FITC), BFM with variable length scales (GPVL), a BFM with variable diagonal covariances (GPVD) and a BFM with variable full covariances (GPVC). The transparency of the lines is proportional to the relevance of the basis, as determined by Equation (4.37). Note that as we make the covariances more rich, and variable, the better the model fit is as measured by the log marginal likelihood, and the sparser the solution is. The BFM with variable covariances performed the best, we note it also learned to ignore extra basis functions

Table 4.2: The time complexity of each approach.

Method	Time complexity
ANN (l-layers)	$O\left(nmd + (l-1)(nm^2)\right)$
FITC	$O\left(nmd + nm^2\right)$
GPVL	$O\left(nmd + nm^2\right)$
GPVD	$O\left(nmd + nm^2\right)$
GPVC	$O\left(nmd^2 + nm^2\right)$
Full GP	$O\left(n^3\right)$

that are not needed due to the use of automatic relevance determination. It is worth mentioning that this comes with extra cost in time complexity that is proportional to the dimensionality of the input d. Table 4.2 shows the time complexities for the different methods discussed in this chapter. Note that if $m \geq d$, the dominant factor will be the term nm^2 for FITC, GPVL and GPVD; thus, the upper bound for these models is $O(nm^2)$. On the other hand, GPVC has complexity $O(nmd^2 + nm^2)$; thus, the time complexity will be $O(nm^2)$ if $m \geq d^2$ and $O(nmd^2)$ otherwise. Note that the time complexity will not differ between a variable versus a global covariance of the same type. One can achieve an intermediate time complexity of $O(nmdq + nm^2)$ for GPVC if one sets $\mathbf{\Gamma}_j \in \mathbb{R}^{q \times d}$, where $q \leq d$. This can be viewed as finding a low rank approximation of the precision matrix $\mathbf{\Sigma}_j^{-1} = \mathbf{\Gamma}_j^T \mathbf{\Gamma}_j$ and it is referred to as *factor analysis* distance (Rasmussen and Williams, 2006, p. 107).

4.5 Summary

This chapter has outlined the optimisation procedure for the hyper-parameters of any basis function definition using a gradient-based optimisation procedure and provided the gradient calculations for several varieties of sigmoid-like activation functions and radial basis functions. We have shown that both types of basis functions have universal approximation capabilities, although some types of basis functions might be more suitable for certain types of functions more than others. In addition, we have

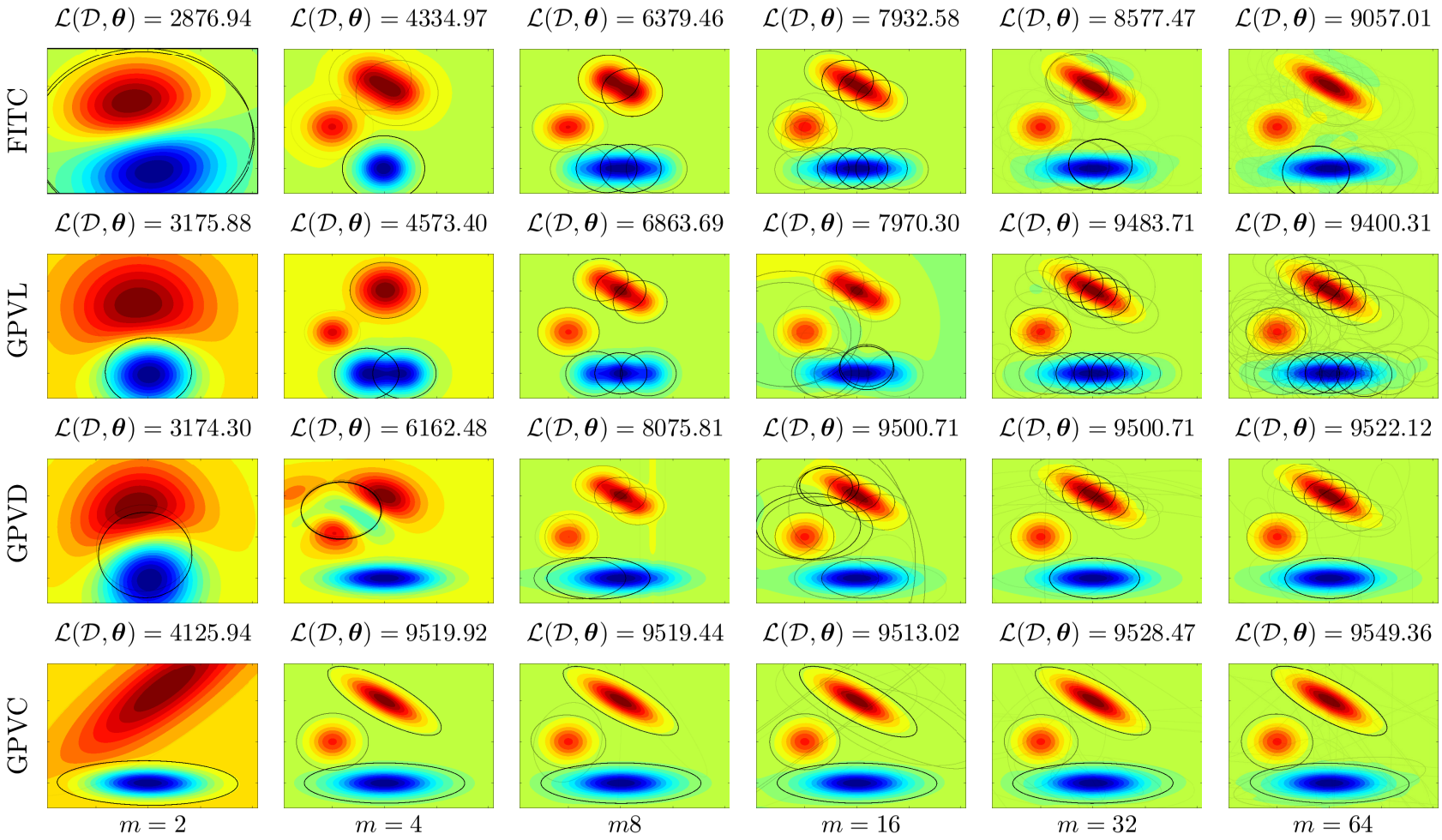


Figure 4.3: The results of training FITC, GPVL, GPVD and GPVC with different numbers of basis functions (m) on the same data collected using Equation (4.35). The ellipses represent the learned covariances of the RBFs, where the degree of transparency is proportional to the relevance (Equation (4.37)). The log marginal likelihoods are shown above each plot.

shown that using radial basis functions with more flexible covariances can potentially enhance the performance and reduce the required number of basis functions to reach the same accuracy compared to a shared global length-scale.

Chapter 5

Noisy or Missing Variables

Thus, far in our discussion, we made the implicit assumption that there is absolute certainty regarding the values of the input (observed) variables and, further, that these variables will continue to be fully observed in any future analysis. In many applications this is unlikely. For example, a model trained using data obtained via one set of equipment might be presented with future observations collected from a different set of equipment which may have less precise measurement capabilities. Also, we might not be able to observe all the variables that were used to train the model for every test sample. Ideally, we require methods that provide the best estimation possible using observed data without resorting to retraining models on the limited data available. In addition to the inevitable increase in computational cost, this will require transporting the data along with the model; which might not be feasible for security, privacy or technical concerns. In the following sections, we will discuss how we can predict and train with missing and/or noisy variables in the context of basis function models with radial basis functions; which, as we have discussed, we may treat as sparse Gaussian Processes. We will first address how to predict with known input noise in Section 5.1.1, since this can be addressed independently of the training data, then in Section 5.1.2 we will show how predicting with missing variables is a special case of noisy input prediction and how can we construct from the learned RBFs a probability density model to convert the missing input problem into a noisy

input problem. Cases where some of the inputs are noisy or missing is considered in Section 5.1.3. In Section 5.2.1, we consider the case in which training data inputs are uncertain by approximating the expected value of the marginal likelihood. We then compare to other techniques for training with noisy inputs. Finally, in Section 5.2.2 we propose a method for training with noisy and/or missing variables that is effective for high percentages of missing data. This chapter is a novel contribution in the thesis.

5.1 Prediction

5.1.1 Predicting with Noisy Variables

Without loss of generality, and for the purposes of simplifying the derivations, we formulate the predictive distribution in this chapter as follows:

$$p\left(y|\mathbf{x}\right) = \mathcal{N}\left(y|f\left(\mathbf{x}\right), \nu^2\left(\mathbf{x}\right) + \sigma^2\left(\mathbf{x}\right)\right), \tag{5.1}$$

where

$$f\left(\mathbf{x}\right) = \quad \boldsymbol{\phi}\left(\mathbf{x}\right)^T \bar{\mathbf{w}} \quad = \sum_{i=1}^{m} \bar{w}_i z_i \mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right), \tag{5.2}$$

$$\sigma^2\left(\mathbf{x}\right) = \exp\left(\boldsymbol{\phi}\left(\mathbf{x}\right)^T \mathbf{v} + b\right) = \exp\left(\sum_{i=1}^{m} v_i z_i \mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right) + b\right), \tag{5.3}$$

$$\nu^2\left(\mathbf{x}\right) = \boldsymbol{\phi}\left(\mathbf{x}\right)^T \boldsymbol{\Sigma}_{\mathbf{w}}^{-1} \boldsymbol{\phi}\left(\mathbf{x}\right) = \sum_{i=1}^{m}\sum_{j=1}^{m} \boldsymbol{\Sigma}_{\mathbf{w}}^{-1}[i,j] z_i z_j \mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right) \mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right), \tag{5.4}$$

and

$$z_i = \sqrt{\left(2\pi\right)^d \left|\boldsymbol{\Sigma}_i\right|}, \tag{5.5}$$

$$\boldsymbol{\Sigma}_i = \left(\boldsymbol{\Gamma}_i^T \boldsymbol{\Gamma}_i\right)^{-1}. \tag{5.6}$$

In case the input is uncertain with probability $p(\mathbf{x})$, the expected value of $f\left(\mathbf{x}\right)$, from Equation (A.7), is as follows:

$$\mathbb{E}\left[f\left(\mathbf{x}\right)\right] = \int f\left(\mathbf{x}\right) p\left(\mathbf{x}\right) \mathrm{d}\mathbf{x}. \tag{5.7}$$

If the input $\mathbf{x} \in \mathbb{R}^d$, it is reasonable to assume that $p(\mathbf{x})$ is Gaussian, or can be approximated using a mixture of Gaussians for any other non-standard distribution (Murphy, 2012, p. 341). In this section, we will consider the case where $\mathbf{x} \sim \mathcal{N}(\mathbf{x}|\bar{\mathbf{x}}, \boldsymbol{\Psi})$, then in the next section we will show how this can be trivially extended to a Gaussian mixture model (GMM). The expected value of $f(\mathbf{x})$, if $p(\mathbf{x})$ is Normal, is as follows:

$$\begin{aligned}\mathbb{E}[f(\mathbf{x})] &= \sum_{i=1}^{m} \bar{w}_i z_i \int \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\, \mathcal{N}(\mathbf{x}|\bar{\mathbf{x}}, \boldsymbol{\Psi})\, \mathrm{d}\mathbf{x}, \\ &= \sum_{i=1}^{m} \bar{w}_i z_i \mathcal{N}(\bar{\mathbf{x}}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i + \boldsymbol{\Psi}). \end{aligned} \tag{5.8}$$

This induces a variance on $f(\mathbf{x})$, as follows:

$$\mathbb{V}[f(\mathbf{x})] = \mathbb{E}\left[f(\mathbf{x})^2\right] - \mathbb{E}[f(\mathbf{x})]^2, \tag{5.9}$$

$$\begin{aligned}\mathbb{E}\left[f(\mathbf{x})^2\right] &= \int \left(\sum_{i=1}^{m} \bar{w}_i z_i \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\right)^2 p(\mathbf{x})\, \mathrm{d}\mathbf{x}, \\ &= \int \sum_{i=1}^{m}\sum_{j=1}^{m} \bar{w}_i \bar{w}_j z_i z_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\, \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)\, \mathcal{N}(\mathbf{x}|\bar{\mathbf{x}}, \boldsymbol{\Psi})\, \mathrm{d}\mathbf{x}, \\ &= \sum_{i=1}^{m}\sum_{j=1}^{m} \bar{w}_i \bar{w}_j z_i z_j \mathcal{N}(\boldsymbol{\mu}_i|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_i + \boldsymbol{\Sigma}_j)\, \mathcal{N}(\bar{\mathbf{x}}|\mathbf{c}_{ij}, \mathbf{C}_{ij} + \boldsymbol{\Psi}), \end{aligned} \tag{5.10}$$

where

$$\mathbf{C}_{ij} = \left(\boldsymbol{\Sigma}_i^{-1} + \boldsymbol{\Sigma}_j^{-1}\right)^{-1}, \tag{5.11}$$

$$\mathbf{c}_{ij} = \mathbf{C}_{ij}\left(\boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i + \boldsymbol{\Sigma}_j^{-1}\boldsymbol{\mu}_j\right). \tag{5.12}$$

Similarly, the expected value of the model variance is as follows

$$\mathbb{E}\left[\nu^2(\mathbf{x})\right] = \sum_{i=1}^{m}\sum_{j=1}^{m} \boldsymbol{\Sigma}_{\mathbf{w}}^{-1}[i,j] z_i z_j \mathcal{N}(\boldsymbol{\mu}_i|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_i + \boldsymbol{\Sigma}_j)\, \mathcal{N}(\bar{\mathbf{x}}|\mathbf{c}_{ij}, \mathbf{C}_{ij} + \boldsymbol{\Psi}), \tag{5.13}$$

The expected value of the noise variance, $\mathbb{E}[\sigma^2(\mathbf{x})]$, cannot be computed directly. Hence, we use a Taylor series expansion to approximate it, noting that for any function

$g(x)$:

$$\begin{aligned}\mathbb{E}[g(x)] &= \mathbb{E}\left[g\Big(\mathbb{E}[x] + (x - \mathbb{E}[x])\Big)\right], \\ &\approx \mathbb{E}\left[g(\mathbb{E}[x]) + g'(\mathbb{E}[x])(x - \mathbb{E}[x]) + \frac{1}{2}g''(\mathbb{E}[x])(x - \mathbb{E}[x])^2\right], \\ &\approx g(\mathbb{E}[x]) + \frac{1}{2}g''(\mathbb{E}[x])\,\mathbb{V}[x]. \end{aligned} \tag{5.14}$$

Therefore, we can compute the variance and the expectation of the logarithm of $\sigma^2(\mathbf{x})$, then approximate $\mathbb{E}[\sigma^2(\mathbf{x})]$ using Equation (5.14):

$$\mathbb{E}\left[\ln \sigma^2(\mathbf{x})\right] = \sum_{i=1}^{m} v_i z_i \mathcal{N}(\bar{\mathbf{x}}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i + \boldsymbol{\Psi}) + b, \tag{5.15}$$

$$\begin{aligned}\mathbb{V}\left[\ln \sigma^2(\mathbf{x})\right] = &\sum_{i=1}^{m}\sum_{j=1}^{m} v_i v_j z_i z_j \mathcal{N}(\boldsymbol{\mu}_i|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_i + \boldsymbol{\Sigma}_j)\,\mathcal{N}(\bar{\mathbf{x}}|\mathbf{c}_{ij}, \mathbf{C}_{ij} + \boldsymbol{\Psi}) \\ &- \left(\mathbb{E}\left[\ln \sigma^2(\mathbf{x})\right] - b\right)^2. \end{aligned} \tag{5.16}$$

Substituting Equations (5.15) and (5.16) into Equation (5.14), and letting $g(x) = \exp(x)$, we obtain

$$\mathbb{E}\left[\sigma^2(\mathbf{x})\right] \approx \exp\left(\mathbb{E}\left[\ln \sigma^2(\mathbf{x})\right]\right)\left(1 + \frac{1}{2}\mathbb{V}\left[\ln \sigma^2(\mathbf{x})\right]\right). \tag{5.17}$$

It is worth noting that in the case $g(x) = x^2$, the Taylor series expansion is no longer an approximation but rather an exact solution. However, in practice, using the exponential as the link function has a better and more stable performance during optimisation. Finally, we arrive at the predictive distribution, in the presence of noise, as follows:

$$p(y|\bar{\mathbf{x}}, \boldsymbol{\Psi}) = \int p(y|f(\mathbf{x}))\, p(f(\mathbf{x})|\bar{\mathbf{x}}, \boldsymbol{\Psi})\, \mathrm{d}f(\mathbf{x}), \tag{5.18}$$

$$= \mathcal{N}\left(y|\mathbb{E}[f(\mathbf{x})]\,|\mathbb{V}[f(\mathbf{x})] + \mathbb{E}\left[\nu^2(\mathbf{x})\right] + \mathbb{E}\left[\sigma^2(\mathbf{x})\right]\right), \tag{5.19}$$

where for the noise and model variance functions, we have simply replaced them with their expected values.

To test this approach, consider the artificial dataset from Figure 3.2. The estimated values for $\mathbb{E}[f(x)]$, $\mathbb{E}[f(x)^2]$, $\mathbb{E}[\nu^2(x)]$ and $\mathbb{E}[\sigma^2(x)]$ were computed for 4000

Table 5.1: The mean z-scores, plus or minus one standard deviation, for $\mathbb{E}[f(x)]$, $\mathbb{E}[f(x)^2]$, $\mathbb{E}[\nu^2(x)]$ and $\mathbb{E}[\sigma^2(x)]$ based on the analytical method described in this section for noisy input prediction and their estimated values based on a sampling method.

	Mean $\pm$ 1 Standard Deviation
$\mathbb{E}[f(x)]$	$7.9 \times 10^{-3} \pm 6.0 \times 10^{-3}$
$\mathbb{E}[f(x)^2]$	$7.8 \times 10^{-3} \pm 6.0 \times 10^{-3}$
$\mathbb{E}[\nu^2(x)]$	$8.0 \times 10^{-3} \pm 6.1 \times 10^{-3}$
$\mathbb{E}\left[\sigma^2(x) = (\boldsymbol{\phi}(x)^T \mathbf{v} + b)^2\right]$	$7.9 \times 10^{-3} \pm 6.0 \times 10^{-3}$
$\mathbb{E}\left[\sigma^2(x) = \exp(\boldsymbol{\phi}(x)^T \mathbf{v} + b)\right]$	$1.5 \times 10^{-1} \pm 1.3 \times 10^{-1}$

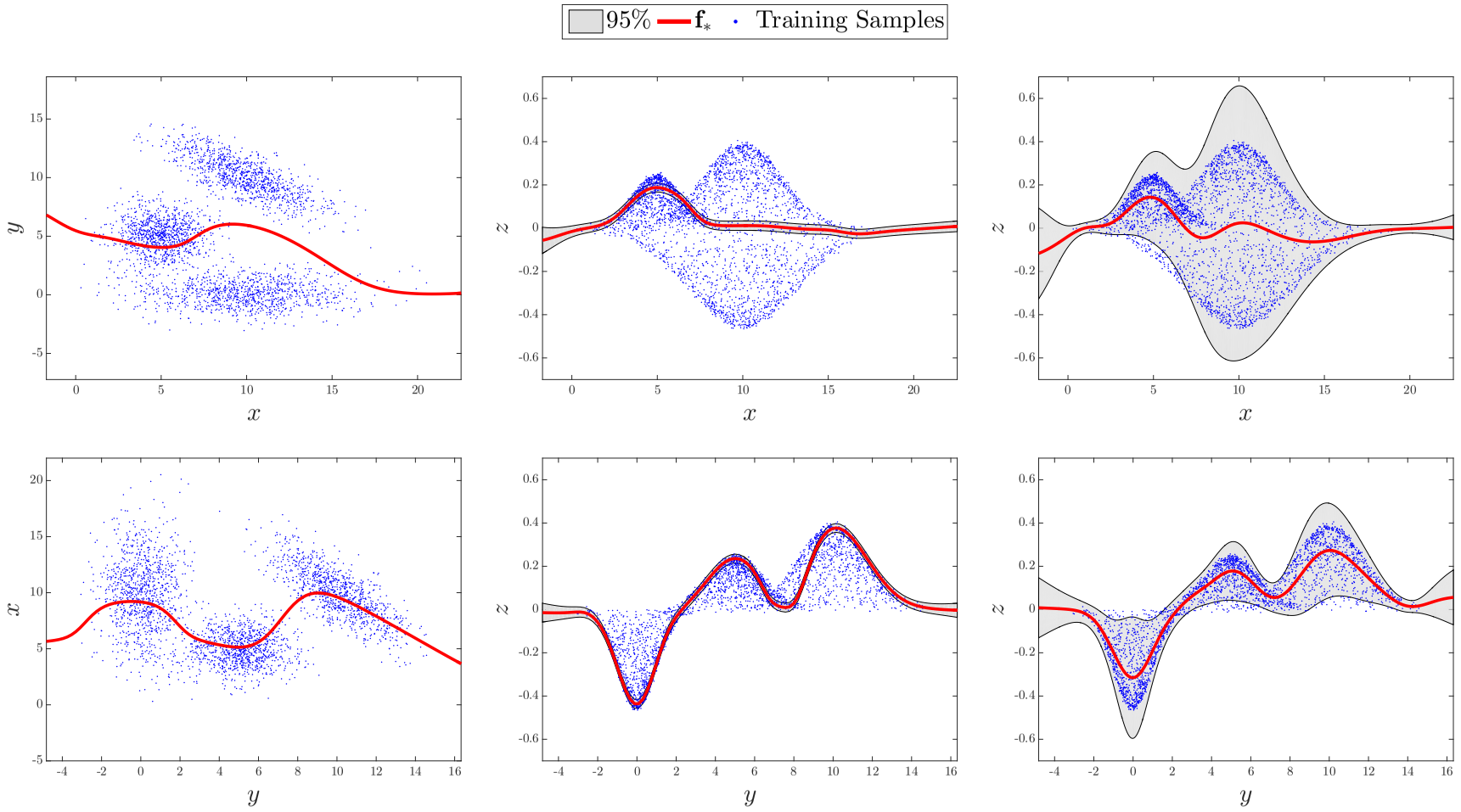
uniformly spaced points from $x = -12$ to $x = 12$. Each input x_i was associated with a random variance $\psi_i \sim \text{Gamma}(\psi_i|\alpha = 1, \beta = 2)$, such that ψ_i has a mean of 0.5 and a variance of 0.25. As a baseline comparison, we sampled 1000 points from the distribution $\mathcal{N}(x_i, \psi_i)$, for each sample i, in order to compare the sample statistics with the analytical computation described above. We report in Table 5.1 the mean and the variance of the z-scores, the absolute difference between the predicted values obtained from the analytical method and the mean of the sample divided by the standard deviation of the sample. We note that the deviations in the results are very small, on average less than 0.01. The area between -0.01 and 0.01 z-scores in a Normal distribution is less than 1%. The results when $g(x) = \exp(x)$ are slightly worse, ~ 0.15, but within acceptable range of z-score values, i.e. $\sim 12\%$ of the area is between -0.15 and 0.15 z-scores.

5.1.2 Predicting with Missing Variables

Let o and u be the sets indicating the locations of the *observed* and *unobserved* variables in $\mathbf{x}$ respectively. For simplicity of notation, let $\mathbf{o} = \mathbf{x}[o]$ and $\mathbf{u} = \mathbf{x}[u]$. A common way to address such a problem interpolates the regions of missing values using the expectations under a model, i.e. $\mathbf{u} = \mathbb{E}[\mathbf{u}]$. This process can be as simple

as assigning a constant value, e.g. zero, or the mean of the missing variable, or as complicated as building a separate model to compute the expected value of the missing given the observed, i.e. $\mathbf{u} = \mathbb{E}[\mathbf{u}|\mathbf{o}]$. However, building a separate model for every possible combination of missing variables is impractical. For the sake of demonstration in this section, we use the artificial dataset from the example in Figure 4.1 and consider every possible input combination, i.e. $p(z|x,y)$, $p(z|x)$, $p(z|y)$, where $p(z|x)$ and $p(z|y)$ are used as reference models to evaluate how well can we derive them from the original model, $p(z|x,y)$, without re-optimisation; where the target output in this case is the variable z, whereas x and y are the input variables.

Consider the approximation approach of using the expected value of the missing variable given the observed one, i.e. $p(z|x) \approx p(z|x, \mathbb{E}[y|x])$ and $p(z|y) \approx p(z|\mathbb{E}[x|y], y)$. Thus, we require access to two additional models, $p(x|y)$ and $p(y|x)$, so that in case of a missing variable, say y, we can use the appropriate model to interpolate missing data using the observations, i.e. $\mathbb{E}[y|x]$. Even though this would incur additional cost in training, we consider it here to study the performance under the ideal scenario of having access to the optimal $\mathbb{E}[x|y]$ and $\mathbb{E}[y|x]$ values. Figure 5.1 shows the result of applying this method to the toy example from Figure 4.1. It is evident from the plots that even with our best attempt of estimating the missing variable, the performance is very poor compared to the reference models for both cases. This is because the underlying assumption in this approach is that $\mathbb{E}[f(x)] = f(\mathbb{E}[x])$, which is not universally true. To demonstrate this, consider again the Taylor series expansion in Equation (5.14). Note that, for the second order approximation, $\mathbb{E}[f(x)] = f(\mathbb{E}[x])$ is true only if the variance or the second derivative of the function at x is zero. Thus, once uncertainty is introduced, this approximation no longer holds for any twice differentiable functions. In this example, the basis function is infinitely differentiable and there is always a degree of uncertainty associated with $\mathbb{E}[y|x]$, we therefore need to take the variance $\mathbb{V}[y|x]$ into account. We argue that we should use



(a) $\mathbb{E}[\mathbf{u}|\mathbf{o}]$ (b) $p(z|\mathbf{o}, \mathbb{E}[\mathbf{u}|\mathbf{o}])$ (c) $p(z|\mathbf{o})$

Figure 5.1: The result of using the expected value of a missing variable on the 2D artificial example from Figure 4.1 compared to the reference model that is trained only on the observed variable.

$p(y|x)$ as a prior and marginalise out y from $f(x, y)$. For a basis function model with a Radial Basis Function (RBF) kernel,

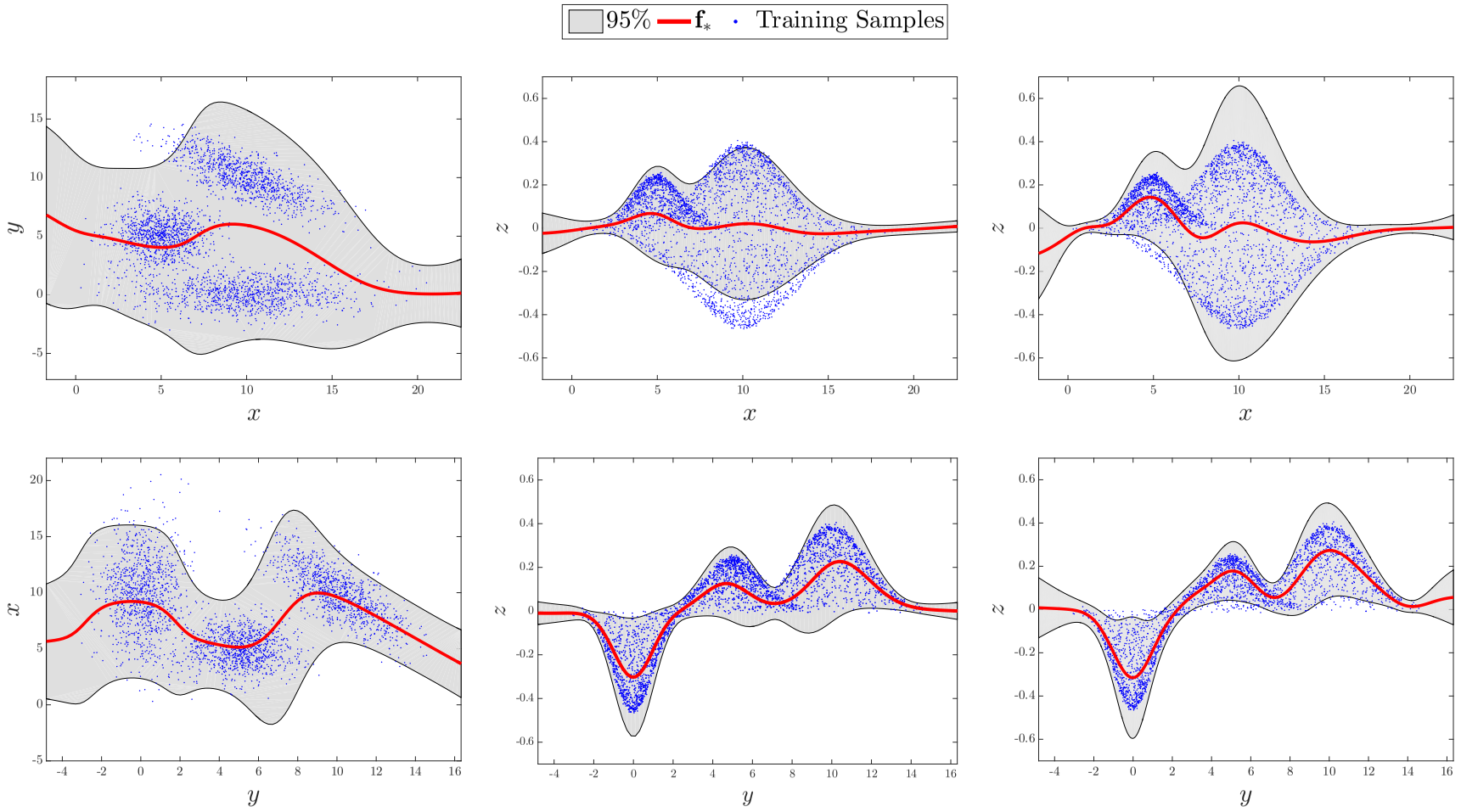
$$\begin{aligned}
\mathbb{E}\left[f(x,y)\right] &= \int \sum_{i=1}^{m} \bar{w}_i z_i \mathcal{N}\left(\begin{bmatrix} x \\ y \end{bmatrix} \middle| \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right) \mathcal{N}\left(y|\mathbb{E}\left[y|x\right], \mathbb{V}\left[y|x\right]\right) \mathrm{d}y, \\
&= \int \sum_{i=1}^{m} \bar{w}_i z_i \mathcal{N}\left(x|\boldsymbol{\mu}_i[o], \boldsymbol{\Sigma}_i[o,o]\right) \\
&\quad \times \mathcal{N}\left(y|\mathbb{E}\left[y|x,i\right], \mathbb{V}\left[y|x,i\right]\right) \mathcal{N}\left(y|\mathbb{E}\left[y|x\right], \mathbb{V}\left[y|x\right]\right) \mathrm{d}y, \\
&= \sum_{i=1}^{m} \bar{w}_i z_i \mathcal{N}\left(x|\boldsymbol{\mu}_i[o], \boldsymbol{\Sigma}_i[o,o]\right) \mathcal{N}\left(\mathbb{E}\left[y|x\right]|\mathbb{E}\left[y|x,i\right], \mathbb{V}\left[y|x,i\right] + \mathbb{V}\left[y|x\right]\right), \\
&= \sum_{i=1}^{m} \bar{w}_i z_i \mathcal{N}\left(\begin{bmatrix} x \\ \mathbb{E}\left[y|x\right] \end{bmatrix} \middle| \begin{bmatrix} \boldsymbol{\mu}_i[o] \\ \boldsymbol{\mu}_i[u] \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_i[o,o] & \boldsymbol{\Sigma}_i[o,u] \\ \boldsymbol{\Sigma}_i[u,o] & \boldsymbol{\Sigma}_i[u,u] + \mathbb{V}\left[y|x\right] \end{bmatrix}\right), \\
&= \sum_{i=1}^{m} \bar{w}_i z_i \mathcal{N}\left(\begin{bmatrix} x \\ \mathbb{E}\left[y|x\right] \end{bmatrix} \middle| \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbb{V}\left[y|x\right] \end{bmatrix}\right). \qquad (5.20)
\end{aligned}$$

Thus, predicting with a missing variable can be treated as a noisy input problem where the input is drawn from the following distribution:

$$p\left(\mathbf{x}|\mathbf{o}\right) = \mathcal{N}\left(\mathbf{x}|\bar{\mathbf{x}}, \boldsymbol{\Psi}\right), \qquad (5.21)$$

$$\bar{\mathbf{x}} = \begin{bmatrix} \mathbf{o} \\ \mathbb{E}\left[\mathbf{u}|\mathbf{o}\right] \end{bmatrix}, \boldsymbol{\Psi} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbb{V}\left[\mathbf{u}|\mathbf{o}\right] \end{bmatrix}.$$

Figure 5.2 shows the effect of applying this approach to the example of Figure 4.1, the results now are much closer to the ground truth. However, it requires us to have a probabilistic model for each possible set of missing variables, which grows exponentially large as the dimensionality of the input increases. Moreover, we make the tacit assumption that the probability density of the missing variable is unimodal and Normal, which may not be the case. For example, the probability of y given $x = 10$ has two peaks, and the overall expected value does not even pass through the training samples, hence the model is far from the ground truth in this region. Providing a richer density function should provide more accurate estimation. In order to achieve these goals in an efficient manner, we use the parameters of the trained



(a) $p(\mathbf{u}|\mathbf{o})$ (b) $p(z|\mathbf{o}, \mathbb{E}[\mathbf{u}|\mathbf{o}], \mathbb{V}[\mathbf{u}|\mathbf{o}])$ (c) $p(z|\mathbf{o})$

Figure 5.2: The result of treating the missing variable as a noisy input with probability $\mathcal{N}(\mathbf{u}|\mathbb{E}[\mathbf{u}|\mathbf{o}], \mathbb{V}[\mathbf{u}|\mathbf{o}])$ on the 2D artificial example from Figure 4.1 compared to the reference model trained only on the observed variable.

$p(z|x,y)$ model as parameters of a Gaussian mixture model (GMM), hence producing the conditional probability of any subset of the input given any other subset. If $\mathbf{x}$ is distributed as a GMM,

$$\begin{aligned} p(\mathbf{x}) &= \sum_{i=1}^{m} p(\mathbf{x}|i)p(i), \\ &= \sum_{i=1}^{m} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\, p(i), \end{aligned} \tag{5.22}$$

we can use the means $\{\boldsymbol{\mu}_i\}_{i=1}^m$ and the covariances $\{\boldsymbol{\Sigma}_i\}_{i=1}^m$ learned from fitting $p(z|x,y)$ and learn the mixture weights $\{p(i)\}_{i=1}^m$ using Expectation Maximisation (EM) (Murphy, 2012, p. 352) while holding the other parameters fixed. The probability of $\mathbf{u}$ given $\mathbf{o}$ is hence found as follows:

$$\begin{aligned} p(\mathbf{o},\mathbf{u}) &= \sum_{i=1}^{m} p(\mathbf{o},\mathbf{u}|i)\, p(i), \\ p(\mathbf{u}|\mathbf{o})\, p(\mathbf{o}) &= \sum_{i=1}^{m} p(\mathbf{u}|\mathbf{o},i)\, p(\mathbf{o}|i)p(i), \\ p(\mathbf{u}|\mathbf{o}) &= \sum_{i=1}^{m} p(\mathbf{u}|\mathbf{o},i)\, \frac{p(\mathbf{o}|i)p(i)}{p(\mathbf{o})}, \\ &= \sum_{i=1}^{m} p(\mathbf{u}|\mathbf{o},i)\, \frac{p(\mathbf{o}|i)p(i)}{\sum_{j=1}^{m} p(\mathbf{o}|j)p(j)}, \\ &= \sum_{i=1}^{m} p(\mathbf{u}|\mathbf{o},i)\, p(i|\mathbf{o}), \end{aligned} \tag{5.23}$$

where, from Equations (A.10) to (A.13), we obtain

$$p(\mathbf{o}|i) = \mathcal{N}(\mathbf{o}|\boldsymbol{\mu}_i[o], \boldsymbol{\Sigma}_i[o,o]), \tag{5.24}$$

$$p(\mathbf{u}|\mathbf{o},i) = \mathcal{N}(\mathbf{u}|\mathbb{E}[\mathbf{u}|\mathbf{o},i], \mathbb{V}[\mathbf{u}|\mathbf{o},i]), \tag{5.25}$$

$$\mathbb{E}[\mathbf{u}|\mathbf{o},i] = \boldsymbol{\mu}_i[u] + \boldsymbol{\Sigma}_i[u,o]\,\boldsymbol{\Sigma}_i[o,o]^{-1}(\mathbf{o} - \boldsymbol{\mu}_i[o]), \tag{5.26}$$

$$\mathbb{V}[\mathbf{u}|\mathbf{o},i] = \boldsymbol{\Sigma}_i[u,u] - \boldsymbol{\Sigma}_i[u,o]\,\boldsymbol{\Sigma}_i[o,o]^{-1}\,\boldsymbol{\Sigma}_i[o,u]. \tag{5.27}$$

We can also express Equation (5.23) jointly with $\mathbf{o}$, or in terms of $\mathbf{x}$, as

$$p(\mathbf{x}|\mathbf{o}) = \sum_{i=1}^{m} p(\mathbf{x}|\mathbf{o},i)\, p(i|\mathbf{o}), \tag{5.28}$$

such that

$$p\left(\mathbf{x}|\mathbf{o},i\right)=\mathcal{N}\left(\mathbf{x}|\bar{\mathbf{x}}_i,\boldsymbol{\Psi}_i\right), \tag{5.29}$$

$$\bar{\mathbf{x}}_i=\begin{bmatrix}\mathbf{o}\\ \mathbb{E}\left[\mathbf{u}|\mathbf{o},i\right]\end{bmatrix}, \tag{5.30}$$

$$\boldsymbol{\Psi}_i=\begin{bmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbb{V}\left[\mathbf{u}|\mathbf{o},i\right]\end{bmatrix}, \tag{5.31}$$

Further, if we approximate Equation (5.28) as a single Normal distribution, $p\left(\mathbf{x}|\mathbf{o}\right)\approx\mathcal{N}\left(\mathbf{x}|\mathbb{E}\left[\mathbf{x}|\mathbf{o}\right],\mathbb{V}\left[\mathbf{x}|\mathbf{o}\right]\right)$, then the overall expected value of the mixture can be computed as follows

$$\mathbb{E}\left[\mathbf{x}|\mathbf{o}\right]=\sum_{i=1}^{m}\bar{\mathbf{x}}_i p(i|\mathbf{o}), \tag{5.32}$$

and the overall covariance, from Equation (A.8), follows,

$$\mathbb{V}\left[\mathbf{x}|\mathbf{o}\right]=\mathbb{E}\left[\mathbf{x}\mathbf{x}^T|\mathbf{o}\right]-\mathbb{E}\left[\mathbf{x}|\mathbf{o}\right]\mathbb{E}\left[\mathbf{x}|\mathbf{o}\right]^T, \tag{5.33}$$

$$\mathbb{E}\left[\mathbf{x}\mathbf{x}^T|\mathbf{o}\right]=\sum_{i=1}^{m}\left(\boldsymbol{\Psi}_i+\bar{\mathbf{x}}_i\bar{\mathbf{x}}_i^T\right)p\left(i|\mathbf{o}\right). \tag{5.34}$$

However, we can use the rich probability density function of Equation (5.28) to compute a more accurate estimate

$$\begin{aligned}\mathbb{E}\left[f(\mathbf{x})\right]&=\int\sum_{i=1}^{m}\bar{w}_i z_i\mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i\right)p(\mathbf{x}|\mathbf{o})\mathrm{d}\mathbf{x},\\ &=\sum_{i=1}^{m}\bar{w}_i z_i\mathbb{E}\left[\mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i\right)|\mathbf{o}\right],\end{aligned} \tag{5.35}$$

where

$$\begin{aligned}\mathbb{E}\left[\mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}\right)|\mathbf{o}\right]&=\int\sum_{j=1}^{m}\mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}\right)\mathcal{N}\left(\mathbf{x}|\bar{\mathbf{x}}_j,\boldsymbol{\Psi}_j\right)p(j|\mathbf{o})\mathrm{d}\mathbf{x},\\ &=\sum_{j=1}^{m}\mathcal{N}\left(\bar{\mathbf{x}}_j|\boldsymbol{\mu},\boldsymbol{\Sigma}+\boldsymbol{\Psi}_j\right)p(j|\mathbf{o})\end{aligned} \tag{5.36}$$

Similarly, we can apply the same principle to obtain:

$$\mathbb{E}\left[f\left(\mathbf{x}\right)^2\right] = \sum_{i=1}^{m}\sum_{j=1}^{m} \bar{w}_i \bar{w}_j z_i z_j \mathcal{N}\left(\boldsymbol{\mu}_i | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_i + \boldsymbol{\Sigma}_j\right) \mathbb{E}\left[\mathcal{N}\left(\mathbf{x} | \mathbf{c}_{ij}, \mathbf{C}_{ij}\right) | \mathbf{o}\right], \tag{5.37}$$

$$\mathbb{E}\left[\nu^2\left(\mathbf{x}\right)\right] = \sum_{i=1}^{m}\sum_{j=1}^{m} \boldsymbol{\Sigma}_{\mathbf{w}}^{-1}[i,j] z_i z_j \mathcal{N}\left(\boldsymbol{\mu}_i | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_i + \boldsymbol{\Sigma}_j\right) \mathbb{E}\left[\mathcal{N}\left(\mathbf{x} | \mathbf{c}_{ij}, \mathbf{C}_{ij}\right) | \mathbf{o}\right], \tag{5.38}$$

$$\mathbb{E}\left[\ln \sigma^2\left(\mathbf{x}\right)\right] = \sum_{i=1}^{m} v_i z_i \mathbb{E}\left[\mathcal{N}\left(\mathbf{x} | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right) | \mathbf{o}\right], + b, \tag{5.39}$$

$$\begin{aligned}\mathbb{V}\left[\ln \sigma^2\left(\mathbf{x}\right)\right] = &\sum_{i=1}^{m}\sum_{j=1}^{m} v_i v_j z_i z_j \mathcal{N}\left(\boldsymbol{\mu}_i | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_i + \boldsymbol{\Sigma}_j\right) \mathbb{E}\left[\mathcal{N}\left(\mathbf{x} | \mathbf{c}_{ij}, \mathbf{C}_{ij}\right) | \mathbf{o}\right] \\ &- \left(\mathbb{E}\left[\ln \sigma^2\left(\mathbf{x}\right)\right] - b\right)^2.\end{aligned} \tag{5.40}$$

We compare the proposed approach with the following methods:

1. **Reference Model**: A heteroscedastic basis function model trained only on the observed variable.

2. **Random Forest**: An ensemble learning method based on aggregating the results of many learners, decision trees, trained on random sub-selection of the features and different subsamples of the data with replacement (Breiman, 2001). The subdivision of the data is referred to as bagging (Breiman, 1996), where the final prediction is computed by averaging the results of all learners, e.g. for m learners

$$f(\mathbf{x}) = \frac{1}{m}\sum_{i=1}^{m} f_i(\mathbf{x}) \tag{5.41}$$

Using bagging alone might result in finding correlated learners, restricting each learner to a sub-selection of the features helps de-correlate the learners and avoid high variance, or overfitting (Caruana and Niculescu-Mizil, 2006). The learners can be any machine learning algorithm; however, decision trees are often used due to their speed. A decision tree generates predictions by subdividing the data based on its features until a termination leaf is reached, determined using an

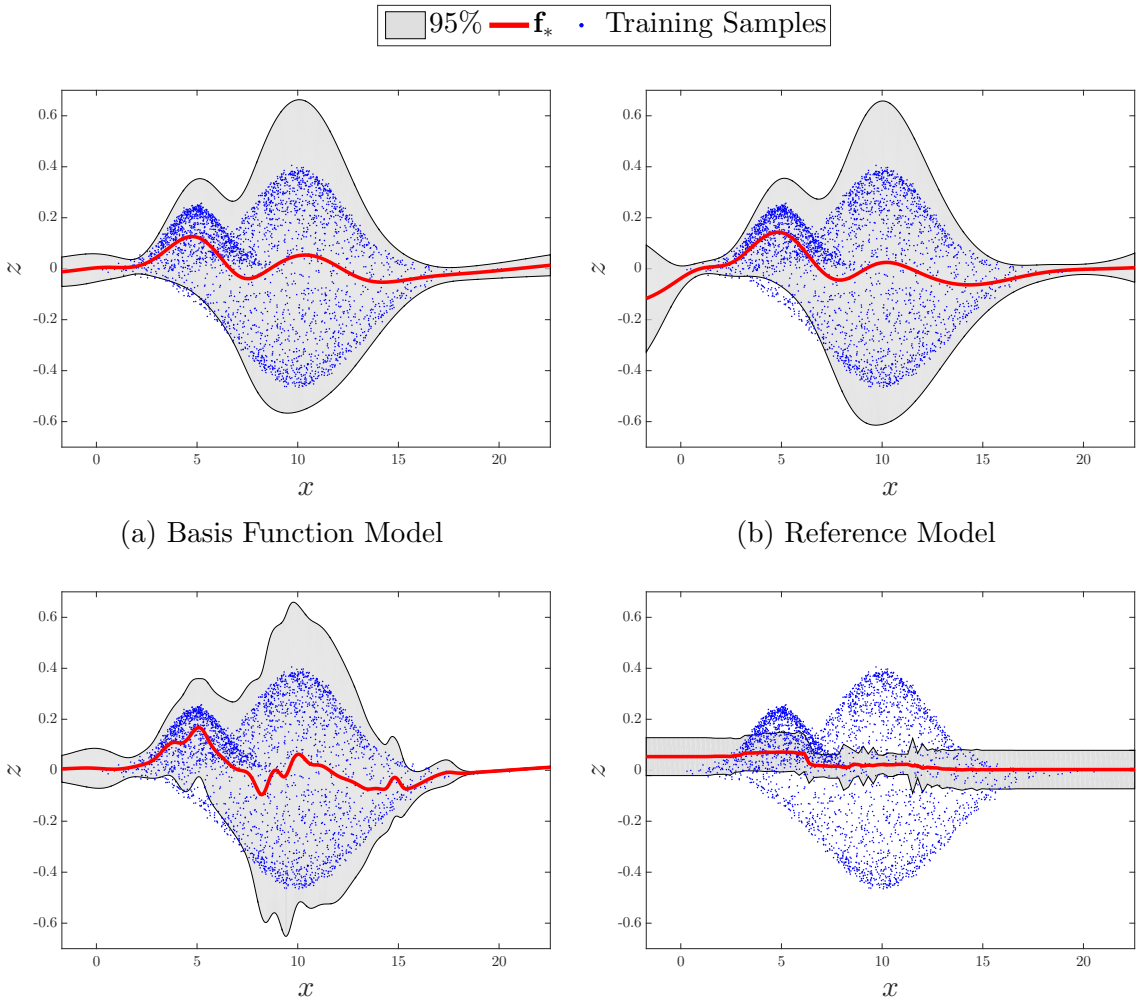
information gain metric that measures the information quality of each feature and its ability to predict the desired output. In case of missing variables, the subtrees that were trained on the observed variables will not be affected; if a subtree expect one of the missing variables, one option is to a make a decision based on the average of that variable, or some other imputation technique. An alternative method is to report the average outputs of the training samples that satisfy the conditions traversed so far.

3. **Gaussian Mixture Model**: We also use a Gaussian mixture model (GMM) to fit a joint distribution of the input and output. This method is a generalisation of the single Normal Gaussian distribution as follows:

$$p(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{m} p(\mathbf{x}, \mathbf{y}|i)p(i), \tag{5.42}$$

where each $p(\mathbf{x}, \mathbf{y}|i) = \mathcal{N}(\mathbf{x}, \mathbf{y}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$, is a joint distribution of both the input $\mathbf{x}$ and the output $\mathbf{y}$, and $\{p(i)\}_{i=1}^{m}$ are the mixture weights that are multinomial distributed such that $p(i) \geq 0 \ \forall i \in \{1, \ldots, m\}$ and $\sum_{i=1}^{m} p(i) = 1$. The parameters of the models, the m means, covariances and mixture weights, are found via the EM algorithm, which maximises the likelihood in Equation (5.42) (Murphy, 2012, p. 352). This is a maximum likelihood estimation procedure (MLE) which is prone to overfitting because it sets no prior probabilities on the parameters. The conditional probability of any missing subset of the variables given the observed set can be found in the same way described earlier in Equation (5.23).

We split the data into three sets of 1000 samples each for training, validation and testing. The training set used to optimise the models, the validation set for model selection and the testing set to report the results. To determine the number of basis functions, trees and mixtures, we train each model on the training set, with no missing variables, and select the model that best performed on the validation set as



(a) Basis Function Model

(b) Reference Model

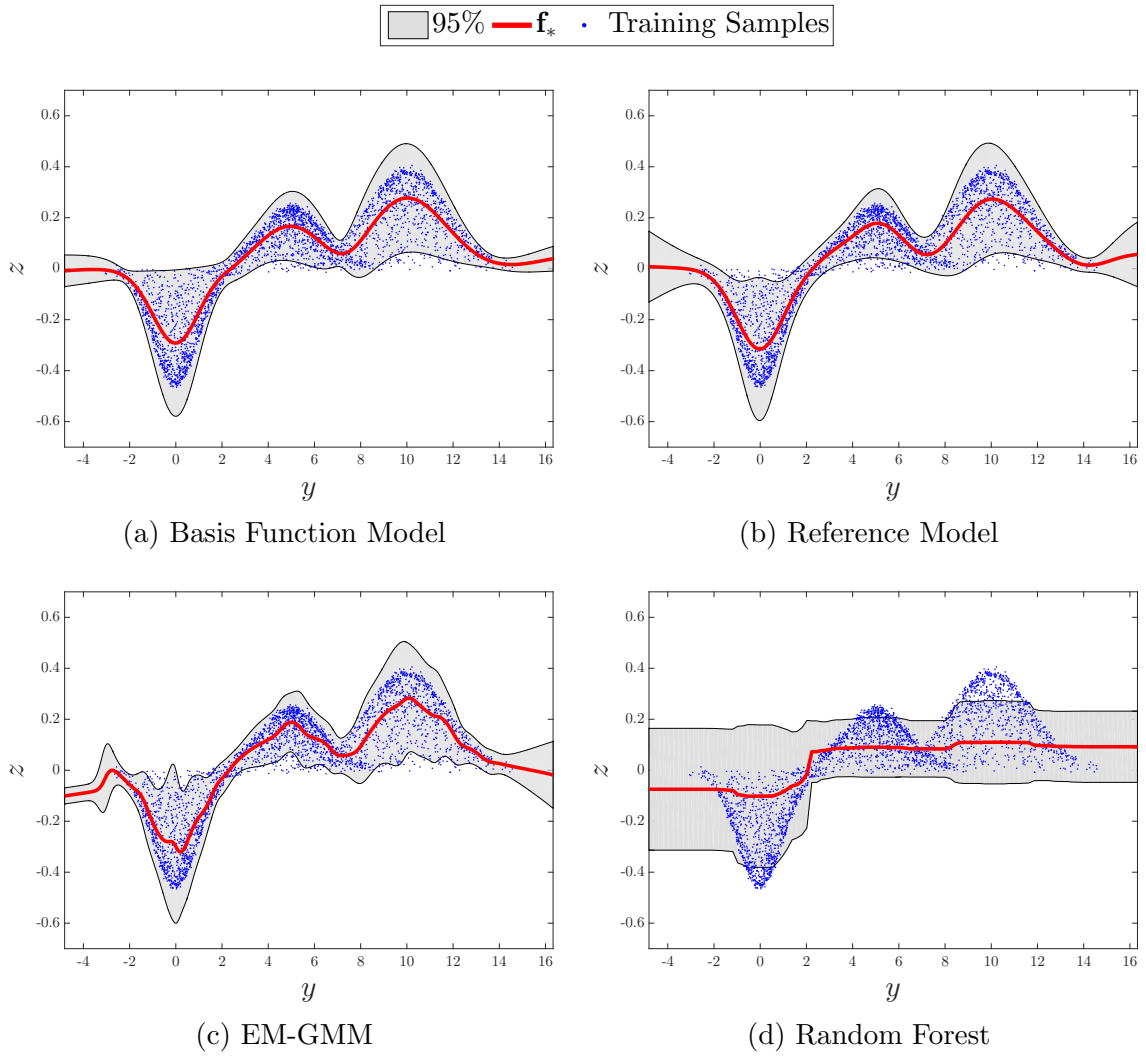
(c) EM-GMM

(d) Random Forest

Figure 5.3: The result of (a) treating the missing variable y as a noisy input, distributed as in Equation (5.23), on the 2D artificial example from Figure 4.1 compared to (b) the reference model trained only on the observed variable, (c) a Gaussian mixture model optimised using Expectation Maximisation and (d) a random forest model.

determined by the log likelihood. The validation set is also used to keep track of the best performing parameterisation of the model during the optimisation procedure. All the methods performed best around 50 basis functions/trees/mixtures. We hence fix the number to 50 for all methods and test their performance on the test set using only one observed variable. Figure 5.3 and Figure 5.4 show the results of all methods for both cases of missing variables, y and x respectively.

To evaluate the performance of the methods in recovering the reference models,



(a) Basis Function Model

(b) Reference Model

(c) EM-GMM

(d) Random Forest

Figure 5.4: The result of (a) treating the missing variable x as a noisy input, distributed as in Equation (5.23), on the 2D artificial example from Figure 4.1 compared to (b) the reference model trained only on the observed variable, (c) a Gaussian mixture model optimised using Expectation Maximisation and (d) a random forest model.

$p(z|x)$ and $p(z|y)$, from $p(z|x, y)$, we compute the average log-likelihood of the predictive distributions on the held-out test set. Each model provides a single Normal predictive distribution of the output for each sample, e.g. if y is missing $p(z_i|x_i) = \mathcal{N}\left(f\left(x_i\right), \sigma^2\left(x_i\right)\right)$. We report in Table 5.2 the mean and the standard deviation of the log-likelihood on the held-out test set for each method. The Basis Function Model (BFM) using the full mixture is the closest to the ground truth, random forest on the other hand is significantly worse than the rest. Note that in the missing y variable

Table 5.2: The mean log likelihood, plus or minus one standard deviation, of the predictive distribution for each method in handling missing values compared to the reference model on the held-out test set.

	$p(z\|x)$	$p(z\|y)$
Reference Model	0.4316±0.8591	1.2716±0.7648
BFM-Full	0.4312±0.7980	1.2664±0.7064
BFM-Single	0.1448±1.2755	1.1994±0.6694
EM-GMM	0.4179±0.8952	1.2591±0.7730
Random Forest	-20.3906±33.9660	0.4307±1.3319

case, BFM-Single is significantly worse than BFM-Full and EM-GMM compared to when x is missing. This indicates that the ability to model more complex and multimodal distributions significantly affects the performance of missing value inference. The results indicate that the proposed approach recovers the reference model accurately. Note that the EM-GMM has a higher model variance, and overfitted more, even with the use of a validation set. In the next section, we extend this approach to handle a mixture of missing and noisy variables during prediction.

5.1.3 Predicting with Missing and Noisy Variables

In the previous section, we addressed estimating the expected value in the presence of unobserved variables $\mathbf{u}$ given a *noiseless* set of observed inputs $\mathbf{o}$. The solution obtained leads to a noisy input problem where $\mathbf{x} \sim p(\mathbf{x}|\mathbf{o})$ (Equation (5.28)). For a noisy input $\mathbf{o} \sim \mathcal{N}(\mathbf{o}|\bar{\mathbf{o}}, \mathbf{O})$, we find $p(\mathbf{x}|\bar{\mathbf{o}}, \mathbf{O})$ as follows:

$$\begin{aligned} p(\mathbf{x}|\bar{\mathbf{o}}, \mathbf{O}) &= \int p(\mathbf{x}|\mathbf{o}) p(\mathbf{o}|\bar{\mathbf{o}}, \mathbf{O}) \mathrm{d}\mathbf{o}, \\ &= \sum_{i=1}^{m} \int p(\mathbf{x}|\mathbf{o}, i) p(i|\mathbf{o}) p(\mathbf{o}|\bar{\mathbf{o}}, \mathbf{O}) \mathrm{d}\mathbf{o}. \end{aligned} \tag{5.43}$$

However, the integration in Equation (5.43) is intractable. We therefore approximate it as follows:

$$\begin{aligned} p(\mathbf{x}|\bar{\mathbf{o}}, \mathbf{O}) &= \sum_{i=1}^{m} \int p(\mathbf{x}|\mathbf{o}, i) p(i|\mathbf{o}) p(\mathbf{o}|\bar{\mathbf{o}}, \mathbf{O}) \mathrm{d}\mathbf{o}, \\ &= \sum_{i=1}^{m} \mathbb{E}\left[p(\mathbf{x}|\mathbf{o}, i) p(i|\mathbf{o})\right], \\ &\approx \sum_{i=1}^{m} \mathbb{E}\left[p(\mathbf{x}|\mathbf{o}, i)\right] \mathbb{E}\left[p(i|\mathbf{o})\right], \\ &= \sum_{i=1}^{m} p(\mathbf{x}|\bar{\mathbf{o}}, \mathbf{O}, i) p(i|\bar{\mathbf{o}}, \mathbf{O}), \end{aligned} \tag{5.44}$$

i.e. we approximate the expectation of the product with the product of the expectations. For $p(i|\bar{\mathbf{o}}, \mathbf{O})$ we get

$$\begin{aligned} p(i|\bar{\mathbf{o}}, \mathbf{O}) &= \int p(i|\mathbf{o}) p(\mathbf{o}|\bar{\mathbf{o}}, \mathbf{O}) \mathrm{d}\mathbf{o}, \\ &= \int \frac{p(\mathbf{o}|i) p(i)}{\sum_{j=1}^{m} p(\mathbf{o}|j) p(j)} p(\mathbf{o}|\bar{\mathbf{o}}, \mathbf{O}) \mathrm{d}\mathbf{o}. \end{aligned} \tag{5.45}$$

This is also intractable; however, we can use multivariate Taylor series expansion to estimate a good approximation (Seltman, 2012). Let $X = p(\mathbf{o}|i)p(i)$ and $Y = \sum_{j=1}^{m} p(\mathbf{o}|j)p(j)$ be two random variables

$$\begin{aligned} p(i|\bar{\mathbf{o}}, \mathbf{O}) &= \int \frac{p(\mathbf{o}|i) p(i)}{\sum_{j=1}^{m} p(\mathbf{o}|j) p(j)} p(\mathbf{o}|\bar{\mathbf{o}}, \mathbf{O}) \mathrm{d}\mathbf{o}, \\ &= \mathbb{E}\left[\frac{X}{Y}\right], \\ &\approx \frac{\mathbb{E}\left[X\right]}{\mathbb{E}\left[Y\right]} - \frac{\mathbb{C}\left[X, Y\right]}{\mathbb{E}\left[Y\right]^2} + \frac{\mathbb{E}\left[X\right]}{\mathbb{E}\left[Y\right]^3} \mathbb{V}\left[Y\right], \end{aligned} \tag{5.46}$$

where $\mathbb{E}[X]$, $\mathbb{E}[Y]$, $\mathbb{V}[Y]$ and $\mathbb{C}[X,Y]$ are computed as follows:

$$\mathbb{E}[X] = \mathcal{N}(\bar{\mathbf{o}}|\boldsymbol{\mu}_i[o], \boldsymbol{\Sigma}_i[o,o] + \mathbf{O})\, p(i), \tag{5.47}$$

$$\mathbb{E}[Y] = \sum_{j=1}^{m} \mathcal{N}(\bar{\mathbf{o}}|\boldsymbol{\mu}_j[o], \boldsymbol{\Sigma}_j[o,o] + \mathbf{O})\, p(j), \tag{5.48}$$

$$\mathbb{E}\left[Y^2\right] = \sum_{i=1}^{m}\sum_{j=1}^{m} p(i)p(j)\mathcal{N}(\boldsymbol{\mu}_i[o]|\boldsymbol{\mu}_j[o], \boldsymbol{\Sigma}_i[o,o] + \boldsymbol{\Sigma}_j[o,o])\, \mathcal{N}(\bar{\mathbf{o}}|\mathbf{m}_{ij}, \mathbf{M}_{ij} + \mathbf{O}), \tag{5.49}$$

$$\mathbb{E}[XY] = \sum_{j=1}^{m} p(i)p(j)\mathcal{N}(\boldsymbol{\mu}_i[o]|\boldsymbol{\mu}_j[o], \boldsymbol{\Sigma}_i[o,o] + \boldsymbol{\Sigma}_j[o,o])\, \mathcal{N}(\bar{\mathbf{o}}|\mathbf{m}_{ij}, \mathbf{M}_{ij} + \mathbf{O}), \tag{5.50}$$

$$\mathbb{C}[X,Y] = \mathbb{E}[XY] - \mathbb{E}[X]\,\mathbb{E}[Y], \tag{5.51}$$

$$\mathbb{V}[Y] = \mathbb{E}\left[Y^2\right] - \mathbb{E}[Y]^2, \tag{5.52}$$

$$\mathbf{M}_{ij} = \left(\boldsymbol{\Sigma}_i[o,o]^{-1} + \boldsymbol{\Sigma}_j[o,o]^{-1}\right)^{-1}, \tag{5.53}$$

$$\mathbf{m}_{ij} = \mathbf{M}_{ij}\left(\boldsymbol{\Sigma}_i[o,o]^{-1}\boldsymbol{\mu}_i[o] + \boldsymbol{\Sigma}_j[o,o]^{-1}\boldsymbol{\mu}_j[o]\right). \tag{5.54}$$

For the product of two random variables, the Taylor series expansion to the second order of $\mathbb{E}[XY]$ is exact, and also exact to the first order if X and Y are independent. We now turn to the second term $p(\mathbf{x}|\bar{\mathbf{o}}, \mathbf{O}, i)$:

$$p(\mathbf{x}|\bar{\mathbf{o}}, \mathbf{O}, i) = \int p(\mathbf{x}|\mathbf{o}, i)p(\mathbf{o}|\bar{\mathbf{o}}, \mathbf{O})\mathrm{d}\mathbf{o}. \tag{5.55}$$

We solve for this by first computing the expected value, then from it derive the covariance as follows:

$$\begin{aligned}
\mathbb{E}[\mathbf{x}|\mathbf{o}, \mathbf{O}, i] &= \iint \mathbf{x}p(\mathbf{x}|\mathbf{o}, i)p(\mathbf{o}|\bar{\mathbf{o}}, \mathbf{O})\mathrm{d}\mathbf{x}\mathrm{d}\mathbf{o}, \\
&= \int \begin{bmatrix} \mathbf{o} \\ \mathbb{E}[\mathbf{u}|\mathbf{o}, i] \end{bmatrix} \mathcal{N}(\mathbf{o}|\bar{\mathbf{o}}, \mathbf{O})\,\mathrm{d}\mathbf{o}, \\
&= \begin{bmatrix} \int \mathbf{o}\mathcal{N}(\mathbf{o}|\bar{\mathbf{o}}, \mathbf{O})\,\mathrm{d}\mathbf{o} \\ \int \left(\boldsymbol{\mu}_i[u] + \mathbf{R}[u, o|i]\left(\mathbf{o} - \boldsymbol{\mu}_i[o]\right)\right)\mathcal{N}(\mathbf{o}|\bar{\mathbf{o}}, \mathbf{O})\,\mathrm{d}\mathbf{o} \end{bmatrix}, \\
&= \begin{bmatrix} \bar{\mathbf{o}} \\ \boldsymbol{\mu}_i[u] + \mathbf{R}[u, o|i]\left(\bar{\mathbf{o}} - \boldsymbol{\mu}_i[o]\right) \end{bmatrix}, \\
&= \begin{bmatrix} \bar{\mathbf{o}} \\ \mathbb{E}[\mathbf{u}|\bar{\mathbf{o}}, i] \end{bmatrix}.
\end{aligned} \tag{5.56}$$

where $\mathbf{R}[u, o|i]$ is a short hand notation for $\mathbf{\Sigma}_i[u, o]\mathbf{\Sigma}_i[o, o]^{-1}$. We can now find the covariance of $\mathbf{x}$ as follows

$$\begin{aligned}
\mathbb{V}\left[\mathbf{x}|\bar{\mathbf{o}}, \mathbf{O}, i\right] &= \mathbb{E}\left[\mathbf{x}\mathbf{x}^T|\bar{\mathbf{o}}, \mathbf{O}, i\right] - \mathbb{E}\left[\mathbf{x}|\bar{\mathbf{o}}, \mathbf{O}, i\right]\mathbb{E}\left[\mathbf{x}|\bar{\mathbf{o}}, \mathbf{O}, i\right]^T, \\
&= \iint \mathbf{x}\mathbf{x}^T p(\mathbf{x}|\mathbf{o}, i)\mathcal{N}\left(\mathbf{o}|\bar{\mathbf{o}}, \mathbf{O}\right) \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{o} - \mathbb{E}\left[\mathbf{x}|\bar{\mathbf{o}}, \mathbf{O}, i\right]\mathbb{E}\left[\mathbf{x}|\bar{\mathbf{o}}, \mathbf{O}, i\right]^T, \\
&= \int \left(\mathbf{\Psi}_i + \bar{\mathbf{x}}_i\bar{\mathbf{x}}_i^T\right)\mathcal{N}\left(\mathbf{o}|\bar{\mathbf{o}}, \mathbf{O}\right) \mathrm{d}\mathbf{o} - \mathbb{E}\left[\mathbf{x}|\bar{\mathbf{o}}, \mathbf{O}, i\right]\mathbb{E}\left[\mathbf{x}|\bar{\mathbf{o}}, \mathbf{O}, i\right]^T, \\
&= \mathbf{\Psi}_i + \int \left(\bar{\mathbf{x}}_i\bar{\mathbf{x}}_i^T - \mathbb{E}\left[\mathbf{x}|\bar{\mathbf{o}}, \mathbf{O}, i\right]\mathbb{E}\left[\mathbf{x}|\bar{\mathbf{o}}, \mathbf{O}, i\right]^T\right)\mathcal{N}\left(\mathbf{o}|\bar{\mathbf{o}}, \mathbf{O}\right) \mathrm{d}\mathbf{o}, \\
&= \mathbf{\Psi}_i + \int \left(\bar{\mathbf{x}}_i - \mathbb{E}\left[\mathbf{x}|\bar{\mathbf{o}}, \mathbf{O}, i\right]\right)\left(\bar{\mathbf{x}}_i + \mathbb{E}\left[\mathbf{x}|\bar{\mathbf{o}}, \mathbf{O}, i\right]\right)^T\mathcal{N}\left(\mathbf{o}|\bar{\mathbf{o}}, \mathbf{O}\right) \mathrm{d}\mathbf{o}, \\
&= \mathbf{\Psi}_i + \int \begin{bmatrix} (\mathbf{o} - \bar{\mathbf{o}}) \\ \mathbf{R}[u, o|i]\,(\mathbf{o} - \bar{\mathbf{o}}) \end{bmatrix}\begin{bmatrix} (\mathbf{o} + \bar{\mathbf{o}}) \\ \mathbf{R}[u, o|i]\,(\mathbf{o} + \bar{\mathbf{o}}) \end{bmatrix}^T\mathcal{N}\left(\mathbf{o}|\bar{\mathbf{o}}, \mathbf{O}\right) \mathrm{d}\mathbf{o}, \\
&= \mathbf{\Psi}_i + \int \begin{bmatrix} \mathbf{I}_{d_o} \\ \mathbf{R}[u, o|i] \end{bmatrix}(\mathbf{o} - \bar{\mathbf{o}})\,(\mathbf{o} + \bar{\mathbf{o}})^T\begin{bmatrix} \mathbf{I}_{d_o} \\ \mathbf{R}[u, o|i] \end{bmatrix}^T\mathcal{N}\left(\mathbf{o}|\bar{\mathbf{o}}, \mathbf{O}\right) \mathrm{d}\mathbf{o}, \\
&= \mathbf{\Psi}_i + \int \begin{bmatrix} \mathbf{I}_{d_o} \\ \mathbf{R}[u, o|i] \end{bmatrix}\left((\mathbf{o} - \bar{\mathbf{o}})\,(\mathbf{o} - \bar{\mathbf{o}})^T + 2\mathbf{o}\bar{\mathbf{o}}^T - 2\bar{\mathbf{o}}\bar{\mathbf{o}}^T\right)\begin{bmatrix} \mathbf{I}_{d_o} \\ \mathbf{R}[u, o|i] \end{bmatrix}^T \\
&\quad \times \mathcal{N}\left(\mathbf{o}|\bar{\mathbf{o}}, \mathbf{O}\right) \mathrm{d}\mathbf{o}, \\
&= \mathbf{\Psi}_i + \begin{bmatrix} \mathbf{I}_{d_o} \\ \mathbf{R}[u, o|i] \end{bmatrix}\mathbf{O}\begin{bmatrix} \mathbf{I}_{d_o} \\ \mathbf{R}[u, o|i] \end{bmatrix}^T, \\
&= \mathbf{\Psi}_i + \begin{bmatrix} \mathbf{O} & \mathbf{O}\mathbf{R}[u, o|i]^T \\ \mathbf{R}[u, o|i]\mathbf{O} & \mathbf{R}[u, o|i]\mathbf{O}\mathbf{R}[u, o|i]^T \end{bmatrix},
\end{aligned} \tag{5.57}$$

where d_o is the number of observed variables. Therefore, the probability of the input $\mathbf{x}$, with partially missing and partially noisy variables, with respect to a specific mixture i, is distributed as follows:

$$p\left(\mathbf{x}|\bar{\mathbf{o}}, \mathbf{O}, i\right) = \mathcal{N}\left(\mathbf{x}|\hat{\bar{\mathbf{x}}}_i, \hat{\mathbf{\Psi}}_i\right), \tag{5.58}$$

where

$$\hat{\bar{\mathbf{x}}}_i = \mathbb{E}\left[\mathbf{x}|\bar{\mathbf{o}}, \mathbf{O}, i\right], \tag{5.59}$$

$$\hat{\mathbf{\Psi}}_i = \mathbb{V}\left[\mathbf{x}|\bar{\mathbf{o}}, \mathbf{O}, i\right], \tag{5.60}$$

hence,

$$\mathbb{E}\left[\mathbf{x}\mathbf{x}^T|\bar{\mathbf{o}}, \mathbf{O}, i\right] = \hat{\mathbf{\Psi}}_i + \hat{\bar{\mathbf{x}}}_i\hat{\bar{\mathbf{x}}}_i^T. \tag{5.61}$$

Putting everything together, we obtain:

$$p\left(\mathbf{x}|\bar{\mathbf{o}},\mathbf{O}\right)=\sum_{i=1}^{m}p\left(\mathbf{x}|\bar{\mathbf{o}},\mathbf{O},i\right)p(i|\bar{\mathbf{o}},\mathbf{O}). \tag{5.62}$$

We can now treat the problem as one of noisy input inference where $\mathbf{x}\sim p\left(\mathbf{x}|\bar{\mathbf{o}},\mathbf{O}\right)$. We can either approximate Equation (5.62) with a single Normal distribution $p\left(\mathbf{x}|\bar{\mathbf{o}},\mathbf{O}\right)\approx \mathcal{N}\left(\mathbf{x}|\mathbb{E}\left[\mathbf{x}|\bar{\mathbf{o}},\mathbf{O}\right],\mathbb{V}\left[\mathbf{x}|\bar{\mathbf{o}},\mathbf{O}\right]\right)$, where

$$\mathbb{V}\left[\mathbf{x}|\bar{\mathbf{o}},\mathbf{O}\right]=\mathbb{E}\left[\mathbf{x}\mathbf{x}^{T}|\bar{\mathbf{o}},\mathbf{O}\right]-\mathbb{E}\left[\mathbf{x}|\bar{\mathbf{o}},\mathbf{O}\right]\mathbb{E}\left[\mathbf{x}|\bar{\mathbf{o}},\mathbf{O}\right],^{T} \tag{5.63}$$

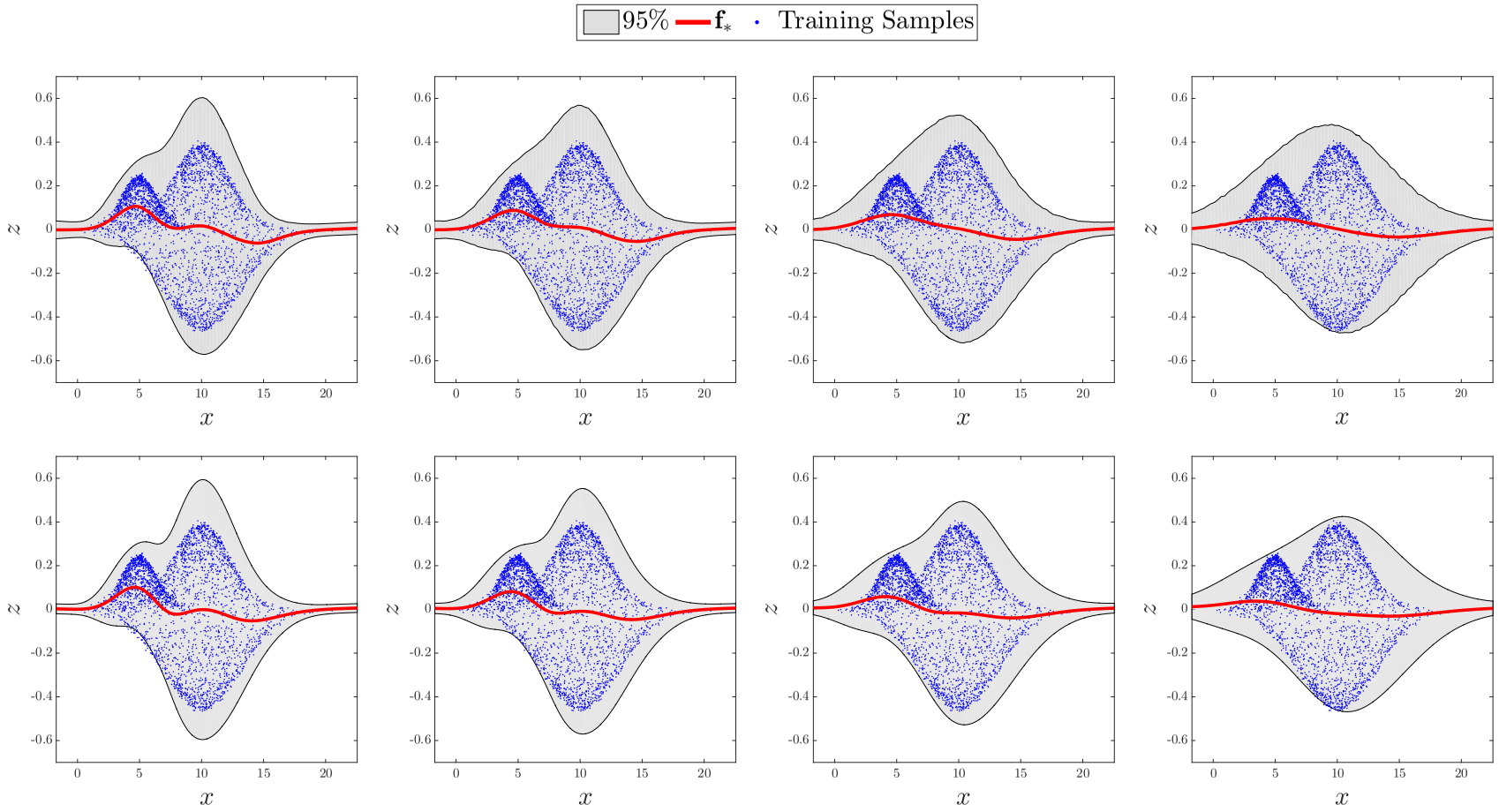
$$\mathbb{E}\left[\mathbf{x}|\bar{\mathbf{o}},\mathbf{O}\right]=\sum_{i=1}^{m}\hat{\bar{\mathbf{x}}}_{i}p(i|\bar{\mathbf{o}},\mathbf{O}), \tag{5.64}$$

$$\mathbb{E}\left[\mathbf{x}\mathbf{x}^{T}|\bar{\mathbf{o}},\mathbf{O}\right]=\sum_{i=1}^{m}\left(\hat{\mathbf{\Psi}}_{i}+\hat{\bar{\mathbf{x}}}_{i}\hat{\bar{\mathbf{x}}}_{i}^{T}\right)p(i|\bar{\mathbf{o}},\mathbf{O}), \tag{5.65}$$

or use the full mixture. This will result in updating $\mathbb{E}\left[f(\mathbf{x})\right]$, $\mathbb{E}\left[f(\mathbf{x})^{2}\right]$, $\mathbb{E}\left[\nu^{2}(\mathbf{x})\right]$, $\mathbb{E}\left[\ln\sigma^{2}(\mathbf{x})\right]$ and $\mathbb{V}\left[\ln\sigma^{2}(\mathbf{x})\right]$ by replacing the term $\mathbb{E}\left[\mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}_{i},\boldsymbol{\Sigma}_{i}\right)|\mathbf{o}\right]$ in Equation (5.36) with

$$\mathbb{E}\left[\mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}\right)|\bar{\mathbf{o}},\mathbf{O}\right]=\sum_{i=1}^{m}\mathcal{N}\left(\hat{\bar{\mathbf{x}}}_{i}|\boldsymbol{\mu},\boldsymbol{\Sigma}+\hat{\mathbf{\Psi}}_{i}\right)p(i|\bar{\mathbf{o}},\mathbf{O}). \tag{5.66}$$

We demonstrate the effectiveness of this approximation on the same data from Figure 4.1. A model is trained to fit $p(z|x,y)$, then during prediction is used to estimate $p(z|x)$, where $x\sim\mathcal{N}\left(x,\psi\right)$. The number of test samples is 100, uniformly spaced from -5 to 18, and we apply the same degree of input noise variance, ψ, to all test cases. We use as a reference model a sampling technique, where the distribution $p(z|x)$ is estimated by averaging 5000 estimates using inputs sampled from $\mathcal{N}\left(x,\psi\right)$ for each test case; each one of which is treated as a missing variable problem with noiseless observed inputs. The values of $\mathbb{E}\left[f\left(x\right)\right]$, $\mathbb{V}\left[f\left(x\right)\right]$, $\mathbb{E}\left[\sigma^{2}\left(x\right)\right]$ and $\mathbb{E}\left[\nu^{2}\left(x\right)\right]$ are then estimated from the 5000 generated predictions of $f\left(x\right)$, $\sigma^{2}\left(x\right)$ and $\nu^{2}\left(x\right)$ respectively. The comparison between the proposed approximation and the sampling method is shown in Figure 5.5 with different values of ψ.



(a) $\psi = 1$ (b) $\psi = 2$ (c) $\psi = 4$ (d) $\psi = 8$

Figure 5.5: A comparison between a sampling method (top) and the proposed approximation in this section (bottom) in predicting with a missing y variable and a noisy x variable with probability $\mathcal{N}(x, \psi)$ on the example from Figure 4.1. The variance ψ is varied from (a) 1 to (d) 8 (in multiples of 2).

Table 5.3: The mean z-scores, plus or minus one standard deviation, for $\mathbb{E}[f(x,y)]$, $\mathbb{E}[f(x,y)^2]$, $\mathbb{E}[\nu^2(x,y)]$ and $\mathbb{E}[\sigma^2(x,y)]$ based on the analytical method described in this section for noisy and missing input and their estimated values based on a sampling method.

	x is missing	y is missing
$\mathbb{E}[f(x,y)]$	0.21±0.25	0.0088±0.11
$\mathbb{E}[f(x,y)^2]$	0.26±0.25	0.11±0.15
$\mathbb{E}[\nu^2(x,y)]$	0.29±0.30	0.18±0.17
$\mathbb{E}\left[\sigma^2(x,y) = (\boldsymbol{\phi}(x,y)^T \mathbf{v} + b)^2\right]$	0.21±0.22	0.19±0.34
$\mathbb{E}\left[\sigma^2(x,y) = \exp(\boldsymbol{\phi}(x,y)^T \mathbf{v} + b)\right]$	0.26±0.28	0.14±0.21

To evaluate the performance of this approximation, we use the same models from the previous section, optimised on the full noiseless training set, then we add random noise, $\epsilon_i \sim \mathcal{N}(0, \psi_i)$, to each sample in the test set, where $\psi_i \sim \text{Gamma}(\psi_i|\alpha = 1, \beta = 2)$, so that ψ_i has a mean of 0.5 and a variance of 0.25. As a baseline comparison, we sampled 1000 points from the distribution $\mathcal{N}(x_i, \psi_i)$, for each sample i, in order to compare the sample statistics with the analytical computation described in this section. The average z-scores, plus or minus one standard deviation, of the proposed method compared to the sampling technique are shown in Table 5.3. The results indicate that this approach achieves an acceptable approximation to the sampling method with an average z-score deviation of ~ 0.25 when x is missing and ~ 0.13 when y is missing. The area between -0.25 and 0.25 z-scores in a Normal distribution is less than 20%.

5.2 Training

5.2.1 Training with Noisy Variables

We now consider optimisation in the presence of known input noise. Let $\mathbf{x}_i \sim \mathcal{N}(\mathbf{x}_i|\bar{\mathbf{x}}_i, \boldsymbol{\Psi}_i)$, $\overline{\mathbf{X}} = \{\bar{\mathbf{x}}_i\}_{i=1}^n$ and $\boldsymbol{\Psi} = \{\boldsymbol{\Psi}_i\}_{i=1}^n$. The most direct approach is to use $\overline{\mathbf{X}}$ in place of $\mathbf{X}$, i.e. to maximise the objective at the expected values of the inputs.

However, this has proven ineffective during prediction. More precisely, approximating $\mathbb{E}\left[f\left(X\right)\right]$ with $f\left(\mathbb{E}\left[X\right]\right)$ is equivalent to a first order approximation using a Taylor series expansion. Ideally we would like to marginalise out the input

$$p\left(\mathbf{y}|\overline{\mathbf{X}}, \boldsymbol{\Psi}, \boldsymbol{\theta}\right) = \int p\left(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}\right) p\left(\mathbf{X}|\overline{\mathbf{X}}, \boldsymbol{\Psi}\right) \mathrm{d}\mathbf{X}, \tag{5.67}$$

$$p\left(\mathbf{X}|\overline{\mathbf{X}}, \boldsymbol{\Psi}\right) = \prod_{i=1}^{n} \mathcal{N}\left(\mathbf{x}_i|\bar{\mathbf{x}}_i, \boldsymbol{\Psi}_i\right). \tag{5.68}$$

This integration is intractable however, so we must again resort to approximations. The approach we use here is to use the expected value of the highest order function. For example, if $f(x)$ can be expressed as $f(g(z(x)))$, one extreme is to use $\mathbb{E}\left[f(g(z(x)))\right]$ (intractable), and the other extreme is to approximate it with $f(g(z(\mathbb{E}\left[x\right])))$ (inaccurate). We therefore aim for the highest function in the chain that we can approximate. Consider again the form of the marginal likelihood in Equation (2.56):

$$\ln p\left(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}\right) = -\frac{1}{2}\mathbf{y}^T\mathbf{S}^{-1}\mathbf{y} - \frac{1}{2}\left|\mathbf{S}\right| - \frac{n}{2}\ln\left(2\pi\right), \tag{5.69}$$

$$\mathbf{S} = \boldsymbol{\Phi}_{\mathbf{x}}{}^T\mathbf{A}^{-1}\boldsymbol{\Phi}_{\mathbf{x}} + \sigma^2\mathbf{I}_n. \tag{5.70}$$

We can express $\ln p\left(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}\right)$ as a function of $\mathbf{S}$ and parameterised by $\boldsymbol{\theta}$, i.e. $f\left(\mathbf{S}|\boldsymbol{\theta}\right)$, then approximate it as follows $f\left(\mathbf{S}|\boldsymbol{\theta}\right) \approx f\left(\mathbb{E}\left[\mathbf{S}\right]|\boldsymbol{\theta}\right)$, where

$$\begin{aligned}
\mathbb{E}\left[\mathbf{S}[i,j]\right] &= \iint \mathbf{S}[i,j] p\left(\mathbf{x}_i\right) p\left(\mathbf{x}_j\right) \mathrm{d}\mathbf{x}_i \mathrm{d}\mathbf{x}_j, \\
&= \iint \sum_{k=1}^{m} \alpha_k^{-1} \phi_k\left(\mathbf{x}_i\right) \phi_k\left(\mathbf{x}_j\right) \mathcal{N}\left(\mathbf{x}_i|\bar{\mathbf{x}}_i, \boldsymbol{\Psi}_i\right) \mathcal{N}\left(\mathbf{x}_j|\bar{\mathbf{x}}_j, \boldsymbol{\Psi}_j\right) \mathrm{d}\mathbf{x}_i \mathrm{d}\mathbf{x}_j, \\
&= \sum_{k=1}^{m} \alpha_k^{-1} \bar{\phi}_k\left(\bar{\mathbf{x}}_i, \boldsymbol{\Psi}_i\right) \bar{\phi}_k\left(\bar{\mathbf{x}}_j, \boldsymbol{\Psi}_j\right),
\end{aligned} \tag{5.71}$$

$$\bar{\phi}_k\left(\bar{\mathbf{x}}_i, \boldsymbol{\Psi}_i\right) = \sqrt{\frac{\left|\boldsymbol{\Sigma}_k\right|}{\left|\boldsymbol{\Sigma}_k + \boldsymbol{\Psi}_i\right|}} \exp\left(-\frac{1}{2}\left(\bar{\mathbf{x}}_i - \boldsymbol{\mu}_k\right)^T\left(\boldsymbol{\Sigma}_k + \boldsymbol{\Psi}_i\right)^{-1}\left(\bar{\mathbf{x}}_i - \boldsymbol{\mu}_k\right)\right). \tag{5.72}$$

Therefore, we can express the full matrix $\mathbb{E}\left[\mathbf{S}\right]$ as follows:

$$\mathbb{E}\left[\mathbf{S}\right] = \bar{\boldsymbol{\Phi}}_{\mathbf{x}}^T\mathbf{A}^{-1}\bar{\boldsymbol{\Phi}}_{\mathbf{x}} + \sigma^2\mathbf{I}_n, \tag{5.73}$$

$$\bar{\boldsymbol{\Phi}}_{\mathbf{x}}[j,i] = \bar{\phi}_j\left(\bar{\mathbf{x}}_i, \boldsymbol{\Psi}_i\right). \tag{5.74}$$

This has transformed the problem into a new basis function model with a new basis function definition as described in Equation (5.72), where $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j = \left(\boldsymbol{\Gamma}_j^T \boldsymbol{\Gamma}_j\right)^{-1}$ are the centre and the covariance of the j-th basis function respectively. The new basis function we refer to as the expected RBF or (ERBF), is equivalent to the RBF kernel when $\boldsymbol{\Psi}_i = \mathbf{0}$, i.e. $\bar{\phi}_j\left(\bar{\mathbf{x}}_i, \mathbf{0}\right) = \phi_j\left(\bar{\mathbf{x}}_i\right)$, which plays the role of down-weighting the sample for large values of $|\boldsymbol{\Psi}_i|$, approaching zero as $|\boldsymbol{\Psi}_i| \to \infty$; hence, its gradient will approach zero which will eliminate its contribution to the learning process. The ratio $|\boldsymbol{\Sigma}_j| \,/\, |\boldsymbol{\Sigma}_j + \boldsymbol{\Psi}_i|$ is bounded between zero and one (see Equation (A.29)) since both $\boldsymbol{\Sigma}_j$ and $\boldsymbol{\Psi}_i$ are PSD matrices. This is a desirable effect so that it will not conflict with our prior on the weights. Without priors on the hyper-parameters, the optimisation will increase the magnitude of the basis function response in order to compensate for any reduction in its corresponding weight. Having the basis function bounded helps restrict such unwanted effect. Moreover, for a basis function model with a prior mean function, the predictive mean approaches the prior mean as the magnitude of the input noise increases. We see this as a desirable effect as the model falls back to its prior as the uncertainty about the input increases.

The gradients now must be updated to account for the new input noise term. We only need however to provide the gradient of the basis function with respect to its logarithm. For a specific sample i and basis function j, we have

$$\begin{aligned}
\ln \bar{\boldsymbol{\Phi}}_{\mathbf{x}}[j,i] &= \frac{1}{2}\ln|\boldsymbol{\Sigma}_j| - \frac{1}{2}\ln|\boldsymbol{\Sigma}_j + \boldsymbol{\Psi}_i| \\
&\quad - \frac{1}{2}\left(\bar{\mathbf{x}}_i - \boldsymbol{\mu}_j\right)^T \left(\boldsymbol{\Sigma}_j + \boldsymbol{\Psi}_i\right)^{-1} \left(\bar{\mathbf{x}}_i - \boldsymbol{\mu}_j\right), && (5.75)\\
\frac{\partial \ln \bar{\boldsymbol{\Phi}}_{\mathbf{x}}[j,i]}{\partial \boldsymbol{\mu}_j} &= \left(\boldsymbol{\Sigma}_j + \boldsymbol{\Psi}_i\right)^{-1} \left(\bar{\mathbf{x}}_i - \boldsymbol{\mu}_j\right), && (5.76)\\
\frac{\partial \ln \bar{\boldsymbol{\Phi}}_{\mathbf{x}}[j,i]}{\partial \boldsymbol{\Sigma}_j} &= \frac{1}{2}\boldsymbol{\Sigma}_j^{-1} - \frac{1}{2}\left(\boldsymbol{\Sigma}_j + \boldsymbol{\Psi}_i\right)^{-1} \\
&\quad + \frac{1}{2}\left(\boldsymbol{\Sigma}_j + \boldsymbol{\Psi}_i\right)^{-1} \left(\bar{\mathbf{x}}_i - \boldsymbol{\mu}_j\right)\left(\bar{\mathbf{x}}_i - \boldsymbol{\mu}_j\right)^T \left(\boldsymbol{\Sigma}_j + \boldsymbol{\Psi}_i\right)^{-1}, && (5.77)\\
\frac{\partial \ln \bar{\boldsymbol{\Phi}}_{\mathbf{x}}[j,i]}{\partial \boldsymbol{\Gamma}_j} &= -2\boldsymbol{\Gamma}_j \boldsymbol{\Sigma}_j \frac{\partial \ln \bar{\boldsymbol{\Phi}}_{\mathbf{x}}[j,i]}{\partial \boldsymbol{\Sigma}_j} \boldsymbol{\Sigma}_j. && (5.78)
\end{aligned}$$

Alternatively, one can also optimise with respect to the covariance matrix $\boldsymbol{\Sigma}_j = \boldsymbol{\Lambda}_j^T \boldsymbol{\Lambda}_j$, where

$$\frac{\partial \ln \bar{\boldsymbol{\Phi}}_{\mathbf{x}}[j,i]}{\partial \boldsymbol{\Lambda}_j} = 2\boldsymbol{\Lambda}_j \frac{\partial \ln \bar{\boldsymbol{\Phi}}_{\mathbf{x}}[j,i]}{\partial \boldsymbol{\Sigma}_j}. \tag{5.79}$$

In order to evaluate the performance of this model we compare the proposed approach with the following five models:

1. **Deterministic**: Treating the uncertain inputs as if they were exact.

2. **Sampling**: We create a new dataset by sampling 100 points from the distribution $\mathcal{N}(x_i, \psi_i)$ for each input and assign them the same target y_i.

3. **Noise as input**: The input noise is treated as an additional input dimension.

4. **Noise as cost**: We use case-sensitive learning where the input noise variance is treated as cost, i.e. $\omega_i = 1/\sqrt{\psi_i}$.

5. **Noisy input Gaussian process (NIGP)**: A Gaussian process framework to deal with input noise (Mchutchon and Rasmussen, 2011). This assumes that all inputs are corrupted by a random noise process drawn from a single shared distribution which is not known a priori.

NIGP uses a Taylor series expansion to approximate the noisy input problem. Consider for example that $\mathbf{x} = \bar{\mathbf{x}} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(0|\boldsymbol{\Psi})$, then the expansion of $f(\mathbf{x})$

$$f(\mathbf{x}) = f(\bar{\mathbf{x}} + \boldsymbol{\epsilon}) \approx f(\bar{\mathbf{x}}) + \boldsymbol{\epsilon}^T \frac{\partial f(\bar{\mathbf{x}})}{\partial \bar{\mathbf{x}}}. \tag{5.80}$$

This will result in changing the likelihood of the GP to

$$p(\mathbf{y}|\mathbf{f}_{\mathbf{x}}) = \mathcal{N}\left(\mathbf{y}|\mathbf{f}_{\mathbf{x}}, \sigma^2 \mathbf{I}_n + \operatorname{diag}\left[\boldsymbol{\nabla}_{\mathbf{x}}^T \boldsymbol{\Psi} \boldsymbol{\nabla}_{\mathbf{x}}\right]\right), \tag{5.81}$$

where $\boldsymbol{\nabla}_{\mathbf{x}} = \left[\frac{\partial f(\bar{\mathbf{x}}_1)}{\partial \bar{\mathbf{x}}_1}, \ldots, \frac{\partial f(\bar{\mathbf{x}}_n)}{\partial \bar{\mathbf{x}}_n}\right] \in \mathbb{R}^{d \times n}$. Similar to a heteroscedastic model, this treatment adds an extra input-dependent variance in the likelihood, which is the

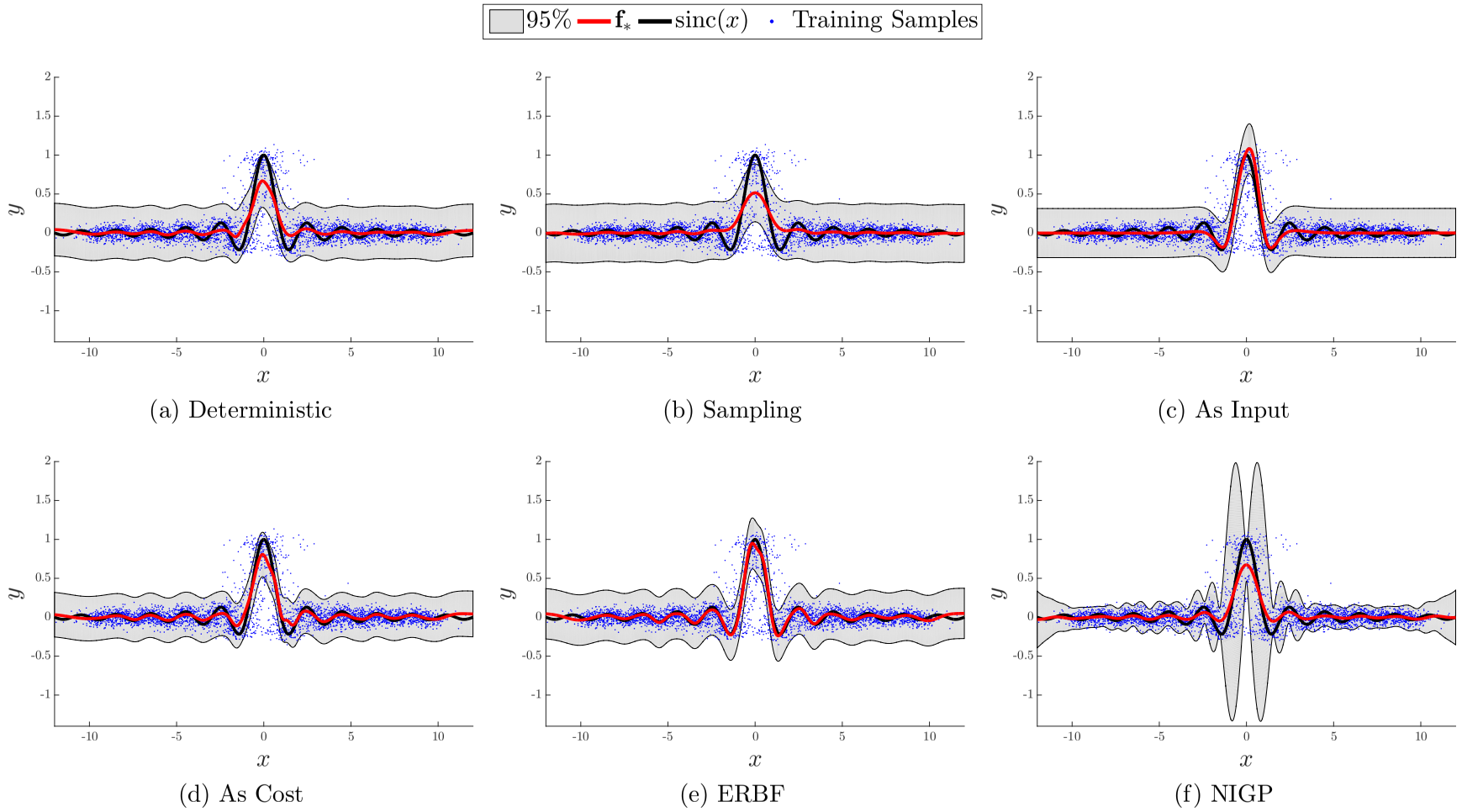
Table 5.4: The RMSE between the tested models and the true underlying function using both homoscedastic and heteroscedastic variance estimations. The results are reported on 1000 test samples that are linearly spaced across the domain of the input.

	Deterministic	Sampling	As Input	As Cost	ERBF	NIGP
Homoscedastic	0.0830	0.1110	0.0547	0.0543	0.0299	0.0757
Heteroscedastic	0.0829	0.1109	0.0523	0.0516	0.0211	N/A

product of the input's noise variance and its gradient. Thus, as the input noise increases, extra uncertainty will be added to the total variance that is proportional to the gradient of the function at that location. Therefore, in areas that are relatively flat, the input noise will have little effect on the inference. The covariance $\mathbf{\Psi}$ in NIGP is considered unknown, diagonal and shared for all inputs. In addition to using homoscedastic models, we also fit heteroscedastic versions for all the methods, except for NIGP, on the same dataset used in Section 5.1.1 and all using the RBF kernel. The results are shown in Figures 5.6 and 5.7 respectively. Performances in terms of the differences between the predictions and the true function, $\text{sinc}(x)$, are reported in Table 5.4. The results are reported on 1000 samples that are linearly spaced to cover the domain of the training set. The results from Figure 5.6, Figure 5.6 and Table 5.4 indicate that the proposed approach significantly and consistently outperforms all the tested methods.

5.2.2 Training with Missing and Noisy Variables

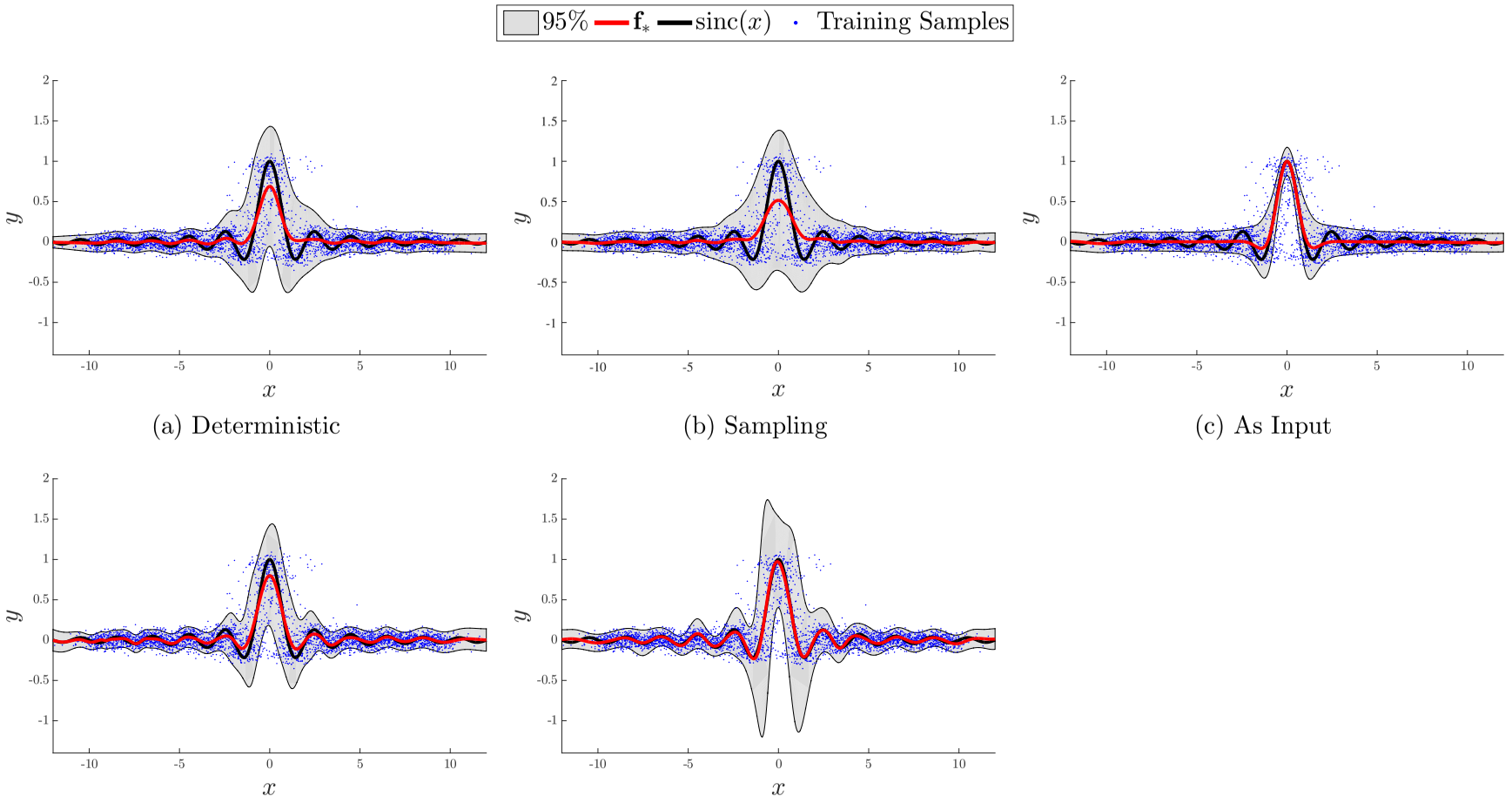
We can extend our process for noisy inputs to the missing variable case by treating the missing variables as noisy inputs drawn from the conditional distribution of the missing given the observed. We have shown that using a basis function model with a RBF kernel has the advantage of utilising its learned parameters to model a distribution over inputs as a Gaussian mixture model. For the training case however, these parameters, by definition, are yet to be inferred; thus, attempting to learn and use



(a) Deterministic (b) Sampling (c) As Input

(d) As Cost (e) ERBF (f) NIGP

Figure 5.6: Comparisons between six different methods for training with input noise and homoscedastic output noise, (a) a deterministic model, (b) a sampling method where we draw 100 samples from each input distribution, (c) treating the noise as input, (d) training with cost-sensitive learning such that $\omega_i = 1/\sqrt{\psi_i}$, (e) using the ERBF method and (f) using NIGP, all using 100 RBFs.



(a) Deterministic (b) Sampling (c) As Input

(d) As Cost (e) ERBF

Figure 5.7: Comparisons between five different methods for training with input noise and heteroscedastic output noise, (a) a deterministic model, (b) a sampling method where we draw 100 samples from each input distribution, (c) treating the noise as input, (d) training with cost-sensitive learning such that $\omega_i = 1/\sqrt{\psi_i}$ and (e) with ERBF, all using 100 RBFs.

them simultaneously for missing value inference is not only computationally expensive but extremely sensitive to initial conditions. In this section, we provide a method that has proved effective and does not add to the complexity cost or the gradient calculations. In the previous section we showed that, for a noisy input, the problem transforms into a new basis function model definition. The new basis function is the expected response value given a distribution over the input. For a partially complete and noisy input $\mathbf{x}$, where the observed variable $\mathbf{o} \sim p(\mathbf{o}|\bar{\mathbf{o}}, \mathbf{O})$, we can replace that prior with the distribution from Equation (5.62). The new expected RBF will be updated as follows:

$$\begin{aligned}
\bar{\phi}_i(\bar{\mathbf{o}}, \mathbf{O}) &= \mathbb{E}\left[\phi_i(\mathbf{x})|\bar{\mathbf{o}}, \mathbf{O}\right], \\
&= \int z_i \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\, p(\mathbf{x}|\bar{\mathbf{o}}, \mathbf{O}) \mathrm{d}\mathbf{x}, \\
&= \int z_i \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \sum_{j=1}^{m} \mathcal{N}\left(\mathbf{x}|\hat{\bar{\mathbf{x}}}_j, \hat{\boldsymbol{\Psi}}_j\right) p(j|\bar{\mathbf{o}}, \mathbf{O}) \mathrm{d}\mathbf{x}, \\
&= \sum_{j=1}^{m} z_i \mathcal{N}\left(\hat{\bar{\mathbf{x}}}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i + \hat{\boldsymbol{\Psi}}_j\right) p(j|\bar{\mathbf{o}}, \mathbf{O}).
\end{aligned} \tag{5.82}$$

However, the parameters of the model have not been optimised and the computational cost has increased by a factor of m. The approximation we hence use replaces the full mixture with its i-th component

$$\begin{aligned}
\bar{\phi}_i(\bar{\mathbf{o}}, \mathbf{O}) &\approx z_i \mathcal{N}\left(\hat{\bar{\mathbf{x}}}_i|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i + \hat{\boldsymbol{\Psi}}_i\right), \\
&= z_i \mathcal{N}\left(\hat{\bar{\mathbf{x}}}_i[o]|\boldsymbol{\mu}_i[o], \boldsymbol{\Sigma}_i[o,o] + \hat{\boldsymbol{\Psi}}_i[o,o]\right) \\
&\quad \times \mathcal{N}\left(\hat{\bar{\mathbf{x}}}_i[u]|\mathbb{E}\left[\hat{\bar{\mathbf{x}}}_i[u]|\hat{\bar{\mathbf{x}}}_i[o]\right], \mathbb{V}\left[\hat{\bar{\mathbf{x}}}_i[u]|\hat{\bar{\mathbf{x}}}_i[o]\right]\right),
\end{aligned} \tag{5.83}$$

where, from Equations (5.59) and (5.60), we obtain

$$\begin{aligned}
\mathbb{E}\left[\hat{\hat{\mathbf{x}}}_i[u]|\hat{\hat{\mathbf{x}}}_i[o]\right] &= \boldsymbol{\mu}_i[u] + \left(\boldsymbol{\Sigma}_i[u,o] + \hat{\boldsymbol{\Psi}}_i[u,o]\right)\left(\boldsymbol{\Sigma}_i[o,o] + \hat{\boldsymbol{\Psi}}_i[o,o]\right)^{-1}(\bar{\mathbf{o}} - \boldsymbol{\mu}_i[o]), \\
&= \boldsymbol{\mu}_i[u] + \left(\boldsymbol{\Sigma}_i[u,o] + \boldsymbol{\Sigma}_i[u,o]\boldsymbol{\Sigma}_i[o,o]^{-1}\mathbf{O}\right)\left(\boldsymbol{\Sigma}_i[o,o] + \mathbf{O}\right)^{-1}(\bar{\mathbf{o}} - \boldsymbol{\mu}_i[o]), \\
&= \boldsymbol{\mu}_i[u] + \boldsymbol{\Sigma}_i[u,o]\boldsymbol{\Sigma}_i[o,o]^{-1}\left(\boldsymbol{\Sigma}_i[o,o] + \mathbf{O}\right)\left(\boldsymbol{\Sigma}_i[o,o] + \mathbf{O}\right)^{-1}(\bar{\mathbf{o}} - \boldsymbol{\mu}_i[o]), \\
&= \boldsymbol{\mu}_i[u] + \boldsymbol{\Sigma}_i[u,o]\boldsymbol{\Sigma}_i[o,o]^{-1}(\bar{\mathbf{o}} - \boldsymbol{\mu}_i[o]), \\
&= \hat{\hat{\mathbf{x}}}_i[u], \qquad (5.84)
\end{aligned}$$

and

$$\begin{aligned}
\mathbb{V}\left[\hat{\hat{\mathbf{x}}}_i[u]|\hat{\hat{\mathbf{x}}}_i[o]\right] &= \boldsymbol{\Sigma}_i[u,u] + \hat{\boldsymbol{\Psi}}_i[u,u] \\
&\quad - \left(\boldsymbol{\Sigma}_i[u,o] + \hat{\boldsymbol{\Psi}}_i[u,o]\right)\left(\boldsymbol{\Sigma}_i[o,o] + \hat{\boldsymbol{\Psi}}_i[o,o]\right)^{-1}\left(\boldsymbol{\Sigma}_i[o,u] + \hat{\boldsymbol{\Psi}}_i[o,u]\right), \\
&= \boldsymbol{\Sigma}_i[u,u] + \hat{\boldsymbol{\Psi}}_i[u,u] - \boldsymbol{\Sigma}_i[u,o]\boldsymbol{\Sigma}_i[o,o]^{-1}\left(\boldsymbol{\Sigma}_i[o,u] + \hat{\boldsymbol{\Psi}}_i[o,u]\right), \\
&= \boldsymbol{\Sigma}_i[u,u] + \hat{\boldsymbol{\Psi}}_i[u,u] - \mathbf{R}[u,o|i]\boldsymbol{\Sigma}_i[o,u] - \mathbf{R}[u,o|i]\mathbf{O}\mathbf{R}[u,o|i]^T, \\
&= \mathbb{V}\left[\mathbf{u}|\bar{\mathbf{o}},\mathbf{O},i\right] + \hat{\boldsymbol{\Psi}}_i[u,u] - \mathbf{R}[u,o|i]\mathbf{O}\mathbf{R}[u,o|i]^T, \\
&= 2\mathbb{V}\left[\mathbf{u}|\bar{\mathbf{o}},\mathbf{O},i\right]. \qquad (5.85)
\end{aligned}$$

We can thus ignore the exponent of the Gaussian PDF with respect to $\hat{\hat{\mathbf{x}}}_i[u]$ in Equation (5.83) since $\mathbb{E}\left[\hat{\hat{\mathbf{x}}}_i[u]|\hat{\hat{\mathbf{x}}}_i[o]\right] - \hat{\hat{\mathbf{x}}}_i[u]$ is always zero. Using Equation (A.27) in the appendix, we can further simplify Equation (5.83) to:

$$\begin{aligned}
\bar{\phi}_i(\bar{\mathbf{o}},\mathbf{O}) &\approx \frac{z_i\mathcal{N}\left(\mathbf{o}|\boldsymbol{\mu}_i[o],\boldsymbol{\Sigma}_i[o,o]+\mathbf{O}\right)}{\sqrt{(2\pi)^{d_u}\left|2\left(\boldsymbol{\Sigma}_i[u,u] - \boldsymbol{\Sigma}_i[u,o]\boldsymbol{\Sigma}_i[o,o]^{-1}\boldsymbol{\Sigma}_i[o,u]\right)\right|}}, \\
&= \frac{z_i\mathcal{N}\left(\mathbf{o}|\boldsymbol{\mu}_i[o],\boldsymbol{\Sigma}_i[o,o]+\mathbf{O}\right)}{\sqrt{(2\pi)^{d_u}\,2^{d_u}\left|\boldsymbol{\Sigma}_i[u,u] - \boldsymbol{\Sigma}_i[u,o]\boldsymbol{\Sigma}_i[o,o]^{-1}\boldsymbol{\Sigma}_i[o,u]\right|}}, \\
&= \frac{z_i\mathcal{N}\left(\mathbf{o}|\boldsymbol{\mu}_i[o],\boldsymbol{\Sigma}_i[o,o]+\mathbf{O}\right)}{\sqrt{(2\pi)^{d_u}\,2^{d_u}\left|\boldsymbol{\Sigma}_i\right|\left|\boldsymbol{\Sigma}_i[o,o]\right|^{-1}}}, \\
&= \sqrt{\frac{\left|\boldsymbol{\Sigma}_i[o,o]\right|}{2^{d_u}\left|\boldsymbol{\Sigma}_i[o,o]+\mathbf{O}\right|}}\exp\left(-\frac{1}{2}(\bar{\mathbf{o}} - \boldsymbol{\mu}_i[o])\left(\boldsymbol{\Sigma}_i[o,o]+\mathbf{O}\right)^{-1}(\bar{\mathbf{o}} - \boldsymbol{\mu}_i[o])^T\right). \qquad (5.86)
\end{aligned}$$

Using this approximation, the expected RBF can be computed by ignoring the missing values and scaling by a factor of $2^{-\frac{d_u}{2}}$, where d_u is the number of missing variables. When $\boldsymbol{\Sigma}_j$ is parameterised as $\boldsymbol{\Lambda}_j^T\boldsymbol{\Lambda}_j$, the gradients are computed simply by considering only the elements indexed by the set of observed values o.

$$\frac{\partial \ln \bar{\boldsymbol{\Phi}}_{\mathbf{x}}[j,i]}{\partial \boldsymbol{\mu}_j[o]} = \left(\boldsymbol{\Sigma}_j[o,o] + \mathbf{O}[o,o]\right)^{-1}\left(\bar{\mathbf{x}}_i[o] - \boldsymbol{\mu}_j[o]\right), \tag{5.87}$$

$$\begin{aligned}\frac{\partial \ln \bar{\boldsymbol{\Phi}}_{\mathbf{x}}[j,i]}{\partial \boldsymbol{\Sigma}_j[o,o]} &= \frac{1}{2}\boldsymbol{\Sigma}_j^{-1}[o,o] - \frac{1}{2}\left(\boldsymbol{\Sigma}_j[o,o] + \mathbf{O}\right)^{-1} \\ &\quad + \frac{1}{2}\left(\boldsymbol{\Sigma}_j[o,o] + \mathbf{O}\right)^{-1}\left(\bar{\mathbf{x}}_i[o] - \boldsymbol{\mu}_j[o]\right)\left(\bar{\mathbf{x}}_i[o] - \boldsymbol{\mu}_j[o]\right)^T\left(\boldsymbol{\Sigma}_j[o,o] + \mathbf{O}\right)^{-1},\end{aligned} \tag{5.88}$$

$$\frac{\partial \ln \bar{\boldsymbol{\Phi}}_{\mathbf{x}}[j,i]}{\partial \boldsymbol{\Lambda}_j[:,o]} = 2\boldsymbol{\Lambda}_j[:,o]\frac{\partial \ln \bar{\boldsymbol{\Phi}}_{\mathbf{x}}[j,i]}{\partial \boldsymbol{\Sigma}_j[o,o]}. \tag{5.89}$$

When $\boldsymbol{\Sigma}_j = \left(\boldsymbol{\Gamma}_j^T\boldsymbol{\Gamma}_j\right)^{-1}$ the gradient is not as straight forward, as $\boldsymbol{\Sigma}_j[o,o]$ depends on all the elements of $\boldsymbol{\Gamma}_j$. Let $\mathbf{G}_j = \boldsymbol{\Gamma}_j^T\boldsymbol{\Gamma}_j$, then $\boldsymbol{\Sigma}_j[o,o]^{-1}$ can be expressed in terms of $\mathbf{G}_j$, using Equation (A.23), as follows:

$$\boldsymbol{\Sigma}_j[o,o]^{-1} = \mathbf{G}_j[o,o] - \mathbf{G}_j[o,u]\mathbf{G}_j[o,u]^{-1}\mathbf{G}_j[u,o]. \tag{5.90}$$

Note that $\mathbf{G}_j[u,o] = \boldsymbol{\Gamma}_j[:,u]^T\boldsymbol{\Gamma}_j[:,o]$. Finally, using the chain rule and after simplification we arrive at

$$\frac{\partial \ln \bar{\boldsymbol{\Phi}}_{\mathbf{x}}[j,i]}{\partial \boldsymbol{\Sigma}_j[o,o]^{-1}} = -\boldsymbol{\Sigma}_j[o,o]\frac{\partial \ln \bar{\boldsymbol{\Phi}}_{\mathbf{x}}[j,i]}{\partial \boldsymbol{\Sigma}_j[o,o]}\boldsymbol{\Sigma}_j[o,o], \tag{5.91}$$

$$\frac{\partial \ln \bar{\boldsymbol{\Phi}}_{\mathbf{x}}[j,i]}{\partial \boldsymbol{\Gamma}_j[:,o]} = 2\left(\boldsymbol{\Gamma}_j[:,o] - \boldsymbol{\Gamma}_j[:,u]\mathbf{G}_j[u,u]^{-1}\mathbf{G}_j[u,o]\right)\frac{\partial \ln \bar{\boldsymbol{\Phi}}_{\mathbf{x}}[j,i]}{\partial \boldsymbol{\Sigma}_j[o,o]^{-1}}, \tag{5.92}$$

$$\frac{\partial \ln \bar{\boldsymbol{\Phi}}_{\mathbf{x}}[j,i]}{\partial \boldsymbol{\Gamma}_j[:,u]} = -\frac{\partial \ln \bar{\boldsymbol{\Phi}}_{\mathbf{x}}[j,i]}{\partial \boldsymbol{\Gamma}_j[:,o]}\mathbf{G}_j[o,u]\mathbf{G}_j[u,u]^{-1}. \tag{5.93}$$

We compare this approach with a random forest model on the example from Figure 4.1. The data is split into 2000 samples for training and 1000 for testing, we then declare one of the two variables as missing, at random from each sample, from a portion of the training set selected at random. We then vary this percentage from 0% to 100% by an increment of 1% and use the trained model to generate predictions

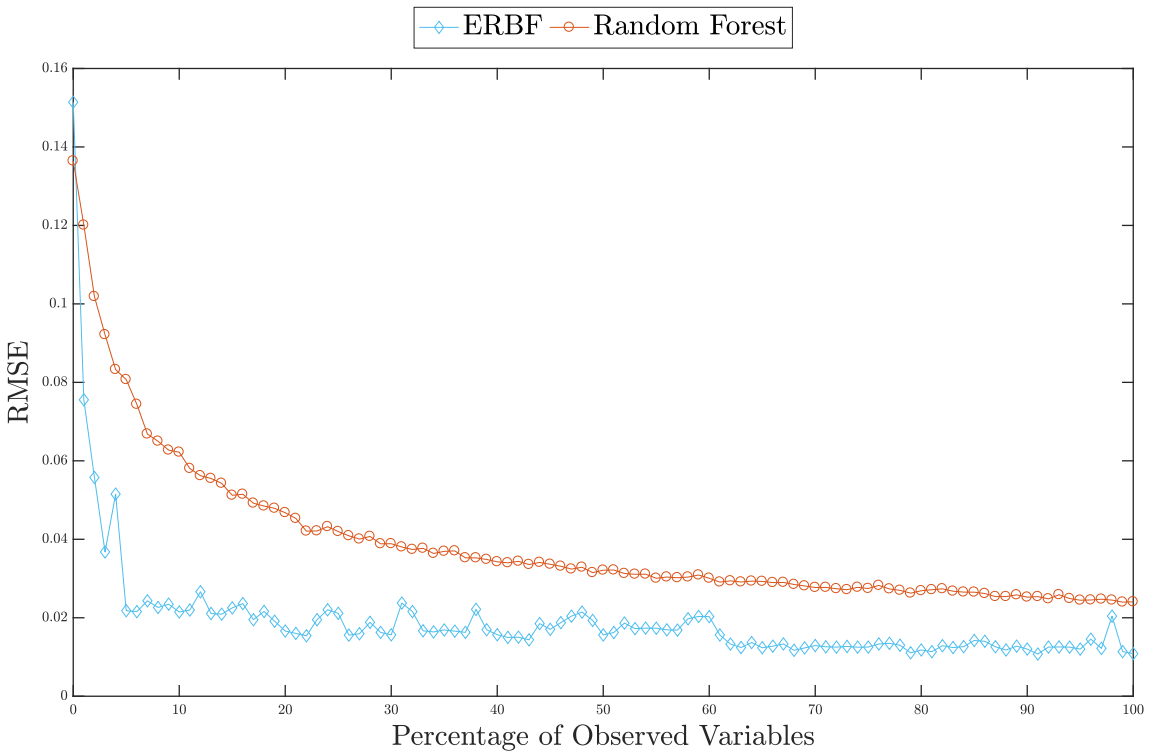


Figure 5.8: A comparison between the proposed approach and a random forest implementation, using 100 basis functions and 100 trees respectively, on the example from Figure 4.1 trained on a varying percentage of observed data. The y-axis reports the RMSE on the test set which has no missing variables.

for the held-out test set using the full data with no missing inputs. Figure 5.8 shows a comparison between the proposed approach and a random forest implementation using 100 basis functions and 100 trees respectively. The performance of our model consistently outperformed the random forest, except in the worst case when none of the samples in the training set observed both variables.

5.3 Summary

We have shown in this chapter how the capabilities of handling noisy and missing inputs can be added to sparse GPs from a BFM perspective during both prediction and training. Incorporating known input noise during prediction translates into adding the noise variance to the RBF's covariances. During prediction this treatment is exact, during training however this is an approximation. Nevertheless, this approx-

imation proved very effective and outperformed other approaches such as sampling, cost-sensitive learning and using the known variances as additional inputs. We have shown that missing variables can be viewed as a special case of noisy data and took advantage of the RBFs to construct a joint distribution of the input in order to obtain the conditional probability distribution of the missing given the observed for any arbitrary segmentation of the input. The proposed solution is exact for prediction and empirically matches a reference model specifically trained using the observed values only; and outperformes Gaussian mixture models and random forests. During training however, we have made some approximations to ease computational concerns. The proposed training procedure is competitive with random forests for up to 50% of missing data at random. In the following chapter, we move from demonstrating the proposed approaches on artificial univariate and bivariate data to a real world problem in astrophysics.

Chapter 6

Photometric Redshift Estimation

6.1 Background

The nearest star to our solar system, Alpha Centauri A, is around 4.2 lightyears away from Earth. To put this in perspective, the farthest human-made object, Voyager 1, was launched in 1977 is now about 20.65 billion kilometres, or 0.002 light-years, away from Earth and has just ventured into interstellar space. Clearly, direct experimentation is infeasible; hence, scientists have to rely only on the light emitted from the stars and gas, or more precisely their electromagnetic radiation, in order to study the Universe. Stars, galaxies and nebulae all emit radiation over a wide electromagnetic spectrum, from radio waves to gamma rays. It is through the analysis of these emissions that cosmologists and astrophysicists study the properties and evolution of astrophysical objects. When studying the geometry and the evolution of the Universe, two important measures are the distances between galaxies and their velocities relative to each other; which can be measured by their redshifts. The redshift of a light source, e.g. star, galaxy or nebula, is a measure of the velocity of an object relative to a frame of a reference, e.g. the Earth. Due to the expansion of the Universe, the redshift of a galaxy is proportional to its distance for extragalactic objects (Hubble, 1929). The measure of redshift is largely analogous to the Doppler effect, which is the change in the frequency of a signal that a moving object emits relative to an observer. If an object is moving away, its frequency will decrease (shifted towards the red), and

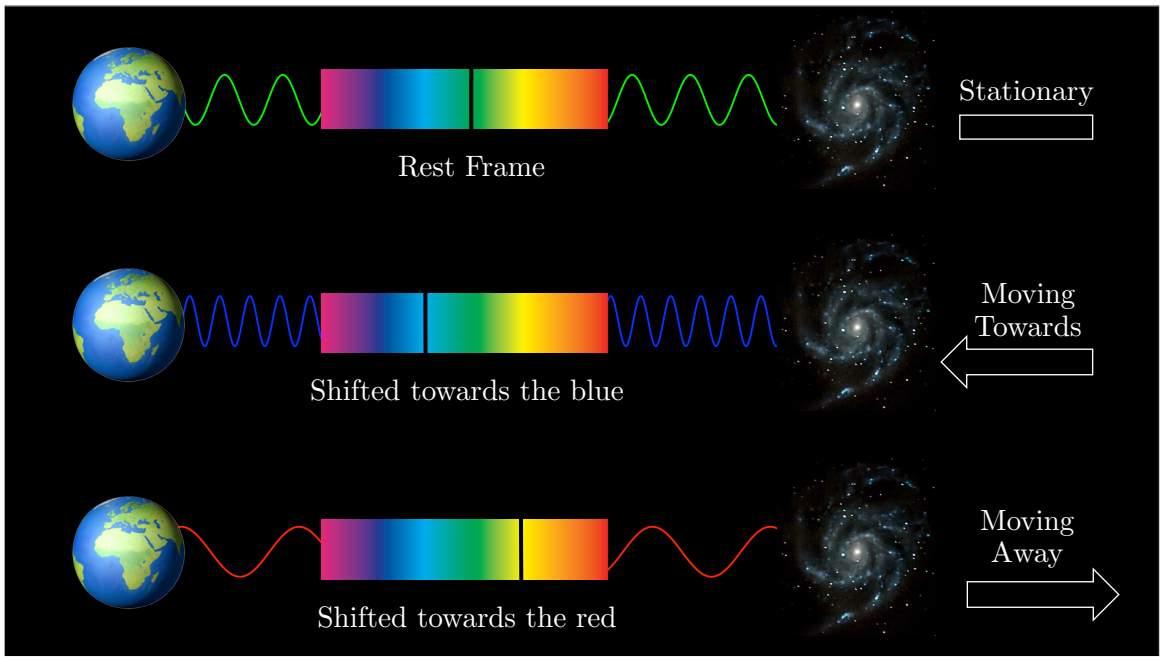


Figure 6.1: A demonstration of how the Doppler effect changes the frequency, and consequently the position of the absorption line, of the light emitted from a galaxy moving away from Earth (bottom) and a galaxy moving towards Earth (middle) compared to the reference frequency at rest (top).

will increase if it is moving towards the Earth (shifted towards the blue), see Figure 6.1. In the case of cosmological distances, the redshifting of the light is due to the expansion of space itself. The redshift of an object can be found through analysing its spectrum and measuring the shift of the absorption, or emission, lines compared to the signature of a known chemical element at a stationary point. For example, hydrogen is the most abundant element in the universe and has known absorption wavelengths to a high degree of accuracy. Comparing the *shift* in these absorption lines, from the rest-frame, allows us to infer whether or not the light source is moving towards or away from Earth, see for example Figure 6.2. The redshift is unit-less, commonly denoted as z, and is calculated as follows:

$$z = \frac{\lambda_{obs} - \lambda_{rest}}{\lambda_{rest}}, \tag{6.1}$$

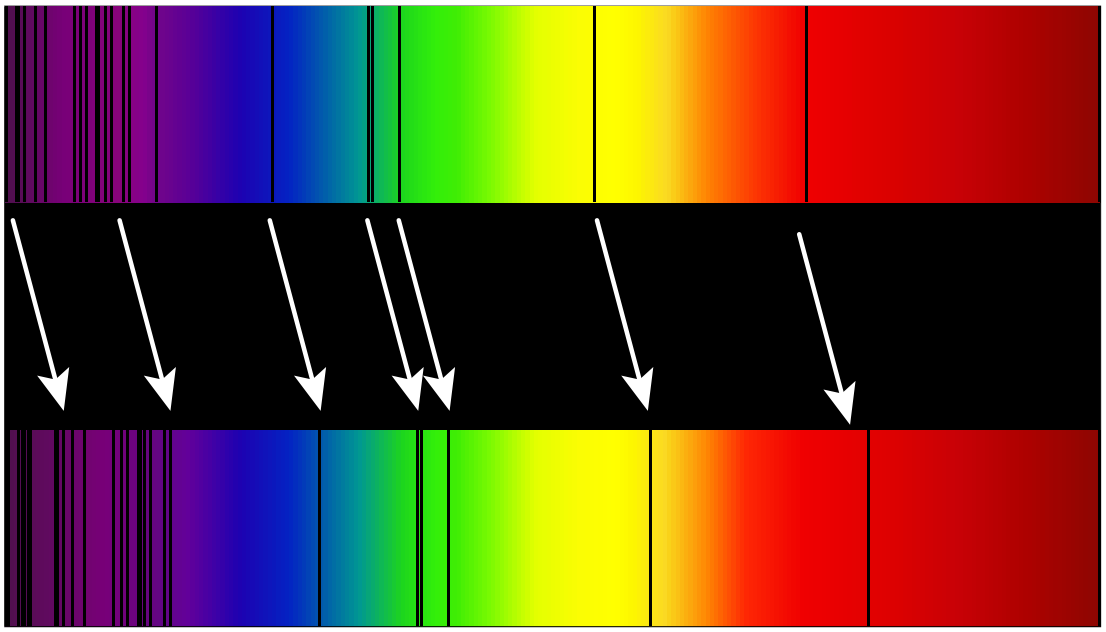


Figure 6.2: A comparison between the absorption lines of a supercluster of distant galaxies (bottom) and that of the Sun (top) in the visible spectrum (Wikimedia Commons, 2005).

also commonly expressed as

$$1 + z = \frac{\lambda_{obs}}{\lambda_{rest}}, \tag{6.2}$$

where λ_{obs} is the observed wavelength of a feature in the spectrum from the source, such as an absorption line, and λ_{rest} is the wavelength of the feature emitted from a light source at rest, or the rest-frame. This is known as the spectroscopic redshift, and provides the most accurate method for measuring redshift. However, this method is time consuming for large scale surveys, because the light is split up into typical spectral bin widths of ~ 1 Å, and only relatively small area spectroscopic campaigns can reach faint magnitudes (Le Fèvre et al., 2013, 2015; Lilly et al., 2009), or at the other extreme, over larger areas for bright magnitudes (Alam et al., 2015; Colless et al., 2003; Driver et al., 2011).

These limitations force cosmologists and astrophysicists to consider alternative means that are cheaper and faster to achieve in order to cope with the demands of current and future large scale surveys such as *Euclid* (Laureijs et al., 2011), the Sloan

Digital Sky Survey (SDSS; Alam et al., 2015) and the Large Synoptic Survey Telescope (LSST; Ivezic et al., 2008) aimed at understanding the reasons behind the accelerating expansion rate of the Universe and the nature of the physical processes behind it, for example *dark energy* and *dark matter*. Together, dark energy and dark matter make up around 94% of the energy density of the Universe, but scientists currently have very little understanding of their nature. The goal is to study their effects through their gravitational effects on the clustering of galaxies in space and time, and through a phenomenon called gravitational lensing. Gravitational lensing occurs when a source of gravity bends the light passing near it due to the distortion it creates in the space-time around it. The foreground source will change the apparent position and shape of the background source when observed from Earth; which Einstein's theory of General Relativity predicted, and was confirmed during the solar eclipse of May 29, 1919 (Dyson et al., 1920). The positions of the stars near the edge of the Sun's disk shifted by the same amount predicted by the theory. Figure 6.3 illustrates the effect of gravitational lensing when observing a distance galaxy behind a foreground source of strong gravity. Using the same concept, one can invert the calculations to derive the mass of the object between the observer and the background source from lensing. When the source of gravity is massive, e.g. galaxy clusters or black holes, the lensing effect is very noticeable from the formation of arcs, full arcs (also known as Einstein rings, see Figure 6.4a) or multiple copies of the background source (quad-systems are also known as Einstein's cross, see Figure 6.4b).

Weak-lensing on the other hand, creates slight distortions to the background source's orientation and shape; which is not sufficient to determine with certainty the mass of the foreground object from one or few background galaxies. Hence, determining the mass of the foreground source of gravity using weak-lensing, a large percentage of which is potentially dark matter, is in its essence a statistical problem that needs more data to obtain a more precise estimate of the foreground mass.

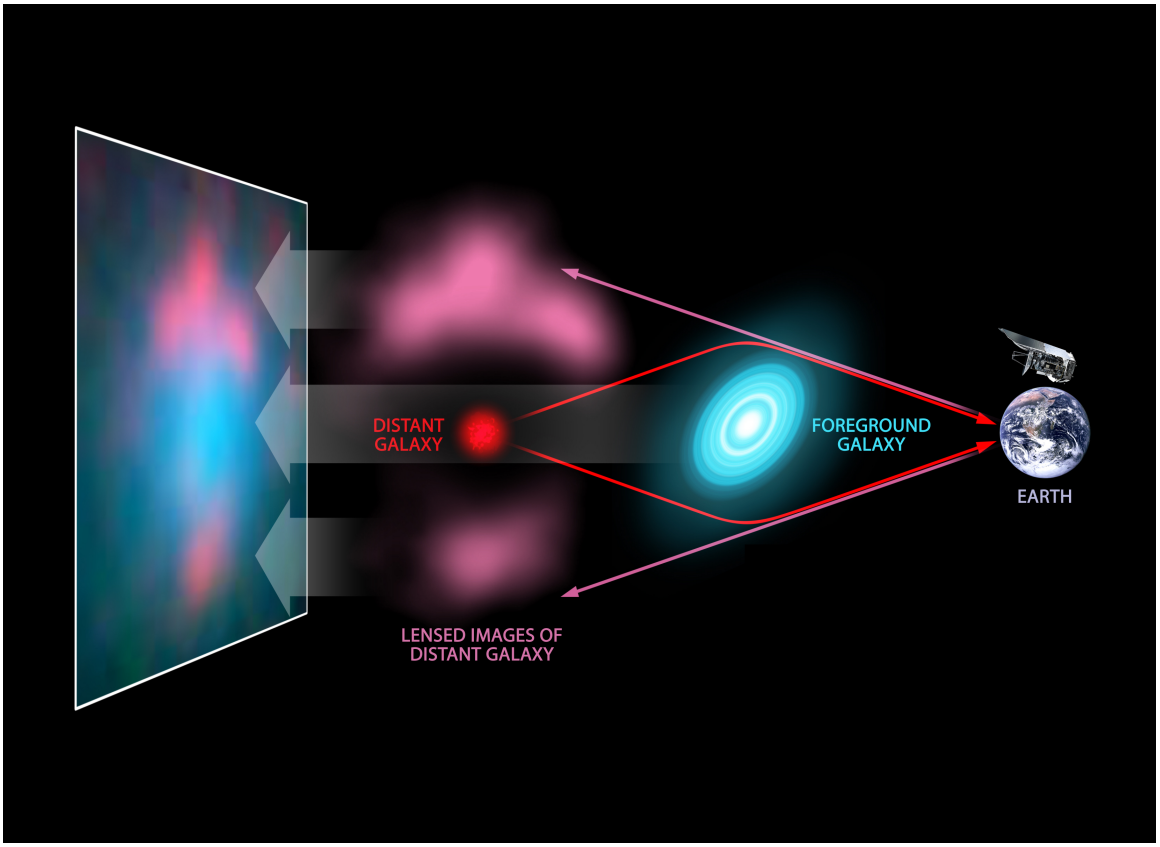


Figure 6.3: A gravitational lensing illustration. A distant galaxy (red) behind a foreground source (blue) where its light is bent due to gravity creating the effect of observing two distinct sources (magenta) in the recorded image in the left panel (NASA/JPL-Caltech, 2010).

Furthermore, the accuracy of the mass determination of the foreground lensing structures is greatly enhanced when accurate distances are available for both the galaxies that trace the lensing mass and also the background galaxies that are being lensed. However, as previously mentioned, spectroscopy is very time consuming and is less accurate for dim sources (due to the dispersion of light). Alternatively, rather than measuring individual spectra, the emission from an object is measured using broad filters (typically $\sim$ 1000 Å) covering different ranges of wavelengths, which together sample the spectral energy distribution (SED) of fainter sources, at the expense of fine spectral resolution. Using this photometry to estimate redshifts, or photometric redshifts (hereafter photo-z), is an essential component to the success of many cur-

(a) Einstein's Ring (b) Einstein's Cross

Figure 6.4: An example of (a) an Einstein's ring (LRG 3-757; ESA/Hubble & NASA, 2011) and (b) an Einstein's cross (HE0435-1223; ESA/Hubble & NASA, 2017). The gravity of the luminous red galaxy in (a) curved the light of the background blue galaxy, creating the illusion of of ring-shaped galaxy. The foreground galaxy in (b) creates four copies of a distant quasar.

rent and future missions. The advantage of using photo-z is to cover a larger redshift range as well as the ability to collect sufficient amount of data in reasonable time. Although photometric redshifts are less accurate than spectroscopy, as long as the estimation performance satisfies some target thresholds set by the various missions, they are sufficient enough to meet the science objectives.

Photo-z estimation appeared as early as 1962 (Baum, 1962) and the techniques largely fall into two methodological classes, machine learning and template fitting. Machine learning methods (MLM), such as artificial neural networks (Collister and Lahav, 2004; Firth et al., 2003; Sadeh et al., 2015), nearest-neighbour (Ball et al., 2008), genetic algorithms (Hogan et al., 2015), self-organised maps (Geach, 2012; Masters et al., 2015), random forest (Carrasco Kind and Brunner, 2013) and Gaussian Processes (Almosallam et al., 2016a,b; Bonfield et al., 2010; Foster et al., 2009; Way et al., 2009), have all been applied to photo-z estimation. Artificial neural networks

dominate the most commonly used MLM (Brescia et al., 2014; Firth et al., 2003; Sadeh et al., 2015; Vanzella, E. et al., 2004). Some MLM for photo-z can also provide a degree of uncertainty in their predictions, or a full multimodal probability distribution (Bonnett et al., 2015; Carrasco Kind and Brunner, 2013; Rau et al., 2015).

Template fitting methods, on the other hand, do not learn a model from a training sample but rather use purely synthetic or empirical templates of galaxy SEDs for different galaxy types that can be redshifted to fit the photometry. Some limitations of template fitting methods are whether the templates are representative of the galaxies observed at high redshift and how emission lines affect the photometry. Some allow spectroscopic data to be used to adjust the zero-points on the photometry to compensate for any slight mismatch between SED templates and the observations. Examples of template fitting software include HYPERZ; (Bolzonella et al., 2000), ZEBRA; (Feldmann et al., 2006), EAZY; (Brammer et al., 2008) and LE PHARE (Ilbert et al., 2006). There have been comprehensive evaluations and comparative studies in (Abdalla et al., 2011; Bonnett et al., 2015; Hildebrandt et al., 2010; Sánchez et al., 2014) of different photometric redshift estimation techniques, using both machine learning and template fitting methods.

In the following sections, we present previously published results from Almosallam et al. (2016b) and Almosallam et al. (2016a) applied to both simulated and real astronomical datasets. In Section 6.2, we present comparisons between the proposed approach and state of the art photo-z code (at the time of publishing Almosallam et al. (2016b)) on a synthetic photometric dataset focusing on point estimation, extrapolation and meeting the special requirements of the *Euclid* mission through cost-sensitive learning. In Section 6.3, we present comparisons between the proposed approach and newly published photo-z codes focusing on variable variance estimation using a real photometric survey. Finally, recent results on handling missing photometry will be presented and discussed in Section 6.4.

6.2 Results on Simulated Data

In this section, we specifically address the photometric bands and depths planned for *Euclid* (Laureijs et al., 2011). *Euclid* is a space-telescope mission, planned to be launched in 2020, with the aim to provide imaging data in a broad RIZ band (wavelength range 550-900 nm) and the more standard near-infrared Y (920-1146 nm), J (1146-1372 nm) and H (1372-2000 nm) bands. Supporting ground-based photometry is expected in the optical g, r, i and z bands (non-overlapping covering the wavelength range 420-930 nm). The total number of features is therefore 16, i.e. the 8 g, r, i, z, RIZ, Y, J and H bands alongside their associated errors and the target output is the spectroscopic redshift z.

6.2.1 Models and Performance Metrics

In this section, we compare the proposed GP models with variable length-scales (GPVL), diagonals (GPVD) and covariances (GPVC), from Section 4.4, with the following models:

ANNz: An artificial neural network based code for photo-z estimation (Collister and Lahav, 2004). The code takes as input for training the magnitudes and their associated errors as well as a validation set with the same set of inputs. The validation set is used for model selection but not for training, i.e. to keep track of the best performing model to avoid over fitting. The code allows for multi-layer architectures to be specified and can merge (average the results of) different models with varying architectures.

stableGP: A low rank approximation Gaussian process (Foster et al., 2009), where the selection of the representative points in the "active set" is done using partial Cholesky factorisation with pivoting, which selects the most linearly independent columns of the covariance matrix. The motivation in this approach is to find a

Table 6.1: Performance metrics used to evaluate the models, where n is the number of samples, z_i is the spectroscopic redshift of sample i and $\acute{z}_i$ is its predicted photometric redshift.

Metric	Equation	Description
δ_i	$z_i - \acute{z}_i$	Error for the i-th sample
$\hat{\delta}_i$	$\delta_i / (1 + z_i)$	Normalised error for the i-th sample
Δz	$\sqrt{\frac{1}{n}\sum_{i=1}^{n} \delta_i^2}$	Root mean squared error
$\widehat{\Delta z}$	$\sqrt{\frac{1}{n}\sum_{i=1}^{n} \hat{\delta}_i^2}$	Normalised root mean squared error
max	$\max(\lvert\delta_1\rvert, \ldots, \lvert\delta_n\rvert)$	Maximum error
$\widehat{\max}$	$\max(\lvert\hat{\delta}_1\rvert, \ldots, \lvert\hat{\delta}_n\rvert)$	Maximum normalised error
μ	$\frac{1}{n}\sum_{i=1}^{n} \delta_i$	Bias
$\hat{\mu}$	$\frac{1}{n}\sum_{i=1}^{n} \hat{\delta}_i$	Normalised bias
σ	$\sqrt{\frac{1}{n}\sum_{i=1}^{n} (\delta_i - \mu)^2}$	Standard deviation of the errors
$\hat{\sigma}$	$\sqrt{\frac{1}{n}\sum_{i=1}^{n} (\hat{\delta}_i - \hat{\mu})^2}$	Standard deviation of the normalised errors

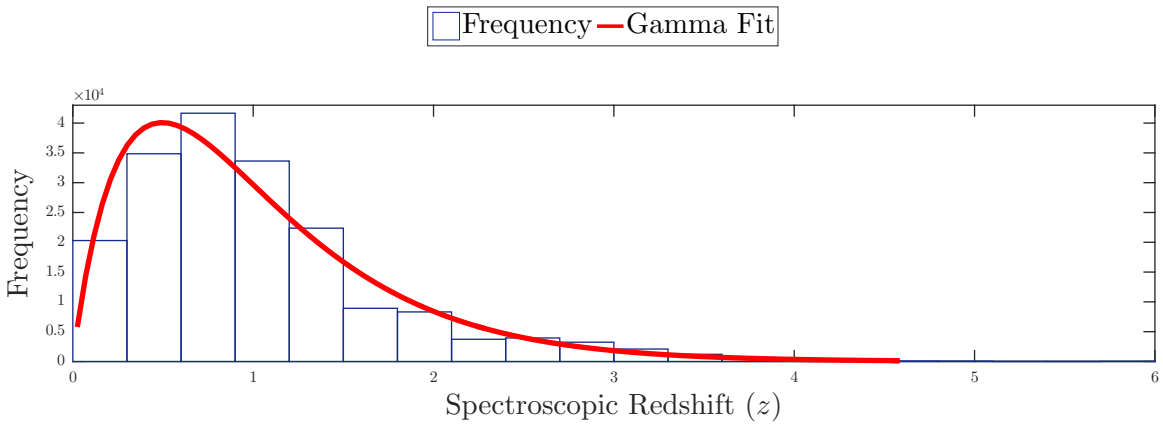
subset of m samples from the training set such that the condition number of the covariance function with respect to the active set is minimised for computational stability. We therefore use the SR-VP version of the code published by the authors (Ashok Srivastava, 2010) that shows the most stable performance.

SPGP: The sparse pseudo-input Gaussian process introduced by Snelson and Ghahramani (2006). This is a fully independent training conditional sparse GP approximation previously described in Section 2.3.2, but learns a separate length-scale per input dimension for the squared exponential kernel; this is close to the global diagonal version of the proposed approach (GPGD). We use the Matlab implementation published by the authors (Edward Snelson, 2007).

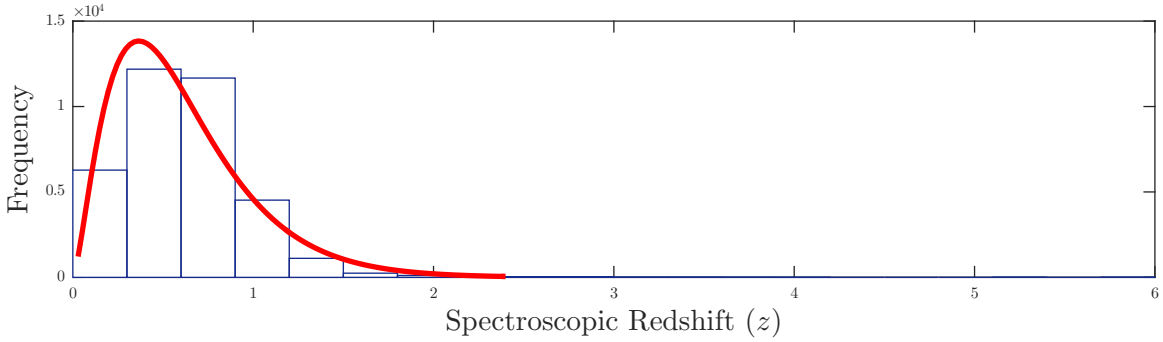
In photometric redshift estimation, the aim is to minimise the *normalised error* $|z - \acute{z}|/(1+z)$, where z is the spectroscopic redshift and $\acute{z}$ is the predicted photometric redshift. We thus consider normalised versions of the standard metrics, shown in Table 6.1.

6.2.2 The Dataset

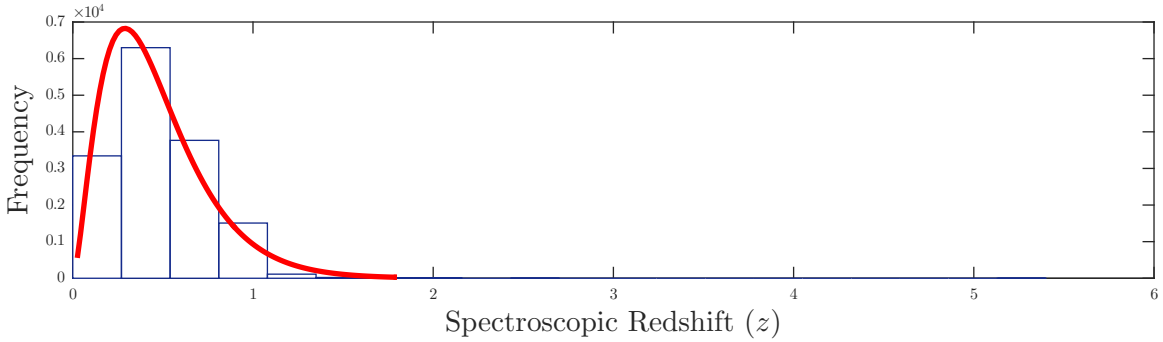
We consider the simulated survey from Jouvel et al. (2009), where we remove all sources with $RIZ > 25$, to target magnitudes set for *Euclid*, and sources with any missing measurements (only 15 sources). The total number of sources after the cuts is 185,253. Further cuts are imposed in Section 6.2.4 to evaluate the extrapolation performance of the models ($RIZ < 23$ and $RIZ < 22$). The spectroscopic redshift distributions, with the RIZ cuts, are shown in Figure 6.5. The data is pre-processed using Principal Component Analysis (PCA; Jolliffe, 1986) to de-correlate the features. De-correlation improves the rate of convergence of the optimisation routine, which is particularly important in this dataset as the magnitudes of the filters are strongly correlated with each other. The uncertainties on the photometry in each band were removed because, unlike real surveys, the log of the uncertainties are almost perfectly correlated with their respective filter, especially in the target range, hence adding no additional information. However, they were used as inputs to ANNz to satisfy the input format of the code. In subsequent tests, we use m to refer to the number of hidden units in ANNz, the rank in stableGP, the number of pseudo-points in SPGP, and the number of basis functions in GPVL, GPVD and GPVC. We split the data randomly into three sets; 80%, 10% and 10% for training, validation and testing respectively. The entire redshift range is used to train the models, but the performances are reported on the range $0 \leq z \leq 2$ to target the parameter space set out in Laureijs et al. (2011). Each model is trained for 500 iterations in each run and the validation set is used for model selection and parameter tuning, the results however are reported on the held-out test set, which is not used in either training, pre-processing or parameter tuning.



(a) Full



(b) RIZ <23



(c) RIZ <22

Figure 6.5: Histograms and gamma distribution fits of the spectroscopic sample using (a) the full data set, (b) sources with RIZ magnitude<23 and (c) sources with RIZ magnitude<22 from the simulated data.

6.2.3 Modelling Performance

In the first test, we cross-compare the models using $m = 10$. We set m deliberately low at the beginning to highlight the sparse-limit modelling capabilities of each

method and to make the differences between the models visually more clear, in latter experiments we increase the value of m and compare the models in a more quantified way. The models are optimised without cost-sensitive learning or a prior mean function to evaluate the models purely as different ways of fitting a linear combination of basis functions to the data. The density scatter plots of the spectroscopic redshift versus the photometric redshift are shown in Figure 6.6 and the performance metrics for each method are reported in Table 6.2. It is clear from the plots and the metrics the advantage of using inducing points over an active set, especially for small values of m, when we compare the scatter plot of stableGP in Figure 6.6b with that of SPGP in Figure 6.6c. The formulation of the two models are almost identical, the main difference is that the basis functions in SPGP are learned during the optimisation of the hyper-parameters, whereas stableGP's basis set is pre-selected from the training set in an unsupervised fashion using partial Cholesky factorisation with pivoting. The use of a bespoke covariance per basis function also provides an additional advantage, as evident from the improvement of GPVD over SPGP, note that the former uses a different diagonal covariance for each basis function whereas the latter uses a single diagonal covariance for all basis functions. Moreover, richer covariances provide better results, as evident from the consistent improvement of GPVC over GPVD and GPVD over GPVL.

6.2.4 Prior Mean Function

We compare in this section the extrapolation performance of ANNz with GPVC using different prior mean functions, namely a zero mean, a linear regression prior and a joint optimisation approach. In the linear regression prior method, first we fit a linear regression model to the data then subtract the predictions from the ground truth before optimising the GP model. The joint linear optimisation on the other hand, learns a linear model and the non-linear deviations from it jointly, as described

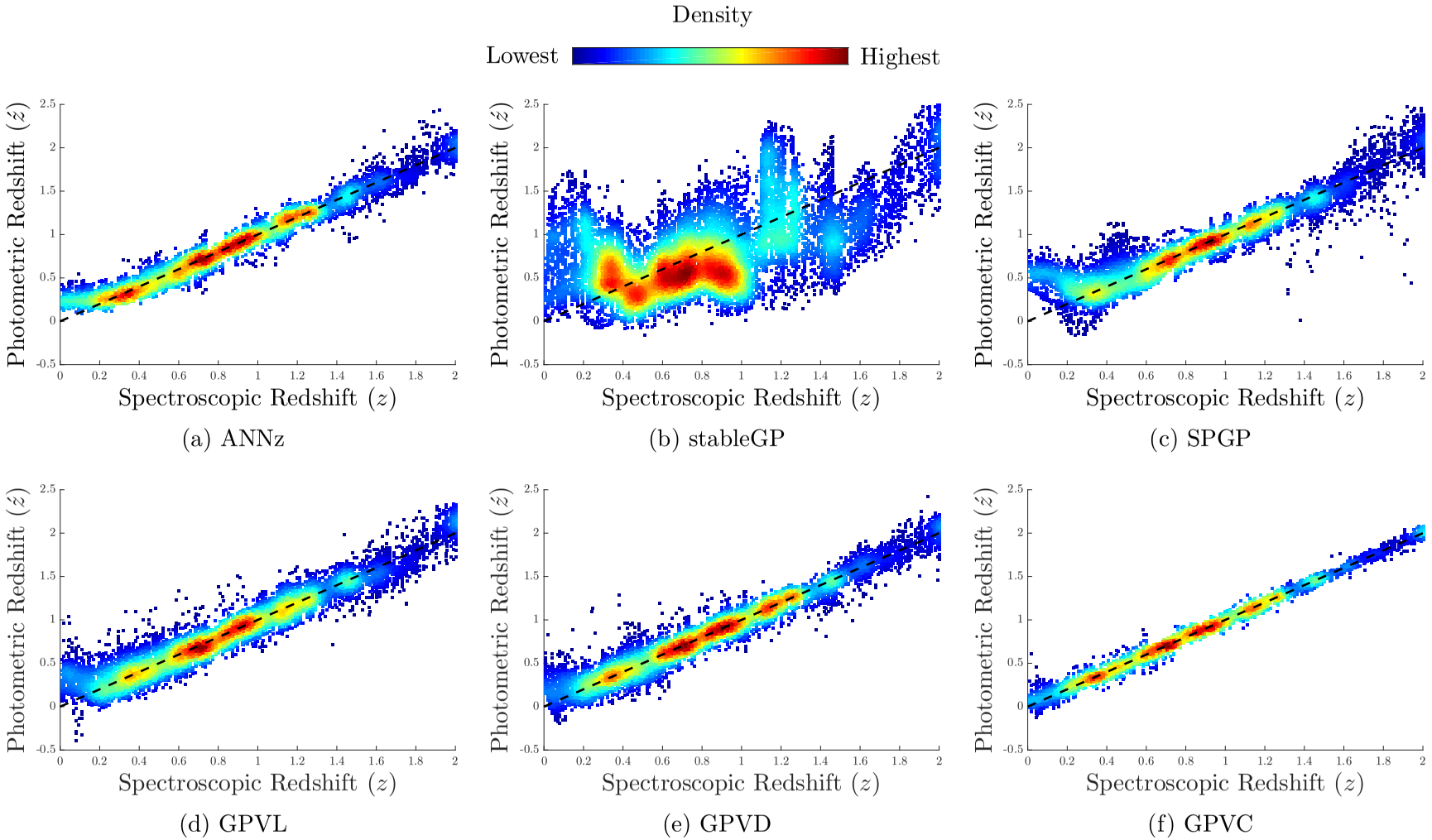


Figure 6.6: Density scatter plots of the true spectroscopic redshift z vs the predicted photometric redshift $\acute{z}$ for (a) ANNz, (b) stableGP, (c) SPGP, (d) GPVL, (e) GPVD and (f) GPVC ($m = 10$). The plots show the performance on the same test set, the colours however are scaled differently according to the density range in each plot to avoid colour saturation.

Table 6.2: Performance metrics for each method (with $m = 10$). The best-performing method is highlighted in bold font.

	ANNz	stableGP	SPGP	GPVL	GPVD	GPVC
Δz	0.0848	0.4399	0.1606	0.1280	0.1004	**0.0378**
$\widehat{\Delta z}$	0.0568	0.2836	0.1222	0.0890	0.0687	**0.0260**
max	1.0126	1.5906	1.4951	0.6632	0.6181	**0.2199**
$\widehat{\max}$	0.4199	1.4441	0.7962	0.4411	0.3127	**0.1595**
μ	-0.0025	-0.0085	-0.0230	-0.0071	-0.0050	**-0.0007**
$\hat{\mu}$	-0.0048	-0.0365	-0.0245	-0.0092	-0.0050	**-0.0008**
σ	0.0847	0.4399	0.1589	0.1278	0.1003	**0.0377**
$\hat{\sigma}$	0.0566	0.2812	0.1197	0.0885	0.0685	**0.0260**

in Section 3.5. In order to effectively evaluate the different approaches, we train each model using different cuts on the RIZ band. First, we train using sources with $RIZ < 23$ (29,024 samples in the training set), then evaluate the performance of the model on the test set with $RIZ < 23$, $RIZ \geq 23$, and the full test set. We also perform a similar test using a split of $RIZ < 22$ (12,056 samples in the training set). This is critical to such missions in order to evaluate the model's performance in scenarios where bright sources dominate the training set, as may well be the case in reality. The density scatter plots for GPVC, using different prior mean methods, and ANNz are shown in Figure 6.7 for the different RIZ cuts and the results on the metrics are reported in Table 6.3. Jointly optimising the prior mean function outperformed consistently the other methods in both extrapolation and interpolation. Furthermore, the "Joint" optimisation method, as evident from the plots in Figure 6.7, results in fewer catastrophic and systematic errors than the other methods with a factor of ~ 2 improvement over ANNz when $RIZ < 22$ and ~ 2.5 when $RIZ < 23$.

6.2.5 Size of the Basis Set

We investigate in this section the relationship between the models' complexity and their accuracy. The complexity of the models is determined by the parameter m, i.e.

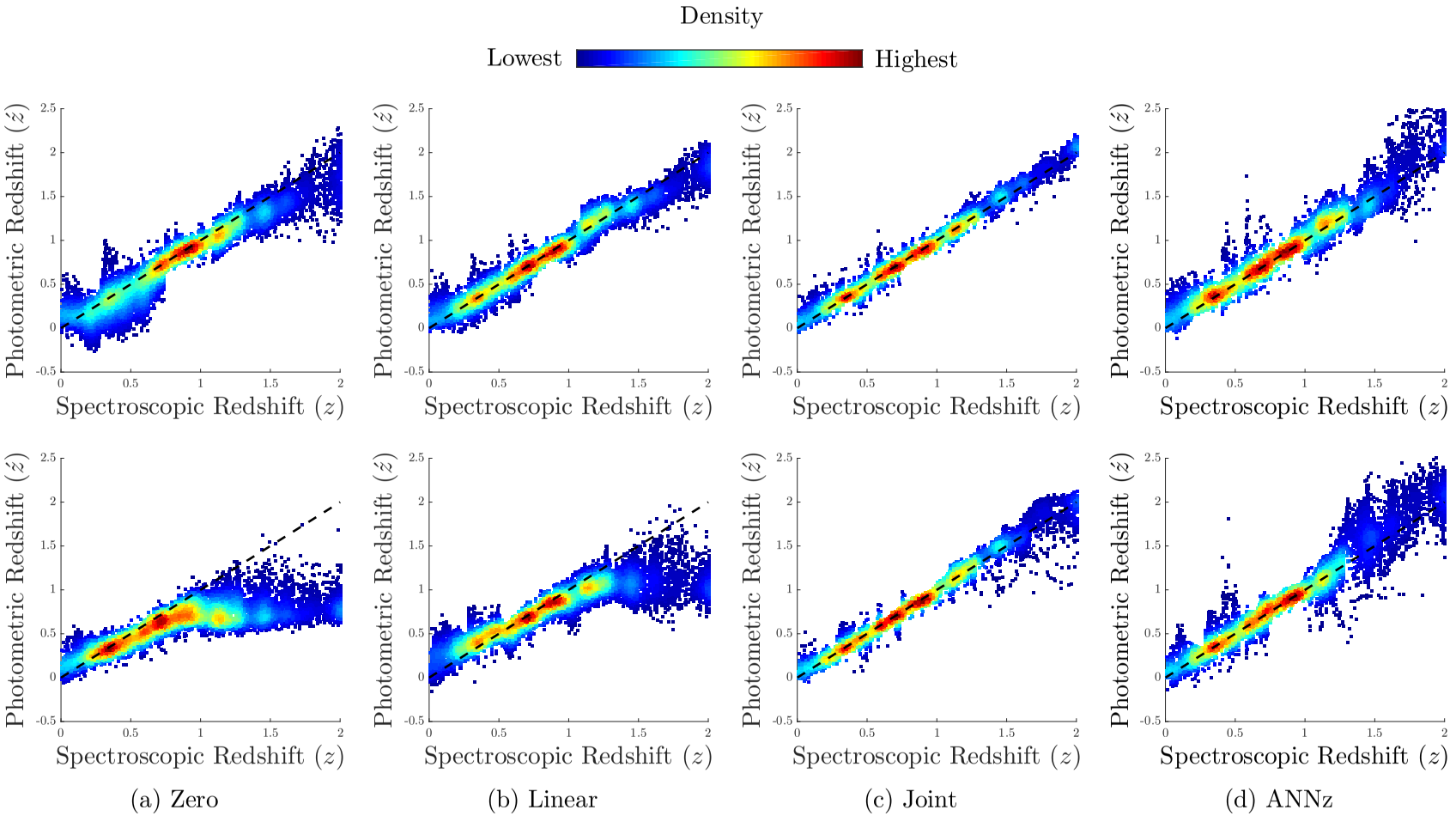


Figure 6.7: Density scatter plots of the true z versus the predicted $\acute{z}$ after training the GPVC model on samples with $RIZ < 23$ (top) and $RIZ < 22$ (bottom) using $m = 10$ basis functions with (a) a zero-mean prior, (b) linear regression prior and (c) a joint prior optimisation and (d) ANNz. The plots show the performance on the same test set, the colours however are scaled differently according to the density range in each plot to avoid colour saturation.

Table 6.3: The Δz score for the GPVC model when trained using $m = 10$ basis functions with different prior mean functions and RIZ splits. The results for ANNz are shown for comparison.

Trained on	$RIZ < 22$			$RIZ < 23$		
Tested on	< 22	≥ 22	Full	< 23	≥ 23	Full
ANNz	0.0385	0.1383	0.1325	0.0537	0.1458	0.1315
Zero	0.0446	0.4081	0.3898	0.0640	0.1698	0.1533
Linear	0.0433	0.2573	0.2459	0.0578	0.1012	0.0936
Joint	**0.0177**	**0.0688**	**0.0659**	**0.0266**	**0.0571**	**0.0521**

the number of neurons, rank, pseudo-inputs and basis functions; which in practice is only limited by the available computational resources and the need to avoid overfitting for non-Bayesian methods such as ANNs. In this experiment, we vary m from 5 to 200 (in an increment of 5) to study its effect on the accuracy as measured by Δz. Figure 6.8 shows Δz as a function of m for all the models. GPVC consistently outperforms the other methods, most notably when m is small, whereas stableGP shows the worst performance across the board. ANNz initially outperformed GPVL and is comparable to GPVD, but it started to overfit by $m = 30$ and by $m = 60$ it gave worse results than GPVL. The performance of SPGP is always poorer than the proposed approaches and better than stableGP. Even though SPGP is more robust against overfitting, the best performing SPGP model is not better than the best ANNz model. The best GPVL model, on the other hand, is better than the best ANNz model.

We also generated photometric redshifts from a committee of five neural networks using a two-layer architecture, each layer with twice the number of hidden units as the number of filters as recommended in Collister and Lahav (2004) and has become a standard for most ANNz users. The results of the final models, with $m = 200$, are summarised in Table 6.4 and the density scatter plots are shown in Figure 6.9. GPVC outperforms all the methods on all the metrics, with a factor of ~ 3 improvement in

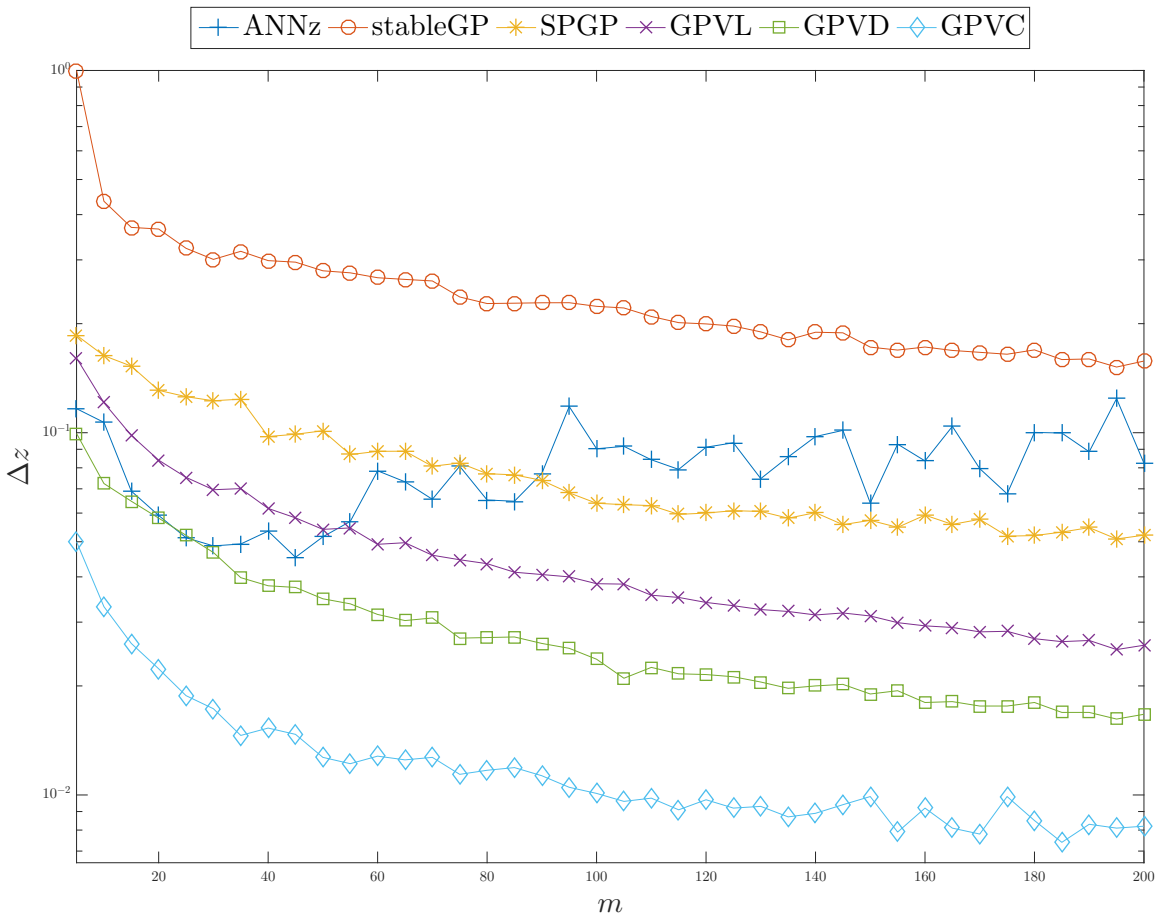


Figure 6.8: Δz as a function of m for all the methods. The y-axis is shown on a log scale to enhance the visualisation.

the accuracy of Δz and $\widehat{\Delta z}$ over ANNz and a factor of ~ 6 improvement over SPGP for this simulated data set.

6.2.6 Cost-sensitive Learning

Most machine learning methods for photo-z analysis suffer from distribution bias, i.e. the optimisation routine will sacrifice accuracy in order to avoid complicating the model just to fit few examples in less represented regions of the space. Given the nature of the problem, where the majority of the available data for training is in the low redshift range, machine learning models will tend to find very good fits for low redshifts, but poor fits for high redshifts. Moreover, the lack of strong emission lines that are detectable with visible-wavelength spectrographs in the 'redshift-desert'

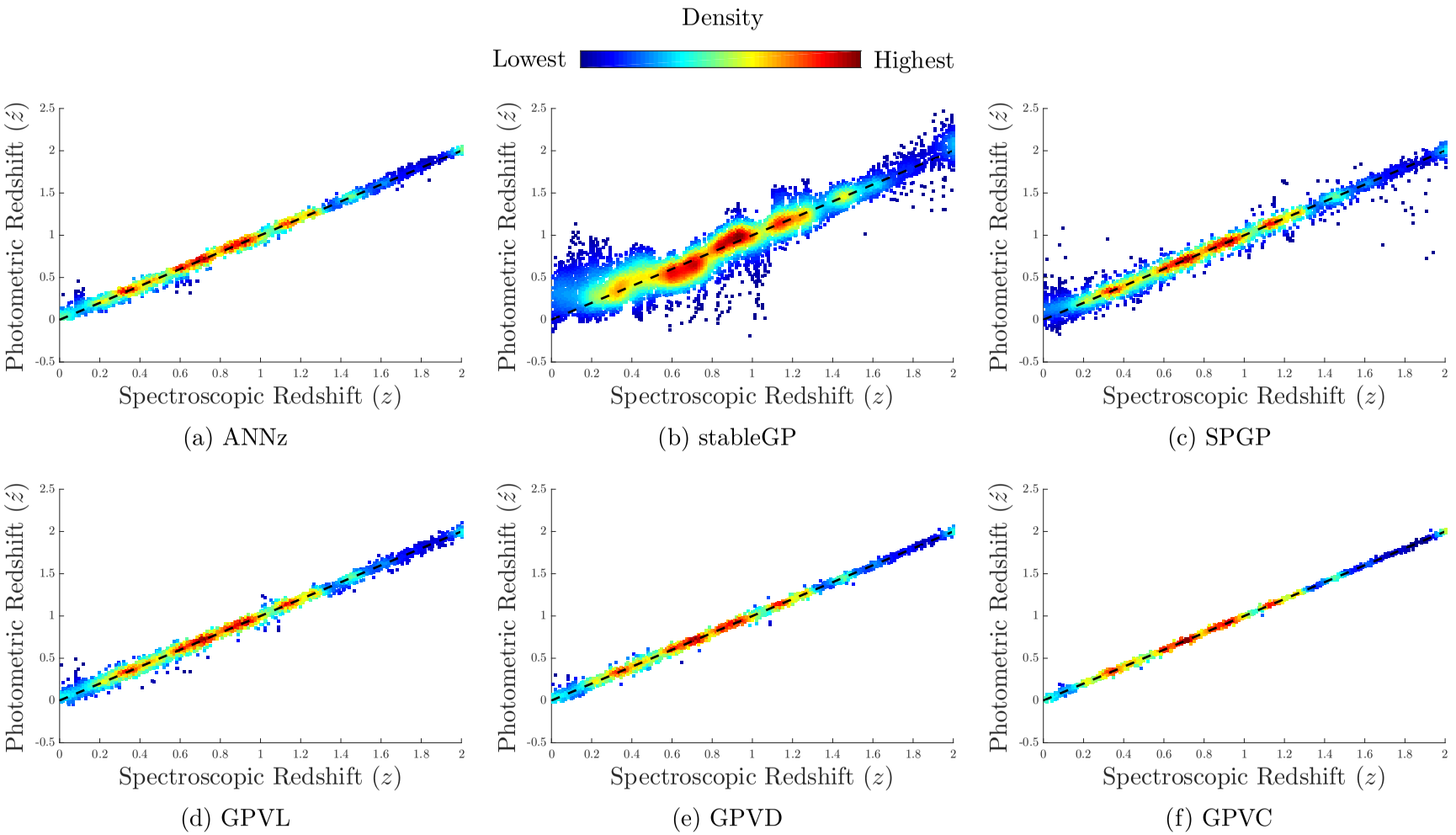


Figure 6.9: The density scatter plot for the final (a) ANNz, (b) stableGP, (c) SPGP, (d) GPVL, (e) GPVD and (f) GPVC models on the simulated survey. ANNz is based on a committee of 5 networks with 8:16:16:1 architectures, whereas the rest of the models are based on $m = 200$.

Table 6.4: Performance measures for the final ANNz model using a committee of 5 networks with 8:16:16:1 architectures and the final models using $m = 200$ basis functions, with a jointly optimised linear function for GPVL, GPVD and GPVC, on the simulated survey.

	ANNz	stableGP	SPGP	GPVL	GPVD	GPVC
Δz	0.0262	0.16377	0.04999	0.02690	0.01691	**0.00846**
$\widehat{\Delta z}$	0.0180	0.11379	0.03594	0.01863	0.01164	**0.00568**
max	0.3696	1.18522	1.18498	0.30934	0.20474	**0.10847**
$\widehat{\max}$	0.3391	0.59559	0.40721	0.18748	0.12408	**0.07280**
μ	-0.0004	-0.01584	-0.00011	-0.00025	-0.00019	**-0.00004**
$\hat{\mu}$	-0.0007	-0.01628	-0.00128	-0.00043	-0.00021	**-0.00006**
σ	0.0262	0.16301	0.04999	0.02690	0.01691	**0.00846**
$\hat{\sigma}$	0.0180	0.11262	0.03592	0.01862	0.01164	**0.00568**

at $1.2 < z < 1.8$ means that this redshift range is often under-represented in the spectroscopic sample. In addition to accounting for variable data densities, one of the requirements for the *Euclid* mission is to minimise the normalised sum of errors, $\sum_{i=1}^{n} |z_i - \acute{z}_i|/(1+z_i)$, rather than the normal sum of squared errors. Thus, low redshift samples are weighted more heavily by a specific scale. To address these issues, we use cost-sensitive learning with two different weight configurations. The first is to assign each sample i an error cost $\omega_i = (1 + z_i)^{-2}$, which we denote "Normalised", to target the objective of the mission, and the second configuration assigns each sample a weight according to the frequency of their true redshift to ensure balanced learning, denoted "Balanced"; in addition to the "Normal" weighting scheme, i.e. $\omega_i = 1 \; \forall i \in \{1, \ldots, n\}$. To mimic a balanced data set in our setup, the galaxies are grouped by their spectroscopic redshift using non-overlapping bins of width $\delta_z = 0.1$. The weights are then assigned as follows for balanced training:

$$\omega_i = \frac{\max(f_1, \ldots, f_{n_b})}{f_j : i \in \mathcal{S}_j}, \tag{6.3}$$

where f_j is the frequency of samples in bin number j, n_b is the number of bins and $\mathcal{S}_j$ is the set of samples in bin number j. Equation (6.3) assigns a weight to each training

point which is the maximum bin frequency over the frequency of the bin where the source belongs to, this ensures that the error cost of source i is inversely proportional to its spectroscopic redshift frequency in the training set.

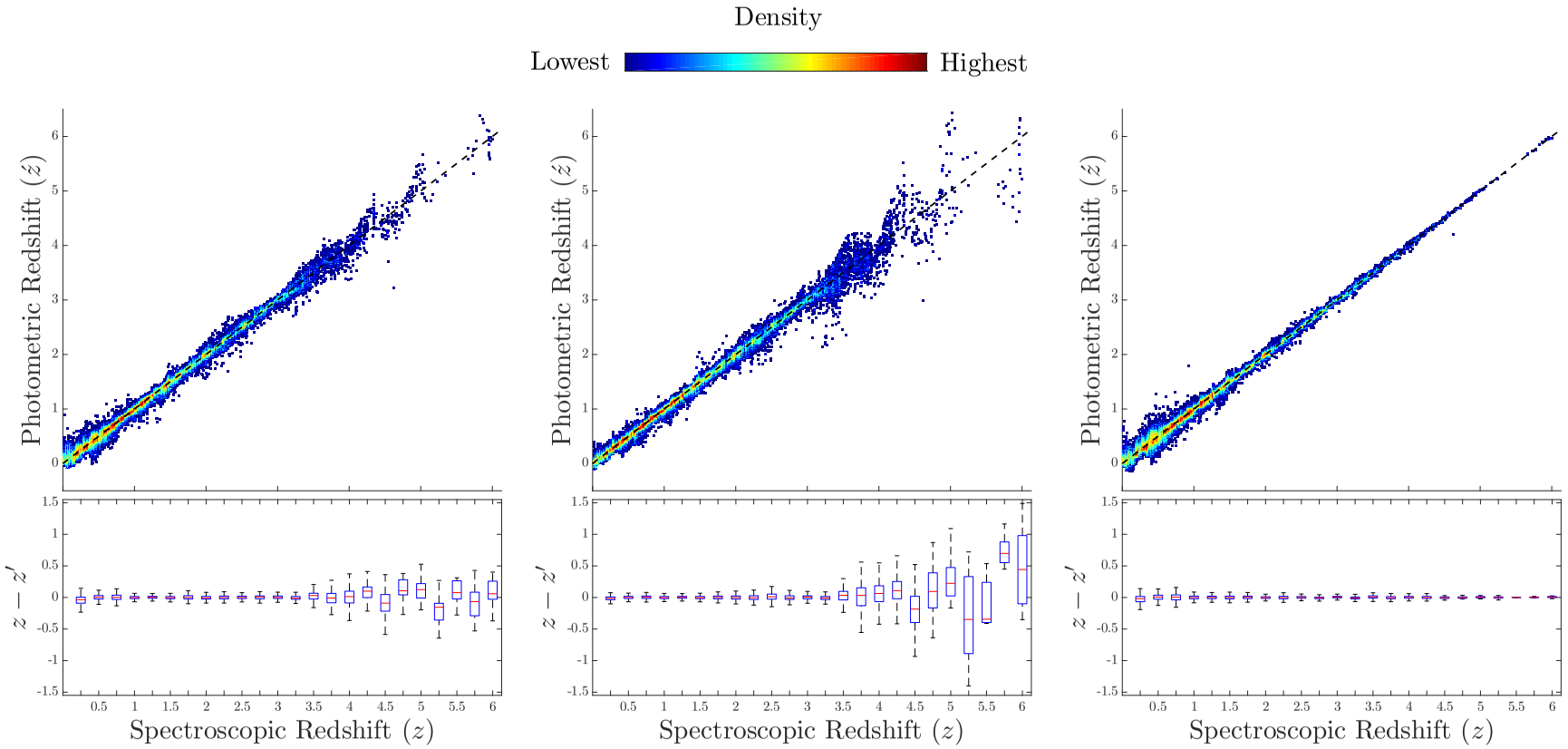
To compare the three weighting schemes, we train a GPVC model with $m = 100$ for each type of cost-sensitive learning. The models were trained such that they have almost equal Δz score, ~ 0.05, to examine the differences between the other metrics and the resulting error distributions. The box plots of the residual errors and the density scatter plots for the "Normal", "Normalised" and "Balanced" on the test set are shown in Figures 6.10a to 6.10c respectively. In this experiment, we train and test on the entire redshift range in order to create a larger distribution bias so that the differences are highlighted more clearly. Using cost-sensitive learning to virtually balance the distribution resulted in a more consistent and uniform error bias across the redshift range, with considerably smaller error bars, as opposed to the "Normal" method, especially at higher redshifts where data is more sparse. The "Normalised" version of cost-sensitive learning results in a systematic bias, as expected, with higher accuracy in lower redshifts. The performance metrics for the three types of cost-sensitive learning are summarised in Table 6.5. The generalisation performance is better when using "Balanced" training over "Normal", when both have similar Δz, with lower maximum errors.

6.3 Results on Real Data

6.3.1 The Dataset

We use the Sloan Digital Sky Survey's 12th Data Release (SDSS-DR12; Alam et al., 2015) to train and test the models. Galaxies with available u,g,r,i and z magnitudes, associated errors (the 10 input features) and spectroscopic redshifts (target output) were retrieved from the database using the CasJobs service provided by SDSS[1]. The

[1] `casjobs.sdss.org`



(a) Normal (b) Normalised (c) Balanced

Figure 6.10: The density scatter plot of z vs $\acute{z}$ and a box plot of the residual errors on the test set, showing median (bar), inter-quartile range (box) and range (whiskers) for a GPVC model with $m = 100$ using (a) "Normal", (b) "Normalised" and "Balanced" cost-sensitive learning weights.

Table 6.5: Performance measures of training a GPVC model with $m = 100$ and different weighting schemes trained on the simulated survey.

	Normal	Normalised	Balanced
Δz	**0.04926**	0.04938	0.04986
$\widehat{\Delta z}$	0.03124	**0.02137**	0.03340
max	0.56632	1.34341	**0.43961**
$\widehat{\max}$	0.29225	**0.19302**	0.23272
μ	0.00003	0.00163	**-0.00002**
$\hat{\mu}$	-0.00066	0.00051	**-0.00034**
σ	**0.04926**	0.04936	0.04986
$\hat{\sigma}$	0.03123	**0.02137**	0.03340

following SQL statement is used to create the dataset:

```
SELECT
p.objid,
p.modelMag_u, p.modelMag_g,
p.modelMag_r, p.modelMag_i,
p.modelMag_z, p.modelMagerr_u,
p.modelMagerr_g, p.modelMagerr_r,
p.modelMagerr_i, p.modelMagerr_z,
s.z as zspec
INTO
mydb.modelmag_dataset
FROM
PhotoObjAll as p, SpecObj as s
WHERE
p.SpecObjID = s.SpecObjID AND
s.class = 'GALAXY' AND s.zWarning = 0 AND p.mode = 1 AND
dbo.fPhotoFlags('PEAKCENTER') != 0 AND
dbo.fPhotoFlags('NOTCHECKED') != 0 AND
```

```
dbo.fPhotoFlags('DEBLEND_NOPEAK') != 0 AND
dbo.fPhotoFlags('PSF_FLUX_INTERP') != 0 AND
dbo.fPhotoFlags('BAD_COUNTS_ERROR') != 0 AND
dbo.fPhotoFlags('INTERP_CENTER') != 0
```

We use the same quality flags from Brescia et al. (2014) to ensure a clean spectroscopic sample and remove any samples with any missing magnitudes (only 54 sources). The total number of sources after the cuts is 2,120,465, from which we randomly sample 300,000 sources and create three sets for training, validation, and testing consisting of 100,000 samples each. The associated photometric errors are replaced by their natural log to transform their domain from $\mathbb{R}_+$ to $\mathbb{R}$, which has the advantage of reducing the skewness of these features and allows for a fully unconstrained optimisation. The data is also de-correlated using PCA, but all the features are kept with no dimensionality reduction.

6.3.2 Models and Metrics

We move the focus in this section to input-dependent noise prediction and evaluate the models' predictive distribution as a whole, i.e. both the predicted point estimate, as well as the predicted uncertainty (typically reflects a predictive variance) for each source given its photometry. We include additional metrics in this section, shown in Table 6.6, to evaluate the models' predictive distribution (mean and variance): The thresholds of interest in photometric redshift estimation, used for missions' requirements and goals, are $\mathcal{F}_{0.15}$ and $\mathcal{F}_{0.05}$. The log likelihood has the advantage of evaluating the predicted point estimate and variance at the same time. Even though the first term improves with a large variance, it is compensated by subtracting the log of the variance in the second term. The optimal variance for this metric is exactly equal to the squared error, if everything else were to set fixed, which is the desired objective. In this section, we only compare against models that also produce

Table 6.6: Performance metrics used to evaluate the models, where n is the number of samples, z_i is the spectroscopic redshift of sample i, $\acute{z}_i$ is the predicted photometric redshift and $\acute{\sigma}_i^2$ is the predicted variance.

Metric	Equation	Description
$\mathcal{L}$	$\frac{1}{n}\sum_{i=1}^{n} -\frac{(z_i-\acute{z}_i)^2}{2\acute{\sigma}_i^2} - \frac{1}{2}\ln\acute{\sigma}_i^2 - \frac{1}{2}\ln 2\pi$	The mean log likelihood
$\mathcal{F}_\epsilon$	$\frac{100}{n}\left\|\left\{i : \left\|\frac{z_i-\acute{z}_i}{1+z_i}\right\| < \epsilon\right\}\right\|$	Fraction of sources with normalised error$< \epsilon$

predictive variance estimation, we therefore replace the previously used ANNz and stableGP models with the following models:

Artificial Neural Networks for Photo-z v2 (ANNz2): ANNz2 (Sadeh et al., 2015) is an extension to ANNz that utilises many machine learning models such as ANNs, decision trees and k-nearest neighbours. In addition to aggregating results from different machine learning models, ANNz2 trains many versions of the same model but with various configurations, initialisations and optimisation techniques. For instance, the output of many ANNs with different number of layers, number of hidden units and input pre-processing; or the number of trees and sampling methods in random forests. We use the recommended randomised regression script provided with the software release[2], where we include ANNs and boosted decision trees as part of the machine learning models to be trained. The `ANNZ_best` score provided by the output of the code is used as the predictive mean and the square of `ANNZ_best_err` as the predictive variance.

Trees for Photo-z (TPZ): TPZ (Carrasco Kind and Brunner, 2013) is a random forest implementation[3], see Section 5.1.2 for a brief description of random forest models, optimised for photometric datasets. The number of sub-features to randomly

[2] `https://github.com/IftachSadeh/ANNZ`
[3] `https://github.com/mgckind/MLZ`

select for each tree is set to the suggested value of $\sqrt{d} \simeq 4$, where d is the number of input features, and the remaining options are set to their default values since the code is configured for SDSS-like surveys. We use `zmean1` as the predictive mean and the square of `err1` as the predictive variance.

In the experiments that follow, we use m to refer to the number of machine learning models (MLMs) in ANNz2, the number of trees in TPZ and the number of basis functions in SPGP, GPVL, GPVD and GPVC.

6.3.3 Model Complexity

In this subsection, we study the change in the performance metrics as functions of m. The models were trained using $m = \{5, 10, 25, 50, 100, 250, 500\}$. The ANNz2 software was not able to train (crashes) past $m = 100$ for this dataset, hence we only include up to 100 machine learning models for ANNz2. The performance metrics on the test set as m is increased are shown in Figure 6.11. GPVC consistently outperforms the other methods in all metrics (reaching $\Delta z \sim 0.0533$, $\mathcal{L} \sim 1.99$, $\mathcal{F}_{0.05} \sim 91.88$ per cent and $\mathcal{F}_{0.15} \sim 98.8$ per cent). TPZ on the other hand is significantly worse in all metrics ($\Delta z \sim 0.0853$, $\mathcal{L} \sim 1.21$, $\mathcal{F}_{0.05} \sim 68.45$ per cent and $\mathrm{FR}_{0.15} \sim 98.67$ per cent). ANNz2 did not improve for $m > 10$ on all metrics.

6.3.4 Performance Analysis

In this experiment, we analyse the performance of the models when $m = 100$ (the limit of ANNz2). The results are reported in Table 6.7 and the scatter plots for each model are shown in Figure 6.12, where the scatter plots for each method are colour coded by the predictive variance. We also show in Figure 6.13 the $\mathcal{F}_\epsilon$ score for all the models using different values of ϵ. GPVC outperformed all the models on all the metrics, and we also note that richer covariances provide better performances, i.e. GPVL<GPVD<GPVC. SPGP outperformed GPVL on some metrics but not all, whereas TPZ underperformed all the models on all metrics, apart from $\mathcal{F}_{0.15}$ where

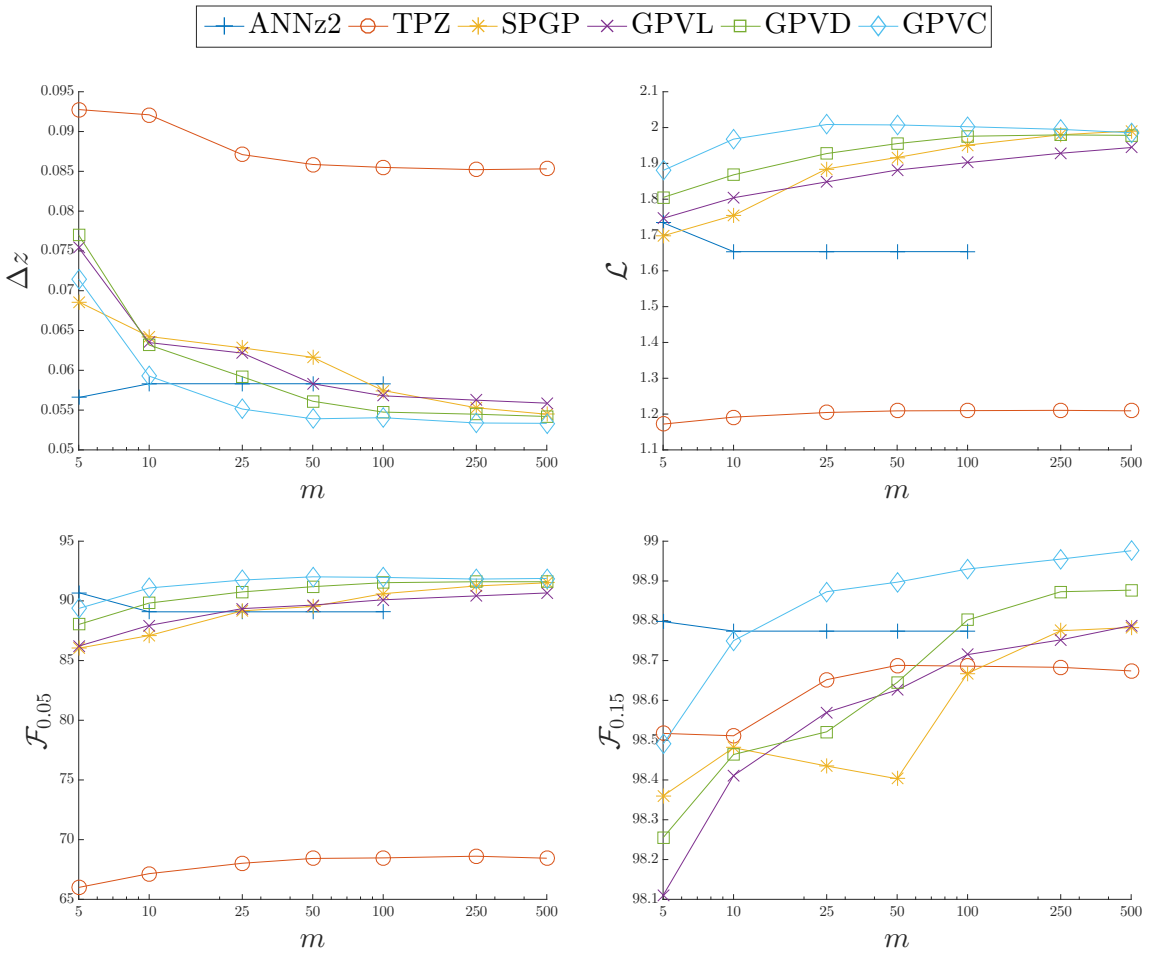


Figure 6.11: The performance metrics of each method on the test set using different values of m.

it slightly outperformed SPGP. ANNz2 underperformed all the proposed GP models (GPVL, GPVD and GPVC) on all performance metrics. For the general $\mathcal{F}_\epsilon$ metrics, TPZ provides the poorest results by a significant margin for low values of ϵ, but asymptotes towards the $\mathcal{F}_{0.15}$ values for the other methods at $\epsilon > 0.1$.

6.3.5 Selection Performance

As indicated from previous experiments, on simulated and real photometric datasets, the various sources of uncertainty (and noise) in measurements and subsequent calculations of photometry and spectroscopy put a lower bound to the accuracy that any model can hope to achieve. Hence, the only way going forward might have to

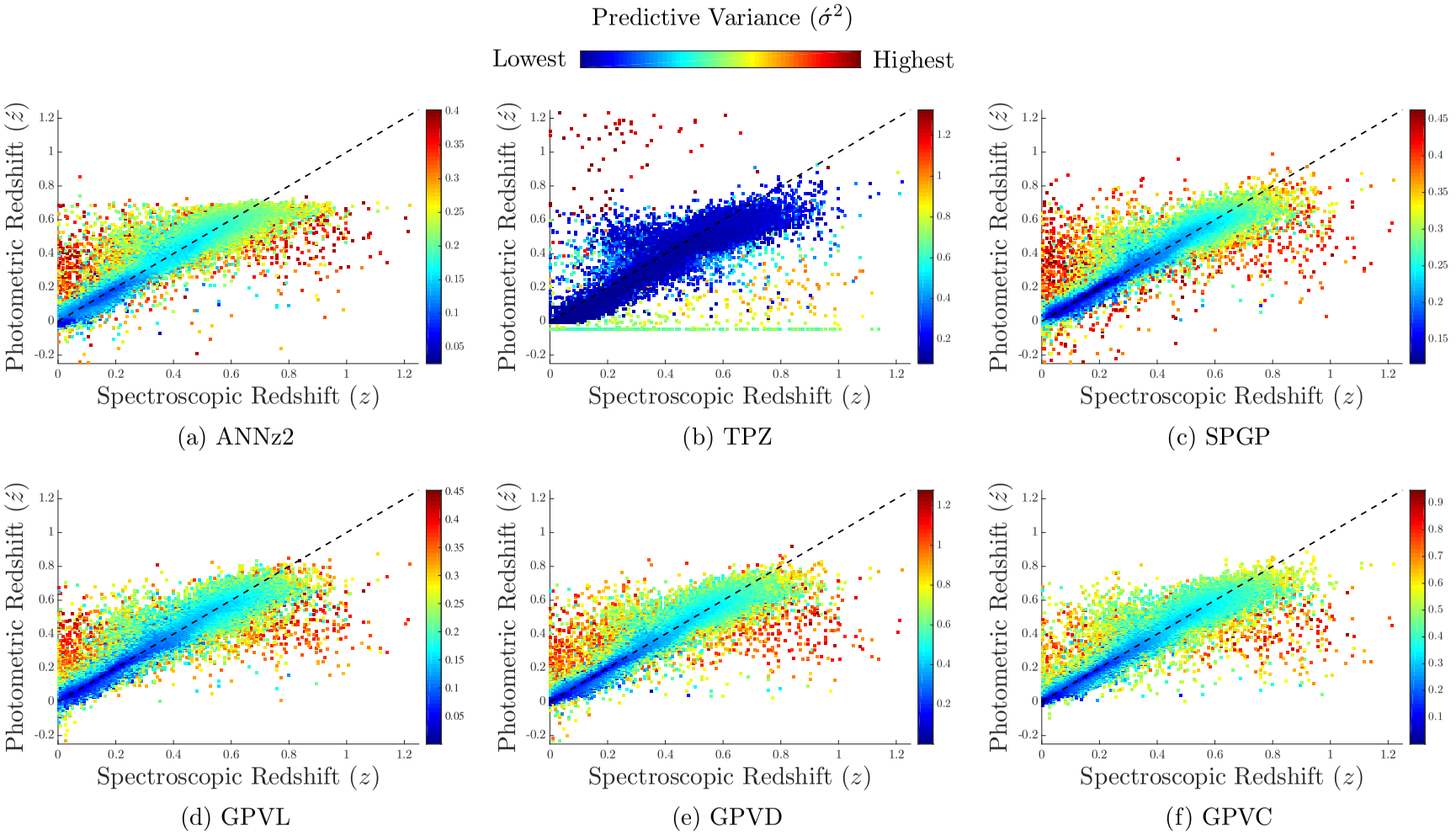


Figure 6.12: The scatter plots of the spectroscopic redshift z versus the predicted photometric redshift $\acute{z}$ on the test set for (a) ANNz2, (b) TPZ, (c) SPGP, (d) GPVL, (e) GPVD and (f) GPVC using $m = 100$. The predictive variance is colour coded, on a log scale, by the value of the predictive variance of each method to avoid colour saturation and enhance visualisation.

Table 6.7: Performance measures for each algorithm trained using $m = 100$ for all the models on the held-out test set. The best-performing method is highlighted in bold font.

	Δz	$\mathcal{L}$	$\mathcal{F}_{0.05}$	$\mathcal{F}_{0.15}$
ANNz2	0.0583	1.6536	89.08%	98.77%
TPZ	0.0855	1.2096	68.47%	98.69%
SPGP	0.0575	1.9509	90.60%	98.67%
GPVL	0.0568	1.9021	90.09%	98.72%
GPVD	0.0547	1.9758	91.52%	98.80%
GPVC	**0.0540**	**2.0024**	**91.97%**	**98.93%**

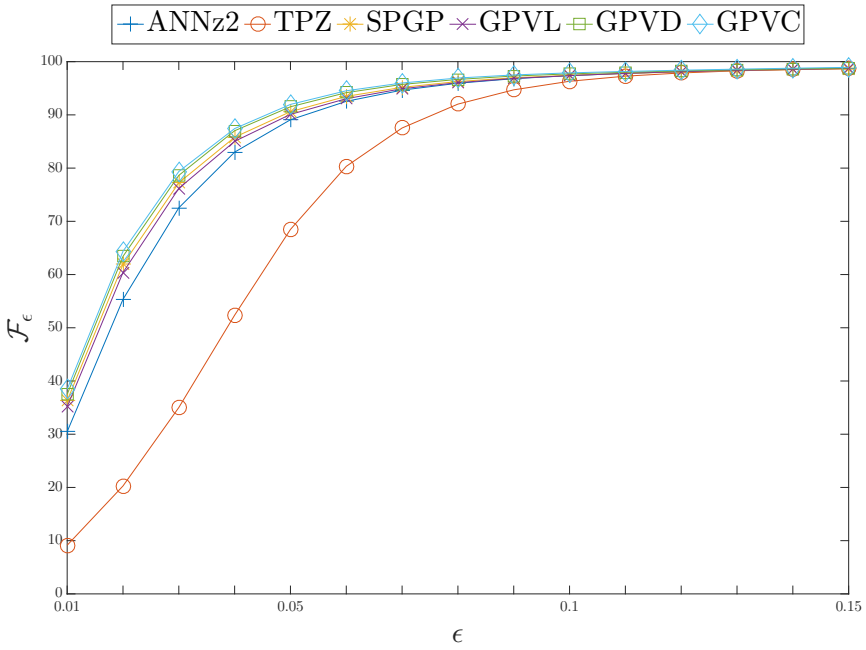


Figure 6.13: The $\mathcal{F}_\epsilon$ for different values of ϵ on the test set for each method with $m = 100$.

rely on input-dependent noise estimation to trade off completeness for accuracy in order to meet the goals of future cosmology experiments. In this section, we evaluate the selection performance of the models as determined by their predictive variance. We would expect, that each model will perform at its best on subsamples of the data with higher confidence predictions and that as more unreliable sources are included the performance should degrade monotonically. Figure 6.14 shows the performance

Table 6.8: The average relative improvement of GPVC over other tested methods on all metrics on the test set using $m = 100$.

	ANNz2	TPZ	SPGP	GPVL	GPVD
Δz	27.617%	60.304%	5.566%	11.883%	3.550%
$\mathcal{L}$	59.210%	87.028%	2.664%	6.578%	1.667%
$\mathcal{F}_{0.05}$	2.094%	35.443%	0.390%	0.929%	0.230%
$\mathcal{F}_{0.15}$	0.071%	0.044%	0.009%	0.015%	0.005%

metrics as functions of the percentage of data selected based on the predictive variance produced by each model using $m = 100$. TPZ performed significantly worse than the other models, except for $\mathcal{F}_{0.15}$ where it outperformed ANNz2. On all other metrics, ANNz2 performs better than TPZ but significantly behind the GP-based models. GPVL is clearly the poorest performing model amongst the GP-based methods, whereas GPVC consistently outperformed all the other models. We also provide in Figure 6.15 the relative improvement of GPVC over the other methods. There is very minor and inconsistent advantage for using the other models over GPVC only on the $\mathcal{F}_{0.15}$ score, which all models perform well on across the range (>99.8%), but for all other metrics there is significant relative improvement when using GPVC over the other models. GPVD generally outperforms SPGP for percentages higher than 30 per cent. The results are quantified by computing the average improvement of GPVC relative to each model, over the entire range, and are reported in Table 6.8. We note that the closest models relative to GPVC on average, in order, are GPVD, SPGP, GPVL, ANNz2, then TPZ.

6.3.6 Uncertainty Analysis

One of the advantages of using GP models for uncertainty analysis over other variance prediction methods, is the ability to separately quantify the amount of uncertainty due to each source of noise; the uncertainty about the function due to data density inhomogeneity ($\nu^2(x)$ from Equation (5.4)) and the noise uncertainty due to lack of

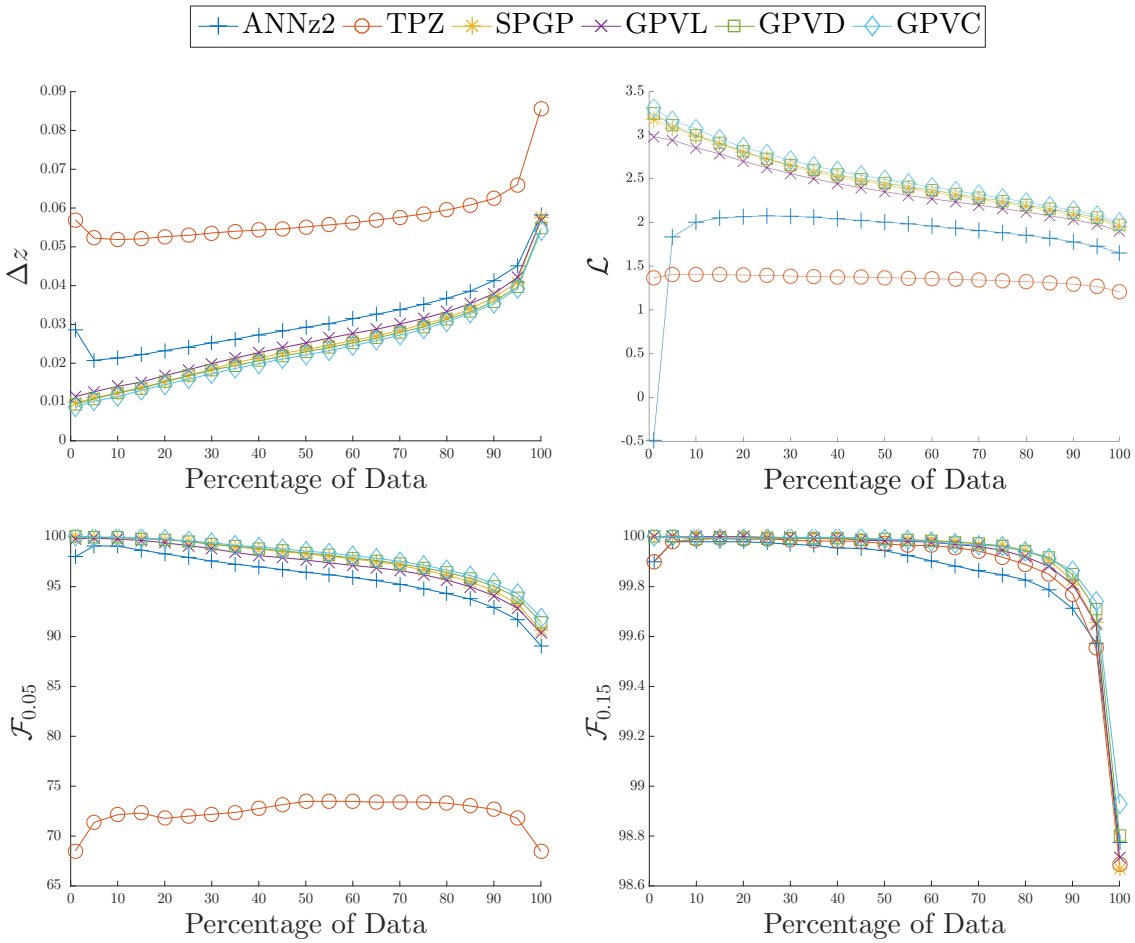


Figure 6.14: The performance metrics as functions of the percentage of data selected based on the predictive variance generated by each method using $m = 100$ on the test set.

information ($\sigma^2(x)$ from Equation (5.3)). The former, the model uncertainty, helps in identifying regions where more data is needed, whereas the latter, the intrinsic noise uncertainty, helps in identifying regions where better or more precise features are needed. The ability to answer these two questions separately is essential to the design of future photometric datasets, the equipment that would acquire that data and the mapping area for future surveys and missions.

In this experiment, we use the GPVC model with $m = 100$ and analyse each component of the variance as functions of redshift. Figure 6.16 shows the model and noise uncertainties as functions of the spectroscopic redshift (z) using uniformly

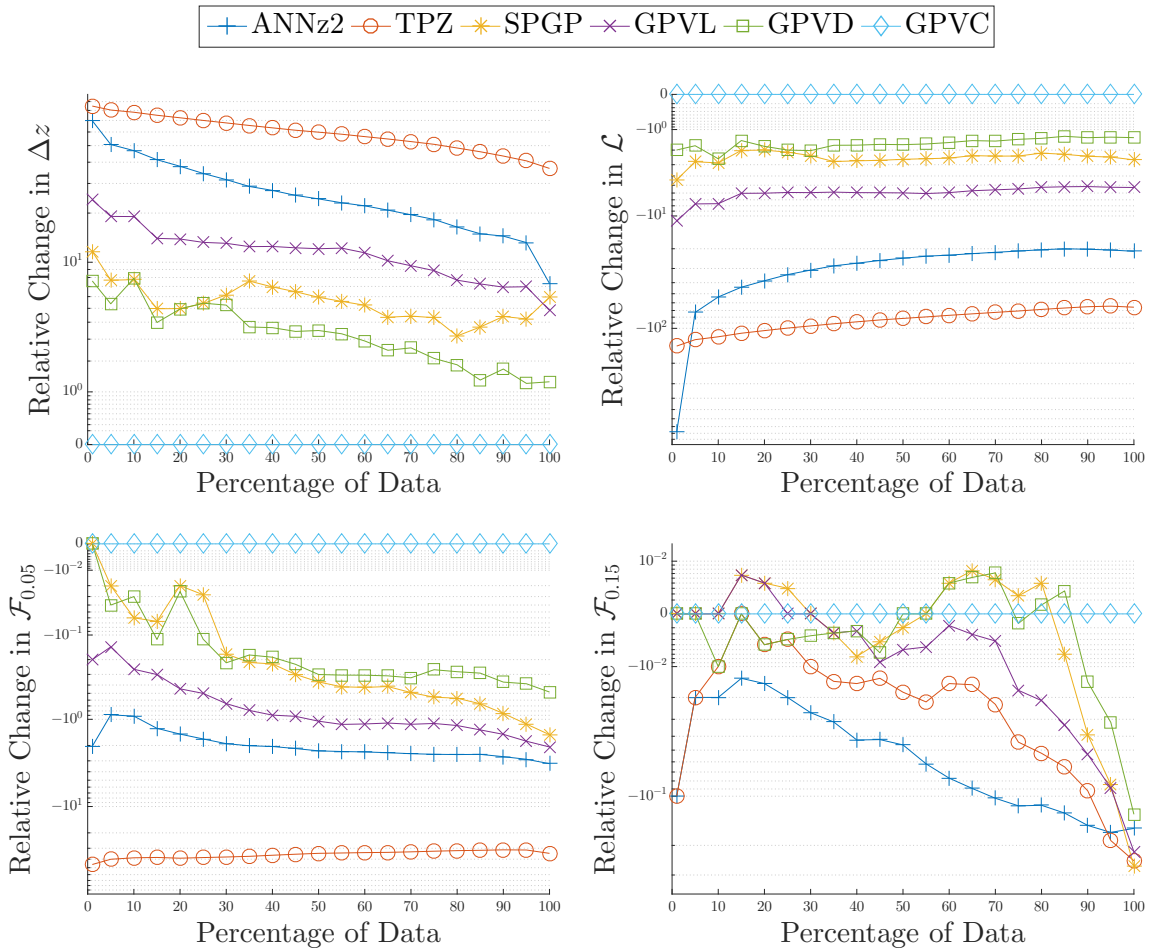
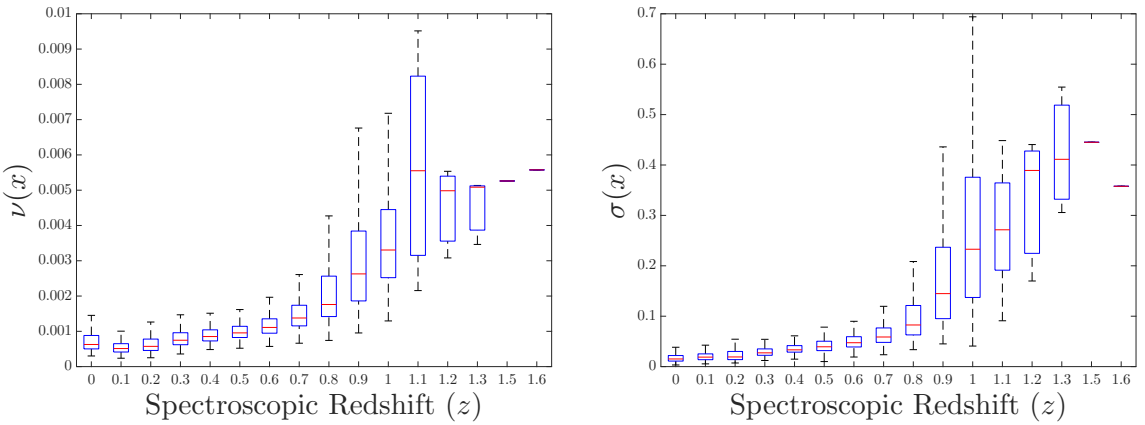


Figure 6.15: The percentage of difference between GPVC and the other methods, computed as $100 \times (\text{Method} - \text{GPVC}) \,/\, |\text{Method}|$ as a function of the percentage of data selected based on the predictive variance generated by each method using $m = 100$ on the test set. The values are plotted on a log-scaled y-axis to enhance visibility.

spaced bins of width 0.1. In both cases, the uncertainty increases drastically beyond $z \sim 0.5$. This is an expected result given the distribution of the spectroscopic sample. However, the results show that the overwhelming contribution to the overall uncertainty for high redshifts is due to the noisy features rather than data density. This indicates that the amount of data is sufficient for the model to be confident about its mean function and we have precise enough features for redshifts < 0.5. For higher redshifts, the results indicate that obtaining more precise, or additional, features (e.g. near-infrared photometry) is a better investment than obtaining, or training on, more



(a) Model uncertainty (b) Noise uncertainty

Figure 6.16: Box plots showing the square root of (a) the model uncertainty, $\nu(x)$, and (b) the noise uncertainty, $\sigma(x)$, as functions of the spectroscopic redshift showing median (bar), inter-quartile range (box) and range (whiskers) on the test set using a GPVC model with 100 basis functions.

samples. However, such a situation will not be the case for most cosmological applications that require photometric redshifts, and having such separable noise terms will aid in determining the optimal approach to ensure that the requisite training samples are in place to address particular scientific problems, from galaxy evolution to various cosmology experiments.

6.4 Missing Photometry

We now evaluate the photometric redshift estimation performance of the GPVC model under scenarios where some of the input photometry magnitudes are missing during training and prediction. We use random forests as a baseline model for comparison due to its ability to handle missing variables during both prediction and training. Both GPVC and random forests are trained using $m = 200$ (number of basis functions/trees) on the same training set used in the previous section.

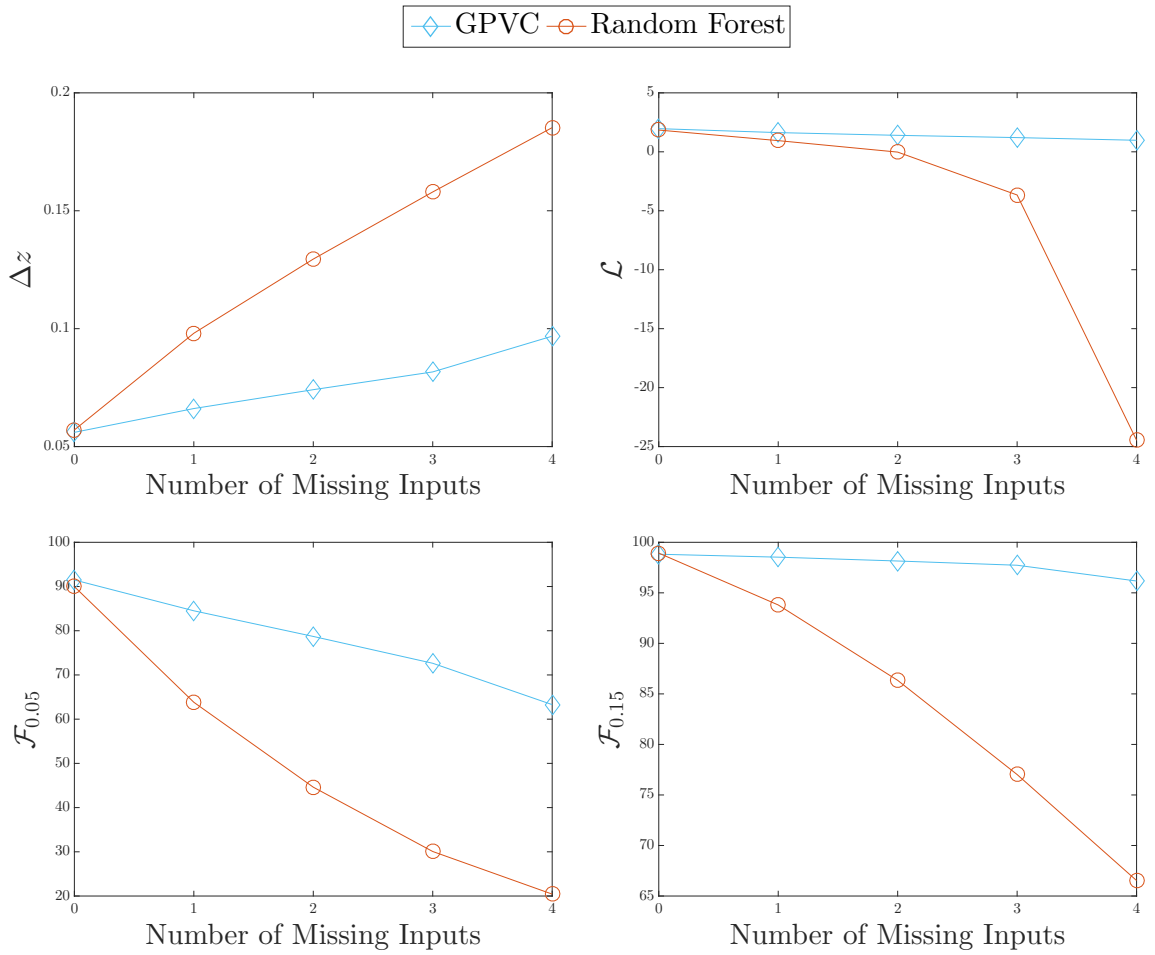
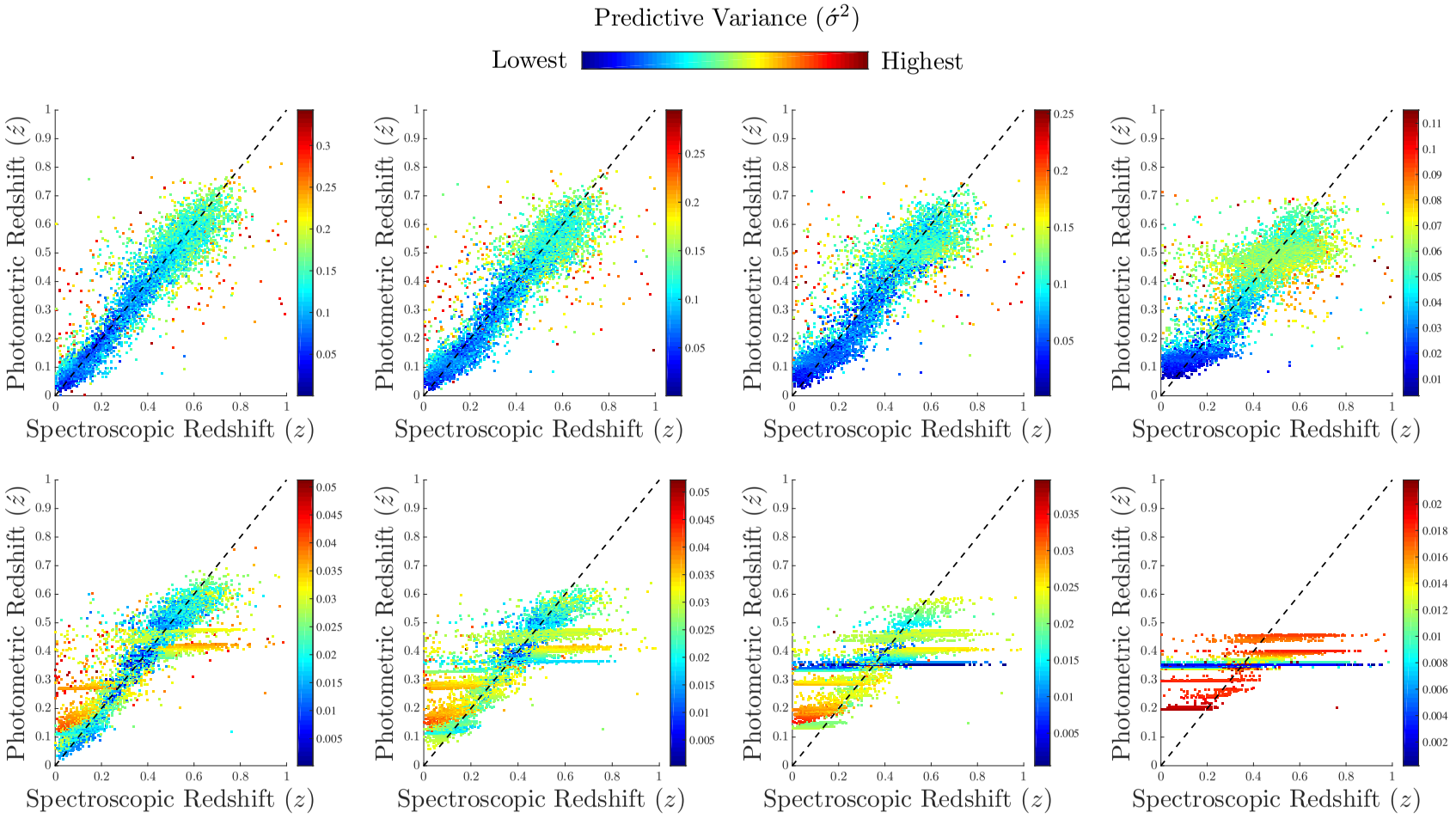


Figure 6.17: The performance metrics of the GPVC and random forests models, using $m = 200$, when trained with the full data and tested on different numbers of missing inputs from 1 to 4.

6.4.1 Predicting with Missing Photometry

In this test, the models are first trained using the full data. Subsequently we remove 1, 2, 3 and 4 filters at random (both the magnitude and the associated error) from all sources. Performance metrics, as the number of missing inputs is increased, are shown in Figure 6.17 and the scatter plots are provided in Figure 6.18. GPVC outperforms random forests on all metrics and the gap increases significantly as the number of missing inputs increase, most notably on the $\mathcal{L}$ metric. From Figure 6.18, random forests appear to cluster sources into bins based on the number of missing inputs, whereas GPVC has a more stable performance.



(b) 1 missing (c) 2 missing (d) 3 missing (e) 4 missing

Figure 6.18: The scatter plots of the GPVC and random forests models, using $m = 200$, when trained with the full data and tested on different numbers of missing inputs from (a) 1 to (d) 4. The plots are colour-coded by the predictive variance, on a log scale, with different scales to avoid colour saturation.

6.4.2 Training with Missing Photometry

In this experiment, we test the photometric redshift estimation performance of GPVC and random forests when trained with missing data and $m = 200$. We remove different percentages of the data at random from 5% to 95% (in an increment of 5%) and test the models using the full data. Different sources might have different numbers of available magnitudes, but the associated error of a removed magnitude is also removed. We remove any source that has all its magnitudes removed before training (both models are trained using the same reduced training sets). Figure 6.19 shows the performance metrics as the percentage of missing data is increased. We note that GPVC maintains its advantage over random forests up to $\sim 60\%$ of missing data on all metrics, except for the $\mathcal{F}_{0.15}$ where they are almost equal up to $\sim 60\%$. Past the 60% threshold, the performance of GPVC drops significantly compared to random forests; thus, for highly sparse datasets random forests still maintains its advantage. This suggests that the proposed approximation, required for training with missing inputs in our model, performs well when the fraction of missing data is not excessive. This appears since the model assumes that the learned parameters may be used to describe a density, against which missing values may be marginalised. To do this sufficient data must be available for it to work robustly. This data percentage varies, as evident from the conducted tests on the synthetic example from **??** and from the experiment conducted here. However, the results suggest that the approach is more accurate than competing approaches for percentages of missing data of up to 50%; which is more than enough for photo-z estimation where the percentage of missing photometry in the training set is often $< 1\%$, but is expected to be much larger in the test set.

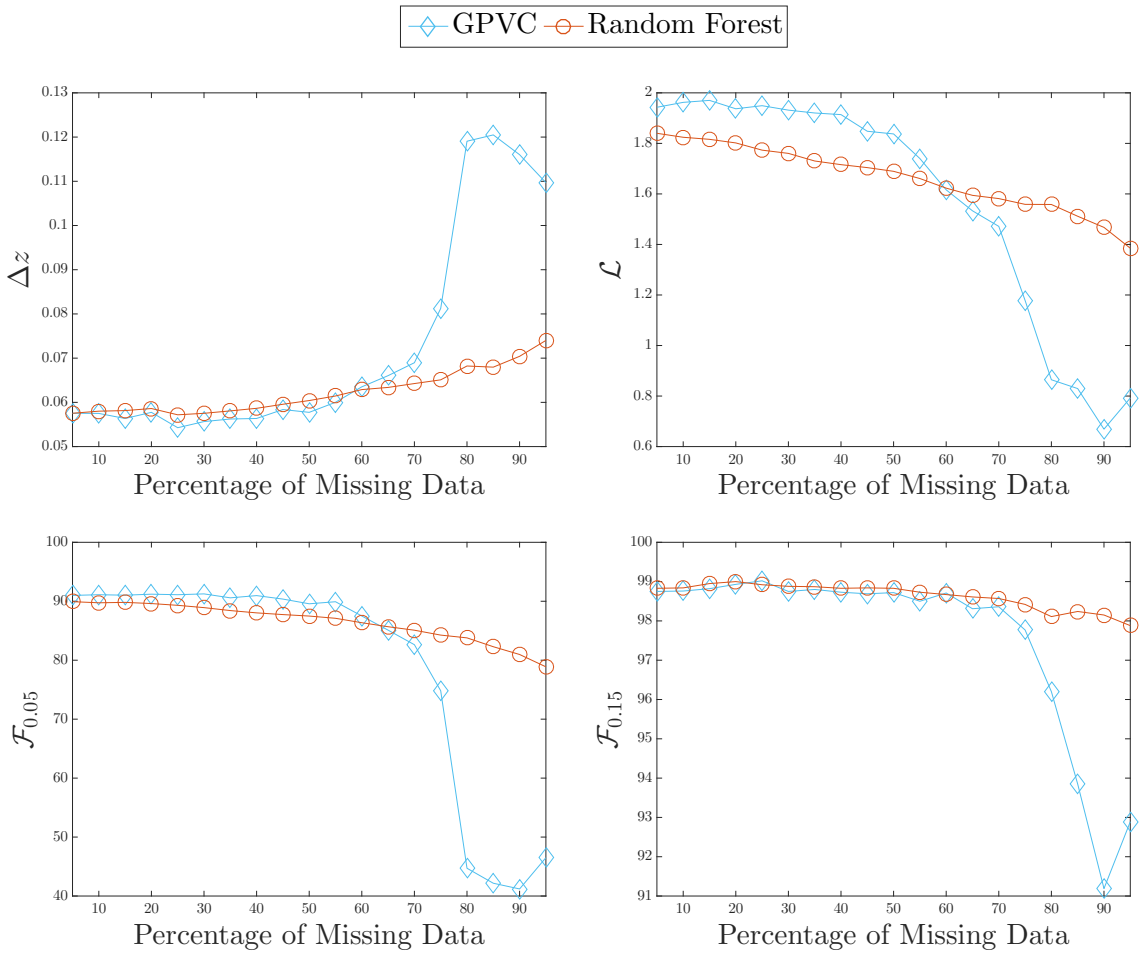


Figure 6.19: The performance metrics of the GPVC and random forests models, using $m = 200$, on predicting with the full data as the percentage of missing data from the training set is increased.

6.5 Summary

This chapter provided an application and testing of the techniques proposed in previous chapters. The methods developed were validated on the problem of photometric redshift estimation and compared against several other methods specifically built for this problem. The proposed model significantly outperforms all the methods in all metrics, which are based on a number of machine learning model such as ANNs, GPs and random forests; and in addition is able to estimate both uncertainty due to density and uncertainty caused by imprecise measurements separately, which can help scientists in the data acquisition process to determine where to collect more data

versus where to collect better data. Moreover, the predictive uncertainty can be used to trade off completeness against accuracy depending on the science objective. The specific photo-z objective of minimising the normalised redshift and the effect of unbalanced training distributions are addressed as a cost-sensitive learning problem and provide significant improvement over minimising the sum of squared errors. The prior mean-function co-optimisation resolved the extrapolation performance in GPs with RBF kernels and now outperforms artificial neural networks. Finally, we introduced the capability of handling missing photometry, not present in any other method apart from random forests. This is particularly useful when using a model trained on one survey to predict the photometric redshift on another survey that does not share the same photometry but overlaps. The proposed approach is orders of magnitude more accurate in recovering the true redshift with missing photometry when compared to random forests. The proposed training procedure with missing variables outperforms random forests for up to 50% of missing data. The implementation of the proposed methods have been published in Python and Matlab packages[4] for the community to use with the ultimate goal of being part of the projects' software pipelines.

[4] `https://github.com/OxfordML/GPz`

Chapter 7

Summary and Future Work

We have introduced in this thesis a number of enhancements to sparse Gaussian processes by recasting them in the context of basis function models to allow for non-stationarity, relevance determination, heteroscedasticity, custom sample weighting and prior mean function co-optimisation. The non-stationarity is achieved by modelling each basis function with bespoke hyper-parameters. The richer covariance function, along with automatic relevance determination, makes it possible to fit more complex models with fewer basis functions which in turn reduces the time complexity even further. By learning an input-dependent noise process jointly with the mean function, the model can dynamically down-weight outliers locally to avoid over-complicating the model unnecessarily and enables the user to dissect the predictive variance into its generating components in order to pinpoint the major contributing source of uncertainty; i.e. whether it is due to lack of data or lack of precision in the data measurements. Furthermore, heteroscedasticity enables us to model strictly positive outputs on the log space more accurately. As shown in Section 3.3, under uncertainty, the best-point estimate of the output depends on both the mean and the variance of its logarithm. Thus, learning variable variance estimation allows us to use the log transform in order to fit functions with strictly positive outputs. The extrapolation performance of sparse GPs is also improved by co-learning a linear mean function that is directly incorporated into the model's formulation rather than

pre-processing the data. This also enables the model to also apply our prior assumption of smoothness directly to the mean function as well during the optimisation procedure and incorporate it into the variance estimate to produce more accurate noise prediction compared to pre-processing the data through linear regression. The cost-sensitive learning framework provides a way to virtually over-sample or under-sample the training set through custom weights in order to apply emphasis to different samples based on the modelling objective.

In addition to improving the model's capabilities in handling output uncertainty, we also provided a principled way to address the problem of noisy and missing inputs by incorporating their uncertainties directly into the model's formulation. Inference from noisy and missing inputs in the context of basis function models with radial basis functions is derived analytically for prediction and, with some approximation, for training. The effectiveness of the proposed treatment is compared against competing models such as random forests, random sampling and Gaussian mixture models on synthetic univariate and bivariate datasets. We also applied the proposed methods to the real-world problem of predicting galactic redshifts from their photometry. The proposed approach achieved state of the art results on both simulated and real photometric redshift datasets across a variety of metrics offering a significant performance advantage over competing methods. The software implementation of the proposed work in this thesis has been published in Matlab and Python on GitHub[1]. The interface can be used for any regression task and we also supplied a workflow for photo-z estimation which we hope will form the basis for Euclid, LSST and future projects.

For future work, we plan to further improve the model's ability to handle missing variables during training. The approximation, although effective for small percentages of missing data, is not effective for extremely sparse datasets. There is room for enhancement in this area and we plan to study how to improve the approximation

[1] `https://github.com/OxfordML/GPz`

further. We also plan to generalise the predictive variance by producing multimodal outputs with more complex distributions. Furthermore, we plan to study how to fit the joint distribution of the inputs and outputs with noisy and/or missing variables so that the predictive distribution of any subset of variables can be inferred from any other given subset instead of having a specific designated output.

Finally, we plan to further investigate other optimisation methods to push the models to their full potential. The focus in this thesis has been on what model is best to optimise rather than how to best optimise a specific model. A possible alternative to optimisation is marginalisation, i.e. imposing a prior on the hyper-parameters and integrating them out from the posterior; which is intractable in this case and, hence, has to be estimated using approximate integration methods such as Markov Chain Monte Carlo (MCMC), variational Bayes (VB) or Bayesian quadrature (BQ).

Appendix A

Probability and Matrix Identities

A.1 Probability Identities

A.1.1 The Chain Rule of Probability

We may factor the joint distribution of n variables, $p\left(x_1, \ldots, x_n\right)$, as follows

$$p(x_1, \ldots, x_n) = p(x_1|x_2, \ldots, x_n)p(x_2|x_3, \ldots, x_n) \ldots p(x_{n-1}|x_n)p(x_n). \tag{A.1}$$

A.1.2 Bayes' Theorem

From the chain rule of probability, the joint probability of x and y can be expressed as:

$$p(x, y) = p(x|y)p(y), \tag{A.2}$$

$$= p(y|x)p(x), \tag{A.3}$$

from which we can derive Bayes' theorem

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}. \tag{A.4}$$

A.1.3 The Marginal Distribution

Given a joint distribution over two variables x and y, the marginal distribution $p(x)$ can be obtain by marginalising the variable y as follows (Rasmussen and Williams,

2006, p. 199):

$$p(x) = \int p(x, y)\mathrm{d}y, \tag{A.5}$$

$$= \int p(x|y)p(y)\mathrm{d}y. \tag{A.6}$$

A.1.4 Expectation and Variance

The expected value of $f(\mathbf{x})$ where $\mathbf{x} \sim p(\mathbf{x})$ is (Papoulis, 1991, p. 105)

$$\mathbb{E}\left[f(\mathbf{x})\right] = \int f(\mathbf{x})p(\mathbf{x})\mathrm{d}\mathbf{x}, \tag{A.7}$$

and the variance is

$$\begin{aligned}\mathbb{V}\left[f(\mathbf{x})\right] &= \mathbb{E}\left[\left(f(\mathbf{x}) - \mathbb{E}\left[f(\mathbf{x})\right]\right)\left(f(\mathbf{x}) - \mathbb{E}\left[f(\mathbf{x})\right]\right)^T\right], \\ &= \mathbb{E}\left[f(\mathbf{x})f(\mathbf{x})^T\right] - \mathbb{E}\left[f(\mathbf{x})\right]\mathbb{E}\left[f(\mathbf{x})\right]^T. \end{aligned} \tag{A.8}$$

A.2 Gaussian Identities

A.2.1 The Marginal Distribution

Consider the following multivariate Gaussian probability density function over $\mathbf{x} \in \mathbb{R}^d$:

$$\mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}\right) = \frac{1}{\sqrt{\left(2\pi\right)^d \left|\boldsymbol{\Sigma}\right|}} \exp\left(-\frac{1}{2}\left(\mathbf{x} - \boldsymbol{\mu}\right)^T \boldsymbol{\Sigma}^{-1}\left(\mathbf{x} - \boldsymbol{\mu}\right)\right), \tag{A.9}$$

for any arbitrary segmentation of $\mathbf{x} = \begin{bmatrix}\mathbf{x}[i] \\ \mathbf{x}[j]\end{bmatrix}$, then the marginal probability distribution is as follows (Rasmussen and Williams, 2006, p. 200):

$$\begin{aligned} p\left(\mathbf{x}[i]\right) &= \int p\left(\mathbf{x}[i], \mathbf{x}[j]\right)\mathrm{d}\mathbf{x}[j], \\ &= \int \mathcal{N}\left(\begin{bmatrix}\mathbf{x}[i] \\ \mathbf{x}[j]\end{bmatrix} \middle| \begin{bmatrix}\boldsymbol{\mu}[i] \\ \boldsymbol{\mu}[j]\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}[i,i] & \boldsymbol{\Sigma}[i,j] \\ \boldsymbol{\Sigma}[j,i] & \boldsymbol{\Sigma}[j,j]\end{bmatrix}\right)\mathrm{d}\mathbf{x}[j], \\ &= \mathcal{N}\left(\mathbf{x}[i]|\boldsymbol{\mu}[i], \boldsymbol{\Sigma}[i,i]\right). \end{aligned} \tag{A.10}$$

A.2.2 The Conditional Distribution

If the probability of $\mathbf{x}[i]$ and $\mathbf{x}[j]$ are jointly Gaussian, then the conditional probability distribution of $\mathbf{x}[i]$ given $\mathbf{x}[j]$ is distributed normal as follows (Rasmussen and Williams, 2006, p. 200):

$$p\left(\mathbf{x}[i]|\mathbf{x}[j]\right) = \mathcal{N}\left(\mathbf{x}[i]|\mathbb{E}\left[\mathbf{x}[i]|\mathbf{x}[j]\right], \mathbb{V}\left[\mathbf{x}[i]|\mathbf{x}[j]\right]\right), \tag{A.11}$$

$$\mathbb{E}\left[\mathbf{x}[i]|\mathbf{x}[j]\right] = \boldsymbol{\mu}[i] + \boldsymbol{\Sigma}[i,j]\boldsymbol{\Sigma}[j,j]^{-1}\left(\mathbf{x}[j] - \boldsymbol{\mu}[j]\right), \tag{A.12}$$

$$\mathbb{V}\left[\mathbf{x}[i]|\mathbf{x}[j]\right] = \boldsymbol{\Sigma}[i,i] - \boldsymbol{\Sigma}[i,j]\boldsymbol{\Sigma}[j,j]^{-1}\boldsymbol{\Sigma}[j,i]. \tag{A.13}$$

A.2.3 The Product of Two Gaussians

The product of two Gaussians with respect to $\mathbf{x}$ is an un-normalised Gaussian with respect to $\mathbf{x}$ (Rasmussen and Williams, 2006, p. 200):

$$\mathcal{N}\left(\mathbf{x}|\mathbf{a},\mathbf{A}\right)\mathcal{N}\left(\mathbf{x}|\mathbf{b},\mathbf{B}\right) = \mathcal{N}\left(\mathbf{a}|\mathbf{b},\mathbf{A}+\mathbf{B}\right)\mathcal{N}\left(\mathbf{x}|\mathbf{c},\mathbf{C}\right), \tag{A.14}$$

$$\mathbf{c} = \mathbf{C}\left(\mathbf{A}^{-1}\mathbf{a} + \mathbf{B}^{-1}\mathbf{b}\right), \tag{A.15}$$

$$\mathbf{C} = \left(\mathbf{A}^{-1} + \mathbf{B}^{-1}\right)^{-1}. \tag{A.16}$$

The product of a Gaussian, with respect to $\mathbf{x}$, with a Gaussian with respect to a linear projection, $\mathbf{Px}$, is an un-normalised Gaussian with respect to $\mathbf{x}$ (Candela and Rasmussen, 2005):

$$\mathcal{N}\left(\mathbf{x}|\mathbf{a},\mathbf{A}\right)\mathcal{N}\left(\mathbf{Px}|\mathbf{b},\mathbf{B}\right) = \mathcal{N}\left(\mathbf{b}|\mathbf{Pa},\mathbf{PAP}^T+\mathbf{B}\right)\mathcal{N}\left(\mathbf{x}|\mathbf{c},\mathbf{C}\right), \tag{A.17}$$

$$\mathbf{c} = \mathbf{C}\left(\mathbf{A}^{-1}\mathbf{a} + \mathbf{P}^T\mathbf{B}^{-1}\mathbf{b}\right), \tag{A.18}$$

$$\mathbf{C} = \left(\mathbf{A}^{-1} + \mathbf{P}^T\mathbf{B}^{-1}\mathbf{P}\right)^{-1}. \tag{A.19}$$

A.2.4 Gaussian Raised to a Power

A Gaussian with respect to $\mathbf{x} \in \mathbb{R}^d$ raised to a power p is an un-normalised Gaussian with respect to $\mathbf{x}$.

$$\begin{aligned}
\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})^p &= \exp\left(-\frac{p}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)(2\pi)^{-\frac{dp}{2}} |\boldsymbol{\Sigma}|^{-\frac{p}{2}}, \\
&= \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \left(p^{-1}\boldsymbol{\Sigma}\right)^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)(2\pi)^{-\frac{dp}{2}} |\boldsymbol{\Sigma}|^{-\frac{p}{2}}, \\
&= \mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}, p^{-1}\boldsymbol{\Sigma}\right)(2\pi)^{\frac{d}{2}} \left|p^{-1}\boldsymbol{\Sigma}\right|^{\frac{1}{2}} (2\pi)^{-\frac{dp}{2}} |\boldsymbol{\Sigma}|^{-\frac{p}{2}}, \\
&= \mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}, p^{-1}\boldsymbol{\Sigma}\right)(2\pi)^{\frac{d(1-p)}{2}} \left|p^{-1}\boldsymbol{\Sigma}\right|^{\frac{1}{2}} |\boldsymbol{\Sigma}|^{-\frac{p}{2}}, \\
&= \mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}, p^{-1}\boldsymbol{\Sigma}\right)(2\pi)^{\frac{d(1-p)}{2}} p^{-\frac{d}{2}} |\boldsymbol{\Sigma}|^{\frac{1}{2}} |\boldsymbol{\Sigma}|^{-\frac{p}{2}}, \\
&= \mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}, p^{-1}\boldsymbol{\Sigma}\right)(2\pi)^{\frac{d(1-p)}{2}} p^{-\frac{d}{2}} |\boldsymbol{\Sigma}|^{\frac{1-p}{2}}, \\
&= \mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}, p^{-1}\boldsymbol{\Sigma}\right)\sqrt{(2\pi)^{d(1-p)} |\boldsymbol{\Sigma}|^{(1-p)} p^{-d}}. \qquad \text{(A.20)}
\end{aligned}$$

A.3 Matrix Identities

A.3.1 Matrix Inversion Lemma

Let $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{D}$ be matrices of sizes $n \times n$, $n \times m$, $m \times m$ and $m \times n$ respectively, if $\mathbf{A}$ and $\mathbf{C}$ are invertible, then the matrix inversion lemma states that (Rasmussen and Williams, 2006, p. 201):

$$(\mathbf{A}+\mathbf{BCD})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}\left(\mathbf{C}^{-1}+\mathbf{DA}^{-1}\mathbf{B}\right)^{-1}\mathbf{DA}^{-1}, \qquad \text{(A.21)}$$

and the determinant can be expressed as (Rasmussen and Williams, 2006, p. 201):

$$|\mathbf{A}+\mathbf{BCD}| = |\mathbf{A}|\,|\mathbf{C}|\left|\mathbf{C}^{-1}+\mathbf{DA}^{-1}\mathbf{B}\right|. \qquad \text{(A.22)}$$

A.3.2 Inverse of a Partitioned Matrix

Let $\mathbf{A}$ be an invertible matrix of size $n \times n$ partitioned as follows:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}[i,i] & \mathbf{A}[i,j] \\ \mathbf{A}[j,i] & \mathbf{A}[j,j] \end{bmatrix},$$

where $\mathbf{A}[i,i]$ is of size $n_i \times n_i$, $\mathbf{A}[i,j]$ is of size $n_i \times n_j$, $\mathbf{A}[j,i]$ is of size $n_j \times n_i$ and $\mathbf{A}[j,j]$ is of size $n_j \times n_j$ such that $n = n_i + n_j$. Then its inverse can also be partitioned into (Rasmussen and Williams, 2006, p. 201):

$$\mathbf{A}^{-1} = \mathbf{B} = \begin{bmatrix} \mathbf{B}[i,i] & \mathbf{B}[i,j] \\ \mathbf{B}[j,i] & \mathbf{B}[j,j] \end{bmatrix},$$

where

$$\mathbf{B}[i,i] = \left(\mathbf{A}[i,i] - \mathbf{A}[i,j]\mathbf{A}[j,j]^{-1}\mathbf{A}[j,i]\right)^{-1}, \tag{A.23}$$

$$\mathbf{B}[i,j] = -\mathbf{B}[i,i]\mathbf{A}[i,j]\mathbf{A}[j,j]^{-1}, \tag{A.24}$$

$$\mathbf{B}[j,i] = -\mathbf{B}[j,j]\mathbf{A}[j,i]\mathbf{A}[i,i]^{-1}, \tag{A.25}$$

$$\mathbf{B}[j,j] = \left(\mathbf{A}[j,j] - \mathbf{A}[j,i]\mathbf{A}[i,i]^{-1}\mathbf{A}[i,j]\right)^{-1}. \tag{A.26}$$

A.3.3 Determinant of a Partitioned Matrix

The determinant of the partitioned matrix $\mathbf{A}$ can be expressed as (Silvester, 2000):

$$|\mathbf{A}| = |\mathbf{A}[i,i]||\mathbf{A}[j,j] - \mathbf{A}[j,i]\mathbf{A}[i,i]^{-1}\mathbf{A}[i,j]|. \tag{A.27}$$

A.3.4 Bounds of a Determinant

If matrix $\mathbf{A}$ is positive definite

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix},$$

then the determinant of $\mathbf{A}$ has the following lower and upper bounds

$$0 < |\mathbf{A}| \leq \prod_{i=1}^{n} a_{ii}. \tag{A.28}$$

This is known as the Hadamard's inequality (Maz'ya and Shaposhnikova, 1999, p. 385).

A.3.5 The Determinant of the Sum of Positive Definite Matrices

Let $\mathbf{A}$ and $\mathbf{B}$ be positive definite matrices of size $n \times n$, then for any vector $\mathbf{x} \in \mathbb{R}^n$

$$\begin{aligned}
\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{x}^T\mathbf{B}\mathbf{x} &> \mathbf{x}^T\mathbf{A}\mathbf{x}, \\
\mathbf{x}^T\left(\mathbf{A}+\mathbf{B}\right)\mathbf{x} &> \mathbf{x}^T\mathbf{A}\mathbf{x}, \\
-\frac{1}{2}\mathbf{x}^T\left(\mathbf{A}+\mathbf{B}\right)\mathbf{x} &< -\frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x}, \\
\exp\left(-\frac{1}{2}\mathbf{x}^T\left(\mathbf{A}+\mathbf{B}\right)\mathbf{x}\right) &< \exp\left(-\frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x}\right), \\
|\mathbf{A}+\mathbf{B}|^{-\frac{1}{2}}\mathcal{N}\left(\mathbf{x}|0,(\mathbf{A}+\mathbf{B})^{-1}\right) &< |\mathbf{A}|^{-\frac{1}{2}}\mathcal{N}\left(\mathbf{x}|0,\mathbf{A}^{-1}\right), \\
\int |\mathbf{A}+\mathbf{B}|^{-\frac{1}{2}}\mathcal{N}\left(\mathbf{x}|0,(\mathbf{A}+\mathbf{B})^{-1}\right)\mathrm{d}\mathbf{x} &< \int |\mathbf{A}|^{-\frac{1}{2}}\mathcal{N}\left(\mathbf{x}|0,\mathbf{A}^{-1}\right)\mathrm{d}\mathbf{x}, \\
|\mathbf{A}+\mathbf{B}|^{-\frac{1}{2}} &< |\mathbf{A}|^{-\frac{1}{2}}, \\
|\mathbf{A}+\mathbf{B}| &> |\mathbf{A}|. \qquad \text{(A.29)}
\end{aligned}$$

Bibliography

F. B. Abdalla, M. Banerji, O. Lahav, and V. Rashkov. A Comparison of Six Photometric Redshift Methods Applied to 1.5 Million Luminous Red Galaxies. *MNRAS*, 417:1891–1903, 2011.

S. Alam, et al. The Eleventh and Twelfth Data Releases of the Sloan Digital Sky Survey: Final Data from SDSS-III. *APJs*, 219:12, July 2015. doi: 10.1088/0067-0049/219/1/12.

I. A. Almosallam, M. J. Jarvis, and S. J. Roberts. GPz: Non-stationary Sparse Gaussian Processes for Heteroscedastic Uncertainty Estimation in Photometric Redshifts. *MNRAS*, 462:726–739, October 2016a. doi: 10.1093/mnras/stw1618.

I. A. Almosallam, S. N. Lindsay, M. J. Jarvis, and S. J. Roberts. A Sparse Gaussian Process Framework for Photometric Redshift Estimation. *MNRAS*, 455:2387–2401, January 2016b. doi: 10.1093/mnras/stv2425.

Ashok Srivastava. stableGP. `https://c3.nasa.gov/dashlink/resources/123/`, October 2010.

N. M. Ball, R. J. Brunner, A. D. Myers, N. E. Strand, S. L. Alberts, and D. Tcheng. Robust Machine Learning Applied to Astronomical Data Sets. III. Probabilistic Photometric Redshifts for Galaxies and Quasars in the SDSS and GALEX. *APJ*, 683:12-21, August 2008. doi: 10.1086/589646.

W. A. Baum. Photoelectric Magnitudes and Red-Shifts. In G. C. McVittie, editor, *Problems of Extra-Galactic Research*, volume 15 of *IAU Symposium*, page 390, 1962.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy Layer-Wise Training of Deep Networks. In Bernhard Schölkopf, John C. Platt, and Thomas Hoffman, editors, *NIPS*, pages 153–160. MIT Press, 2006. ISBN 0-262-19568-2. URL `http://papers.nips.cc/book/advances-in-neural-information-processing-systems-19-2006`.

Chris M Bishop. Training with Noise is Equivalent to Tikhonov Regularization. *Neural Computation*, 7(1):108–116, 1995.

M. Bolzonella, J.-M. Miralles, and R. Pelló. Photometric Redshifts Based on Standard SED Fitting Procedures. *A&A*, 363:476–492, November 2000.

D. G. Bonfield, Y. Sun, N. Davey, M. J. Jarvis, F. B. Abdalla, M. Banerji, and R. G. Adams. Photometric Redshift Estimation using Gaussian Processes. *MNRAS*, 405: 987–994, June 2010. doi: 10.1111/j.1365-2966.2010.16544.x.

C. Bonnett, et al. Redshift Distributions of Galaxies in the DES Science Verification Shear Catalogue and Implications for Weak Lensing. *ArXiv e-prints*, July 2015.

G. B. Brammer, P. G. van Dokkum, and P. Coppi. EAZY: A Fast, Public Photometric Redshift Code. *APJ*, 686:1503-1513, October 2008. doi: 10.1086/591786.

Leo Breiman. Bagging Predictors. *Machine Learning*, 24(2):123–140, 1996. ISSN 1573-0565. doi: 10.1007/BF00058655. URL `http://dx.doi.org/10.1007/BF00058655`.

Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001. ISSN 1573-0565. doi: 10.1023/A:1010933404324. URL `http://dx.doi.org/10.1023/A:1010933404324`.

M. Brescia, S. Cavuoti, G. Longo, and V. De Stefano. A Catalogue of Photometric Redshifts for the SDSS-DR9 Galaxies. *A&A*, 568:A126, August 2014. doi: 10.1051/0004-6361/201424383.

Charles George Broyden. The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations. *IMA Journal of Applied Mathematics*, 6(1): 76–90, 1970.

Joaquin Quiñonero Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959, 2005. URL `http://www.jmlr.org/papers/v6/quinonero-candela05a.html`.

M. Carrasco Kind and R. J. Brunner. TPZ: Photometric Redshift PDFs and Ancillary Information by Using Prediction Trees and Random Forests. *MNRAS*, 432:1483–1501, June 2013. doi: 10.1093/mnras/stt574.

Rich Caruana and Alexandru Niculescu-Mizil. An Empirical Comparison of Supervised Learning Algorithms. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 161–168, New York, NY, USA, 2006. ACM. ISBN 1-59593-383-2. doi: 10.1145/1143844.1143865. URL `http://doi.acm.org/10.1145/1143844.1143865`.

M. Colless, et al. The 2dF Galaxy Redshift Survey: Final Data Release. *ArXiv Astrophysics e-prints*, June 2003.

A. A. Collister and O. Lahav. ANNz: Estimating Photometric Redshifts Using Artificial Neural Networks. *PASP*, 116:345–351, April 2004. doi: 10.1086/383254.

George Cybenko. Approximation by Superpositions of a Sigmoidal Function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Alexander Davies. *Effective Implementation of Gaussian Process Regression for Machine Learning*. PhD thesis, University of Cambrige, 2014.

S. P. Driver, et al. Galaxy and Mass Assembly (GAMA): Survey Diagnostics and Core Data Release. *MNRAS*, 413:971–995, May 2011. doi: 10.1111/j.1365-2966.2010.18188.x.

F. W. Dyson, A. S. Eddington, and C. Davidson. A Determination of the Deflection of Light by the Sun's Gravitational Field, from Observations Made at the Total Eclipse of May 29, 1919. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 220(571-581):291–333, 1920. ISSN 0264-3952. doi: 10.1098/rsta.1920.0009. URL `http://rsta.royalsocietypublishing.org/content/220/571-581/291`.

Edward Snelson. Sparse pseudo-input Gaussian processes (SPGP). `http://www.gatsby.ucl.ac.uk/~snelson/`, April 2007.

Dumitru Erhan, Yoshua Bengio, Aaron C. Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why Does Unsupervised Pre-training Help Deep Learning? *Journal of Machine Learning Research*, 11:625–660, 2010. URL `http://doi.acm.org/10.1145/1756006.1756025`.

ESA/Hubble & NASA. LRG 3-757. `http://www.spacetelescope.org/images/potw1151a/`, December 2011.

ESA/Hubble & NASA. HE0435-1223. `https://www.spacetelescope.org/images/heic1702a/`, January 2017.

R. Feldmann, et al. The Zurich Extragalactic Bayesian Redshift Analyzer and its First Application: COSMOS. *MNRAS*, 372:565–577, October 2006. doi: 10.1111/j.1365-2966.2006.10930.x.

A. E. Firth, O. Lahav, and R. S. Somerville. Estimating Photometric Redshifts with Artificial Neural Networks. *MNRAS*, 339:1195–1202, March 2003. doi: 10.1046/j.1365-8711.2003.06271.x.

Roger Fletcher. A New Approach to Variable Metric Algorithms. *The Computer Journal*, 13(3):317–322, 1970.

Leslie Foster, et al. Stable and Efficient Gaussian Process Calculations. *Journal of Machine Learning Research*, 10:857–882, 2009. doi: 10.1145/1577069.1577100. URL `http://doi.acm.org/10.1145/1577069.1577100`.

J. E. Geach. Unsupervised Self-Organized Mapping: a Versatile Empirical Tool for Object Selection, Classification and Redshift Estimation in Large Surveys. *MNRAS*, 419:2633–2645, January 2012. doi: 10.1111/j.1365-2966.2011.19913.x.

M. Gibbs and D. J. C. MacKay. Efficient Implementation of Gaussian Processes. Technical report, Cavendish Laboratory, Cambridge, UK, 1997.

Donald Goldfarb. A Family of Variable-metric Methods Derived by Variational Means. *Mathematics of Computation*, 24(109):23–26, 1970.

H. Hildebrandt, et al. PHAT: PHoto-z Accuracy Testing. *A&A*, 523:A31, November 2010. doi: 10.1051/0004-6361/201014885.

Hinton and Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *SCIENCE: Science*, 313, 2006.

Geoffrey Hinton, et al. Deep Neural Networks for Acoustic Modeling in Speech Recognition. *IEEE Signal Processing Magazine*, 29(6):82–97, November 2012. URL `http://research.microsoft.com/apps/pubs/default.aspx?id=171498`.

R. Hogan, M. Fairbairn, and N. Seeburn. GAz: a Genetic Algorithm for Photometric Redshift Estimation. *MNRAS*, 449:2040–2046, May 2015. doi: 10.1093/mnras/stv430.

Edwin Hubble. A Relation Between Distance and Radial Velocity Among Extra-galactic Nebulae. *Proceedings of the National Academy of Sciences*, 15(3):168–173, 1929. doi: 10.1073/pnas.15.3.168. URL `http://www.pnas.org/content/15/3/168.short`.

O. Ilbert, et al. Accurate Photometric Redshifts for the CFHT Legacy Survey Calibrated Using the VIMOS VLT Deep Survey. *A & Ap*, 457:841–856, October 2006. doi: 10.1051/0004-6361:20065138.

Z. Ivezic, et al. LSST: from Science Drivers to Reference Design and Anticipated Data Products. *ArXiv e-prints*, May 2008.

I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, New York, 1986.

S. Jouvel, et al. Designing Future Dark Energy Space Missions. I. Building Realistic Galaxy Spectro-Photometric Catalogs and their First Applications. *A&A*, 504: 359–371, September 2009. doi: 10.1051/0004-6361/200911798.

Kristian Kersting, Christian Plagemann, Patrick Pfaff, and Wolfram Burgard. Most Likely Heteroscedastic Gaussian Process Regression. In Zoubin Ghahramani, editor, *Proceedings of the 24th Annual International Conference on Machine Learning (ICML 2007)*, page 393, Corvallis, OR, 2007. Omnipress.

Alex Krizhevsky and Geoffrey E. Hinton. Using very Deep Autoencoders for Content-based Image Retrieval. In *Proceedings of the 19th European Symposium on Artificial Neural Networks*, Bruges, Belgium, April 2011. ESANN. URL `https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2011-10.pdf`.

Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. Sampling Methods for the Nyström Method. *Journal of Machine Learning Research*, 13(Apr):981–1006, 2012.

R. Laureijs, et al. Euclid Definition Study Report. *ArXiv e-prints*, October 2011.

O. Le Fèvre, et al. The VIMOS VLT Deep Survey Final Data Release: a Spectroscopic Sample of 35016 Galaxies and AGN out to $z \sim 6.7$ Selected with $17.5 \leq i_{AB} \leq 24.75$. *A&A*, 559:A14, November 2013. doi: 10.1051/0004-6361/201322179.

O. Le Fèvre, et al. The VIMOS Ultra-Deep Survey: ∼10,000 Galaxies with Spectroscopic Redshifts to Study Galaxy Assembly at Early Epochs $2 < z \simeq 6$. *A&A*, 576: A79, April 2015. doi: 10.1051/0004-6361/201423829.

Simon J. Lilly, et al. The zCOSMOS 10k-Bright Spectroscopic Sample. *The Astrophysical Journal Supplement Series*, 184(2):218, 2009. URL `http://stacks.iop.org/0067-0049/184/i=2/a=218`.

D. Masters, et al. Mapping the Galaxy Color-Redshift Relation: Optimal Photometric Redshift Calibration Strategies for Cosmology Surveys. *APJ*, 813:53, November 2015. doi: 10.1088/0004-637X/813/1/53.

Vladimir G Maz'ya and Tatiana Olegovna Shaposhnikova. *Jacques Hadamard: a Universal Mathematician*. Number 14 in History of Mathematics. American Mathematical Soc., 1999.

Andrew Mchutchon and Carl E. Rasmussen. Gaussian Process Training with Input Noise. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 1341–1349. Curran Associates, Inc., 2011. URL `http://papers.nips.cc/paper/4295-Gaussian-process-training-with-input-noise.pdf`.

James Mercer. Functions of Positive and Negative Type, and their Connection with the Theory of Integral Equations. *Philosophical transactions of the royal society of London*, pages 415–446, 1909.

Kevin P Murphy. *Machine Learning: A Probabilistic Perspective*. Adaptive Computation and Machine Learning. MIT press, 2012.

NASA/JPL-Caltech. Gravitational Lensing Diagram. `http://herschel.cf.ac.uk/results/herschel-atlas-gravitational-lenses`, 2010.

Jorge Nocedal. Updating Quasi-Newton Matrices with Limited Storage. *Mathematics of Computation*, 35:773–773, 1980. doi: 10.2307/2006193.

Bruno A. Olshausen and Dacid J. Field. Emergence of Simple-Cell Receptive Field Properties by Learning a Sparse Code for Natural Images. *Nature*, 381(6583): 607–609, 1996.

Athanasios Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, New York, NY, US, 3rd edition, December 1991.

Jooyoung Park and Irwin W Sandberg. Universal Approximation using Radial-Basis-Function Networks. *Neural Computation*, 3(2):246–257, 1991.

William D Penny and Stephen J Roberts. Neural Network Predictions with Error Bars. Technical Report TR-97-1, Imperial College London, Department of Electrical and Electronic Engineering, London, UK, February 1997.

K. B. Petersen and M. S. Pedersen. The Matrix Cookbook. `http://www2.imm.dtu.dk/pubdb/p.php?3274`, November 2012. Version 20121115.

Marc'Aurelio Ranzato, Christopher S. Poultney, Sumit Chopra, and Yann LeCun. Efficient Learning of Sparse Representations with an Energy-Based Model. In Bernhard Schölkopf, John C. Platt, and Thomas Hoffman, editors, *NIPS*, pages

1137–1144. MIT Press, 2006. ISBN 0-262-19568-2. URL http://papers.nips.cc/book/advances-in-neural-information-processing-systems-19-2006.

Carl Edward Rasmussen and Hannes Nickisch. Gaussian Processes for Machine Learning (GPML) Toolbox. *Journal of Machine Learning Research*, 11:3011–3015, 2010. URL http://portal.acm.org/citation.cfm?id=1953029.

C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. Adaptative computation and machine learning series. MIT Press, Cambridge, MA, 2006.

M. M. Rau, S. Seitz, F. Brimioulle, E. Frank, O. Friedrich, D. Gruen, and B. Hoyle. Accurate Photometric Redshift Probability Density Estimation - Method Comparison and Application. *MNRAS*, 452:3710–3725, October 2015. doi: 10.1093/mnras/stv1567.

S Roberts, M Osborne, M Ebden, S Reece, N Gibson, and S Aigrain. Gaussian Processes for Time-Series Modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1984):20110550, 2013.

Stephen J Roberts, William Penny, and David Pillot. Novelty, Confidence and Errors in Connectionist Systems. In *Intelligent Sensors (Digest No: 1996/261), IEE Colloquium on*, pages 10/1–10/6. IET, September 1996.

I. Sadeh, F. B. Abdalla, and O. Lahav. ANNz2 - Photometric Redshift and Probability Density Function Estimation using Machine Learning Methods. *ArXiv e-prints*, July 2015.

C. Sánchez, et al. Photometric Redshift Analysis in the Dark Energy Survey Science Verification Data. *MNRAS*, 445:1482–1506, December 2014. doi: 10.1093/mnras/stu1836.

Anton Schwaighofer and Volker Tresp. Transductive and Inductive Methods for Approximate Gaussian Process Regression. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *NIPS*, pages 953–960. MIT Press, 2002. ISBN 0-262-02550-7. URL `http://papers.nips.cc/book/advances-in-neural-information-processing-systems-15-2002`.

Matthias W. Seeger, Christopher K. I. Williams, and Neil D. Lawrence. Fast Forward Selection to Speed Up Sparse Gaussian Process Regression. In Christopher M. Bishop and Brendan J. Frey, editors, *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics, AISTATS 2003, Key West, Florida, USA, January 3-6, 2003*. Society for Artificial Intelligence and Statistics, 2003. ISBN 0-9727358-0-1. URL `http://research.microsoft.com/conferences/aistats2003/`.

Howard Seltman. Approximations for Mean and Variance of a Ratio. *unpublished note*, 2012.

David F Shanno. Conditioning of Quasi-Newton Methods for Function Minimization. *Mathematics of Computation*, 24(111):647–656, 1970.

John R Silvester. Determinants of Block Matrices. *The Mathematical Gazette*, 84 (501):460–467, 2000.

Alexander J. Smola and Peter L. Bartlett. Sparse Greedy Gaussian Process Regression. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *NIPS*, pages 619–625. MIT Press, 2000. URL `http://papers.nips.cc/book/advances-in-neural-information-processing-systems-13-2000`.

Edward Snelson and Zoubin Ghahramani. Sparse Gaussian Processes using Pseudo-inputs. In Y. Weiss, B. Schölkopf, and J. C. Platt, editors, *Advances in Neural*

Information Processing Systems 18, pages 1257–1264. MIT Press, Cambridge, MA, 2006.

Edward Snelson, Carl Edward Rasmussen, and Zoubin Ghahramani. Warped Gaussian Processes. *Advances in Neural Information Processing Systems*, 16:337–344, 2004.

Richard Socher. *Recursive Deep Learning for Natural Language Processing and Computer Vision*. PhD thesis, Stanford University, 2014.

Michael E. Tipping. Sparse Bayesian Learning and the Relevance Vector Machine. *Journal of Machine Learning Research*, 1:211–244, 2001. URL `http://www.ai.mit.edu/projects/jmlr/papers/volume1/tipping01a/abstract.html`.

T. Tsiligkaridis and A.O. Hero. Covariance Estimation in High Dimensions Via Kronecker Product Expansions. *Signal Processing, IEEE Transactions on*, 61(21): 5347–5360, Nov 2013. ISSN 1053-587X. doi: 10.1109/TSP.2013.2279355.

Vanzella, E., et al. Photometric redshifts with the Multilayer Perceptron Neural Network: Application to theHDF-S and SDSS. *Astronomy & Astrophysics*, 423(2): 761–776, 2004. doi: 10.1051/0004-6361:20040176.

Christian Walder, Kwang In Kim, and Bernhard Schölkopf. Sparse Multiscale Gaussian Process Regression. In William W. Cohen, Andrew McCallum, and Sam T. Roweis, editors, *ICML*, volume 307 of *ACM International Conference Proceeding Series*, pages 1112–1119. ACM, 2008. ISBN 978-1-60558-205-4. URL `http://doi.acm.org/10.1145/1390156.1390296`.

M. J. Way, L. V. Foster, P. R. Gazis, and A. N. Srivastava. New Approaches to Photometric Redshift Prediction Via Gaussian Process Regression in the Sloan Digital Sky Survey. *APJ*, 706:623–636, November 2009. doi: 10.1088/0004-637X/706/1/623.

Gary Weiss, Kate McCarthy, and Bibi Zabar. Cost-Sensitive Learning vs. Sampling: Which is Best for Handling Unbalanced Classes with Unequal Error Costs? In Robert Stahlbock, Sven F. Crone, and Stefan Lessmann, editors, *DMIN*, pages 35–41. CSREA Press, 2007.

Wikimedia Commons. Absorption Lines of BAS11 and the Sun. `https://commons.wikimedia.org/wiki/File:Redshift.png`, 2005.

WA Wright. Bayesian Approach to Neural-Network Modeling with Input Uncertainty. *IEEE Transactions on Neural Networks*, 10(6):1261–1270, 1999.

Y. Zhang, W.E. Leithead, and D.J. Leith. Time-series Gaussian Process Regression Based on Toeplitz Computation of $O(N^2)$ Operations and $O(N)$-level Storage. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 3711–3716, Seville, Spain, 2005. IEEE. ISBN 0-7803-9567-0. doi: 10.1109/CDC.2005.1582739.