Noise Modelling and Outlier Detection in Satellite Scatterometer Data

Robert Jon Bullen

MSc (by Research) in Pattern Analysis and Neural Networks

ASTON UNIVERSITY

October 2001

Acknowledgements

I would like to acknowledge the help of the individuals who made the undertaking of this project and the writing of this thesis possible.

Primarily I would like to thank my supervisor, Dr Dan Cornford for his guidance, support and patience throughout this year. In addition, also for his supervisory role, Dr Ian Nabney who was my primary supervisor during the early stages of the project but whose continued support was most appreciated.

During the course of the project, I have made much use of NetLab, written by Ian Nabney and Chris Bishop. The existence of this suite of programmes has saved me a lot of work.

On a personal note, I would also like to pay tribute to my girlie, Christine Styles for her patience and encouragement and to my Mother, who, like all mothers, never doubted my ability to succeed for a moment.

Finally, I wish to acknowledge all the members of the Neural Computing Research Group for providing a friendly environment in which to work.

This document was compiled using MiKTeX 2.1 which in turn uses LaTeX2e. It was written using the enormously friendly WinEdt 5.

ASTON UNIVERSITY

Noise Modelling and Outlier Detection in Satellite Scatterometer Data

Robert Jon Bullen

MSc (by Research) in Pattern Analysis and Neural Networks, 2001

Thesis Summary

This thesis describes a project in two parts. The project is an application of data modelling techniques to satellite scatterometer data. Scatterometer data is computed from radiation returned to a satellite as a result of its emitted radar beam interacting with the surface of oceans. The frequency of the beam is fixed to react to ripples on the water surface which are a result of the surface wind conditions, therefore the amount of power reflected back to the satellite is dependent on the surface wind vectors. The satellite has three radar beams so the scatterometer data is 3-dimensional and is believed to lie close to a cone-like manifold.

The two parts of the project are:

1. The application of a Generative Topographic Mapping (GTM) to the data in order to create a manifold in data space which can be used identify outliers for the purposes of exclusion from further processing.

2. The modelling of the noise variance in order to see if there is any variation depending on the position within the data. The noise is modelled using an iterative technique involving 2 interacting neural networks, one modelling the mean of the data and one modelling the noise.

Keywords: pattern analysis, RBF, GTM, scatterometer, outlier

Contents

List of Figures

List of Tables

Chapter 1

Overview

This thesis is split into two parts:

- Application of a Generative Topographic Mapping (GTM) to satellite data with the intention of ascertaining whether it is a suitable technique for identifying/excluding outliers.
- Modelling the output noise of the satellite data, treating it as input dependent.

The thesis will begin with an introduction, covering the basics of satellite scatterometer theory including the forward model which will be used in the project, the principle of GTM and its proposed application to the scatterometer problem which is the aim of the first part of the project.

In Chapter 3 changes to the latent space of the GTM are detailed to take account of the prior knowledge of the shape of the satellite data. In addition the method of initialization and training of the model will be detailed and an investigation into ways of maximizing the effectiveness by varying the GTM will follow. Targeting of outliers using the trained model will then be dealt with. Finally in this chapter, the results of the application of the GTM are discussed; this concludes the first part of the project.

In the second part, the output noise variance will be modelled as input dependent. Chapter 4 describes adaptations to the currently utilized scatterometer model, 2 methods of modelling input dependent noise and their application to a toy problem and to the scatterometer data. The results of this part of the project are discussed at the end of this chapter.

Finally the two parts of the project are brought together in Chapter 5 in which what has been learned from the project is discussed and some ideas for future work are proposed.

Chapter 2

Introduction

2.1 Scatterometer

Satellite scatterometer observations are derived from the backscatter radiation 'reflected' from ocean surfaces. The satellite emits radiation at a frequency set to interact with the ripples on the ocean's surface which are dependent on the surface wind conditions. The amount of backscattered radiation received by the satellite varies according to the surface wind vectors. The observations are measured by the satellite's on-board scatterometer. In this case information from the ERS-1 satellite, which was launched in July 1991 is used. The information is backscatter radiation at a frequency of $5.3\,GHz$, the power of which is affected by ripples on the ocean's surface of approximately $5\,cm$. Figure 2.1 shows the geometry of the radar scan; a swathe of $500\,km$ is swept in nineteen $50\,km$ tracks which gives rise to some overlap. Individual readings come from a $50\,km$ cell within each track.

There are 3 radar beams mounted on the satellite. The fore and aft beams are angled at + and − 45 deg respectively to the mid beam. The incidence angles (to the normal) of tracks $1-19$ are $25-59$ deg for the fore and aft beams and $18-47$ deg for the mid beam. In the course of experiments carried out during this project, models are constructed for tracks 1, 10 and 19. All diagrams are taken from the model trained on the data from track 10.

Each cell is therefore scanned 3 times from different angles and the resulting aggregate reading is referred to as the σ^0 triple which comprises σ^0_f, σ^0_m and σ^0_a. When

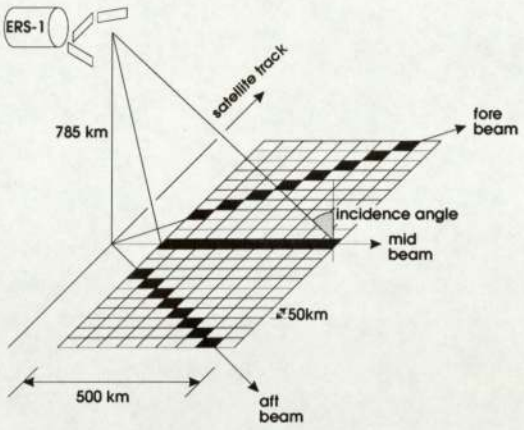


Figure 2.1: ERS-1 satellite showing geometry of fore, aft and mid beams, the incidence angle (to the normal) and the scanning pattern.

plotted, the σ^0 readings appear to lie in a self intersecting double cone [6] (Figure 2.2).

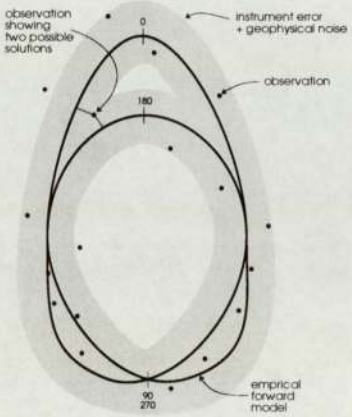


Figure 2.2: A theoretical cross-section of the scatterometer data showing the relationship between relative wind direction and position around the conical surface. The wind speed is related to the position of the data along the axis of the cone. Outliers may fall inside or outside the cone, or between its surfaces.

Outliers are due to various geophysical phenomena (e.g. ice) and to measurement noise and it is these which the first part of this project aims to identify in order to exclude them from further processing.

The scatterometer forward model, which will be used in the second part of the project, is a variation of NN3CMOD [2]. This is a hybrid model based on the functional forward model CMOD4 [6]. The aim of a forward model is to predict the scatterometer

data as accurately as possible from various wind and observation parameters. The nature of the double cone of backscatter data may be understood from the basic functional form established by empirical studies

$$\sigma^0_{\text{lin}} \sim b_0(s,\theta) + b_1(s,\theta)\cos(\chi) + b_2(s,\theta)\cos(2\chi), \tag{2.1}$$

where s is the wind speed, θ the satellite beam incidence angle and χ is the wind direction, relative to the satellite azimuth angle. This model computes σ^0_{lin}, the linear scaled output; as the model develops, the log scaled output, σ^0_{dB} will be used. The position along the axis of the cone is mainly dominated by s while the $\cos(2\chi)$ term generates the resultant σ^0 value around the cone surface, the dual path coming from a moderating $\cos(\chi)$ term which contributes the upwind or downwind component[1]. CMOD4 is the most commonly used functional mapping between the wind vectors and the scatterometer data and develops the form of (2.1)

$$\sigma^0_{\text{lin}} = B_0[1 + B_1(s,\theta)\cos(\chi) + B_3(s,\theta)\cos(2\chi)]^{1.6}, \tag{2.2}$$

the result being raised to the power 1.6 compensate for the series (effectively a Fourier series) being truncated to two terms.

The hybrid model NN3CMOD retains the Fourier terms of CMOD4 but improves upon the calculation of the coefficients from the original method, which used Legendre polynomials, by using an MLP network [1] in order to learn them,

$$\sigma^0_{lin} = a_0[1 + 0.37\tanh(a_1)\cos(\chi) + 0.62\tanh(a_2)\cos(2\chi)]^p, \tag{2.3}$$

where p, a_0, a_1 and a_2 are the outputs from the the neural network and the tanh function ensures σ^0_{lin} remains real for all inputs. If this is now converted to log space, the noise, which is assumed to be a function of σ^0_{lin}, becomes additive.

$$\sigma^0_{dB} = \frac{10}{\ln(10)}\Big(a_0 + p\ln[1 + 0.37\tanh(a_1)\cos(\chi) + 0.62\tanh(a_2)\cos(2\chi)]\Big). \tag{2.4}$$

[1]This explains the maximum discrepancy between 0 deg and 180 deg positions around the cone section (Figure 2.2).

This was called NN2CMOD where the inputs to the neural net were $\log(s)$ and $\sin(\theta)$ and the outputs were as stated above. These outputs, in addition to χ were then the inputs to the functional part of the model, the output from which was σ^0_{dB}.

The improvement in NN3CMOD was to use two models, one for fore and aft satellite beams and one for the mid beam; it has been estimated [2] that the noise in the data from the mid beam was higher than that of the fore and aft ones and was not therefore modelled well by NN2CMOD. It is NN3CMOD which is the starting point for the second part of the project.

2.2 Generative Topographic Mapping

2.2.1 The GTM Model

The Generative Topographic Mapping (GTM) [7] is traditionally used for visualizing high dimensional data in a lower dimensional space. The principle, illustrated in Figure 2.3, is that all points in an L-dimensional latent space are mapped non-linearly to the D-dimensional data space[2]. In this instance the latent space is 2-dimensional and the data space 3-dimensional.

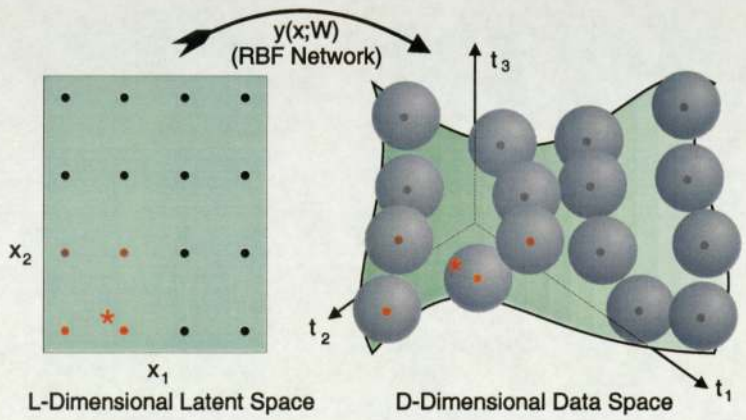


Figure 2.3: The non-linear mapping between the latent and data space is shown with the noise model in data space varying with the density of the data. A data point (represented by the (red) star) will have high posterior probability towards its closest noise model centres.

[2]Normally $L \leq D$.

A grid of K selected points $\{\mathbf{x_1}, \ldots, \mathbf{x}_K\}$ is set up in the latent space and mapped to the centres of a Gaussian Mixture Model (GMM) [1] in data space $\{\mathbf{y_1}, \ldots, \mathbf{y}_K\}$, the ordering of the mixture model is constrained by the grid of points in latent space such that they lie on an L-dimensional manifold within the data. The mapping can be performed by any continuous differentiable function but the usual and most convenient form is of a Radial Basis Function (RBF) network [1] with M fixed basis functions ϕ_m, $m = 0, \ldots, M-1$

$$y(\mathbf{x}, \mathbf{W}) = \sum_{m}^{M} \phi(\mathbf{x}) w_m. \tag{2.5}$$

The network has L input units, therefore $\mathbf{x}$ will be a set of L column vectors, each containing K elements. ϕ is referred to as the activation and is usually a non-linear function. In this project two functions will be considered. The Gaussian

$$\phi(\mathbf{x}) = \exp\left(-\frac{\mathbf{x}^2}{2\sigma^2}\right), \tag{2.6}$$

in which σ is the variance of the function. This is the most common form of activation and is an example of a localized function with the property that $\phi \to 0$ as $|x| \to \infty$. In addition the Thin Plate Spline (TPS) will be used for comparison purposes

$$\phi(x) = \mathbf{x}^2 \ln(\mathbf{x}), \tag{2.7}$$

which has the property that $\phi \to \infty$ as $|x| \to \infty$.

The centres of the activation functions, more commonly known as RBF centres, are usually positioned to span the data. This can be adapted so that they take account of different densities in the data, that is that they are closer together in areas of high data density and vice versa. The term (ϕ_0) is a bias term having a value 1 and is included to account for an offset of the function.

Referring back to (2.5), w_m is the set of weights and biases which act on the activation to produce the final mapping. Part of the training of the GTM, which is

discussed in Section 2.3, is to optimize the weights in the RBF network. The latent space mapping through the RBF network may be expressed in matrix form as

$$\mathbf{Y} = \mathbf{\Phi}\mathbf{W}, \tag{2.8}$$

where $\mathbf{Y}$ is a $K \times D$ matrix of mixture component centres[3] , $\mathbf{\Phi}$ is a $K \times M$ matrix of basis function activations and $\mathbf{W}$ an $M \times D$ matrix of weights and biases.

The mixture model in data space has an isotropic Gaussian noise distribution, given by

$$\begin{aligned} p(\mathbf{t}|\mathbf{x}, \mathbf{W}, \beta) &= \mathcal{N}(\mathbf{y}(\mathbf{x}, \mathbf{W}), \beta) \\ &= \left(\frac{\beta}{2\pi}\right)^{D/2} \exp\left[-\frac{\beta}{2}\sum_{d}^{D}(t_d - y_d(\mathbf{x}, \mathbf{W}))^2\right]. \end{aligned} \tag{2.9}$$

This gives a probability of a data point $\mathbf{t}$ dependent upon its proximity to the manifold[4]. The variance of the noise is given by β^{-1}; this will vary the probability of a data point for a given distance from the manifold. By integrating out $\mathbf{x}$, the probability distribution in data space can be expressed in terms of the parameters β and $\mathbf{W}$

$$p(\mathbf{t}|\mathbf{W}, \beta) = \int p(\mathbf{t}|\mathbf{x}, \mathbf{W}, \beta)p(\mathbf{x})\, \mathrm{d}\mathbf{x}. \tag{2.10}$$

If the prior, $p(\mathbf{x})$ is taken as a set of K equally weighted delta functions

$$p(\mathbf{x}) = \frac{1}{K}\sum_{k=1}^{K}\delta(\mathbf{x} - \mathbf{x}_k), \tag{2.11}$$

the integral becomes a sum

$$p(\mathbf{t}|\mathbf{W}, \beta) = \sum_{k=1}^{K} p(\mathbf{t}|\mathbf{x}_k, \mathbf{W}, \beta). \tag{2.12}$$

The log-likelihood may then be expressed as

[3]The network has D output units.

[4]The manifold is the result of the mapping $(\mathbf{y})$ of $\mathbf{x}$ into data space.

$$l = \sum_{n=1}^{N} \ln\left(\frac{1}{K}\sum_{k=1}^{K} p(\mathbf{t}_n|\mathbf{x}_k, \mathbf{W}, \beta)\right), \tag{2.13}$$

assuming an i.i.d. set of target data points $\{\mathbf{t}_1, \ldots, \mathbf{t}_N\}$.

If, for instance, a Multi Layer Perceptron (MLP) [1] is used in place of the RBF, the optimization of the mixture model could be carried out using any non-linear optimization technique. However, it is more efficient when using a mixture of Gaussians and an RBF network to use an Expectation Maximization (EM) algorithm to fit the manifold to the data.

2.3 The GTM EM Algorithm

2.3.1 Expectation

This step involves the calculation of the responsibilities of the k^{th} Gaussian mixture component for the generation of the n^{th} data point and, using (2.11), (2.12) and Bayes' Theorem, it is expressed as

$$r_{kn} = p(\mathbf{x}_k|\mathbf{t}_n, \mathbf{W}, \beta) = \frac{p(\mathbf{t}_n|\mathbf{x}_k, \mathbf{W}, \beta)p(\mathbf{x}_k)}{\sum_{k'} p(\mathbf{t}_n|\mathbf{x}_{k'}, \mathbf{W}, \beta)p(\mathbf{x}_{k'})}. \tag{2.14}$$

The prior terms will cancel as they were taken to be 'flat', i.e. an evenly weighted sum of $1/K$. So r_{kn} represents the responsibilities of centre k for generating data point n.

2.3.2 Maximization

Using (2.5), (2.9) and (2.14), the derivatives of the log-likelihood (2.13) with respect to the weights may be expressed as

$$\frac{\partial l}{\partial w_m} = \sum_{n,k=1}^{N,K} r_{kn}\beta\left(\sum_{m'}^{M} \phi_{m'}(\mathbf{x}_k)w_{m'} - t_n\right)\phi_m(\mathbf{x}_k). \tag{2.15}$$

A detailed explanation of the differentiation process is given in Appendix A of [7]. By setting (2.15) to zero and using (2.8) we can get the update formula for $\mathbf{W}$ as

$$\mathbf{\Phi}^T\mathbf{G}\mathbf{\Phi}\mathbf{W} = \mathbf{\Phi}^T\mathbf{R}\mathbf{T}, \tag{2.16}$$

where $\mathbf{T}$ is an $N \times D$ matrix containing the data points, $\mathbf{R}$ is a $K \times N$ matrix of responsibilities and $\mathbf{G}$ is a $K \times K$ diagonal matrix with responsibilities summed over all points and components defined by

$$g_{kk} = \sum_{n=1}^{N} r_{kn}. \tag{2.17}$$

In addition, it is possible to add a regularization (weight decay) factor α in order to improve generalization in training. This modifies (2.16) in the following way

$$(\mathbf{\Phi}^T\mathbf{G}\mathbf{\Phi} + \alpha\mathbf{I})\mathbf{W} = \mathbf{\Phi}^T\mathbf{R}\mathbf{T}, \tag{2.18}$$

where α is the inverse variance of the prior and $\mathbf{I}$ is the identity matrix with the same dimensions as $\mathbf{\Phi}^T\mathbf{G}\mathbf{\Phi}$. This is equivalent to restricting $\mathbf{W}$ by placing an isotropic prior upon it of the following form

$$p(\mathbf{W}) = \left(\frac{\alpha}{2\pi}\right)^{W/2} \exp\left(-\frac{\alpha}{2}\|\mathbf{W}\|^2\right). \tag{2.19}$$

Equation (2.18) can then be solved for $\mathbf{W}$ updating the weights in the RBF and thus changing the mapping such that the mixture model centres are globally closer to the data. Effectively, the aim is to move the centres of the mixture model towards the point where they have maximum responsibility for generating the data.

The partial derivative of the log-likelihood (error) function (2.14) with respect to the noise variance β may be used in a similar way, yielding

$$\frac{1}{\beta} = \frac{1}{ND}\sum_{n,k=1}^{N,K} r_{kn}\|\boldsymbol{y}(\boldsymbol{x}_k, \widehat{\boldsymbol{W}}) - \boldsymbol{t}_n\|^2, \tag{2.20}$$

where $\widehat{\mathbf{W}}$ stands for the updated weights.

This outlines one step of the EM algorithm. By iterating the steps in the following way, the GTM is trained until the probability of the data given the model is at a maximum.

1. Generate latent points $\mathbf{x}$. The latent points (and the subsequent RBF centres) are usually arranged in a regular grid spanning the latent space. The particular latent space in these experiments will be described in Section 3.2

2. Create RBF network, setting basis function centres (and widths, σ, in the case of the Gaussian activations). Compute activations using $\mathbf{x}$.

3. Create an initial set of mixture model centres using the target data if possible and compute the matrix of weights $\mathbf{W}$, from (2.8).

4. Initialize β; a suitable value for this is half the average distance between initialized mixture centres.

5. If applicable, assign a regularization factor, α. If no regularization is required, then $\alpha = 0$.

6. Compute the responsibilities for the model generating the data using (2.14). *E-step*

7. Update $\mathbf{W}$ and β using (2.18) and (2.20). *M-step.*

8. Repeat 6 and 7 until convergence.

2.4 Using GTM for Outlier Detection

The GTM is a density model which lends itself to visualizing data in high dimensions, and this is indeed a common use. However it seems feasible that, if the manifold is to be trained to the solution which gives the highest probability of the data given the model (the maximum likelihood solution) then points in a test data set with low probability,

$p(\mathbf{t}|\mathbf{W}, \beta)$, can be considered outliers. We have said that the satellite scatterometer data is conical in shape so the intention is to train a GTM using this prior knowledge. Outliers in a test set of data are then identified/excluded on the basis of their low probability given the model.

The above sounds simple enough; however it may be worth taking some time to discuss what an outlier is (there may be several conflicting definitions) and to look at the mechanics of training which may be affected by them.

An outlier, from the point of view of a GTM is purely judged on its weighted distance to the sum of the GMM centres, as it is from this that its probability is computed. However more generally one might say that an outlier is a point that is not in the 'correct' position, given its peers. It may be therefore that there are points which lie in the main cloud of scatterometer data σ^0, and therefore be close to the manifold, but that are in the 'incorrect' position using some other criterion. In fact, as will be discussed later, in wind vector space, it perfectly possible for this to happen. It is worth bearing this in mind when we come to a critical appraisal of using the GTM for this purpose.

If a set of data contains outliers, by any definition, the maximum likelihood method of training (a GTM for instance) using the sum-of-squares error function ($\frac{1}{2}\sum_{n=1}^{N}\{y_n - t_n\}^2$) will certainly be influenced by them. This may also have some bearing on the effectiveness of this method and could be cause for caution when evaluating the results. Even if one is fortunate enough to have a training set which has had outliers manually removed (as we have), the criteria used to identify them may still have a bearing on the effectiveness of training.

Chapter 3

Outlier Detection Using GTM

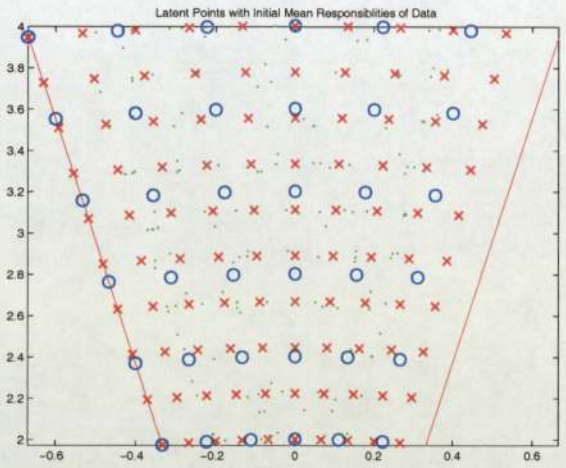
3.1 Introduction

We have said that the satellite scatterometer data lies close to a conical manifold. It seems sensible, therefore, to use this information to tailor the GTM. To this end, it was decided that instead of arranging the latent space points and RBF centres in a rectangular lattice, as is traditional for a 2-dimensional latent space, the latent space would be laid out as the development of a cone and that distances between points would be made as if across the conical surface. $p(\mathbf{x})$ still retains the property defined in (2.11) and so we can still treat the probability distribution in data space as a sum. In this chapter these changes will be detailed along with the initialization and training of the GTM. In addition, the results of experiments involving changes in the parameters and mapping function of the GTM will be shown.

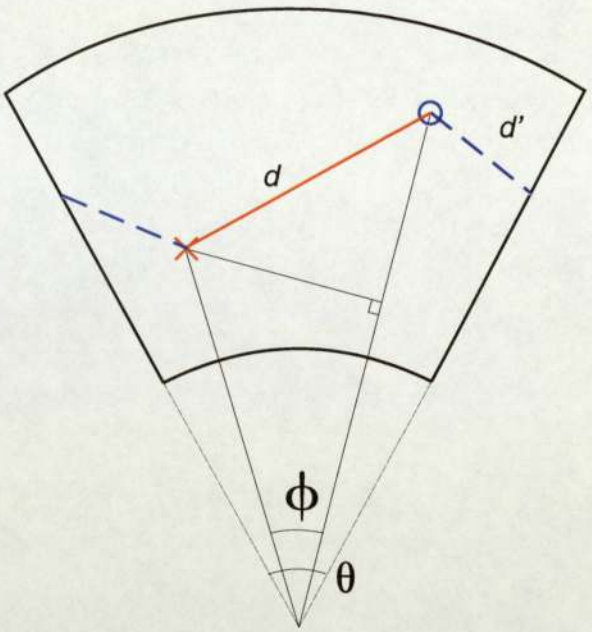
3.2 Changes in Latent Space

Essentially it is the input side of the RBF network which is concerned with measurement in latent space. This is required during any mapping to determine distances from latent points to RBF centres for the purposes of calculating activations. Figure 3.1(a) shows the revised conical development layout of the latent space.

The latent space was constructed as the development of the cone used in the ini-



(a) Latent Space Geometry



(b) Measurement in Latent Space

Figure 3.1: In the left figure the (red) crosses show the arrangement of latent points, the (blue) circles, a similar arrangement of RBF centres and the (green) dots the mean responsibility of the currently positioned centres in data space for the data points. The two edges of the developed cone are coincident in data space; therefore there is a space, equivalent to a column of latent points before the right hand edge to avoid duplication of points along this line. In the right hand figure the distance between (for instance) a latent point (shown as a (red) cross) and an RBF centre (the (blue) circle) is measured as the lesser of distances d and d', calculated using ϕ and θ.

tialization of the GTM (see Section 3.3.2), scaled to fit in a square of side 2 units. The polar coordinates of the relevant points were used to calculate distances and the Cartesian coordinates stored only for displaying the latent space. Because the two sides of the conical development are concurrent, a space, the equivalent of an extra column of points, was included on one side; this avoided duplicated columns of points where the cone joins in data space. When calculating distances, the distance across the conical surface in both directions must be considered, so two calculations were made as shown in Figure 3.1(b).

Comparison was made between the included angle of the two points being considered (ϕ) and the total included angle of the conic surface development (θ).

$$\phi = \begin{cases} \phi & \text{if } \phi \leq \theta/2, \\ \theta - \phi & \text{otherwise.} \end{cases} \tag{3.1}$$

The simple geometry involved in the calculation of the distance d or d' is based on the resultant angle ϕ and the shortest distance around the cone's surface is calculated.

3.3 Initialization and Training of GTM Manifold

3.3.1 Introduction

The above technique was initially developed on a toy problem, using a bullet shaped set of 163 synthetically generated points based on a cone with a tanh profiled surface including additive Gaussian noise with a variance of 0.01. The technique was then transferred to the scatterometer data, using training data sets of 964, 954 and 936 points for tracks 1, 10 and 19 respectively. Outliers had been manually removed on this training set. The test error was evaluated on a test set, also with outliers removed, comprising 478, 476 and 464 points for each of the 3 tracks.

3.3.2 Initialization

An initial conical set of target coordinates for the RBF was created in data space using the Principal Components (PCs, see Appendix A) of the data scaled to give reasonable alignment. This technique is commonly used for dimensionality reduction, where the data is mapped onto a lower dimension spanned by the eigenvectors associated with the largest eigenvalues. In our case, however it is utilized to construct a framework of vectors on which to build an initialized conic manifold. Again, we can make use of our prior knowledge of the shape of the data; the largest PC will be roughly along the axis of the cone (the cone is longer than its base is wide) and the second on a plane which intersects the data approximately parallel to its base. The third will lie on the same plane, orthogonal to the second. Initially a cone is constructed, with

arbitrary[1] alignment and with length of axis and radius of base taken from the two largest eigenvalues. This is then mapped onto the three eigenvectors of the covariance matrix (PCs) with the mean of the grid of GMM centres coincident with that of the data. The cone in 3-space is an example of a data set with an intrinsic dimensionality of 2. If PCA was being used for dimensionality reduction, it would not be able to detect the intrinsic dimensionality as the 2 PCs with the smallest eigenvalues are so similar. In our case, since we are not using it for this purpose, this is not a problem.

The noise variance parameter of the GMMs (β) was set to the smaller of the following

- Half the distance between centres
- The magnitude of the third PC[2].

Figure 3.2 shows the initialized cone.

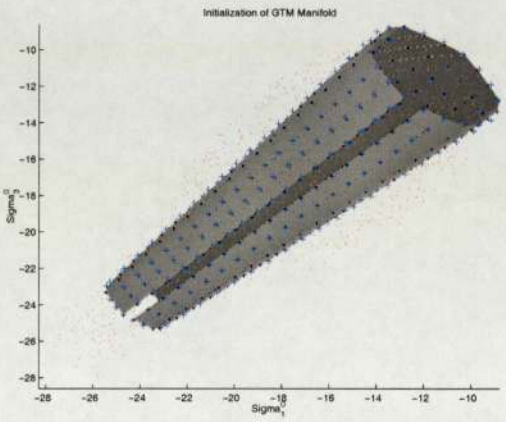


Figure 3.2: The initialized conical manifold lying along the main axis (first principal component) with large and small diameters computed using scaled second principal component. The (blue) crosses represent the targets, to which the initialized RBF was mapped; these are coincident (nearly) to the mixture model centres. The view is along the mid beam axis. The cloud of data points are shown as (red) dots.

[1]With its axis running along the y axis of the data coordinates.

[2]In practice, the magnitude of the third PC will always be similar to the second due to the conical shape of the data.

3.3.3 Training

Training of the GTM used 30 iterations[3] of the standard GTM EM algorithm as detailed in [7]. Figure 3.3 shows the manifold after training. The model was then applied to test data ($\mathbf{t}^t = \mathbf{t}_1^t, \ldots, \mathbf{t}_N^t$) which had been manually processed to remove outliers, and the log likelihood test error computed (2.13).

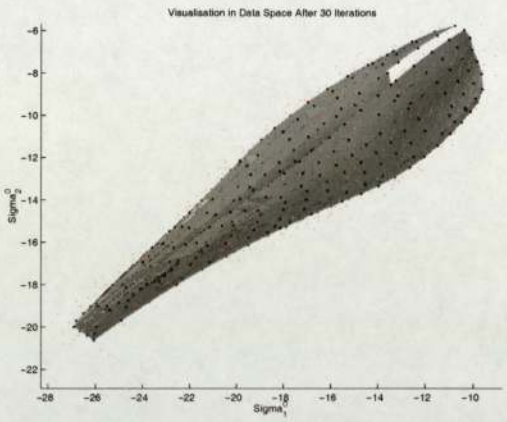


Figure 3.3: After 30 iterations of the EM algorithm the manifold has aligned itself with the data.

Experimentation resulted in the choice of a combination of 36 RBF centres on a 6 by 6 grid (which seemed to adequately span the dimension and scale of the data) and 400 latent points (GMM components) which together proved to be a good compromise between computational effort and minimum test error (See Section 3.5). Experiments were carried out with increasing numbers of latent points, including some in which there were more GMM components than points in the data set. This did not result in a significant reduction in training error, nor in final variance of the elements of the GMM (β), which would have indicated a tighter, more accurate noise model. With fewer components in the model, β rapidly increased indicating that the convergence of the GTM was subject to a higher degree of uncertainty.

[3]Due to good initialization, convergence can feasibly be achieved, regardless of various considerations (track etc.), within this number.

3.4 Hyperparameters, Activations and Generalization

The character of a GTM is fundamentally controlled by three parameters.

- The number of RBF functions.
- α - The prior distribution over $\mathbf{W}$, introduced in (2.18), which controls the magnitude of the weights and penalizes sharp curvatures. This is often referred to as a weight decay factor.
- σ - The variance of RBF functions (2.6) which controls the 'stiffness' of the manifold in data space. This parameter only exists with Gaussian activations in the RBF.

The parameters in the last two methods are commonly referred to as hyperparameters.

The first of these can be fairly roundly dismissed as a fairly blunt tool for improving generalization and the model can be more flexibly controlled by varying the latter two hyperparameters. Experiments were carried out in hyperparameter space to see how the initialization and training of the model were affected.

3.4.1 Hyperparameters

The GTM with Gaussian RBF activations was trained with all combinations of the following values, determined by experimentation

- $\alpha = 0, 0.001, \ldots, 0.1$
- $\sigma = 0.4, 0.48, \ldots, 1.2$

and the resulting test error plotted.

From Figure 3.4, which shows the effect of the two hyperparameters on the negative log likelihood (2.13) test error, it can be seen that in the area of the minimum test error the effect of α was relatively slight and that the model was much more sensitive to variations in σ.

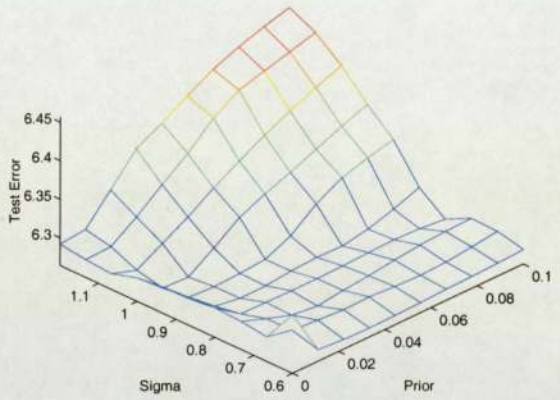


Figure 3.4: The surface shows the variation of test error with the two Gaussian activation hyperparameters. The prior over the weights (α) has very little effect on the error in the region of the minimum error (along the line $\sigma = 0.9$) while the variance (σ) has a more marked effect.

3.4.2 Alternative Activations

Although the most popular activation function for the mapping is Gaussian, experiments were also carried out using the TPS activation to see if this would have any effect on training performance. In addition, experiments were carried out with the similar R4LogR function ($x^4 \ln(x)$) which has similar properties to the TPS.

The thin plate spline does not have any variance (width parameter) σ and therefore its only regularization comes from the prior over the weights α. Figure 3.5 illustrates that, as with the Gaussian activation, the error is very insensitive to the weight decay factor until the point where it becomes so big that any non-zero weights are penalized so heavily that the manifold collapses into a single line along the axis of the cone.

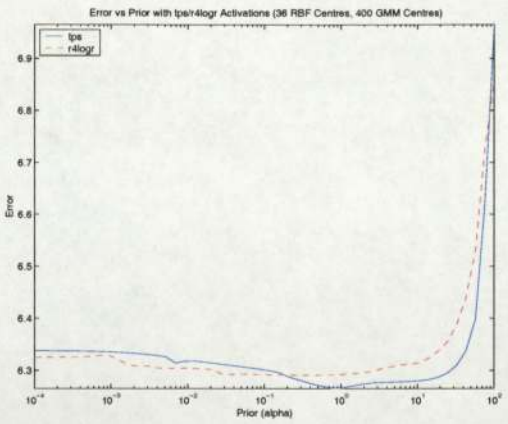


Figure 3.5: Thin Plate Spline and R4LogR activations have no width (variance) parameter. The variation with test error is relatively insensitive to the prior over the weights until the collapse of the manifold to a single line. The solid (blue) line shows TPS performance and the dashed (red) one, that for R4LogR

3.5 Manifold Aligned/More GMM Components

Since we are trying to model off-manifold noise, it seems reasonable that we should investigate a noise model with a full or diagonal covariance matrix, where the noise normal to the manifold is less than that along it and sensitivity to outliers which are removed from the dense layers of the data would be increased. A contrasting method of achieving the same effect is to increase the number of mixture components which will result in each having a reduced variance in the maximum likelihood solution.

A possible scenario is illustrated in an area of low data density in which a relatively low number of spherical Gaussian centres exist (Figure 3.6) where points may have low probability to centres but be close to the line of the manifold.

Full covariance or increased numbers of spherical Gaussians may account for these points more correctly (Figure 3.7).

The first method considered for calculating the full covariance matrices involved the use of directional curvatures of the GTM manifold, as discussed in [8], which requires the partial derivatives of the mapping $\mathbf{y}(\mathbf{x}, \mathbf{W})$ with respect to the latent variable $\mathbf{x}$

$$\frac{\partial \mathbf{y}}{\partial x_r} = \frac{\partial \phi_{\mathbf{m}}}{\partial x_r} \mathbf{W}, \tag{3.2}$$

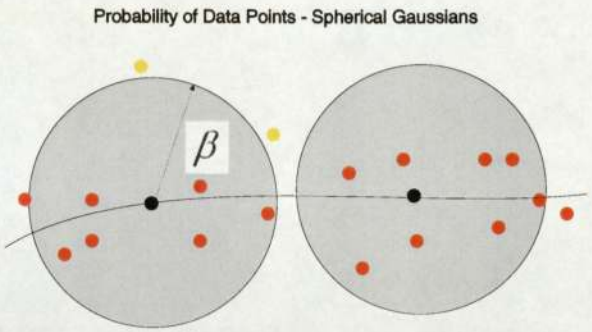
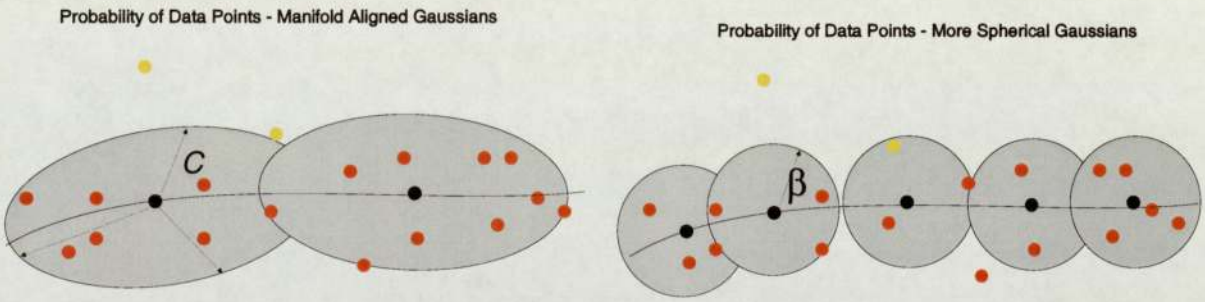


Figure 3.6: Relatively low numbers of spherical covariance noise modelling nodes in areas of low data density may erroneously mis-identify points which lie close to the manifold but not to their nearest GMM centres as outliers. The lightly shaded (yellow) points outside the circles representing the mixture component variances will have disproportionately low probabilities even though they may lie close to the manifold.



(a) Manifold Aligned Noise Model

(b) Increased Centres Spherical Noise Model

Figure 3.7: Increasing the variance along the manifold is one solution to the misrepresentation of data points illustrated in Figure 3.6, as is increasing the number of mixture model components to correctly model the data close to the manifold.

where $r = 1, \ldots, L$ latent space dimensions[4] and $m = 1, \ldots, M-1$, hidden unit activations excluding the bias term ϕ_0. The idea for each latent point was to map a line of unit length, with the latent point at one end, into data space. The alignment of the line in latent space would dictate the direction of the resultant vector in data space. For instance a vertical line in Figure 3.1 would emerge in data space as a tangent running parallel to the axis of the cone. A similar process could be carried out with a horizontal line in latent space resulting in a tangent at right angles to the first. Lastly the line normal to the surface to the cone could be constructed using the

[4] $\mathbf{x}_n = \{x_1, x_2, \ldots, x_L\}$.

first two. From these 3 vectors, which could be scaled as required, a full or diagonal covariance matrix could be constructed for each Gaussian mixture component. In this case, a diagonal covariance matrix could be used as only the variance along the principal axes are of interest and so the off-diagonal elements in the matrix would be redundant. Subsequently a simpler chord-based method, which generated similar vectors purely using the position of neighbouring mixture centres in data space was tried. The vectors could be scaled such that the axis aligned covariance was increased or conversely the covariance normal to the manifold was reduced.

Having full covariance matrices changes the EM algorithm in the following ways.

- Instead of calculating Euclidian distances between data and centres in data space, the Mahalinobis distance (Δ), which takes account of the shape of the covariance matrix, is required in the E-step

$$\Delta^2 = (\mathbf{x} - \mu)^T \mathbf{\Sigma}^{-1} (\mathbf{x} - \mu); \tag{3.3}$$

  where Σ is the $D \times D$ covariance matrix and μ is the centre of the mixture component. This results in increased computational effort.

- The closed form updating of the weights using the original M-step is lost as the weights $\mathbf{W}$ are used in the calculation of the covariance matrix which is required in the E-step. The existing M-step can still be used, however, as an approximation to update the weights.

Experiments using this approach soon confirmed the problems discussed in [7] Chapter 6.

- The added computational effort in calculating the full covariance matrices with less centres was approximately equivalent to increasing the number of centres. This has a similar effect in modelling off-manifold noise as the variance of the mixture components reduces in the maximum likelihood solution and the area of data will contain more mixture components.

- Increasing the number of centres (with spherical covariance matrices) retained the M-step as an exact, closed form method of updating the weights.

- After a point, there was no significant reduction in test error as a result of either increasing the number of centres or using full covariance matrices. In addition, after this limit in the number of centres with spherical covariance matrices had been reached, the final variance of the GMM model components did not decrease indicating that the limit in modelling of the off-manifold noise using GTM had been reached and that using manifold aligned covariance matrices would achieve nothing.

For these reasons, the use of full covariance matrices was abandoned.

Figure 3.8 shows the performance of the GTM manifold with increasing numbers of centres. It can be easily seen that there is a minimal reduction in both error and noise variance for a significant increase in computational effort.

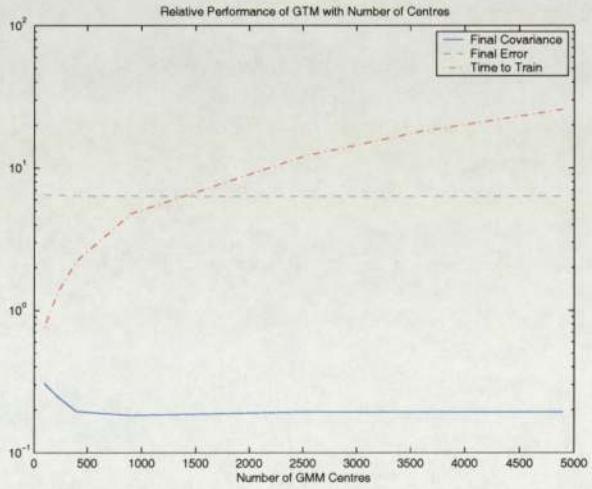


Figure 3.8: Performance of the GTM manifold in training: The reduction in test error is illustrated by the dotted (green) line, the final covariance (β - a measure of the effectiveness with which the noise has been learned) by the solid (blue) line and increase in computational effort in terms of time to train by the chained (red) line.

3.6 Identification of Outliers

The pre-processed test data discussed in Sub-section 3.3.3 was used to choose a probability threshold $p_t(\mathbf{t}|\mathbf{W}, \beta)$, where $\mathbf{W}$ and β are the parameters of the GTM, below which data points (using the same test set without the outliers removed: $\mathbf{t}^o$) were deemed to be outliers. Figure 3.9 shows a plot of the probabilities of the data points and the threshold.

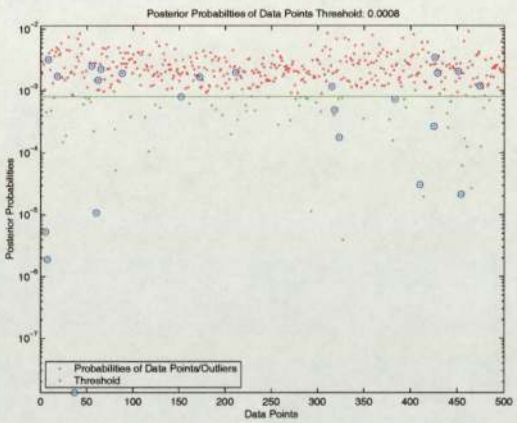


Figure 3.9: Data points and their probability plotted with threshold of 0.004. Those falling below the threshold (shown as a (green) horizontal line) are classified as outliers. The manually identified outliers, which were removed from the test set and used as a basis for scoring the success of the model are shown as (blue) circles.

The success of the model in identifying outliers was scored on the number correctly identified[5] as a proportion of the total identified including erroneous ones. This is a rather misleading measure, as will be discussed in Section 3.7.

Figure 3.10 shows the outliers identified on the plot of the trained manifold; it should be remembered that these outliers are identified on the basis of their probability under the trained model, which is based upon their distance to it. It will be noticed that there are a larger number of identified outliers at the 'nose' of the cone; perhaps this is an indication the cone is not so well defined in this area, that is to say that there may be more noise. This is more evidence for fact that the noise varies over the surface of

[5]Compared with those manually removed from a test set.

the cone; this will be addressed in the second part.

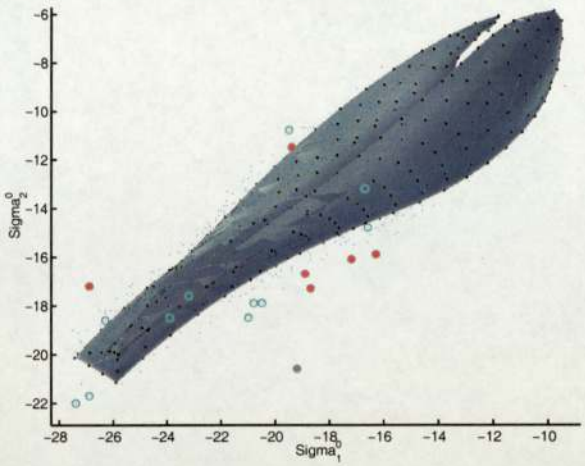


Figure 3.10: Manually identified outliers are shown as (red) stars, those identified by the GTM as (blue) circles. As can be seen, many more GTM outliers are identified towards the 'nose' of the cone, where the shape is less well defined. Also, some manually identified outliers lie close to the GTM manifold but are spurious from the point of view of wind vectors as discussed in Section 3.7.

The quantitative results are tabulated in Table 3.1, based on a comparison between the test set referred to in Subection 3.3.1 and the original set from which it came before excluding the outliers.

Track	Threshold	'True' Outliers	GTM Outliers	Score
1	0.0004	22	26	9
	0.0008		34	11
10	0.0004	24	34	8
	0.0008		72	11
19	0.0004	34	62	18
	0.0008		105	22

Table 3.1: The 'score' based on 2 validation samples of 500 points with and without manually identified outliers removed. The table shows the track number, two threshold values, the number of outliers identified manually and removed from the validation set, the total number of points identified as outliers by the GTM, and finally, the number of GTM outliers which agreed with the manually removed ones.

3.7 Discussion

Having attempted to model the scatterometer data using a GTM, the following points are drawn,

- The use of GTM for the visualization of high dimensional data has been proven to be useful. The first part of this project has borne out that the basic method is valid for the modelling of data noise and identification of outliers although with some reservations as will be discussed.

- The modification of latent space is useful for incorporating prior knowledge of the data being modelled. The more the geometry of the latent space can reflect that of the data, the more accurate the model and the less the mapping function is having to distort the manifold to ensure it fits the data. Comparisons carried out using GTM with a regular flat latent space still managed to model the data after a fashion and even identify outliers, although generally it identified less in total and less of the manually identified ones. The main difference, however, was in the final variance of the noise model which was much greater. This would indicate that the GTM with latent space topography modified to incorporate prior knowledge of the data will model it in a more accurate manner. Table 3.2 shows the comparison of the test error and final variance between the conical and flat GTMs for the 3 sample tracks.

		Track 1	Track 10	Track 19
Test Error	Conical	4.539	6.3524	6.9787
	Flat	4.583	6.489	7.000
Final Variance	Conical	0.109	0.1768	0.1755
	Flat	0.130	0.2536	0.2292

Table 3.2: The comparison between a GTM with a plain rectangular grid of latent points and one with the latent space topography discussed in Section 3.2. The conical GTM outperforms the other marginally in test error and by a more significant amount in terms of the final variance of the GMM components. The data set was the same as used to produce Table 3.1.

- The main weakness of the procedure was that of trying to model a double skinned self intersecting cone with a single GTM manifold. The 'best fit' will be somewhere between the two skins and, in areas where the distance between the two skins is large, this will vary quite considerably from the actual data. This will result in a false representation of the variance of the data at these points and the identification of any outliers will therefore be flawed. Fig 3.11 shows the single path of the GTM manifold through a cross-section of the scatterometer cone.

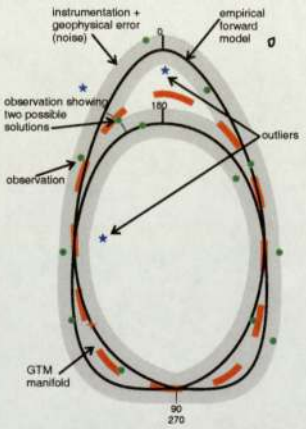


Figure 3.11: True outliers (shown as (blue) stars) are correctly identified by the GTM if they have low probability to the GTM manifold, shown as a (red) dotted line. This will not always coincide with data points which lie a long way from the theoretical cone. Similarly, in areas where the GTM manifold has to find the best fit to areas where the two surfaces of the cone are far apart, valid scatterometer points may be identified as outliers.

- Variation of hyperparameters (see Figure 3.4 in Subsection 3.4.1) obviously has an effect upon how the GTM behaves. However the model is far more sensitive to the variance of the RBFs (σ) mapping the latent to data space than to the prior imposed on the weights α. This would seem reasonable, as the mapping from 2 to 3 dimensions and to the target data is relatively smooth and therefore will not require, nor be affected by such regularization as α imposes. The stiffness of the manifold, however, will affect the manifold's ability to fit to such curves as the data has[6] and this, in turn will affect the test (and training) error.

[6]The data by no means lies on a perfect cone, and its curvature does change dependent upon the track, and therefore incidence angle, of the satellite beam.

- Success of the model. The main fault with the scoring of the model's ability to identify outliers rests with the definition of what is an outlier. When manually removing outliers from the test set, not only was 'distance to cone' taken into account but also readings which were on the cone, but spurious from the point of view of their position in terms of wind speed and direction. The outliers from the training and comparison test set had been removed using visualization of the data relative to a theoretical manifold created by NN3CMOD (described in Section 2.1) and by identifying data points which were at grossly at odds with their neighbours in the wind fields identified by wind speed and direction; it was this latter criterion which identified outliers which could not be subsequently highlighted by GTM. A data point which provides a spurious wind vector due to its position on the cone is rightly deemed an outlier in v, v' space but not σ^0 space and consequently the GTM will not classify it as such, since its probability given the model is high enough to be above the threshold. This is illustrated in Figure 3.9 where the main body of the manually excluded outliers lie well above the threshold in the main body of the data.

- In the initial stages of the project it was questioned whether the data should be modelled using a double-skinned cone. However it was decided that the nature of the theoretical shape of the data was such that, whereas a double-skinned might effectively model areas where the two layers of data were well separated it would have problems where they coincided. In these areas, two layers of mixture components would effectively be competing for a single layer of data points. It is likely then that a situation may arise where component centres would compromise by finding a position either side of the data and be effectively repelling each other. The situation is the converse of the actual result where the single GTM is torn between data in areas where the skins of the cone are separated.

- We were fortunate to have both a training and test set from which outliers had been manually removed. If the GTM had been trained, using the EM algorithm

and sum-of-squares function, on a data set which included outliers this would affect the convergence of the model with the result that data points which were outliers might not get identified as such and vice versa. For this reason alternative error functions (such as Minkowski Error [1], Least Median of Squares [4] or Least Trimmed Squares [5]), which have been investigated in general outlier detection research, might improve the effectiveness of the model in this regard. An alternative is to use an iterative technique in order to progressively remove outliers; training with outliers initially, removing those which are identified, re-training and so on.

- The shape of the GTM manifold seems to be less well defined at either end of the cone (Figure 3.3). This could be because there are less data points in these areas, or because the noise is increased. Hopefully the second part of the project will go some way to clarifying this.

Chapter 4

Input Dependent Noise

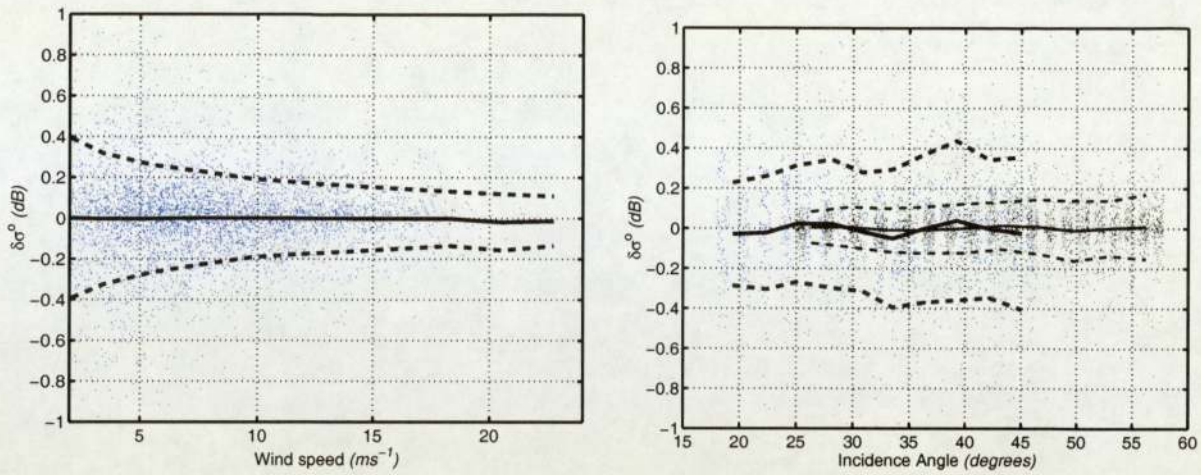
4.1 Introduction

We now come to the prospect of modelling the output noise separately as dependent upon the inputs. This is termed Input Dependent Noise and will be referred to from now on as IDN; it should not be confused with any input noise, which is dealt with separately in the model which will be used to compute σ^0.

The current residual noise projection of the σ^0 data is a function of retrieved wind speed and incidence angle is illustrated in Figure 4.1 and strongly implies that there is at least a trend in the noise variance with speed. This projection uses the NN3CMOD model as described in Section 2.1.

In the first part of the chapter the principle of modelling output noise using 2 networks will be discussed in general. There are two methods of modelling the noise: Using the maximum likelihood method to update the weights of both networks simultaneously, or an iterative approach in which the training of each network is linked with the other and each is updated as a result of the output of the other. In this project, the maximum likelihood method will be discussed but only the iterative technique will be used in the experiments.

Next, some modifications to the scatterometer forward model NN3CMOD, introduced in Section 2.1, will be implemented. Finally, the iterative technique is initially applied to a toy problem and then transferred to the scatterometer model which has a slightly more complex structure.



(a) Variation of Noise Variance with Speed

(b) Variation of Noise Variance with Incidence Angle

Figure 4.1: Residual values $\delta\sigma^0 = \sigma^0_{\text{predicted}} - \sigma^0_{\text{observed}}$ plotted as a function of retrieved wind speed (a) and incidence angle (b). The solid line gives the running mean, the dotted line $\pm$ one standard deviation. In the speed graph, all beams are considered together, however, the mid beam incidence angle range is lower and therefore to the left in (b).

4.2 Input Dependent Noise

In this scenario, it is assumed that the noise in a data set is given by

$$\mathbf{t}_n = f(\mathbf{x}_n) + \nu(\mathbf{x}_n), \tag{4.1}$$

where $\mathbf{x}_n$ and $\mathbf{t}_n$ are an input and target respectively of a set of N data points, $f(\cdot)$ is the function mapping the mean of t_n and $\nu(\cdot)$ is the noise, generated from a random process having a Gaussian distribution with a zero mean and a variance dependent upon $\mathbf{x}$.

4.2.1 Maximum Likelihood

There are two networks, given by $\mathbf{y}_n = y(\mathbf{x}_n; \mathbf{w})$ and $\beta_n = \beta(\mathbf{x}_n; \mathbf{u})$, modelling the mean and the noise of the function respectively. The likelihood function may then be expressed as

$$p(D|\mathbf{w},\mathbf{u}) = \frac{1}{Z_D}\exp\Big(-\sum_{n=1}^{N}\beta_n E_n\Big), \tag{4.2}$$

where D is the data $\{\mathbf{x}_n, \mathbf{t}_n\}$ comprising N points, $E_n = \frac{1}{2}(\mathbf{y}_n - \mathbf{t}_n)^2$ and the normalizing factor is given by

$$Z_D = \frac{(2\pi)^{N/2}}{\prod_{n=1}^{N}\beta_n^{1/2}}. \tag{4.3}$$

It will be remembered that β_n is the inverse noise variance $1/\sigma^2$ associated with a single input $\mathbf{x}_n$. $\boldsymbol{\beta}$, therefore, will be a vector $\{\beta_1, \ldots, \beta_N\}$.

The error (negative log likelihood) function may then be expressed as

$$\begin{aligned} E(\mathbf{w},\mathbf{u}) &= -\ln p(D|\mathbf{w},\mathbf{u}) \\ &= \sum_{n=1}^{N}\beta_n E_n - \frac{1}{2}\sum_{n=1}^{N}\ln\beta_n. \end{aligned} \tag{4.4}$$

The third term $(-\ln(2\pi)^{N/2})$ is constant and, since it is minimization of the error term that we will subsequently be interested in, is ignored. The addition of regularization terms (priors) gives

$$E(\mathbf{w},\mathbf{u}) = \sum_{n=1}^{N}\beta_n E_n - \frac{1}{2}\sum_{n=1}^{N}\ln\beta_n + \alpha_w E_w + \alpha_u E_u, \tag{4.5}$$

where

$$E_w = \frac{1}{2}\|\mathbf{w}\|^2 \qquad E_u = \frac{1}{2}\|\mathbf{u}\|^2. \tag{4.6}$$

In the pure Bayesian framework, the priors (α_w and α_u) would be estimated probabilistically; in the scatterometer case, however we have sufficient prior knowledge to fix them *a priori*. The method used in this project, then, is penalized maximum likelihood.

4.2.2 Bayesian Inference/Penalized Maximum Likelihood

For a data sample, $D = \{t_1, t_2, \ldots, t_N\}$, maximum likelihood will yield the standard results for the most probable values of μ, the mean and σ^2, the variance

$$\widehat{\mu} = \frac{1}{N}\sum_{n=1}^{N} t_n \tag{4.7}$$

and

$$\widehat{\sigma}^2 = \frac{1}{N}\sum_{n=1}^{N}(t_n - \widehat{\mu})^2. \tag{4.8}$$

This value of $\widehat{\sigma}^2$ is known to be biased, the average over the samples being

$$\langle\widehat{\sigma}^2\rangle = \frac{N-1}{N}\sigma_0^2, \tag{4.9}$$

where $\langle\cdot\rangle$ is the expectation or average. This only approaches (4.8) as $N \to \infty$.

This bias may be compensated for by marginalizing over the prior distribution[1] giving a marginal likelihood

$$\begin{aligned} p(D|\sigma^2) &= \int p(D|\sigma^2, \mu)p(\mu)\mathrm{d}\mu \\ &\propto \frac{1}{\sigma^{N-1}}\exp\left[-\frac{1}{2\sigma^2}\sum_{n=1}^{N}(t_n - \widehat{\mu})^2\right], \end{aligned} \tag{4.10}$$

which when maximized with respect to σ^2 gives

$$\widetilde{\sigma}^2 = \frac{1}{N-1}\sum_{n=1}^{N}(t_n - \widehat{\mu})^2. \tag{4.11}$$

which is unbiased.

At this stage it is wise to identify the exact models used for the networks modelling the mean and variance of the data[2] which will facilitate some of the computations later. RBFs are again used with the β-network having an exponential output in order that the noise variance is always positive[3]

[1] A 'flat' prior is used.

[2] Hereafter referred to as the y-network and the β-network.

[3] This is a common procedure and simply means that the output of the β network is $\ln\beta$.

$$y(\mathbf{x};\mathbf{w}) = \phi(\mathbf{x})\mathbf{w} \qquad \beta(\mathbf{x};\mathbf{u}) = \exp[\psi(\mathbf{x})\mathbf{u}]. \tag{4.12}$$

As in the first part of the project (2.5), the RBF mapping comprises activations combined with weights. $\phi(\mathbf{x})$ and $\psi(\mathbf{x})$ are the activations and $\mathbf{w}$ and $\mathbf{u}$ the weights for the y-network and the β-network respectively. As before Gaussian,

$$\phi(\mathbf{x}) = \exp\left(-\frac{\mathbf{x}^2}{2\sigma^2}\right), \tag{4.13}$$

and TPS,

$$\phi(x) = \mathbf{x}^2 \ln(\mathbf{x}), \tag{4.14}$$

activation functions are used.

The next step is to develop the error functions for the two networks and the hierarchy which will be used in their optimization. The first stage is to define the weight priors which are taken as Gaussian and isotropic

$$p(\mathbf{w}|\alpha_w) = \left(\frac{\alpha_w}{2\pi}\right)^{k_w/2} \exp(-\alpha_w E_w), \tag{4.15}$$

$$p(\mathbf{u}|\alpha_u) = \left(\frac{\alpha_u}{2\pi}\right)^{k_u/2} \exp(-\alpha_u E_u), \tag{4.16}$$

where k is the number of elements in the respective weight vectors ($\mathbf{w}$ and $\mathbf{u}$) and E is defined in (4.6).

So, for a given $\widehat{\mathbf{u}}$ (which will be determined in the next stage), $\widehat{\mathbf{w}}$ is found by maximizing the posterior distribution

$$p(\mathbf{w}|D, \widehat{\mathbf{u}}, \alpha_w) = \frac{p(D|\mathbf{w}, \widehat{\mathbf{u}})p(\mathbf{w}|\alpha_w)}{p(D|\widehat{\mathbf{u}}, \alpha_w)}, \tag{4.17}$$

where the denominator is given by

$$p(D|\widehat{\mathbf{u}}, \alpha_w) = \int p(D|\mathbf{w}, \widehat{\mathbf{u}})p(\mathbf{w}|\alpha_w)\, \mathrm{d}\mathbf{w}. \tag{4.18}$$

Using (4.2) and (4.15), taking the negative log and ignoring any constants, an expression for the error of the y-network is obtained

$$S(\mathbf{w}) = \sum_{n=1}^{N} \beta_n E_n + \frac{\alpha_w}{2} \|\mathbf{w}\|^2, \tag{4.19}$$

where E_n is the squared error for a single point $\frac{1}{2}(y_n - t_n)^2$. Here we have the first of the advantages of using RBFs, where the most probable weights can be found by 'one shot' matrix inversion techniques. In this case

$$\widehat{\mathbf{w}} = \mathbf{A}^{-1} \mathbf{\Phi}^T \boldsymbol{\beta} \mathbf{t}, \tag{4.20}$$

where $\boldsymbol{\beta}$ is an $N \times N$ diagonal matrix with the elements $\beta(\mathbf{x}_n; \mathbf{u})$, $\mathbf{\Phi}$ is an $N \times M$ matrix of RBF activations (M is the number of basis functions including the bias), $\mathbf{t}$ is an $N \times D$ matrix of targets (D is the dimension of the data) and $\mathbf{A}$ is the regularized Hessian matrix of the y-network which is discussed in Appendix B. The Hessian matrix is regularized in the same way as the error functions,

$$\mathbf{A}_N = \mathbf{H}_N + \alpha_w \mathbf{I}, \tag{4.21}$$

where H_N is the unregularized data Hessian, α is the weight decay factor for the y-network and $\mathbf{I}$ is the identity matrix with the same dimensions.

Now we develop the error function for the β-network which comes from the marginalized posterior distribution

$$p(\mathbf{u}|D, \alpha_u, \alpha_w) = \frac{p(D|\mathbf{u}, \alpha_w) p(\mathbf{u}, \alpha_u)}{p(D|\alpha_w, \alpha_u)}. \tag{4.22}$$

This is the probability of the β-network weights, given the data and the weight regularizing terms in both networks and uses Bayes' rule, (4.16) and (4.18). The denominator is marginalized in the same way as (4.18) by integrating $\mathbf{w}$ out. In the case of the models and priors we are using, this integral becomes Gaussian and therefore can be performed analytically. Taking logarithms and discarding constants as before, the error function for the β-network may be expressed as

$$M(\mathbf{u}) = \sum_{n=1}^{N} \beta_n E_n + \frac{\alpha_u}{2}\|\mathbf{u}\|^2 - \frac{1}{2}\sum_{n=1}^{N} \ln \beta_n + \frac{1}{2}\ln|\mathbf{A}|, \tag{4.23}$$

where $|\cdot|$ indicates the determinant. The value for $\widehat{\mathbf{u}}$ may be found by minimizing this function using any non linear optimization technique. In this case Scaled Conjugate Gradient (SCG) is used which, in common with most optimization techniques, required an expression for the gradient of the error with respect to the network weights. Thus, an expression for $\partial M(\mathbf{u})/\partial \mathbf{u}$ is required. In the main, the expression is fairly straightforward. The addition of the Hessian term[4] and its derivative with respect to the network weights is the main complication. Utilizing the chain rule of differentiation,

$$\begin{aligned}
\frac{\partial \ln|\mathbf{A}|}{\partial \mathbf{u}} &= \sum_{n=1}^{N} \frac{\partial \ln|\mathbf{A}|}{\partial \beta_n}\frac{\partial \beta_n}{\partial \mathbf{u}} \\
&= \sum_{n=1}^{N} \mathrm{Trace}(\mathbf{A}^{-1}\mathbf{g}_n\mathbf{g}_n^T)\frac{\partial \beta_n}{\partial \mathbf{u}} \\
&= \sum_{n=1}^{N} \mathbf{g}_n^T\mathbf{A}^{-1}\mathbf{g}_n\frac{\partial \beta_n}{\partial \mathbf{u}},
\end{aligned} \tag{4.24}$$

where $\mathbf{g}_n = \partial \mathbf{y}_n/\partial \mathbf{w}$. The whole expression for the gradient of the β-network with respect to the weights is therefore

$$\frac{\partial M(\mathbf{u})}{\partial \mathbf{u}} = \sum_{n=1}^{N}\left(E_n + \frac{1}{2}\mathbf{g}_n^T\mathbf{A}^{-1}\mathbf{g}_n - \frac{1}{2\beta_n}\right)\frac{\partial \beta_n}{\partial \mathbf{u}} + \alpha_u\mathbf{u}. \tag{4.25}$$

The evaluation of the Hessian is discussed in Subsubsection 4.3.2 and Appendix B.

The algorithm, then, is iterative

1. Evaluate $\widehat{\mathbf{w}}$ using an initial value of β and (4.20)
2. Optimize (4.23) to find $\widehat{\mathbf{u}}$
3. Re-evaluate $\widehat{\mathbf{w}}$ using $\widehat{\mathbf{u}}$
4. Repeat 2 and 3 until converged

[4]This is the only difference between the error term in the Bayesian inference (or penalized maximum likelihood) technique and the Maximum Likelihood one.

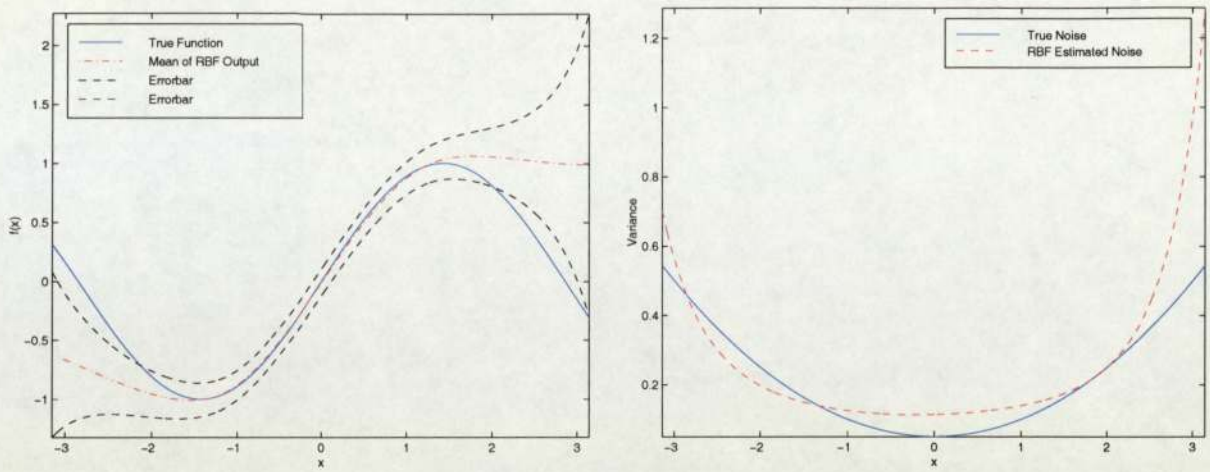
True Bayesian inference would involve third stage described in [3], which would go on to carry out a similar procedure to update values of α_w and α_u, but as stated in Subsection 4.2.1 these will be fixed using prior knowledge.

4.3 Applications

4.3.1 A Toy Problem with IDN

Before applying the above technique to the scatterometer data, it was decided to experiment with the toy sine wave problem used in [3] with some modifications. The target data comprised 100 training sets of 10 data points, each randomly selected from 1000 points based on $\sin(0.35\pi\mathbf{x})$ with zero mean Gaussian noise with variance given by $\sigma^2 = 0.05 + 0.05\mathbf{x}^2$. The modifications concerned a change of activation function in the RBFs (the RBF variant of NN3CMOD currently uses Thin Plate Spline (TPS) activations, as opposed to Gaussian ones in the original toy problem) and to experiment with varying (but not updating) values for $\alpha_{\mathbf{w/u}}$. Figure 4.2 shows the performance of the two networks on the toy sine problem.

The results of the toy experiment are similar to those in [3] although there did seem to be a tendency for the thin plate spline activations to be somewhat unstable with points which are at the extremes of the range spanned by the RBF centres. It was found that this effect could be reduced by setting the RBF centres to span the *expected* range of the data, rather than the actual one for each data set; in this way the data was more effectively encompassed. Because we wanted to model the noise evenly across the whole data set, it was decided to set the RBF centres to evenly span the data limits without taking any account of varying density within it. With this reservation, it seemed fair to transfer the procedure to the scatterometer data.



(a) Y-Network Results with Error-Bars

(b) β-Network Results

Figure 4.2: The results of the toy IDN problem averaged over 100 sets of 10 data points. On the left (blue) solid line shows the original function, the (red) chained line is the mean of the RBF output, the (black) dashed lines represent one standard deviation from the mean. The spacing of the RBF centres is even across the range of the data and the widths (in the case of Gaussian activations) are chosen as the distance between centres. The activations of the y-network are TPS while those in the β network are Gaussian. The prior over the weights in the y-network was 20 and in the β-network 0.01. On the right, the (blue) solid line shows the true function variance of $0.05 + 0.05\mathbf{x}^2$ and the (red) dashed line the output of the β-network.

4.3.2 Application to Scatterometer Data

This section deals with applying the IDN algorithm[5], to the scatterometer data. This will involve a modification to NN3CMOD. In addition the mechanics of computing the Hessian $\mathbf{A}$ will vary from the straightforward RBF Hessian due to the inclusion of the functional part of the model which results from the Fourier terms.

Scatterometer Error Function

The first thing to consider in the application of the IDN technique is the error function which will be used for the training of the mean of the σ^0 data. It not only relies on the data sum-of-squares error and a regularization term involving a prior over the weights

[5]As it has been shown that for limited data sets, the Bayesian iterative model is superior to the Maximum Likelihood approach [3] it is this method only which will be employed on the scatterometer data.

but also the error involving the input noise from the wind vectors[6] from which the speed s and relative wind direction χ are computed. The error is given by 3 terms[7]

$$\begin{aligned} E &= E_1 + E_2 + E_4 \\ &= \sum (f(\widetilde{s}, \widetilde{\chi}, \theta, \mathbf{w}) - \sigma^0)^2/(2\sigma_t^2) \\ &+ \sum \left((\widetilde{v} - v)^2 + (\widetilde{v'} - v')^2 \right)/(2\sigma_v^2) \\ &+ \sum_w \mathbf{w}^2/(2\sigma_w^2). \end{aligned} \tag{4.26}$$

The full derivation of this cost function can be seen in [2], Section 3.4. E_2 represents the error due to input noise, that is the difference between the noise free (ideal) wind vectors and the corresponding noisy ones given in the data set. The distribution, $p(x_n|\widetilde{x}_n)$, where x_n and $\widetilde{x}_n$ are the noiseless and noisy inputs respectively, is assumed to be spherically Gaussian in the wind component vectors (v, v'). σ_v^2 is the variance in the wind vector measurements. The value used in these experiments is estimated [6] at $2.25m^2s^{-2}$. A more in-depth study of modelling data with input uncertainty is carried out in [9].

The abstract y-network error function derived in Subsection 4.2.2 is reproduced here for contrast

$$S(\mathbf{w}) = \sum_{n=1}^{N} \beta_n E_n + \frac{\alpha_w}{2} \|\mathbf{w}\|^2. \tag{4.27}$$

It can be seen that E_1 and E_4 in (4.26) are equivalent to the two terms in (4.27) where $\beta_n = 1/2\sigma_t^2$ and $\alpha_w = 1/\sigma_w^2$ and so the error function for the scatterometer model may be rewritten in the same form as (4.19)

[6] In [2] these are referred to as u and v but, to save confusion with the β-network weights, we will refer to them as v and v'.

[7] A fourth term E_3(!), involving the prior distribution of the noiseless wind vectors is assumed constant and therefore ignored.

$$\begin{aligned} S(\mathbf{w}) &= E_1 + E_2 + E_4 \\ &= \sum_{n=1}^{N} \beta_n E_n \\ &+ \sum \left((\widetilde{v} - v)^2 + (\widetilde{v'} - v')^2 \right) / (2\sigma_v^2) \\ &+ \frac{\alpha_w}{2} \|\mathbf{w}\|^2. \end{aligned} \tag{4.28}$$

Modifying the Scatterometer Model

One modification to the NN3CMOD model was to remove the index (p) from (2.3) which reduces it to a linear equation and simplify subsequent calculations. It will be remembered that the purpose of the index was to compensate for the truncated (2 terms plus a constant) Fourier series. It is now, therefore, necessary to extend the series to include a further 2 terms. The equation now becomes:

$$\sigma^0_{\text{lin}} = a_0 + a_1 \cos(\chi) + a_2 \cos(2\chi) + a_3 \cos(3\chi) + a_4 \cos(4\chi), \tag{4.29}$$

which is a much simplified form and, since the expression will be automatically be real for all inputs, the need for the $\tanh(\cdot)$ term is also removed. The conversion to σ^0_{dB} is equivalent to 2.4. This form, with an RBF network in place of the MLP, and the separate model for fore/aft and mid beams is labelled RBF3CMOD and is illustrated in Figure 4.3. It has the property that it is linear in the network weights which is of use in computing the Hessian as will be seen in Sub-subsection 4.3.2.

The Hessian Matrix for RBF3Cmod

The justification for using the outer product to determine the Hessian matrix for RBF3Cmod is discussed in Appendix B but we need to take into account the functional element of the model, retained from the original CMOD4. (4.29) may be rewritten as

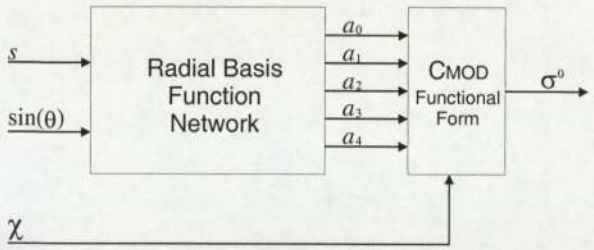


Figure 4.3: A block diagram of the hybrid model RBF3CMOD. The inputs s and χ are computed from the wind vectors (v and v') and θ is the radar beam incidence angle. $a_0, \ldots, a_4$ are the outputs from the RBF network and the coefficients of the Fourier series. σ_0 is the inferred scatterometer value.

$$\sigma^0_{\text{lin}} = \sum_{f=0}^{4} a_f \cos(f\chi). \tag{4.30}$$

From (4.12) we get the expression

$$\frac{\partial \mathbf{a}}{\partial \mathbf{w}} = \phi, \tag{4.31}$$

where a is the output of the y-network. ϕ comprises columns ϕ_k, each an N element column vector with $k = \{0, 1, \ldots, M\}$ where M is the number of hidden units in the network, including the bias term ϕ_0. Because the terms of the Fourier series are summed, the output σ^0_{lin} is affected by each weight in the network; this would not be the case if we were computing the derivative for each of the 5 outputs of the RBF3CMOD network alone where only the weights on the path to the individual output units have an effect. The expression for the derivative with respect to each row of weights associated with individual hidden units (each of which comprises 5 elements), is

$$\mathbf{g}_k = \frac{\partial \sigma^0_{\text{lin}}}{\partial w_k} = \phi_k^T \mathbf{F}, \tag{4.32}$$

where $\mathbf{F}$ is defined as

$$\mathbf{F} = \begin{bmatrix} \cos(0\chi) & \cos(\chi) & \cos(2\chi) & \ldots & \cos(4\chi) \end{bmatrix}, \tag{4.33}$$

an $N \times 5$ matrix and then

$$\mathbf{g} = \{\mathbf{g}_0, \mathbf{g}_1, \ldots, \mathbf{g}_M\}, \tag{4.34}$$

resulting in a row vector with $M \times 5$ elements[8]. It is now possible to compute the Hessian matrix for the whole model using the outer product method.

Training the y-network

It has been said in Subsection 4.2.2 that the y-network can be trained using matrix inversion in a single step (4.20). This is fine with a straight RBF network. RBF3CMOD, however aims to not only learn the mean of the scatterometer data, but also the input noise of the wind vectors, v and v'. The 'true' values $\widetilde{v}$ and $\widetilde{v'}$ can only be learned by non-linear optimization. There are subsequently two options. One is that the matrix inversion technique is used for a initial or interim value of the wind vectors (from which are calculated, the speed and relative direction inputs) in order to obtain $\widetilde{w}$ for those values. Then, an optimization routine could be employed to update the 'true' value of the vectors and the process repeated. The other option is to optimize the whole model using SCG (or similar) converging towards the ideal values for the wind vectors and the network weights at the same time. Preliminary experiments showed that the latter approach was no more computationally expensive and that the risk of converging to spurious weights, for a given set of wind vectors, was reduced. In subsequent work, SCG is used for optimization of both the y and β-network weights.

Results of Scatterometer Experiments

Initially the RBF3CMOD model was completely trained, using 50000 iterations of the SCG method and a fixed variance, estimated by Stoffelen [6], of 0.04 for the fore and aft

[8]In this resulting vector, the elements are ordered with bias term at the front of each $\mathbf{g}_k$ component. In practice this is rearranged so that all the bias terms (the column vectors resulting from the multiplication by ϕ_0) are all grouped at the end of the vector and consequently in one corner of the Hessian matrix.

beams and subsequently Cornford [2] of 0.08 for the mid beam before the IDN algorithm was applied in order to learn the variance more accurately. The subsequent training to learn the noise variance consisted of an outer loop, in which the y-network is trained to learn $\widehat{\boldsymbol{w}}$ for the current value of $\boldsymbol{u}$, the weights of the β-network are then updated using the weights of the previously trained y-network. We used 12 hidden units in both the RBF networks; 3 to cater for the speed input times 4 for the incidence angle. This appeared to be sufficient for both the forward model mapping and that of the noise model. Noticeable during this part of the training was that it seemed that a lot of time was lost unnecessarily training the initial model so thoroughly. When the iterative IDN algorithm was applied to a roughly trained model, as soon as the noise variance was adjusted as a result of learning $\widehat{\mathbf{u}}$ for an interim value of $\mathbf{w}$, training of the y-network advanced in large steps, thus making the training much quicker. This can be seen from the record of the training error for both the y and β networks, Figure 4.4. As will be discussed later, this proved to be a rather false indication.

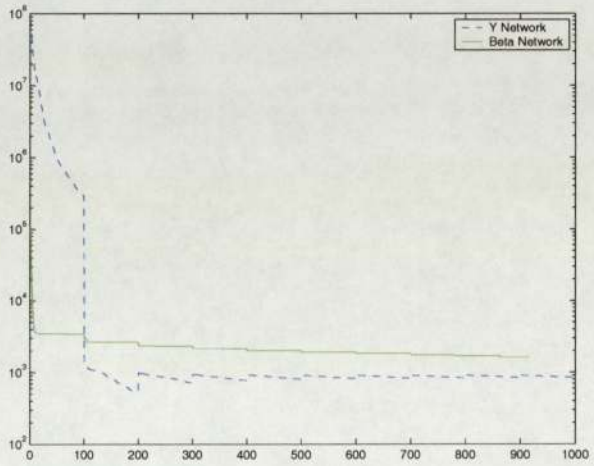
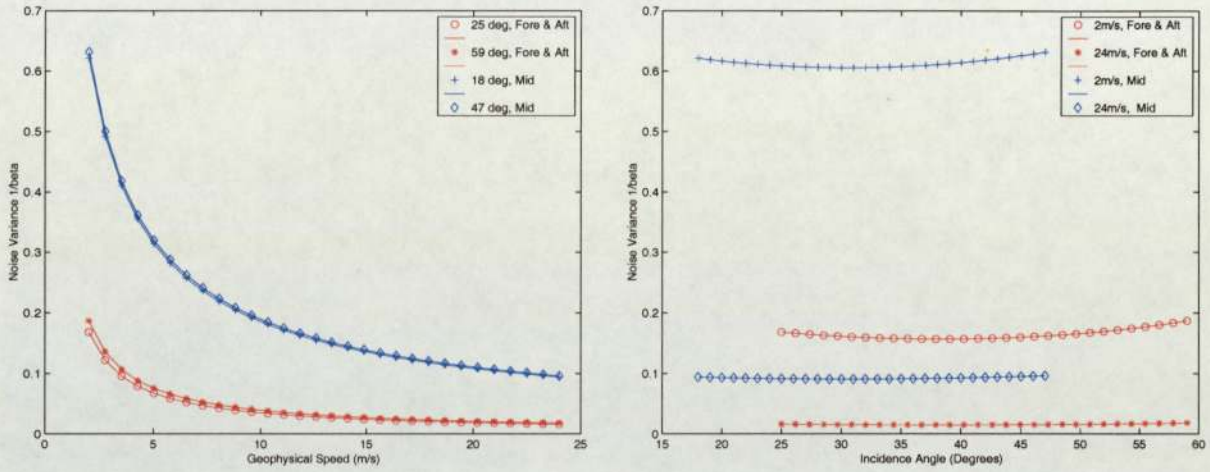


Figure 4.4: A plot showing number of SCG training iterations along the x-axis with training error along the y-axis. The training consisted of an outer loop of 10 cycles, within which the weights of the y-network were updated using 100 cycles of SCG. The weights of the β network were then updated over a further 100 cycles of SCG. The (blue) dashed line shows the progress of the y-network and the (green) solid line, that of the β-network.

The weight decay priors α_w and α_u and, for the y-network, were applied only to the hidden to output layer weights and not the biases. They were arrived at using a

validation data set and were set at 0.1 for the y network and 0.0001 for the β-network. Our prior knowledge of the mean and noise variance of the data was such that we were confident that the mapping in both cases would be reasonably insensitive to any prior over the weights.

The results of the training are illustrated in Figure 4.5 relative to geophysical speed and incidence angle.



(a) Noise Variance with Geophysical Speed

(b) Noise Variance with Incidence Angle

Figure 4.5: Both plots represent noise modelling using Gaussian activations in the β-network and TPS activations in the y-network. Both networks contained 12 hidden units. The networks were trained using 10 iterations of 100 cycles of SCG for the y-network, followed by 100 cycles for the β-network, that is to learn the most probable weights ($\widehat{\mathbf{u}}$ and $\widehat{\mathbf{w}}$). Plot (a) is the variation of noise with speed. The (red) line is readings taken using the fore and aft beam model at fixed incidence angles of 25 degrees ($\circ$) and 59($*$) degrees. The (blue) line represents readings using the mid beam model at angles of 18 degrees ($+$) and 47 degrees ($\diamondsuit$). These plots represent tracks 1 and 19 respectively for both the fore and aft and mid beams. Plot (b) is the variation of noise with incidence angle and shows variance with speeds of $2ms^{-1}/24ms^{-1}$ for the fore and aft beam (red, $\circ$ and $*$) and mid beam (blue, $+$ and $\diamondsuit$) models respectively.

Finally, the IDN method was compared with modelling the data with fixed variance using RBF3CMOD. The variance used was that discussed at the beginning of this section. The measure used for comparison was negative likelihood test error and as an example, in the experiment which produced Figure 4.5 the training error per data point was 2.584 compared to 5.076 for the model with fixed output noise.

4.4 Discussion

- The Qazaz iterative IDN algorithm, when applied to the satellite scatterometer data seems to bear out some of the previous ideas of Cornford and others. Primarily, from Figure 4.5 we see that there appears to be a definite increase in the noise at the nose (low speed) end of the data. This would bear out not only what was previously thought, but the evidence from the GTM phase of the project.

- In addition, there would also seem to be some correlation between incidence angle and noise and it would seem that the mid beam has higher noise associated with although there seems to be no physical explanation as to why.

- An interesting by-product of the IDN technique is the effect of speeding up the initial stages of the training of the function network. Of course, this is an illusion since what is actually happening is that the training error is being reduced by the fact that the noise is being either modelled more accurately, or that the iterative technique is allowing the variance parameter to become too large. In the case of the scatterometer, it seems likely that unless a y-network is used whose weights have fairly well converged upon $\widehat{w}$, then the β-network is accounting for the wind vector input noise.

- It seems that, from the evidence in the toy experiment the TPS model was a little unstable at the extremes of the data range. It was found, however, that if the RBF centres were moved to span the *possible* range of the data (i.e. extend further than many of the data samples) the model performed much better. Gaussian activations appeared to be more stable in these areas.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

So what have we learned from the project?

- The GTM as a technique for identifying outliers has been, in general, validated, although the reservations concerning the definition of an outlier would imply that some caution is needed in its application.
- The scatterometer data set was not ideal for density modelling with a GTM; data with clearly separated layers which do not intersect would be more suited.
- The choice of RBF activations had only little effect on the performance of both the GTM and the IDN techniques. TPS activations, however did seem to be a little more unstable in certain conditions and lacked the extra control of the σ hyperparameter. The GTM is, in this application, insensitive to variations of the weight regularizer α.
- The view that the data output noise varies, dependent on at least one of the inputs to the forward model seems to have been borne out by the results of the IDN part of the project and, based on the comparison of training error discussed in Sub-subsection 4.3.2 the IDN technique does result in a more accurate model of the data in this case.

- It is possible that there is a need to exercise caution when applying IDN Bayesian inference, particularly to a training set with possible input noise. It seems wise to train the function mean network (the y-network) to a stage at least where it is close to convergence before trying to learn the noise. If not, the result could be noise modelling with a misleadingly large variance.

5.2 Future Work

The work of this project could possibly be continued in the following ways.

- Investigation of combining the use of GTMs to identify outliers, with different error functions has the potential to improve performance, especially when using training data sets which contain, or may contain outliers.

- Performance of the IDN technique might be improved by the inclusion of a noise hyper-hyperparameter to the β-network in order apply error bars to the way that the noise is modelled as well as to the mean of the function network. This, however is a never ending road would not, perhaps, reap any great rewards.

- The use of data sets of increased size might yield some interesting results. A comparison of results between the maximum likelihood method of learning IDN and the Bayesian inference method may be worthwhile. With a large data set the under estimation of noise variance using the maximum likelihood method might not manifest itself as it would theoretically do with the small data sets which have been utilized in this project.

Appendix A

Principal Component Analysis - in brief

This Appendix briefly explains the basic technique of Principal Component Analysis [1] for as much as it is relevant to the application of GTM initialization. For this purpose, the specific case of 3-dimensional data is used.

The data vectors, being in 3-dimensional space may be represented as a linear combination of 3 vectors

$$\mathbf{x} = \sum_{n=1}^{3} z_n \mathbf{u}_n, \tag{A.1}$$

where $\mathbf{x} = \{x_1, x_2, x_3\}$ and the vectors $\mathbf{u}_n$ are orthonormal

$$\mathbf{u}_n^T \mathbf{u}_j = \delta_{ij}, \tag{A.2}$$

δ is the Kronecker delta.

$$\delta = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{A.3}$$

To ensure that the vectors $\mathbf{u}$ are the *principal* components of the data, the eigenvalues of the covariance matrix of the data is used

$$\mathbf{\Sigma u}_n = \lambda_n \mathbf{u}_n, \tag{A.4}$$

where Σ is the $D \times D$ covariance matrix of the data and D is the dimension of the

data:

$$\mathbf{\Sigma} = \sum_{n=1}^{N} (\boldsymbol{x}_n - \bar{\boldsymbol{x}})(\boldsymbol{x}_n - \bar{\boldsymbol{x}})^T \tag{A.5}$$

and $\bar{\mathbf{x}}$ is the mean of the data. The order of the principal components will be given by the magnitude of the eigenvalues and their respective eigenvectors. Because the covariance matrix is real and symmetric its eigenvectors are orthonormal.

Appendix B

The Hessian Matrix and the Outer Product Method

The Hessian matrix, which is required for the calculation of the β-network error function (4.23) and therefore the gradient function (4.25) is a matrix of second derivatives of the network error with respect to the weights with elements

$$\mathbf{H} = \frac{\partial^2 E}{\partial w_{ij} \partial w_{kl}}, \tag{B.1}$$

equation

where the subscripts i, j, k and l are nodes in the network between which the weights apply. This is the generic formula for the Hessian and takes into account weights between input and hidden units and hidden and output units. In the case of the RBF network used in RBF3CMOD, the Hessian comprises elements associated with the hidden to output units only.

The sum-of-squares output error function is being used, therefore

$$E = \frac{1}{2} \sum_{n=1}^{N} (y_n - t_n)^2. \tag{B.2}$$

The second derivative with respect to the weights may be expressed as

$$\mathbf{H} = \sum_{n=1}^{N} \left(\frac{\partial y_n}{\partial w_{ij}} \frac{\partial y_n}{\partial w_{kl}} + (y_n - t_n) \frac{\partial^2 y^n}{\partial w_{ij} \partial w_{kl}} \right). \tag{B.3}$$

The second term on the right hand side will be rendered negligible if

- The outputs of the network are close to the target data set.

- The second derivative of the outputs is small.

In the case of RBF3Cmod, where the model (4.29) has been contrived to be linear in the weights, then the second derivative is zero and the second term disappears completely. (B.3) then reduces to the outer product approximation and may be written as

$$\mathbf{H} = \sum_{n=1}^{N} \mathbf{g}_n \mathbf{g}_n^T, \tag{B.4}$$

where $\mathbf{g}$ is defined in (4.24). It is this method which is used in the application of IDN inference to RBF3Cmod.

It is worth noting that the outer product can be used for an MLP but, because it is non linear in the weights (see [1] 4.1.1 and 4.10.2 for a detailed explanation) and the network is not fully trained or the data is noisy, such that $y - t$ is large, then the second term in (B.3) cannot be discounted and the outer product technique becomes an approximation. There are methods of accurately computing the Hessian for an MLP but non to date as computationally efficient as the outer product. It is for this reason that the NN3CMOD was adapted to use RBFs.

Bibliography

[1] C.M. Bishop. *Neural Networks For Pattern Recognition.* Oxford University Press, 1995.

[2] D. Cornford, I.T. Nabney, and G. Ramage. Improved neural network scatterometer forward models. Technical report, Neural Computing Research Group, Aston University, Birmingham B4 7ET, UK, 1999.

[3] Qazhaow Salih Qazaz. *Bayesian Error Bars for Regression.* PhD thesis, The University of Aston in Birmingham, Birmingham B4 7ET, UK, November 1996.

[4] P.J. Rousseeuw. Least median of squares regression. *Journal of the American Statistical Society*, 79:871–880, 1984.

[5] P.J. Rousseeuw. Multivariate estimation with high breakdown point. In W. Grossman, G. Pfling, I. Vincze, and W. Werly, editors, *Mathematical Statistics and Applications*, volume B, pages 293–297. Reidel, Dordecht, Netherlands, 1985.

[6] A. Stoffelen and D. Anderson. Scatterometer Data Interpretation: Estimation and Validation of the Transfer Function CMOD4. *Journal of Geophysical Research*, 8:5767–5780, 1997.

[7] J.F.M. Svensén. *GTM: The Generative Topographic Mapping.* PhD thesis, Neural Computing Research Group, Aston University, Birmingham B4 7ET, UK, 1998.

[8] P. Tiňo, I.T. Nabney, and Yi Sun. Using directional curvatures to visualize folding patterns of the gtm projection manifolds. Technical report, Neural Computing Research Group, Aston University, Birmingham B4 7ET, UK, 2001.

[9] W.A. Wright, G. Ramage, D. Cornford, and I.T. Nabney. Neural network modelling with input uncertainty: Theory and application. *Journal of VLSI Signal Processing*, 26:169–188, 2000.