View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Aston Publications Explorer

Neural Computing Research Group
Dept of Computer Science & Applied Mathematics
Aston University
Birmingham B4 7ET
United Kingdom
Tel: +44 (0)121 333 4631
Fax: +44 (0)121 333 4586
http://www.ncrg.aston.ac.uk/

First Year Qualifying Report: Neural Networks for Extracting Wind Vectors from Satellite Scatterometer Data

David J. Evans
evansdj@aston.ac.uk

Technical Report NCRG/99/005 January 12, 1999

Abstract

The ERS-1 Satellite was launched in July 1991 by the European Space Agency into a polar orbit at about 800 km, carrying a C-band scatterometer. A scatterometer measures the amount of radar back scatter generated by small ripples on the ocean surface induced by instantaneous local winds. Operational methods that extract wind vectors from satellite scatterometer data are based on the local inversion of a forward model, mapping scatterometer observations to wind vectors, by the minimisation of a cost function in the scatterometer measurement space.

This report uses mixture density networks, a principled method for modelling conditional probability density functions, to model the joint probability distribution of the wind vectors given the satellite scatterometer measurements in a single cell (the 'inverse' problem). The complexity of the mapping and the structure of the conditional probability density function are investigated by varying the number of units in the hidden layer of the multi-layer perceptron and the number of kernels in the Gaussian mixture model of the mixture density network respectively. The optimal model for networks trained per trace has twenty hidden units and four kernels. Further investigation shows that models trained with incidence angle as an input have results comparable to those models trained by trace. A hybrid mixture density network that incorporates geophysical knowledge of the problem confirms other results that the conditional probability distribution is dominantly bimodal.

The wind retrieval results improve on previous work at Aston, but do not match other neural network techniques that use spatial information in the inputs, which is to be expected given the ambiguity of the inverse problem. Current work uses the local inverse model for autonomous ambiguity removal in a principled Bayesian framework. Future directions in which these models may be improved are given.

Contents

1 Introduction **7**
1.1 The geophysical problem 7
1.2 Measurement acquisition 7
1.3 Background 8
1.4 The NEUROSAT project at the Neural Computing Research Group (NCRG) 10
1.5 The aims of this research project 11
1.6 Report outline 11

2 Methods and results **12**
2.1 Theory of mixture density networks 12
2.2 Mixture density network implementation 15
2.2.1 Training mixture density networks with 'fast mdn' . 16
2.2.2 Putting mixture density networks into a geophysical context 16
2.3 Data pre-processing 16
2.3.1 Data description 17
2.3.2 Pre-Processing the wind data 17
2.3.3 Pre-processing the scatterometer data 18
2.4 Direct application of mixture density networks 18
2.5 Incorporating geophysical knowledge 19
2.6 Results 21

3 Analysis and discussion **32**
3.1 Analysis of the inverse models 32
3.1.1 The complexity of the mapping $\boldsymbol{\sigma^o} \rightarrow (u, v)$ 32
3.1.2 The complexity of the conditional probability distribution $P(u, v \mid \boldsymbol{\sigma^o})$ 34
3.2 Comparison with other neural network methods 35

4 Conclusions and future work **44**
4.1 Conclusions 44
4.2 On-going work 44
4.3 Potential future work 46
4.3.1 Further investigation on the $\boldsymbol{\sigma^o} \rightarrow (u, v)$ mapping . 46
4.3.2 Further investigation of the structure of the probability distribution $P(u, v \mid \boldsymbol{\sigma^o})$ 46
4.3.3 Further work to improve generalisation 46

A The gradient of the error function of a MDN **49**
A.1 The Error function and its partial derivatives 49
A.1.1 Computing the derivatives of the mixing coefficients 50
A.1.2 Computing the derivatives of the variances 51
A.1.3 Computing the partial derivative with respect to the kernel centres 52
A.2 Summary of results 53

B The gradient of the error function of the hybrid MDN **54**
B.1 The Error function and its partial derivatives 54
B.1.1 Computing the derivatives of the mixing coefficients 54
B.1.2 Computing the derivatives of the kernel variances (widths) 55
B.1.3 Computing the derivatives of the kernel centres (means) 56
B.2 Summary of results 56

C Definitions of the summary measures used in the results **57**
C.1 Disambiguation for model appraisal 57
C.2 Figure of Merit 57
C.3 Predicted wind vector root-mean-square error 58
C.4 *Performance at* 20° 58

List of Tables

1 *FoM* results - for networks trained per trace. 24

2 Vector RMS error results - for networks trained per trace. 25

3 *Performance* @ 20°- for networks trained per trace. 26

4 Performance results over the whole swathe - for networks trained with incidence angle as an input. 27

5 *FoM* results - by trace, for networks trained with incidence as angle as an input. . 28

6 Vector RMS error - by trace, for networks trained with incidence as angle as an input. 29

7 *Performance* @ 20°- by trace, for networks trained with incidence as angle as an input. 30

8 Comparing the direction performance of the best MDNs with published results. . . 36

9 Comparing the speed performance of the best MDNs with published results 38

List of Figures

1 Relationship of the ERS-1 satellite antennas and the ocean surface. For simplicity, only nine non-overlapping satellite tracks are shown. 8

2 A two dimensional sketch of the scatterometer measurement space. 9

3 The structure of a Mixture Density Network. 13

4 The relationship between the wind direction angles, *mdir*, *vdir*, and *rdir* used in pre-processing. 18

5 Conditional probability distribution plots, for a model with 2 kernels and 25 hidden units. 23

6 Conditional probability distribution plots, for a hybrid model with 2 kernels and 25 hidden units. 23

7 Conditional probability distribution plots, for a model with 4 kernels and 25 hidden units. 27

8 Conditional probability distribution plots with incidence angle, for a model with 5 kernels and 50 hidden units. 31

9 Conditional probability distribution plots with incidence angle, for a model with 12 kernels and 50 hidden units. 31

10 Model performance by the number of hidden units in the MLP, for models with two kernels and trained per trace. 33

11 Model performance by the number of hidden units in the MLP, for models with hybrid kernels and trained per trace. 34

12 Model performance by the number of hidden units in the MLP, for models with four kernels and trained per trace. 35

13 Model performance by the number of kernels. For a network with thirty five units in the hidden layer, and trained with incidence angle as an input 37

14 Model performance by the number of kernels. For a network with fifty units in the hidden layer, and trained with incidence angle as an input 40

15 Model performance, comparing networks which are trained by trace, by the number of kernels 40

16 An example of $P(u, v \mid \boldsymbol{\sigma^o})$ where the modes are non-Gaussian 41

17 Model performance, comparing networks which take incidence angle as an input, by the number of kernels, ... 41

18 An example of overfitting in $P(u, v \mid \boldsymbol{\sigma^o})$ space. 42

19 Scatter plots of observed vs predicted wind directions 42

20 Scatter plots of observed vs predicted wind speed 43

1 Introduction

This report investigates a particular inverse modelling problem within the field of the geophysical sciences. It is one of inferring wind vectors that describe wind speed and direction over the ocean surface, from satellite scatterometer data (measured from the environment), and aims to solve this problem using advanced statistical models called neural networks.

In this section the work of other authors in the area is introduced along with a general background to satellite scatterometry.

1.1 The geophysical problem

Numerical Weather Prediction (NWP) models are tools used by oceanographers, meterologists, geophysicists and many other scientists to forecast the future state of the atmosphere. Initial conditions for the NWP model are important, as they are essential for accurate predictions. The initial condition of the NWP model is the current state of the atmosphere described by the parameters of the model including wind speed and wind direction. Depending on the context, wind vectors may be defined in polar coordinates (s, χ) or Cartesian coordinates (u, v). The wind vectors are inferred, by an inverse model, from scatterometer data collected from space borne satellites. Such scatterometers provide fast and accurate global coverage of the world's oceans, providing an up to date 'picture' of the current state of the winds over the ocean. The performance of the NWP model is then dependent on the quality of the initial conditions which are in turn dependent on the quality of the model used to infer the model parameters, which depends on the quality of the measurements collected by the space borne scatterometers. Therefore new or improved methods for solving the inverse problem are of interest to a wide range of scientists and add value to the quality of weather forecasts.

1.2 Measurement acquisition

The ERS-1 Satellite was launched in July 1991 by the European Space Agency into a polar orbit at about 800 km, carrying a C-band scatterometer. The scatterometer has a microwave radar operating at 5.3 GHz, and measures the amount of back scatter generated by small ripples on the ocean surface of about 5 cm wavelength (Robinson, 1985). The scatterometer has three independent antenna which point, on a horizontal plane, in three directions, 45°, 90°, 135°, with respect to the satellite propagation and are referred to as fore, mid and aft beam antennae respectively. The satellite samples a swathe of ocean surface approximately 500 km by 500 km. This swathe is divided into nineteen tracks, where each track is approximately 25 km wide. Each measurement cell is approximately 50 km by 50 km and so there is some overlap between adjacent tracks. Each track is identified by the incidence angle of the mid beam with respect to the local vertical, which varies from 18° to 45°, and is numbered from 1 to 19 respectively (see Figure 1). The odd numbered tracks are referred to as traces, which are identified as trace 0 to trace 9 where trace 0 is the inner most trace with respect to the satellite (has the smallest incidence angle). As the satellite passes over the ocean surface, each cell is illuminated by the footprint of each antenna; fore, mid and aft beam respectively, and a measurement vector for each cell is collected, $(\sigma^o_{\mathrm{f}}, \sigma^o_{\mathrm{m}}, \sigma^o_{\mathrm{a}})$. This is referred to as the normalised radar cross subsection, denoted by $\boldsymbol{\sigma^o}$ and has units of decibels.

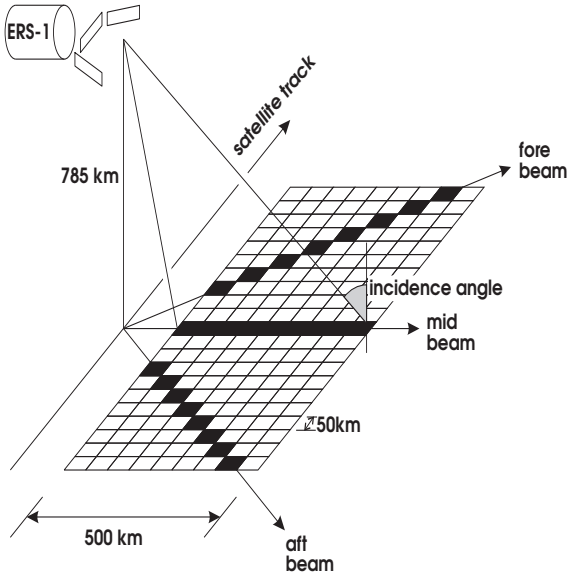


Figure 1: Relationship of the ERS-1 satellite antennas and the ocean surface. For simplicity, only nine non-overlapping satellite tracks are shown.

1.3 Background

There is a unique set of wind vectors, called the *noisy ambiguity set*, which is identifiable from a single scatterometer measurement. This set shows an inverse mapping exists and is multi-valued (Long and Mendel, 1991).

Much effort has been applied to understanding the scatterometer measurement space. An empirical forward model (called the Geophysical Transfer Function (GTF)) has been developed that describes the mapping from wind speed and direction to the scatterometer space. The first model, CMOD2, was calibrated by the RENE-91 campaign of the coast of Norway (Offiler, 1994). The GTF has been further calibrated to the now operational model CMOD4 (Stoffelen and Anderson, 1997b).

Stoffelen and Anderson (1997c) show that scatterometer measurements lie close to a three dimensional manifold defined by CMOD4 in measurement space, and are largely dependent on two geophysical parameters, wind speed and direction. In general the measurements lie within 0.2 dB of the manifold (corresponding to an uncertainty in wind vector rms error of 0.5 ms^{-1}), which is close to the instrumental measurement noise level.

Ambiguity in wind direction arises from noise on the observation and it becomes difficult to distinguish if winds are blowing toward or away from the antenna. This is illustrated in Figure 2, a sketch of a two dimensional slice through the manifold defined by CMOD4 at a roughly constant speed. If the observations were noiseless then they would fall on the surface of the manifold, somewhere on the solid black line. The areas of ambiguity would only exist where the black lines crossed for a few wind directions. Now consider adding noise to the observation. The observation is placed somewhere near the surface of the manifold in the gray area. Theoretically a noisy observation can come from one of the two surfaces of the manifold to which it is normal to. Thus, there is no way to distinguish which is the correct solution, and it follows that there are at least two possible solutions for the wind direction for that observation. Any method of inversion will have multi-valued solutions for wind direction given a single observation.

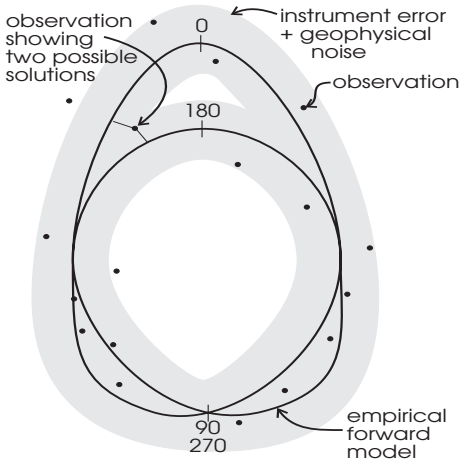


Figure 2: A two dimensional sketch of the scatterometer measurement space. The two dimensional slice is taken through the measurement manifold at constant wind speed. For a noisy observation there are at least two solutions in wind direction.

Most inversion methods which extract wind vectors from scatterometer data are based on local inversions of a GTF. These methods invert the GTF by finding a triplet on the measurement manifold that is closest to the observed measurement triplet by minimising some cost function that describes the distance between the observed and approximated measurements (Stoffelen and Anderson, 1997c). Following inversion, the 'best' solutions are chosen by comparison to a NWP background wind field. A heuristic filter is then applied over the wind field to smooth and remove non-geophysical structures within the wind field (Stoffelen and Anderson, 1997a). These methods successfully invert the measurements and show that scatterometer measurements can provide higher resolution wind fields than those generated at the European Centre for Medium range Weather Forcasting (ECMWF). These methods are not autonomous since they rely on NWP winds for selection of the initial wind fields.

Thiria *et al.* (1993) used neural networks to model wind direction and speed directly from simulated scatterometer data. The model consisted of two neural networks. One network modelled wind speed, the other, a classification network with thirty six bins representing ten degree intervals, modelled wind direction by interpreting the outputs of each bin as probability. The inputs to the neural network took neighbourhood information from the eight surrounding cells of a nine by nine grid, where the centre cell was the measurement of interest. This was found to improve the performance by seventeen percent, showing that a spatial context may well be an important consideration in the inverse model. Another interpretation of this improvement is that this configuration may provide additional wind direction disambiguation skill to the network.

The network for modelling wind direction has inputs of spatial information (the same as those used for the wind speed network) and the wind speed estimate. Wind speed as an input improves the position of the solution on the measurement manifold (the shape of Figure 2 is strongly dependent on wind speed). Taking speed as an additional input was found to improve performance of direction estimation. Simulated data was used because ERS-1 was not fully operational and the results showed neural networks to be a promising avenue of investigation for a solution to this inverse problem.

Cornford *et al.* (1997) applied a feed forward neural network to model wind speed and a mixture density network (see section 2.1) with kernels of circular normal densities (Bishop and Nabney, 1996) to model the full conditional probability density of the wind direction given the scatterometer measurements, $\boldsymbol{\sigma^o}$. Two configurations of mixture density networks were considered for modelling

wind direction. In the first configuration the kernels were free to move, while in the second the centres and variances of the kernels were fixed. A committee of the direction networks (made up of members from the two configurations just mentioned) was also considered. Additional to the scatterometer triplet measurement $(\sigma^o_{\rm f}, \sigma^o_{\rm m}, \sigma^o_{\rm a})$, incidence angle was also included as an input to the networks. The wind speed model performed within the designed specification of the instrument of 2 ms^{-1}, the results being comparable to the inverted CMOD4 model (Stoffelen and Anderson, 1997c), although the model had some difficulty in learning the transfer function at high and low wind speeds. For wind direction, the models learnt the inherent ambiguity in the problem, but did not perform as well as the inverted CMOD4 model. However, these results are encouraging, for the committee of networks the solution for direction (to within 20°) was correct roughly 75% of the time when considering the two most likely solutions. Ambiguity removal was not addressed in this work.

Following the methods of Thiria *et al.* (1993), Richaume *et al.* (1998) continued to address the inverse problem by training the networks using data collected from the ERS-1 satellite. There is a dedicated transfer function for wind speed and wind direction for each trace. The results reported show performance to be better than the methods proposed by wind retrieval systems based on the CMOD4 GTF. Ambiguity removal is achieved by using a NWP background wind to initialise the system, and then applying a three by three cell spatial filter over the wind field to minimise the global wind variance within the spatial filter. The results show wind fields that are consistent, and compare favourably wind fields retrieved from the same measurements by the European Centre for Medium range Weather Forcasting.

1.4 The NEUROSAT project at the Neural Computing Research Group (NCRG)

The NEUROSAT project is concerned with applying neural network approaches to problems in satellite remote sensing to extract wind vectors from satellite scatterometer measurements taken over the ocean. There are three distinct areas of research, although the boundaries are not distinct:

- Solving the forward model, the mapping of $(u, v) \rightarrow \boldsymbol{\sigma^o}$, by building the probabilistic model $P(\boldsymbol{\sigma^o} \mid u, v)$.
- Solving the inverse model, the mapping of $\boldsymbol{\sigma^o} \rightarrow (u, v)$, by building the probabilistic model $P(u, v \mid \boldsymbol{\sigma^o})$.
- Autonomous ambiguity removal. Predicting wind fields without reference to NWP model winds. Both heuristic and Bayesian methods are applied to this problem.

This report contributes towards the NEUROSAT project by investigating the feasibility of building the inverse model by using the mixture density network framework of Bishop (1994). Guillaume Ramage, a fellow research student, has investigated the forward model (Ramage, 1998). Dan Cornford and Ian Nabney oversee the project, and have written several publications about modelling wind speed/direction and generating wind priors. For further information see Cornford and Nabney (1998)[1]

[1] Available from `http://www.ncrg.aston.ac.uk/Papers/`

1.5 The aims of this research project

Probabilistic models provide a general model for uncertainty in the natural world. Our aim is to build a local (that is, for a 50 km by 50 km cell in a trace) probability model that describes the probability of a set of wind vector components, (u, v), given a satellite scatterometer observation, $\boldsymbol{\sigma^o}$, expressed as $P(u, v \,|\, \boldsymbol{\sigma^o})$. In geophysical terms this is called an *inverse model*. The probability model is implemented using a Mixture Density Network (MDN) framework, which provides a principled method for modelling conditional probabilities (Bishop, 1994). Furthermore, a hybrid MDN will be developed that incorporates the geophysical knowledge inherent in the problem, namely the relationship between ambiguous wind directions. This is achieved by fixing the relative ambiguous wind directions within the MDN framework.

Mixture density networks facilitate the investigation of complexity of the mapping from scatterometer data to wind vector component space ($\boldsymbol{\sigma^o} \rightarrow (u, v)$) *and* the complexity of the conditional joint probability distribution ($P(u, v \mid \boldsymbol{\sigma^o})$) itself. This investigation attempts to answer the following questions:

- How difficult is it to model speed and direction simultaneously by directly mapping to the wind component space?
- Can the incidence angle be used as input to the MDN? That is do models trained over all tracks of the swathe perform as well as those models trained on each track within the swathe (and do not take incident angle as an input)?
- Is the conditional probability distribution of the Cartesian wind vectors, (u, v), bimodal or more complex?
- Is the noise on the Cartesian wind vector components Gaussian?
- How well do the hybrid MDN models compare with standard MDN models of similar complexity?

1.6 Report outline

In this section, the introduction to the field has been described, and the aims of this project set out.

In Section 2 the reader is introduced to the mixture density network, and the technical details of its construction. The data pre-processing is described before the experimental details are outlined. The results of the experiments are presented and discussed with reference to summary measures used within the meteorological community.

In Section 3 the results are analysed and discussed with reference to the complexity of the mapping, and the conditional probability distribution. The results are then compared with other published results in the field.

Finally, in Section 4 the conclusions of this project are presented. On-going work is described followed by potential future work which indicates how the models might be improved.

2 Methods and results

In this section the methods and results of experiments to build an inverse model that maps the satellite scatterometer data directly to wind vector component space are presented. The models employ the mixture density network framework (and includes a hybrid mixture density network specifically designed to model the directional ambiguity), which is principled method to model complicated conditional probability density functions.

2.1 Theory of mixture density networks

Mixture Density Networks (MDNs) provide a framework for modelling conditional probability density functions, denoted $p(\boldsymbol{t}|\boldsymbol{x})$ (Bishop, 1994; Bishop, 1995). The distribution of the outputs, $\boldsymbol{t}$, is described by a parametric model whose parameters are determined by the output of a neural network, which takes $\boldsymbol{x}$ as its inputs. The general model is described by the equations

$$p(\boldsymbol{t}|\boldsymbol{x}) = \sum_{j=1}^{M} \alpha_j(\boldsymbol{x})\phi_j(\boldsymbol{t}|\boldsymbol{x}) \tag{1}$$

and

$$\sum_{j=1}^{M} \alpha_j(\boldsymbol{x}) = 1. \tag{2}$$

Here $\alpha_j(\boldsymbol{x})$ represents the mixing coefficients (which depend on $\boldsymbol{x}$), $\phi_j(\boldsymbol{t}|\boldsymbol{x})$ are the kernel distributions of the mixture model (whose parameters also depend on $\boldsymbol{x}$), and M is the number of kernels in the mixture model. There are various choices available for the kernel functions, but for the purposes of this report the choice has been restricted to spherical Gaussians of the form:

$$\phi_j(\boldsymbol{t}|\boldsymbol{x}) = \frac{1}{(2\pi)^{\frac{c}{2}}\sigma_j^c(\boldsymbol{x}_n)} \exp\left(-\frac{\|\boldsymbol{t}_n - \boldsymbol{\mu}_j(\boldsymbol{x}_n)\|^2}{2\sigma_j^2(\boldsymbol{x}_n)}\right), \tag{3}$$

where c is the dimensionality of the target space $\boldsymbol{t}$. This is a valid restriction because in principle a Gaussian Mixture Model (GMM) with sufficiently many kernels of the type given by (3) can approximate arbitrarily closely a density function of any complexity providing the parameters are chosen correctly (McLachlan and Bashford, 1988). It follows then that for any given value of $\boldsymbol{x}$, the mixture model (1) can model the conditional density function $p(\boldsymbol{t}|\boldsymbol{x})$. To achieve this the parameters of the mixture model[2] are taken to be general continuous functions of $\boldsymbol{x}$. These functions are modelled by the outputs of a conventional neural network that takes $\boldsymbol{x}$ as its input. It is this combination of a GMM whose parameters are dependent on the output ofa feed forward neural network that takes $\boldsymbol{x}$ as its inputs that is referred to as a *Mixture Density Network* (MDN) and is represented schematically in Figure 3.

By choosing enough kernels in the mixture model and a neural network with sufficiently many hidden units the MDN can approximate as closely as desired any conditional density, $p(\boldsymbol{t}|\boldsymbol{x})$ (Bishop, 1994). The neural network element of the MDN is implemented with a standard Multi-Layer Perceptron (MLP) with single hidden layer of tanh units and an output layer of linear units. The output vector from the MLP, $\boldsymbol{Z}$, holds the parameters that the define the Gaussian mixture

[2]Choosing a spherical Gaussian kernel determines the parameters to be the mixing coefficients and the variances and centres (or means) of the kernel functions.

model. A single row of the parameter vector, for the n^{th} pattern, takes the following form:

$$[\underbrace{\alpha_{1,n}, \alpha_{2,n}, \cdots, \alpha_{j,n}, \cdots, \alpha_{M,n}}_{M \text{ mixing coefficients}},$$
$$\underbrace{\mu_{11,n}, \mu_{12,n}, \cdots, \mu_{1c,n}}_{1^{st} \text{ kernel centre}}, \cdots, \underbrace{\mu_{j1,n}, \mu_{j2,n}, \cdots, \mu_{jc,n}}_{j^{th} \text{ kernel centre}}, \cdots,$$
$$\underbrace{\mu_{M1,n}, \mu_{M2,n}, \cdots, \mu_{Mc,n}}_{M^{th} \text{ kernel centre}}, \cdots,$$
$$\underbrace{\sigma^2_{1,n}, \sigma^2_{2,n}, \cdots, \sigma^2_{j,n}, \cdots, \sigma^2_{M,n}}_{M \text{ widths}}]. \quad (4)$$

Where the total number of outputs from the MLP is $(c+2) \times M$, as compared with the usual c outputs for a MLP network used in the conventional manner. In the style of Bishop (1994) outputs of the MLP are denoted by z_i. These outputs undergo some transformations to satisfy the constraints of the mixture model. The first constraint is that

$$\sum_{j=1}^{M} \alpha(\boldsymbol{x}) = 1 \quad (5)$$

and

$$0 \leqslant \alpha(\boldsymbol{x}) \leqslant 1 \qquad \text{for} \quad j = 1, \ldots, M. \quad (6)$$

The outputs of the MLP which correspond to the mixing coefficients, z_j^{α}, are constrained using the 'softmax' function:

$$\alpha_j = \frac{\exp(z_j^{\alpha})}{\sum_{i=1}^{M} \exp(z_i^{\alpha})}. \quad (7)$$

This mapping ensures that the mixing coefficients always sum to unity. The variance of the kernel represents a scale parameter and always takes a positive value. The variance parameters of the kernels are represented by exponentials of the corresponding outputs of the MLP, z_j^{σ}:

$$\sigma_i^2 = \exp(z_j^{\sigma}). \quad (8)$$

The centres of the Gaussians represent a location in the target space and can take any value within that space. They are therefore taken directly from the outputs from the MLP, z_{jk}^{μ}:

$$\mu_{jk} = z_{jk}^{\mu} \quad (9)$$

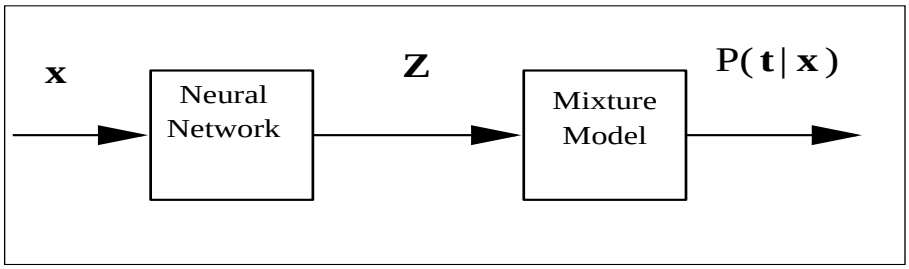


Figure 3: The structure of a Mixture Density Network. The inputs **x** are feed through a neural network. The outputs of the neural network, **Z**, define the parameters of the GMM

In order to optimise the parameters in a MDN, an error function is required that provides an indication of how well the model represents the underlying generating function of the training data. The error function of the mixture density network is motivated from the principle of maximum likelihood (Bishop, 1995). The likelihood of the training data set, $\{\boldsymbol{x}, \boldsymbol{t}\}$, may be written as:

$$\begin{aligned} \mathcal{L} &= \prod_n p(\boldsymbol{x}_n, \boldsymbol{t}_n) \\ &= \prod_n p(\boldsymbol{t}_n|\boldsymbol{x}_n) p(\boldsymbol{x}_n), \end{aligned} \tag{10}$$

where the assumption has been made that each data point has been drawn independently from the same distribution, and so the likelihood is a product of probabilities. Generally one wishes to maximise the likelihood function. However, in practice, it is usual to *minimise* the negative logarithm of the likelihood function (termed the negative log likelihood). These are equivalent procedures, since the negative log likelihood is a monotonic function. The error function E is defined as the negative log likelihood:

$$E = -\ln \mathcal{L} = -\sum_n \ln p(\boldsymbol{t_n}|\boldsymbol{x_n}) - \sum_n p(\boldsymbol{x}_n). \tag{11}$$

The second term in (11) is constant because it is independent of the network parameters and can be removed from the error function. The error function becomes:

$$E = -\sum_n \ln p(\boldsymbol{t}_n|\boldsymbol{x}_n). \tag{12}$$

Comparing (12) with (1), we substitute (1) into (12) and derive the negative log likelihood error function for the mixture density network:

$$E = -\sum_n \ln\left\{\sum_{j=1}^{M} \alpha_j(\boldsymbol{x}_n)\phi_j(\boldsymbol{t}_n|\boldsymbol{x}_n)\right\}. \tag{13}$$

In order to minimise the error function the derivatives of the error E with respect to the network weights must be calculated. Providing that the derivatives can be computed with respect to the output of the neural network, the errors at the network inputs may be calculated using the back-propagation procedure (Bishop, 1995). By first defining the posterior probability of the j^{th} kernel, using Bayes theorem:

$$\pi_j(\mathbf{x}, \mathbf{t}) = \frac{\alpha_j \phi_j}{\sum_{l=1}^{m} \alpha_l \phi_l} \tag{14}$$

the analysis of the error derivatives with respect to the network outputs is simplified. The computation of the error dervivative is further simplified by considering the error derivatiave with respect to each training pattern, n. The total error, E, is defined as a sumation of the error, E_n, for each training pattern:

$$E = \sum_{n=1}^{N} E_n, \tag{15}$$

where

$$E_n = -\ln\left\{\sum_{j=1}^{m} \alpha_j(\mathbf{x}_n)\phi_j(\mathbf{t}_n|\mathbf{x}_n)\right\}. \tag{16}$$

Each of the derivatives of E_n are considered with respect to the outputs of the networks and their respective labels for the mixing coefficients, z_j^α, variance parameters, z_j^σ and centres or position parameters z_{jk}^μ. The derivatives are as follows:

$$\frac{\partial E_n}{\partial z_j^\alpha} = \alpha_j - \pi_j, \tag{17}$$

$$\frac{\partial E_n}{\partial z_j^\sigma} = -\frac{\pi_j}{2}\left\{\frac{\|\mathbf{t}_n - \boldsymbol{\mu}_j\|^2}{\sigma_j^2} - c\right\}, \tag{18}$$

$$\frac{\partial E_n}{\partial z_{jk}^\mu} = \pi_j\left\{\frac{\mu_{jk} - t_k}{\sigma_j^2}\right\}, \tag{19}$$

where a full derivation is given in Appendix A.

2.2 Mixture density network implementation

This algorithm is conveniently implemented in the NETLAB[3] toolbox for MATLAB. The following toolbox functions are of interest to the reader

`mdn`	creates a data structure to model a MDN. This structure comprises of the feed forward network structure (an MLP) and a structure for the mixture model parameters.
`mdninit`	initialises the weights of the network. Uses the target data $\boldsymbol{t}$ to initialise the biases for the output units of the network after initialising the other weights randomly with a Gaussian prior. The biases are initialised by the parameters of a model of the unconditional density of $\boldsymbol{t}$. These parameters are computed using tool box function `gmminit`, which uses the k-means algorithm to compute the parameters of the unconditional model of $\boldsymbol{t}$.
`mdnfwd`	forward propagates the inputs through the model. The output is an array of structures containing a mixture model for each input pattern.
`mdnerr`	computes the error of the model for a set of inputs and targets.
`mdngrad`	computes the error gradient of the model using the results of (17), (18) and (19) and back propagating these results through the network using the tool box function `mlpbkp`
`mdnpak/unpak`	packs the weights of the network into a vector: this is required to use the optimisation routines.
`scg`	implementation of the scaled conjugate gradients algorithm, which is a general purpose optimisation algorithm.

All the MDNs trained in this project were optimised using the Scaled Conjugate Gardient (SCG) algorithm. A demonstration programme `demmdn1` is available from the web site which gives a worked example of training a MDN on the 'toy problem' described by Bishop (1994).

[3]Available from `http://www.ncrg.aston.ac.uk/netlab/`

2.2.1 Training mixture density networks with 'fast mdn'

Initial experiments took some time to train (between a few days to one week depending on the computer) and so some time was spent during this project developing a *fast mdn* NETLAB implementation of MDNs. The focus of this work was re-engineering the computation of the error derivatives with respect to the neural network outputs, so that the computation was carried out in parameter space (4). The necessary expressions for the derivatives are contained in Appendix A. The details of the software re-engineering are given in Evans (1998)[4]. In summary the training time is decreased by a factor of sixty for the example provided in the technical report. The inverse model training time decreased from a few days to a few hours for networks without incidence angle as input. Decreased training time means that problems with larger data sets can be trained in a realistic time frame. For this project it was also possible to train networks that take incidence angle as an input and have a training data set size of ten thousand examples.

2.2.2 Putting mixture density networks into a geophysical context

The aim of this subsection is to show how a MDN may be employed to model the inverse mapping from scatterometer data, $\boldsymbol{\sigma^o}$, directly to the wind vector component space, (u, v). In Subsection 2.1 the inputs and targets of the MDN are labelled as $\boldsymbol{x}$ and $\boldsymbol{t}$ respectively. In the context of this application, each input pattern for the MDN, $\boldsymbol{x}$, will be scatterometer data (the triplet $(\sigma^o_{\mathrm{f}}, \sigma^o_{\mathrm{m}}, \sigma^o_{\mathrm{a}})$), or scatterometer data and incidence angle (the input vector $(\sigma^0_f, \sigma^0_m, \sigma^0_a, \theta)$) if the MDN is being trained over all traces. Modelling the wind vector components directly implies that the targets of the MDN, $\boldsymbol{t}$, are the Cartesian wind vector components (u, v). The general description of the MDN, (1), is then re-expressed using geophysical parameters for a particular wind vector component as

$$p(u, v|\boldsymbol{\sigma^o}) = \sum_{j=1}^{M} \alpha_j(\boldsymbol{\sigma^o})\phi_j(u, v \mid \boldsymbol{\sigma^o}). \tag{20}$$

This project uses data sets generated from real world processes, and so assumptions made in the modelling process need to be validated. There are three main assumptions made about the data set in our approach:

- The noise on the targets in (u, v) space is Gaussian and spherically symmetric.
- The theory in Subsection 2.1 assumes that the inputs, $\boldsymbol{x}$, are noiseless. It is therefore assumed that the inputs, $\boldsymbol{\sigma^o}$, are noiseless. This a reasonable assumption based on the quality of the $\boldsymbol{\sigma^o}$ measurements collected from the ERS-1 satellite (measurements are within $0.2db$ of the CMOD4 manifold, which corresponds to an uncertainty in wind vector rms error of $0.5\ ms^{-1}$) (Stoffelen and Anderson, 1997c) when compared to the errors on numerical weather prediction target winds.
- For computation of the error function (13) it is assumed that all data is independently drawn from the same distribution. The selection of the data ensures that this condition is met.

2.3 Data pre-processing

Before training the MDN the data was pre-processed to create training, validation and test data sets. A data set comprises of pairs of input and target patterns.

[4] Available from `http://www.ncrg.aston.ac.uk/Papers/`

2.3.1 Data description

The input data, $\boldsymbol{\sigma^o}$, is supplied by the European Space Agency (ESA), and is labelled by longitude, latitude and time. The data is then processed at IFREMER where quality control is applied (including sea ice mask filter to remove observations taken over sea ice) to remove low quality observations. The target data is computed from an ECMWF forecast by taking a 250 km by 250 km grid of $10m$ model winds and interpolating in space and time to the satellite observation position. Philippe Richaume (working as part of the NEUROSAT team) selected the data from the North Atlantic regions (1995-96). This data set was then sub-sampled to make training, validation and test data sets. The test data set is generated to have a distribution in wind speed close to a distribution of naturally occurring wind speeds. The training and validation data sets have a wind speed distribution that are an equal combination of uniform distribution over wind speed and a distribution close to that of naturally occurring wind speeds. The wind speed ranges from 4 ms^{-1} to 24 ms^{-1} inclusively. For each measurement in the data set there is a corresponding fore, mid and aft scatterometer measurement, incidence angle, azimuth angle, wind speed and meteo wind direction. It is this data that is referred to as 'raw' data in the scope of this project. Before describing the pre-processing applied to the raw data, some new terminology is needed for the different methods of measuring wind direction (all in degrees):

- *Meteo* wind direction is the angle in which the wind is coming from. Therefore a meteo wind direction of zero degrees describes a wind coming from the north toward the south. This wind direction will be referred to as $mdir$ (see Figure 4).
- *Vector* wind direction is the angle of the direction in which the wind is blowing. Therefore a vector wind direction of zero degrees describes a wind blowing toward the north. This wind direction will be referred to as $vdir$ (see Figure 4).
- *Relative* wind direction is the angle of wind direction relative to the antenna azimuth angle from the vector wind direction. This wind direction will be referred to as $rdir$ (see Figure 4).

All angles given in this document are quoted in degrees, and a clockwise direction from their reference point. Wind speed and direction are resolved into wind vector components, and the input data is pre-processed by a simple linear rescaling.

For each trace (trace 0 to trace 9) there are three data sets. The training set has three thousand examples, whilst the test and validation set have one thousand examples each. There is also a data set that contains data from all traces, and includes incedence angle as an input. This training set has ten thousand examples whilst the validation and test set have five thousand examples.

2.3.2 Pre-Processing the wind data

The data set describes the wind in terms of wind speed and wind direction. To model the wind vector directly, the data is transformed into relative wind vector components $(u_{\mathrm{r}}, v_{\mathrm{r}})$.

The wind direction is transformed from meteo direction, $mdir$, to vector direction , $vdir$, by adding $180°$, and taking the modulus with respect to $360°$. This maintains the convention of wind direction in the range $[0°, 360°)$. Because of the nature of the orbit of the satellite, measurements are taken in two directions (running from north to south, and from south to north). These directions are encoded in the satilite azimuth angle, φ. The vector wind direction is transformed relative to the azimuth angle by subtracting the azimuth angle from vector wind direction, $vdir$, and taking the modulus with respect to $360°$. The wind direction is now relative to the azimuth angle, and the

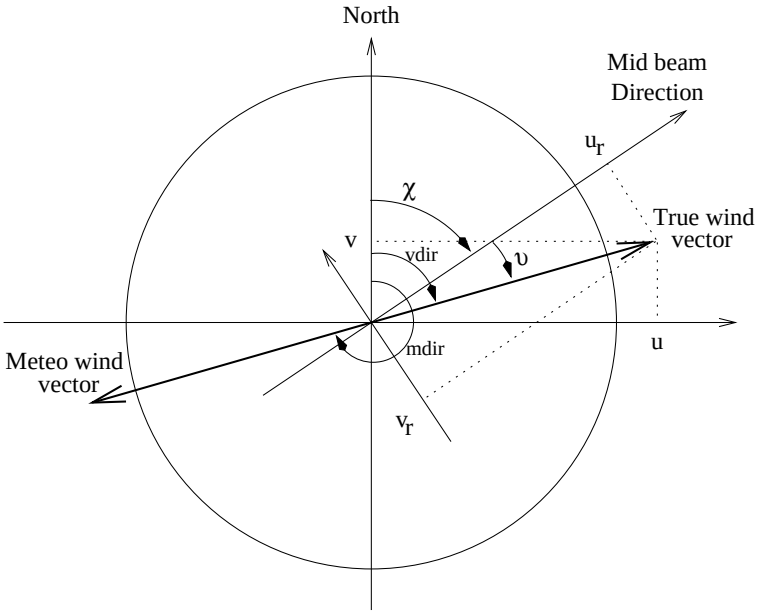


Figure 4: The relationship between the wind direction angles, *mdir*, *vdir*, and *rdir* used in pre-processing.

relative wind vector components can be resolved. To ensure that a geophysical reference of zero degrees for north is maintained *rdir* is subtracted from 90° before either sin or cos are applied. The transformation is summarised below:

1. Compute *vdir*: $vdir = mdir + 180°$ (mod 360°).
2. Compute *rdir*: $rdir = vdir - \varphi$ (mod 360°).
3. Compute u_r: $u_r = s\cos(90° - rdir)$.
4. Compute v_r: $v_r = s\sin(90° - rdir)$.

2.3.3 Pre-processing the scatterometer data

The input data, $\boldsymbol{\sigma^o}$, is pre-processed by a linear transformation to a zero mean unit variance Gaussian distribution. The mean and variance for the rescaling are taking from the whole input data set. For input data that also contains the incidence angle, θ, we pre-process this input by taking the cosine of the incedence angle. This insures that the input is in the range $[-1, 1]$.

2.4 Direct application of mixture density networks

The previous subsections in this section have described the technical detail of how an MDN is implemented, an overview of the implementation of the MDN framework within NETLAB and how the input and target data is pre-processed. This theory is now applied to modelling the mapping of scatterometer data to wind vector components.

The networks are trained by minimising the negative log likelihood error function defined by (13):

$$E = -\sum_{i=1}^{N} \ln\left\{\sum_{j=1}^{M} \alpha_j(\boldsymbol{\sigma_i^o})\phi_j(u_i, v_i \mid \boldsymbol{\sigma_i^o})\right\} \tag{21}$$

where $\boldsymbol{\sigma^o}$ represents either the three or four dimensional input vectors, $(\sigma_{\mathrm{f}}^o, \sigma_{\mathrm{m}}^o, \sigma_{\mathrm{a}}^o)$ and $(\sigma_{\mathrm{f}}^o, \sigma_{\mathrm{m}}^o, \sigma_{\mathrm{a}}^o, \cos(\theta))$ respectively. The network training was regularised by the early stopping technique (Bishop, 1995).

In Subsection 1.5 the areas of investigation were outlined. The first is to investigate the complexity of the mapping between $\boldsymbol{\sigma^o}$ and (u, v) space. This complexity is modelled by the MLP within the MDN structure. The second, the complexity of the (u, v) space, is represented by the number of kernels in the mixture model of the MDN. The third is to train networks with and without incidence angle as an input to the network. The following network configurations were trained to investigate the areas of interest:

- For models without incidence angle as an input:
  - Train a model for each track (0 to 9) and with $10, 15, 20, 25$ hidden units in the MLP.
  - For each model train with two and four centres.
- For models with incidence angle as an input:
  - Train models with 50 and 35 hidden units in the MLP.
  - For each model train with $2, 4, 5$, and 12 centres.

The results of these experiments are presented in Subsection 2.6.

2.5 Incorporating geophysical knowledge

In this subsection we describe a modification to the MDN structure in order to incorporate the knowledge that ambiguities in wind direction exist in the mapping from $\boldsymbol{\sigma^o}$ to (u, v) space. We call this the *hybrid* mixture density network model. This is first and foremost of scientific interest, but also, if successful, will reduce the model complexity by reducing the number of model parameters.

The following expression for MDN with two kernels can be derived from (1):

$$p(\boldsymbol{t}|\boldsymbol{x}) = \alpha(\boldsymbol{x})\phi_1(\boldsymbol{t}|\boldsymbol{x}) + (1 - \alpha(\boldsymbol{x}))\phi_2(\boldsymbol{t}|\boldsymbol{x}). \tag{22}$$

As already established the ambiguity in wind direction arises from the fact that there are alias solutions for the wind direction; that is, it is not known for certain from the $\boldsymbol{\sigma^o}$ data in which of two directions the wind is blowing. This alias is at approximately 180°. The ambiguity is encoded within the MDN framework by two spherical Gaussian kernels with diametrically opposed centres. One kernel is free to move (its parameters are determined during training), whilst the second mirrors the first by taking the negative mean (which is equivalent to an ambiguous direction of 180° in (u, v) space). The centres of the kernels (which correspond to wind vectors in (u, v) space) always represent the ambiguity within the mapping. The noise model for each kernel is assumed to be the same; that is, the variance of the free Gaussian is the same as that of the mirroring Gaussian. The mixing coefficients, then, determine the 'responsibility' that each kernel has for the

probability mass of the given ambiguity within the model. Expressed within the MDN framework the model becomes (for the n^{th} observation):

$$p(\boldsymbol{t_n}|\boldsymbol{x_n}) = \alpha(\boldsymbol{x_n})\phi(\boldsymbol{t_n}|\boldsymbol{x_n}) + (1 - \alpha(\boldsymbol{x_n}))\psi(\boldsymbol{t_n}|\boldsymbol{x_n}), \tag{23}$$

where the kernels are defined by diametrically opposed spherical Gaussians:

$$\phi(\boldsymbol{t_n}|\boldsymbol{x_n}) = \frac{1}{2\pi\sigma^2(\boldsymbol{x_n})} \exp\left(-\frac{\|\boldsymbol{t_n} - \boldsymbol{\mu}(\boldsymbol{x_n})\|^2}{2\sigma^2(\boldsymbol{x_n})}\right), \tag{24}$$

$$\psi(\boldsymbol{t_n}|\boldsymbol{x_n}) = \frac{1}{2\pi\sigma^2(\boldsymbol{x_n})} \exp\left(-\frac{\|\boldsymbol{t_n} + \boldsymbol{\mu}(\boldsymbol{x_n})\|^2}{2\sigma^2(\boldsymbol{x_n})}\right). \tag{25}$$

The target data is two dimensional and therefore the dimension parameter c in the Gaussian model (22) is two. The error for a pattern n is defined as a negative log likelihood function and is derived from (13):

$$E_n = -\ln\left\{\alpha(\boldsymbol{x_n})\phi(\boldsymbol{t_n}|\boldsymbol{x_n}) + (1 - \alpha(\boldsymbol{x_n}))\psi(\boldsymbol{t_n}|\boldsymbol{x_n})\right\}. \tag{26}$$

The training of the network is identical in principle to the general NETLAB MDN framework. Special modifications are needed to compute the function which maps the outputs of the MLP to the parameters of the mixture model and the gradient of the error function with respect to the MLP outputs. The mixing coefficients are no longer constrained by the softmax rule, but by the simpler logistic function:

$$\alpha = \frac{1}{1 + \exp(-z^\alpha)}. \tag{27}$$

To train the hybrid architecture, the derivative of the gradient of the error function with respect to the outputs of the MLP is required. Two posterior probabilities are defined with respect to each kernel, for the free kernel:

$$\pi = \frac{\alpha\phi}{\alpha\phi + (1 - \alpha)\psi}, \tag{28}$$

and for the mirrored kernel:

$$\gamma = \frac{(1 - \alpha)\psi}{\alpha\phi + (1 - \alpha)\psi}. \tag{29}$$

Then the derivatives of the error function with respect to the network outputs are:

$$\frac{\partial E_n}{\partial z^\alpha} = \pi - \alpha, \tag{30}$$

$$\frac{\partial E_n}{\partial z^\sigma} = -\left[\frac{\pi}{2}\left\{\frac{\|\boldsymbol{t_n} - \boldsymbol{\mu}\|^2}{\sigma^2} - c\right\} + \frac{\gamma}{2}\left\{\frac{\|\boldsymbol{t_n} + \boldsymbol{\mu}\|^2}{\sigma^2} - c\right\}\right], \tag{31}$$

$$\frac{\partial E_n}{\partial z_k^\mu} = -\left[\pi\left(\frac{t_k - \mu_k}{\sigma^2}\right) - \gamma\left(\frac{t_k + \mu_k}{\sigma^2}\right)\right]. \tag{32}$$

$$\tag{33}$$

This configuration reduces the number of mixture model parameters by $M(\frac{c}{2} + 1)$. The full detail of the derivation is presented in Appendix B. Software for implementing this architecture is coded in MATLAB and designed to integrate into the NETLAB toolbox (this implementation inspired the general *fast mdn* implementation). The code was tested for accurate implementation using the methods detailed in Evans (1998).

Architectures with the same number of hidden units and input dimensions were trained to compare with the networks trained in Subsection 2.4. For these experimentts the number of kernels is always two (it is possible to have multiple 'centre pairs'). The network architectures trained are as follows:

- For models without incidence angle as an input:
  - Train a model for each trace (0 to 9) and with $10, 15, 20, 25$ hidden units in the MLP.
- For models with incidence angle as an input:
  - Train models with 35 and 50 hidden units in the MLP.

2.6 Results

In total twelve networks were trained for each trace (and so a total of one hundred and twenty networks) and ten networks were trained over all traces. Each network had the same seed in the random number generator when initialised. The results are presented in tabular form, using summary statistics commonly used in the meteorological community; the *Figure of Merit*, an evaluation of how well the predictions compare to the instrument specifications of $\pm 2\ ms^{-1}$ for wind speed and $\pm 20°$ for wind direction; vector Root Mean Square (RMS) error, a measure of how close the predictions are to the target; and *performance at* $20°$ (denoted *perf.* @ $20°$), a measure of the percentage of predicted directions within $20°$ of the target wind directions (Cornford *et al.*, 1997; Richaume *et al.*, 1998). The technical details of these summary measures are detailed in Appendix C. The reported results are computed using the *test set*, which was not used when choosing the model complexity, and so the results are unbiased with respect to the data set. The results are based on wind vector predictions that are derived from a simple ambiguity removal algorithm. The two most likely wind vectors are inferred from the MDN (for models with more than two kernels the position of the two most probable modes, found by a SCG optimisation starting from the positions of each kernel, are inferred as the two most likely wind vectors) and then compared to the target wind vectors. The predicted wind vector closest to the target wind vector is chosen as the disambiguated wind vector. The summary methods are then applied.

Tables 1, 2, and 3 present the summary results for networks trained per trace. Over all tables, there is a general trend of increasing performance from trace 0 to trace 9, which is to be expected because wind vectors are harder to model for the innermost traces of the swathe. Table 1 presents model performance as measured by the FoM evaluation function. An FoM result greater than one indicates the the model is performing to the instrument specifications. Inspection of Table 1 shows that the models are close to this threshold, and all the models except for one meet the specification for trace 9. The performance also exhibits trends over model complexity, where for models with two kernels (including the hybrid configuration) the measure continually improves for increasing complexity, and for models with four kernels the maximum performance is achieved with twenty hidden units in the MLP. The results in Table 2 and Table 3 are strongly correlated to the results in Table 1. For vector RMS errors the current operational model CMOD4 returns wind vector RMS errors of $3\ ms^{-1}$ when compared to ECMWF winds (Stoffelen and Anderson, 1997b). The results presented here show higher values than $3\ ms^{-1}$, but follow the same trends as those in Table 1, the lowest vector RMS error being $3.11\ ms^{-1}$. The results in Table 3 for *perf.* @ $20°$ show similar trends to those for FoM and vector RMS error. Our results are comparable with the results of Cornford *et al.* (1997), where the correct solution, within $20°$, is found more than 70% of the time from the two most probable aliases.

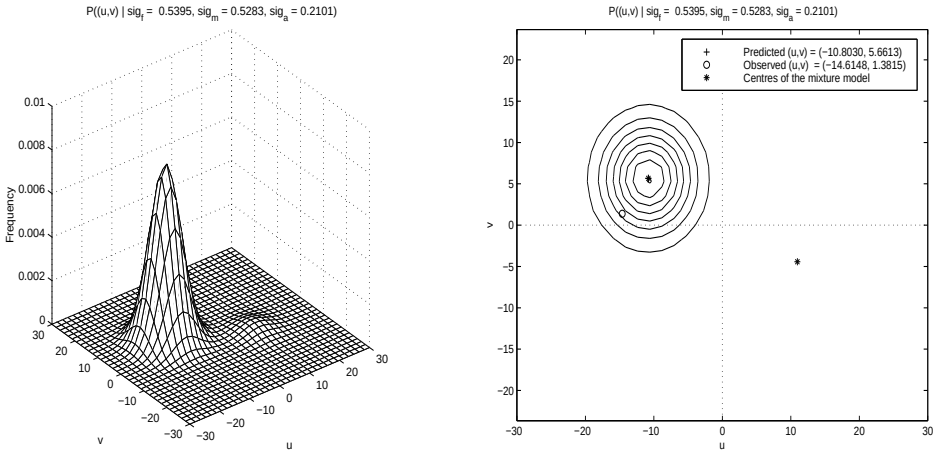
Table 4 presents the results of networks trained over all traces. Model performance is similar to the MDNs with thirty five hidden units in the MLP. In addition, the model performance for MDNs

with twelve kernels is worse than the other kernel configurations, reflecting that there is a limit to the complexity an MDN can have to model this problem.

For comparison between networks with and without incidence angle as an input, results are presented by trace for networks that take incidence angle as an input. The results are generated by using the test sets for models trained by trace and adding the respective incidence angles to the input patterns. Again, the summary results are presented in Tables 5, 6, 7. These tables also show a general trend of improving performance, which is smoother than the networks trained individually per trace. However the best model trained with incidence angle is worse than those trained by trace.

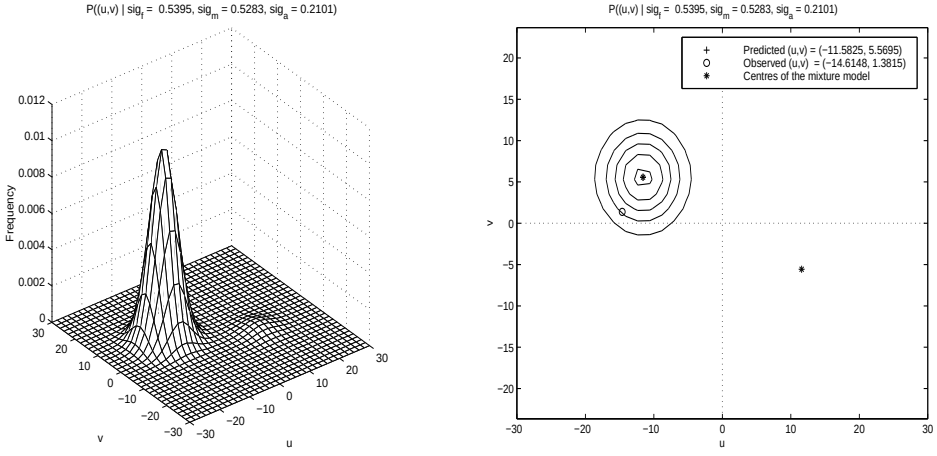
Visualisation of the conditional probability density function modelled by the MDN is of interest. Mesh and contour plots give a good visualisation of the probability density function. Several model architectures have been selected to represent the range of models trained, and to show how the distribution varies for changing kernel configurations. An arbitrary wind vector (that gives a good graphical visualisation of the distribution) was chosen from trace 9 test data (for information the components are $(-14.6, 1.4)$). Figures 5, 6 and 7 are for networks trained per trace and represent kernel configurations of two, two hybrid and four respectively. Figures 8, and 9 are for networks trained with incidence angle as an input and are for MDNs with five and twelve kernels respectively. These plots show that the conditional probability distribution is generally bimodal, closer inspection of Figures 8 and 9 also shows that the modes are not necessarily Gaussian.

In this section the technical detail of MDN*s has been explained including a hybrid* MDN *architecture that models the ambiguity inherent in the mapping from scatterometer space to wind vector space. An overview of the software used to train the* MDN*s has been given by using the* NETLAB *toolbox for* MATLAB*, where* fast mdn *was developed. The data source and pre-processing has been described before it is presented to a* MDN *for training. Using summary tools used within the meteorological community the results have been compiled into several tables in order to compare the performance of the particular network architectures. In the next section the results are analysed and discussed with respect to the aims laid out in section 1.5.*



(a) (b)

Figure 5: Conditional probability distribution plots, for a model with 2 kernels and 25 hidden units.



(a) (b)

Figure 6: Conditional probability distribution plots, for a hybrid model with 2 kernels and 25 hidden units.

FoM- results for networks trained per trace											
MDN architecture	trace 0	trace 1	trace 2	trace 3	trace 4	trace 5	trace 6	trace 7	trace 8	trace 9	average over swathe
Two hybrid Kernels, 10 Hidden Units	0.76	0.80	0.79	0.82	0.83	0.83	0.87	0.81	0.89	1.02	0.84
Two Kernels, 10 Hidden Units	0.82	0.75	0.80	0.78	0.84	0.79	0.79	0.81	0.85	0.93	0.82
Four Kernels, 10 Hidden Units	0.88	0.86	**0.92**	0.97	0.92	0.95	0.99	1.02	**1.02**	**1.19**	0.97
Two hybrid Kernels, 15 Hidden Units	0.90	0.83	0.83	0.90	0.90	0.88	0.88	0.86	0.95	1.03	0.90
Two Kernels, 15 Hidden Units	0.90	0.83	0.81	0.88	0.88	0.87	0.87	0.84	0.95	1.08	0.89
Four Kernels, 15 Hidden Units	**0.95**	0.91	0.87	**1.00**	0.91	0.87	1.00	0.94	**1.02**	1.13	0.96
Two hybrid Kernels, 20 Hidden Units	0.93	0.83	0.85	0.90	0.87	0.89	0.89	0.89	0.97	1.07	0.91
Two Kernels, 20 Hidden Units	0.92	0.84	0.82	0.85	0.88	0.88	0.89	0.90	0.97	1.07	0.90
Four Kernels, 20 Hidden Units	0.93	**0.92**	0.86	0.98	**0.93**	**0.98**	**1.02**	**1.00**	0.99	1.16	**0.98**
Two hybrid Kernels, 25 Hidden Units	0.89	0.82	0.86	0.91	0.91	0.91	0.91	0.91	0.98	1.06	0.92
Two Kernels, 25 Hidden Units	0.93	0.85	0.84	0.92	0.89	0.93	0.93	0.95	0.97	1.07	0.93
Four Kernels, 25 Hidden Units	0.88	0.88	0.87	0.99	0.86	0.97	1.00	0.95	0.97	1.17	0.95

Table 1: *FoM* results - for networks trained per trace. Results in **bold face** indicate best results per column.

Vector RMS error - results for networks trained per trace											
MDN architecture	trace 0	trace 1	trace 2	trace 3	trace 4	trace 5	trace 6	trace 7	trace 8	trace 9	average over swathe
Two hybrid Kernels, 10 Hidden Units	4.71	4.29	4.36	4.29	4.27	4.14	4.09	4.26	3.90	3.50	4.18
Two Centres, 10 Hidden Units	4.25	4.43	4.22	4.45	4.08	4.26	4.42	4.24	4.09	3.74	4.22
Four Centres, 10 Hidden Units	4.27	4.16	**3.77**	3.73	4.06	3.68	3.74	3.60	**3.52**	**3.11**	3.76
Two hybrid Kernels, 15 Hidden Units	3.90	4.17	4.19	3.93	3.88	3.95	4.01	4.06	3.70	3.44	3.92
Two Centres, 15 Hidden Units	3.90	4.09	4.10	4.04	3.91	3.96	3.95	4.05	3.66	3.28	3.90
Four Centres, 15 Hidden Units	3.86	4.00	4.16	**3.66**	4.00	4.31	3.54	4.01	**3.52**	3.25	3.83
Two hybrid Kernels, 20 Hidden Units	3.83	4.17	4.03	3.96	4.04	3.95	3.96	3.98	3.63	3.34	3.89
Two Centres, 20 Hidden Units	3.81	4.04	4.13	4.07	3.90	3.93	3.86	3.86	3.61	3.31	3.85
Four Centres, 20 Hidden Units	3.94	**3.84**	4.08	3.71	3.89	**3.63**	**3.48**	3.67	3.63	3.17	**3.70**
Two hybrid Kernels, 25 Hidden Units	3.97	4.24	3.99	3.94	**3.86**	3.87	3.92	3.87	3.61	3.36	3.86
Two Centres, 25 Hidden Units	**3.79**	4.00	3.99	3.82	3.89	3.76	3.77	**3.65**	3.61	3.28	3.76
Four Centres, 25 Hidden Units	4.20	3.97	4.16	3.67	4.24	3.70	3.63	4.14	3.94	3.15	3.88

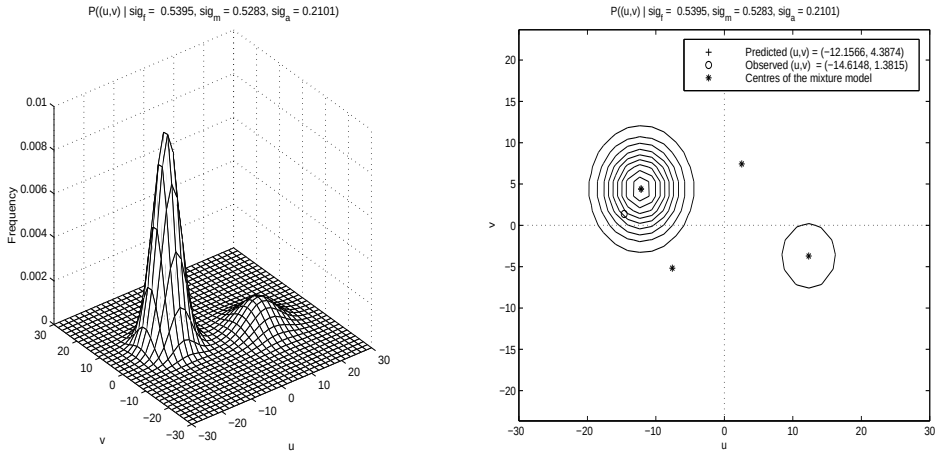
Table 2: Vector RMS error results - for networks trained per trace. Results in **bold face** indicate best results per column.

Performance @ 20° - results for networks trained per trace											
MDN architecture	trace 0	trace 1	trace 2	trace 3	trace 4	trace 5	trace 6	trace 7	trace 8	trace 9	average over swathe
Two hybrid Kernels, 10 Hidden Units	61.60	66.30	67.40	69.90	70.10	72.90	69.60	69.60	73.60	80.30	70.13
Two Centres, 10 Hidden Units	64.40	67.10	67.10	69.50	73.20	73.70	69.50	70.30	74.10	78.10	70.70
Four Centres, 10 Hidden Units	71.00	70.30	**72.60**	74.40	74.70	75.70	75.40	**75.00**	**77.60**	**81.90**	74.86
Two hybrid Kernels, 15 Hidden Units	66.10	68.80	67.50	72.50	72.90	74.20	71.00	72.00	74.90	79.00	71.89
Two Centres, 15 Hidden Units	69.30	68.50	70.70	72.80	74.40	75.10	73.10	72.90	75.80	79.40	73.20
Four Centres, 15 Hidden Units	**72.50**	71.00	72.30	**75.00**	**76.20**	75.10	**76.00**	74.80	77.40	81.10	75.14
Two hybrid Kernels, 20 Hidden Units	68.30	67.20	69.20	72.60	72.50	74.00	72.20	72.10	75.70	81.20	72.50
Two Centres, 20 Hidden Units	69.10	69.10	69.50	72.60	73.80	75.80	74.20	74.80	**77.60**	81.10	73.76
Four Centres, 20 Hidden Units	72.30	**71.40**	71.80	74.20	76.10	75.70	75.30	74.70	76.50	81.00	**74.90**
Two hybrid Kernels, 25 Hidden Units	67.20	67.70	69.80	72.30	73.00	75.00	72.00	71.40	75.80	80.60	72.48
Two Centres, 25 Hidden Units	69.20	69.50	70.70	73.60	73.70	75.10	73.60	74.10	75.90	79.50	73.49
Four Centres, 25 Hidden Units	71.60	69.10	72.50	74.40	75.40	**76.50**	75.70	74.80	76.90	81.70	74.86

Table 3: *Performance* @ 20° - for networks trained per trace. The results are computed using the wind direction obtained by the 'perfect' ambiguity removal algorithm described in the text. Results in **bold face** indicate best results per column.

Performance summary for networks trained with incidence angle as an input			
MDN architecture	*FoM*	RMS Errors	Perf @ 20°
Two Hybrid Kernels, 35 Hidden Units	0.85	4.18	73.54
Two Kernels, 35 Hidden Units	0.88	4.03	73.28
Four Kernels, 35 Hidden Units	0.96	3.83	76.96
Five Kernels, 35 Hidden Units	0.96	**3.84**	76.66
Twelve Kernels, 35 Hidden Units	0.80	5.13	74.82
Two Hybrid Kernels, 50 Hidden Units	0.82	4.33	71.10
Two Kernels, 50 Hidden Units	0.89	4.02	74.58
Four Kernels, 50 Hidden Units	0.96	3.86	77.08
Five Kernels, 50 Hidden Units	**0.97**	**3.84**	**77.14**
Twelve Kernels, 50 Hidden Units	0.82	4.87	75.32

Table 4: Performance results over the whole swathe - for networks trained with incidence angle as an input. These results are generated with the test data set of 5000 examples. Results in **bold face** indicate best results per column.



(a) (b)

Figure 7: Conditional probability distribution plots, for a model with 4 kernels and 25 hidden units.

FoM- results by trace, for networks that take incidence angle as an input											
MDN architecture	trace 0	trace 1	trace 2	trace 3	trace 4	trace 5	trace 6	trace 7	trace 8	trace 9	average over swathe
Two hybrid Kernels, 35 Hidden Units	0.80	0.79	0.82	0.86	0.86	0.88	0.88	0.90	0.92	0.97	0.87
Two Kernels, 35 Hidden Units	0.81	0.81	0.82	0.88	0.88	0.91	0.93	0.92	0.93	0.99	0.89
Four Kernels, 35 Hidden Units	**0.88**	0.84	0.84	0.94	0.92	0.89	0.92	0.95	0.93	**1.12**	0.92
Five Kernels, 35 Hidden Units	0.79	**0.90**	**0.89**	0.97	0.92	**0.98**	0.96	0.98	**1.00**	1.04	0.94
Twelve Kernels, 35 Hidden Units	0.60	0.71	0.79	0.92	0.85	0.81	0.80	0.85	0.90	0.99	0.82
Two hybrid Kernels, 50 Hidden Units	0.79	0.77	0.79	0.84	0.85	0.87	0.88	0.87	0.90	0.97	0.85
Two Kernels, 50 Hidden Units	0.80	0.81	0.83	0.89	0.88	0.92	0.91	0.94	0.95	1.02	0.90
Four Kernels, 50 Hidden Units	0.85	0.85	0.87	**0.99**	0.91	0.96	0.98	0.97	0.99	1.04	0.94
Five Kernels, 50 Hidden Units	0.84	0.89	0.84	0.98	**0.95**	0.97	**0.99**	**1.00**	0.99	1.05	**0.95**
Twelve Kernels, 50 Hidden Units	0.63	0.75	0.84	0.87	0.83	0.80	0.82	0.85	0.82	0.89	0.81

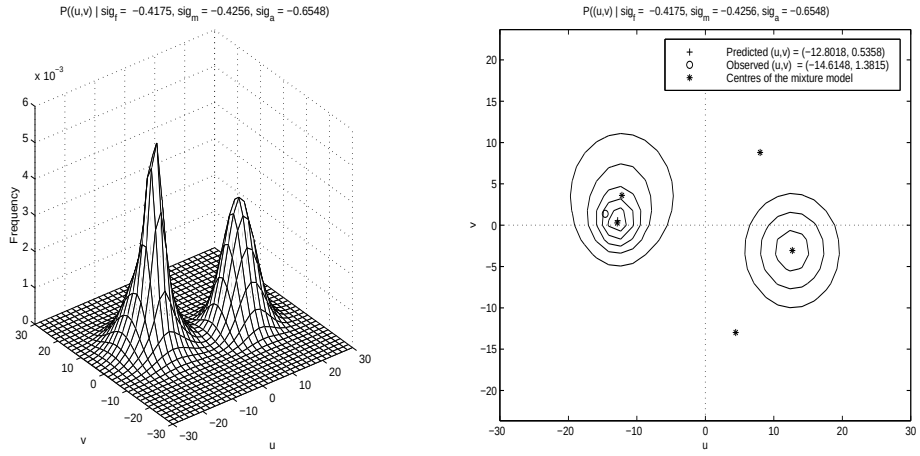
Table 5: *FoM* results - by trace, for networks trained with incidence as angle as an input. Results in **bold face** indicate best results per column.

Vector RMS error - results by trace, for networks that take incidence angle as an input											
MDN architecture	trace 0	trace 1	trace 2	trace 3	trace 4	trace 5	trace 6	trace 7	trace 8	trace 9	average over swathe
Two hybrid Kernels, 35 Hidden Units	4.41	4.45	4.23	4.12	4.05	3.94	3.98	3.94	3.81	3.65	4.06
Two Kernels, 35 Hidden Units	4.28	4.28	4.13	3.97	3.93	3.81	3.74	3.84	3.69	3.46	3.91
Four Kernels, 35 Hidden Units	**4.22**	4.33	4.08	4.11	4.01	4.40	4.10	4.08	3.94	**3.30**	4.06
Five Kernels, 35 Hidden Units	4.91	**4.09**	3.85	3.88	4.12	**3.62**	3.75	3.71	**3.56**	3.45	3.89
Twelve Kernels, 35 Hidden Units	7.89	5.38	4.75	4.08	4.36	4.93	5.24	4.77	4.43	4.05	4.99
Two hybrid Kernels, 50 Hidden Units	4.54	4.60	4.38	4.17	4.12	4.00	3.99	4.03	3.86	3.72	4.14
Two Kernels, 50 Hidden Units	4.33	4.27	4.13	3.99	3.92	3.77	3.79	3.75	3.69	3.49	3.91
Four Kernels, 50 Hidden Units	4.28	4.16	4.03	3.74	4.21	3.90	3.75	3.94	3.69	3.45	3.91
Five Kernels, 50 Hidden Units	4.33	4.16	**3.99**	**3.74**	**3.85**	3.70	**3.65**	**3.68**	3.69	3.54	**3.83**
Twelve Kernels, 50 Hidden Units	7.08	5.16	4.33	4.78	4.90	5.18	5.12	4.67	4.84	4.39	5.04

Table 6: Vector RMS error - by trace, for networks trained with incidence as angle as an input. Results in **bold face** indicate best results per column.

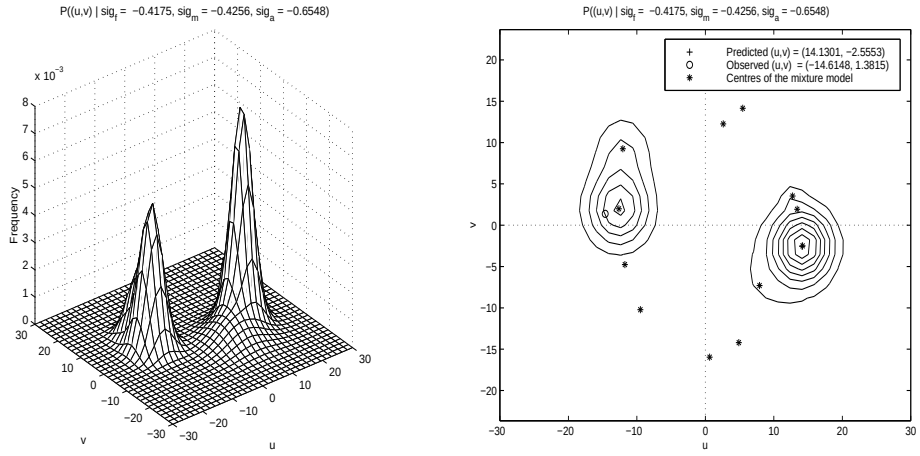
Performance at 20° - results by trace, for networks that take incidence angle as an input											
MDN architecture	trace 0	trace 1	trace 2	trace 3	trace 4	trace 5	trace 6	trace 7	trace 8	trace 9	average over swathe
Two hybrid Kernels, 35 Hidden Units	61.60	63.60	69.50	71.70	71.80	73.90	71.00	72.00	74.50	79.10	70.87
Two Kernels, 35 Hidden Units	63.50	64.20	68.20	71.70	72.40	75.30	71.80	73.10	74.30	79.00	71.35
Four Kernels, 35 Hidden Units	**66.90**	66.30	71.30	73.50	77.00	74.90	74.30	75.60	75.70	**80.50**	73.60
Five Kernels, 35 Hidden Units	63.70	69.00	**71.40**	74.20	76.20	76.40	75.00	**75.80**	77.10	79.30	73.81
Twelve Kernels, 35 Hidden Units	63.10	**69.10**	71.20	73.40	75.10	74.40	74.20	74.80	77.50	79.90	73.27
Two hybrid Kernels, 50 Hidden Units	58.50	60.50	66.80	70.70	70.80	73.00	71.20	72.20	74.60	79.00	69.73
Two Kernels, 50 Hidden Units	62.80	66.10	69.20	72.30	73.30	76.20	73.60	74.40	76.30	79.60	72.38
Four Kernels, 50 Hidden Units	65.80	68.10	71.00	73.30	76.80	**76.80**	**75.90**	74.70	**77.80**	79.60	**73.98**
Five Kernels, 50 Hidden Units	65.00	68.10	71.20	74.00	76.90	76.20	75.00	75.40	76.60	79.20	73.76
Twelve Kernels, 50 Hidden Units	64.60	68.40	**71.40**	**74.10**	**77.10**	75.80	73.30	72.50	75.00	76.20	72.84

Table 7: *Performance @* 20° - by trace, for networks trained with incidence as angle as an input The results are computed using the wind direction obtained by the 'perfect' ambiguity removal algorithm described in the text. Results in **bold face** indicate best results per column.



(a) (b)

Figure 8: Conditional probability distribution plots with incidence angle, for a model with 5 kernels and 50 hidden units.



(a) (b)

Figure 9: Conditional probability distribution plots with incidence angle, for a model with 12 kernels and 50 hidden units.

3 Analysis and discussion

The aims of this project are to investigate the underlying data generator that describes the mapping from satellite scatterometer data to wind vectors, $\boldsymbol{\sigma^o} \rightarrow (u, v)$, using a mixture density network. In Section 2 the methods and results of training the MDN*s, with and without incidence angle as inputs, were presented. Also the structure of the* MDN *was modified to investigate modelling the inherent ambiguity in the wind direction. In this section the results of the experiments in Sections 2 are analysed and discussed.*

3.1 Analysis of the inverse models

Both the complexity of the mapping $\boldsymbol{\sigma^o} \rightarrow (u, v)$ and of the probability density function $P(u, v|\boldsymbol{\sigma^o})$ are of interest. These properties are represented by the MLP and GMM structures within the MDN framework. The mapping $\boldsymbol{\sigma^o} \rightarrow (u, v)$ is modelled by the MLP, whose complexity is controlled by the number of units in the hidden layer. The complexity of $P(u, v \mid \boldsymbol{\sigma^o})$ is modelled by the GMM, whose complexity is controlled by number of kernels in mixture. There are two kinds of model to compare: those which have been trained without incidence angle as an input (trained for a specific trace) and those that take incidence angle as an input and are general models over the whole swathe. The results are presented by plotting the FoM and vector RMS results over all the traces. This format helps to highlight trends in the results.

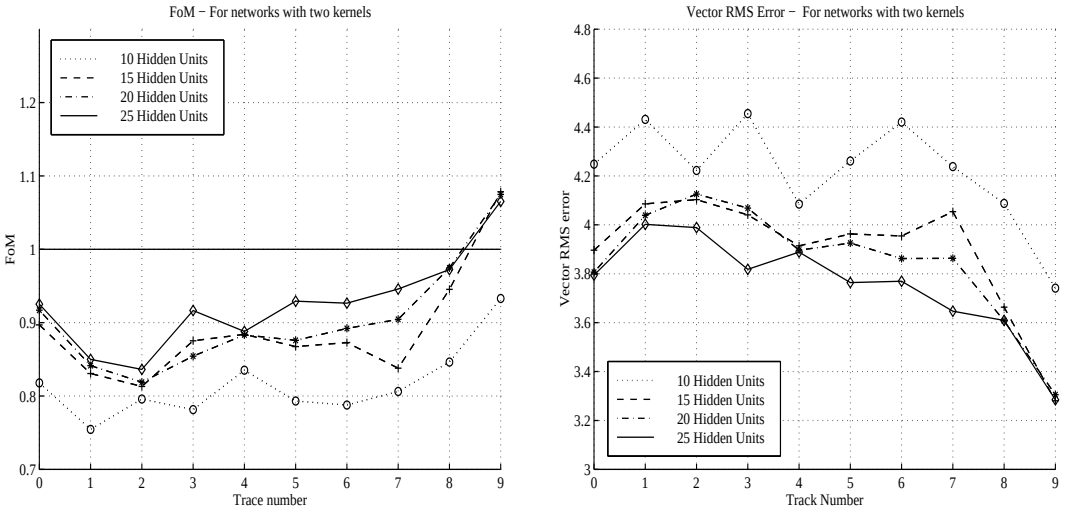
3.1.1 The complexity of the mapping $\boldsymbol{\sigma^o} \rightarrow (u, v)$

In order to investigate the complexity of the mapping $\boldsymbol{\sigma^o} \rightarrow (u, v)$ the MDN architectures were trained with the MLP structure having ten, fifteen, twenty and twenty five units in the hidden layer. Also networks were trained with different kernel configurations (to investigate the complexity of $P(u, v \mid \boldsymbol{\sigma^o})$), which may also affect model performance, and so comparisons in this subsection are made by kernel configuration, over the number of units in the hidden layer of the MLP.

For two kernels, Figure 10 shows that increasing the number of hidden units improves the model performance. Figure 11, shows similar results, but for the hybrid MDN configuration: Again, increasing the number of units in the MLP improves the performance of the model, but in this case the improvement is not as distinct as for those configurations with two free kernels. The results of MDNs with four kernels are plotted in Figure 12, and show that the model performance does not significantly improve by increasing the number of units in the hidden layer of the MLP over the range trained.

For models with two kernels there is a correlation between increasing the number of units in the MLP and improving model performance. The best models have twenty five units in the hidden layer of the MLP. For models with four kernels the results show that the best performance is achieved by a MLP with twenty hidden units (see Figure 12). The model performance for a MLP with twenty five hidden units is worse than that of twenty hidden units. There are two explanations for this. Firstly the complexity of the MLP with twenty five hidden units is sufficient to overfit the training data, decreasing the models ability to generalise, or, secondly because of the complexity of the MLP, the MDN becomes stuck in a local minima in the error surface, and the network fails to find the weights that give optimum generalisation.

For MDNs with a hybrid kernel configuration, the results suggest that the MLP is reaching a limit in its ability to model the mapping $\boldsymbol{\sigma^o} \rightarrow (u, v)$, given the GMM configuration. The improvement in performance does not change significantly for hidden units of twenty and twenty five. Comparing



(a) *FoM* (b) Vector RMS error

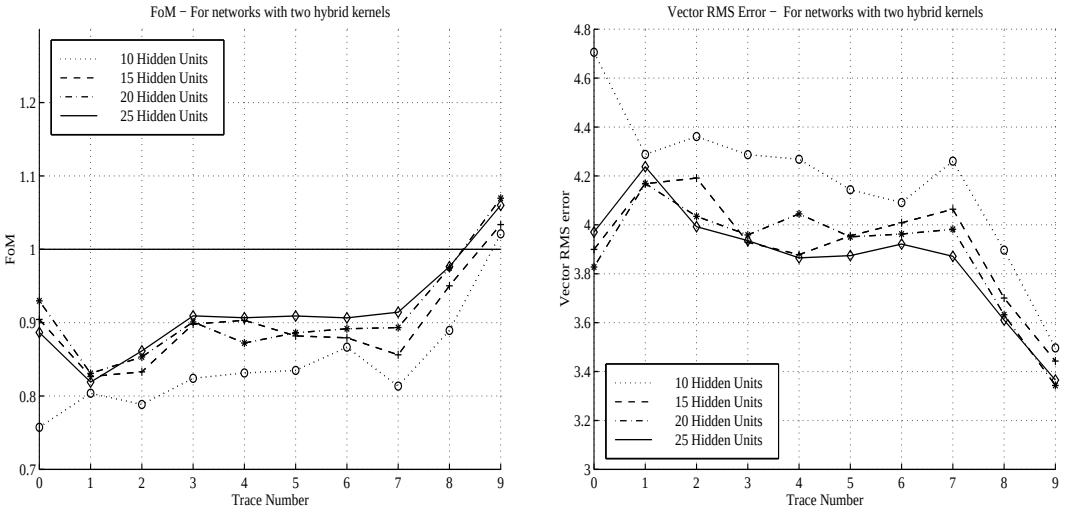
Figure 10: Model performance by the number of hidden units in the MLP, for models with two kernels and trained per trace.

these results with MDN configurations having two free kernels confirms that the assumption that the inherent ambiguity in the problem is generally 180° holds. The slightly lower performance for the hybrid MDN is attributed to fixing the ambiguity to 180° and having equal variances on each kernel, which implies that the model is less flexible than those with two free kernels. Generalising to data that violates these assumptions will be more difficult for the hybrid configuration (that is, if the ambiguity moves away from 180°, and/or the variance of each mode is significantly different).

The MDNs that take incidence angle as an input were trained with two MLP configurations: thirty five, and fifty hidden units. Comparing the results of Table 5 and the graphs in Figures 13 and 14, we see that the performance of the model is affected more by the number of kernels than the number of hidden units in the MLP. A MLP with thirty five hidden units models the mapping as well as an MLP with fifty hidden units. The MLP is also modelling the mapping of incidence angle to the (u, v) space. An extra ten units in the hidden layer (compared with models trained per trace) adequately models this mapping.

In summary, for MDNs trained per trace, with four kernels, the mapping of $\boldsymbol{\sigma^o} \rightarrow (u, v)$ is adequately approximated by an MLP having ten hidden units, although the optimal solution requires twenty hidden units. For MDNs trained per trace with two kernels (including hybrids), twenty five hidden units in the MLP are required. The assumption that the ambiguity in the problem is principally at 180° has been shown to hold by comparing the model performance of the MDNs with hybrid and two free kernel configurations. Models with two kernels and twenty five hidden units in the MLP do not perform as well as models with four kernels and ten hidden units in the MLP. This is attributed to the structure of the probability density $P(u, v \mid \boldsymbol{\sigma^o})$, and is discussed subsection 3.1.2.

For networks trained with incidence angle as an input a MLP with thirty five hidden units satisfactorily models the mapping $(\boldsymbol{\sigma^o}, \theta) \rightarrow (u, v)$. Again, the number of kernels in the MDN contributes more significantly to changes in model performance than changing the number of hidden units in the MLP.



(a) FoM (b) Vector RMS error

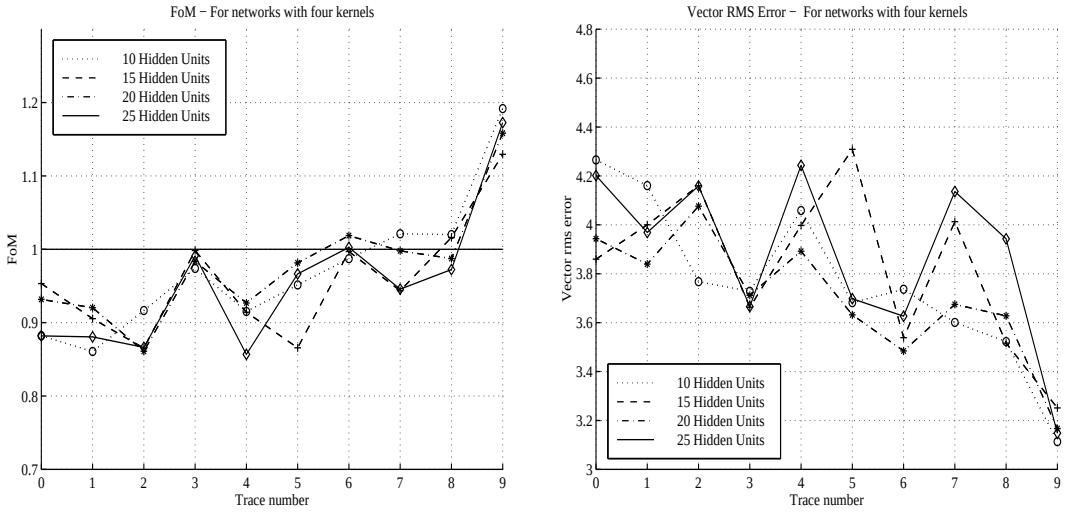
Figure 11: Model performance by the number of hidden units in the MLP, for models with hybrid kernels and trained per trace.

3.1.2 The complexity of the conditional probability distribution $P(u, v \mid \boldsymbol{\sigma^o})$

The complexity of the probability distribution $P(u, v \mid \boldsymbol{\sigma^o})$ is reflected by the number of kernels in the MDN. If $P(u, v \mid \boldsymbol{\sigma^o})$ is a highly complex distribution then a larger number of kernels are required to model the distribution than if $P(u, v \mid \boldsymbol{\sigma^o})$ is relatively simple.

Considering MDNs trained by trace, the graphs in Figure 15 show a trend of increasing performance for an increasing number of kernels in the MDN. The plots for two kernels (red and blue) have a large variance which indicates a significant dependence on the number of hidden units in the MLP. Furthermore the variance across a single plot at constant number of hidden units is large which suggests there is a large amount of noise in the performance estimate. The plots for four kernels have less variance indicating that there is less dependence on the number of hidden units in the MLP; however the noise across a single plot at constant hidden units is consistent with the red and blue plots. The inference from these plots is that a MDN with two kernels is insufficiently complex to model the conditional probability distribution, whereas a MDN with four kernels has increased flexibility and sufficiently models the conditional probability distribution. The plots in Figures 5, 6 and 7 show that the probability distribution is dominantly bimodal. The fact that models with four kernels perform better than models with two kernels is of interest. Two possible explanations are offered; the first is that four centres provide more flexibility when modelling a probability distribution than two; this is shown in Figure 7 where the positions of the kernels are placed in roughly four quadrants of (u, v) space. Models with two kernels do not have that kind of flexibility. The second is that the noise on the targets is not Gaussian or spherically symmetrical. Figure 16 shows such a case where each mode is modelled by two superimposed kernels; again, the MDNs with two kernels do not have the flexibility to model non-Gaussian modes, only being able to approximate each mode by a symmetric Gaussian probability distribution function.

Switching attention to models that take incidence angle as an input the results show an interesting relationship with the number of kernels in the MDN. The MDNs with twelve kernels perform worse than MDNs with five or less kernels (see Figures 17, 14 and 13).



(a) FoM (b) Vector RMS error

Figure 12: Model performance by the number of hidden units in the MLP, for models with four kernels and trained per trace.

Why is this so? The aim of any model is to describe the underlying data generator and not the noise on the targets. The MDNs with twelve kernels are complex enough to model some of the noise on the targets, and overfit the data. The model then is not a general description of the underlying data generator describing the mapping $\boldsymbol{\sigma^o} \rightarrow (u, v)$ but a specific description of the target data, including the noise. A good example is shown by the plots in Figure 18 where the probability distribution has several strong peaks, reflecting the distribution of the target data. Comparing the remaining configurations, performance is best for the MDN with five kernels, for both MLP configurations of thirty five and fifty units in the hidden layer. Comparing the hybrid kernel and two free kernel models, it is clear that the free kernel model performs better than the hybrid model, again, this can be explained by the reduced flexibility of the hybrid model, but suggests that the probability distribution is generally bimodal but the modes are not exactly diametrically opposite one another or are non-Gaussian.

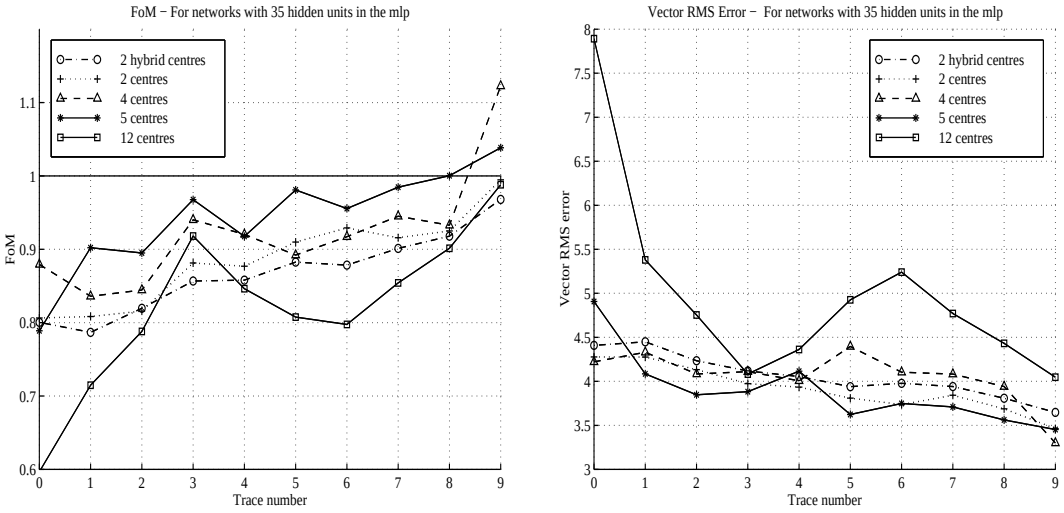
The probability density $P(u, v \mid \boldsymbol{\sigma^o})$ is generally bimodal. The modes, however, are more complex than a spherical Gaussian model. For the networks trained for each trace the MDN with four kernels and twenty hidden units in the MLP is the best model, and is denoted MDN_{trace}. For networks trained with incidence angle as an input, the best MDN has five kernels and fifty hidden units in the MLP, and is denoted $MDN_{incidence}$.

3.2 Comparison with other neural network methods

Cornford *et al.* (1997) give figures for their neural network approach for predicting wind vectors, based on ERS-1 satellite data. They solve the inverse problem by training distinct networks with incidence angle as an input to model wind speed and wind direction. Considering their models on an individual basis they report that the correct solution within 20° is obtained more than 70% of the time for the first two aliases, which is similar to the results obtained with the models in this project. They improved the performance by creating a committee of networks and then obtained the correct solution to within 20° roughly 75% of the time (averaged over the swathe). For wind

Predicted wind direction chosen from the first two most probable aliases, performance at 20° (%)											
MDN architecture	trace 0	trace 1	trace 2	trace 3	trace 4	trace 5	trace 6	trace 7	trace 8	trace 9	average
MDN_{trace}	72.3	71.4	71.8	74.2	76.1	75.7	75.3	74.7	76.5	81.0	74.9
$MDN_{incidence}$	63.7	69.0	71.4	74.2	76.2	76.4	75.0	75.8	77.1	79.3	73.8
$A\text{-}NN_i$	85.1	85.0	86.9	87.7	87.5	87.8	87.6	88.0	88.2	86.9	87.1

Table 8: Comparing the direction performance of the best MDNs with published results.



(a) *FoM* (b) Vector RMS error

Figure 13: Model performance by the number of kernels. For a network with thirty five units in the hidden layer, and trained with incidence angle as an input

speed they obtained a RMS error of 1.8 ms^{-1} which is within the design specifications of the instrument (2.0 ms^{-1}). The networks presented here have an wind speed RMS error of 2.0 ms^{-1} when averaged over the swathe. Richaume *et al.* (1998) give results for their neural network approach to wind vector retrieval based on ERS-1 data (following an initial study by Thiria *et al.* (1993) which was based on simulated scatterometer data). For each trace two networks are trained, one to model wind speed, denoted $S\text{-}NN_i$ (where i corresponds to the trace), and one to model wind direction, denoted $A\text{-}NN_i$. Results are quoted for predicted wind speed bias and RMS error, and wind direction performance @ 20°. The wind direction results are quoted for the first, second, third and forth alias predictions. The second alias method is equivalent to the 'perfect' ambiguity removal method used in this project, and provides a means of comparison. Table 8 shows the results for *perf.* @ 20° of MDN_{trace}, $MDN_{incidence}$, $A\text{-}NN_i$. The $A\text{-}NN_i$ neural network performs better than both MDN networks when predicting direction to the second alias. However, we must not that the imputs to the $A\text{-}NN_i$ network also contain spatial information which gives additional ambiguation skill to the $A\text{-}NN_i$ networks. The MDN networks a purely local models, having no disambiguation skill whatsoever, inverting each scatterometer measurement on a per-cell basis.

Table 9 presents the wind speed bias and RMS error results. Comparing the biases it is interesting to note that $A\text{-}NN_i$ has only negatively biased results, where as the MDN models have both positively and negatively biased networks, and so are less biased over the whole swathe. The RMS error results show that $A\text{-}NN_i$, performs within the instrument specification of 2 ms^{-1}, whereas $MDN_{incidence}$ and MDN_{trace} both fall outside the measurement specification for several of the middle traces.

The superior performance of $S\text{-}NN_i$ and $A\text{-}NN_i$ may be attributed to:

- Larger data sets (the training set contains 24,000 pairs and the test set 5,000 pairs). Large data sets help to regularise the network during training, making it less susceptible to outliers in the data set.
- The spatial information presented on the inputs may provide extra disambiguation skill. The

Predicted wind speed bias (ms^{-1})										
MDN architecture	trace 0	trace 1	trace 2	trace 3	trace 4	trace 5	trace 6	trace 7	trace 8	trace 9
MDN_{trace}	0.0	-0.3	-0.1	0.0	-0.1	0.0	0.1	0.0	-0.1	0.1
$MDN_{incidence}$	0.0	-0.3	-0.1	0.0	-0.1	0.0	0.1	0.0	-0.1	0.1
$A\text{-}NN_i$	-0.2	0.0	-0.1	-0.1	-0.1	-0.1	-0.1	-0.1	0.0	0.0
Predicted wind speed RMS (ms^{-1})										
MDN architecture	trace 0	trace 1	trace 2	trace 3	trace 4	trace 5	trace 6	trace 7	trace 8	trace 9
MDN_{trace}	1.9	2.1	2.1	2.0	2.1	2.1	2.1	2.0	2.0	1.8
$MDN_{incidence}$	1.9	2.1	2.1	2.0	2.1	2.1	2.1	2.0	2.0	1.8
$A\text{-}NN_i$	1.6	1.6	1.5	1.6	1.6	1.6	1.6	1.6	1.6	1.7

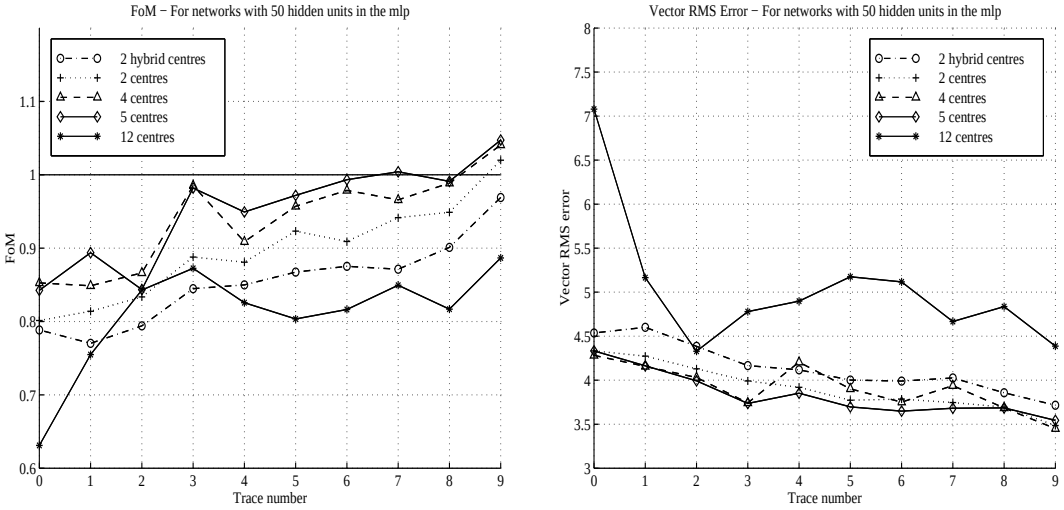
Table 9: Comparing the speed performance of the best MDNs with published results

MDNs are trained with input from a single measurement.

- The range of wind speeds used by Richaume *et al.* (1998) is slightly greater, with a wind speed range of $[3.5\ ms^{-1}, 25.0\ ms^{-1}]$. There are likely to be more training patterns in the wind speed ranges that are more difficult to learn.
- The MDN networks may be getting stuck in local minima on the error surface during training.

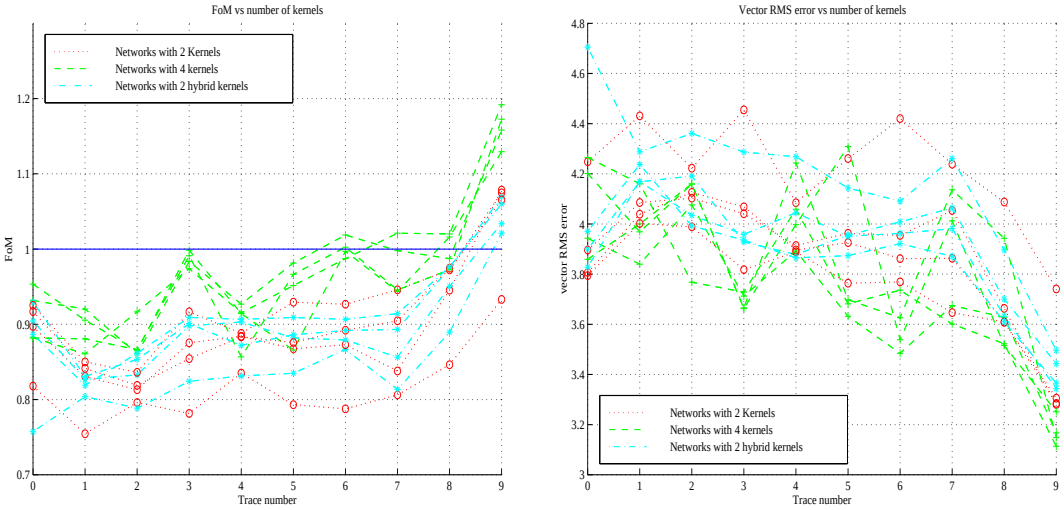
To visualise the alias results and the effect of choosing the two most likely solutions, four graphs are provided. The results are obtained from $MDN_{incidence}$, using the test set with five thousand examples. The Figure 19(a) shows the most probable predicted direction, where the true and alias solutions can be seen. Figure 19(b) shows the de-aliased prediction. The Figure 20(a) shows the most probable predicted wind speed, Figure 20(b) shows the de-aliased prediction, in each case the model is biased high for wind speeds approximately less than 7 ms^{-1}, and biased low at wind speeds approximately greater than 20 ms^{-1}.

In this section the results presented in Section 2 have been discussed. It has been shown that MDN*s model the mapping from scatterometer space directly into wind vector component space with a high degree of success. The probability distribution* $P(u, v|\boldsymbol{\sigma^o})$ *is generally bimodal, but the noise on the targets is more complex than the spherically symmetric Gaussian assumption that was first made. It has also been shown that the mapping* $(\boldsymbol{\sigma^o}, \theta) \rightarrow (u, v)$ *performs similarly to models trained by trace. The best* MDN*s do not perform as well some other neural network methods, but they do have the advantage of directly mapping to* (u, v) *space. In the next section the conclusions of this project are presented along with possible future avenues of investigation.*



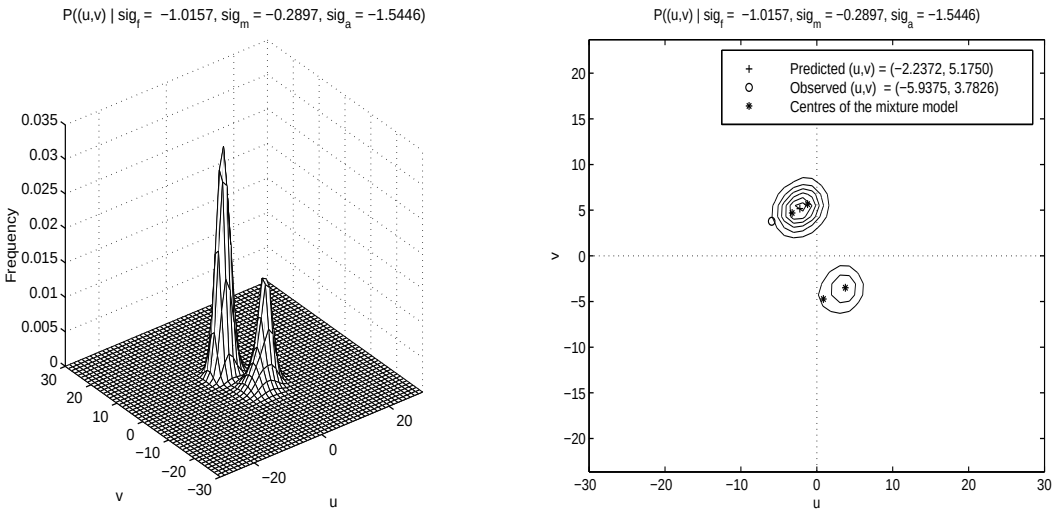
(a) FoM (b) Vector RMS error

Figure 14: Model performance by the number of kernels. For a network with fifty units in the hidden layer, and trained with incidence angle as an input



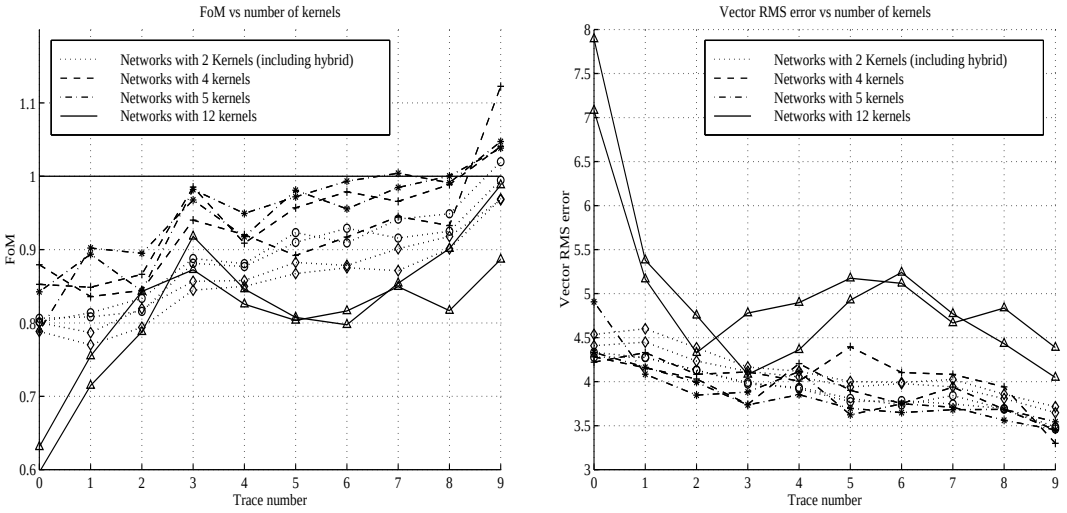
(a) (b)

Figure 15: Model performance, comparing networks which are trained by trace, by the number of kernels



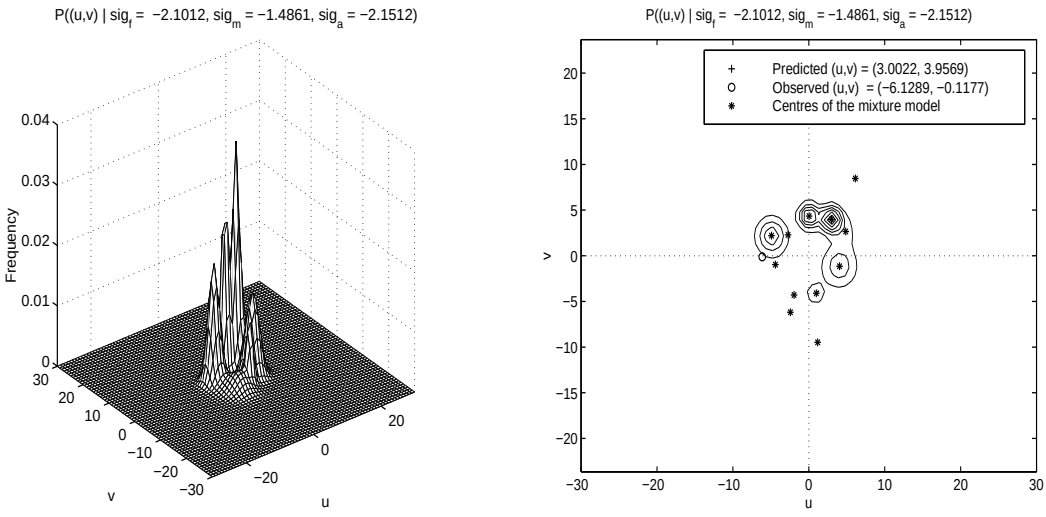
(a) (b)

Figure 16: An example of $P(u, v \mid \boldsymbol{\sigma}^o)$ where the modes are non-Gaussian



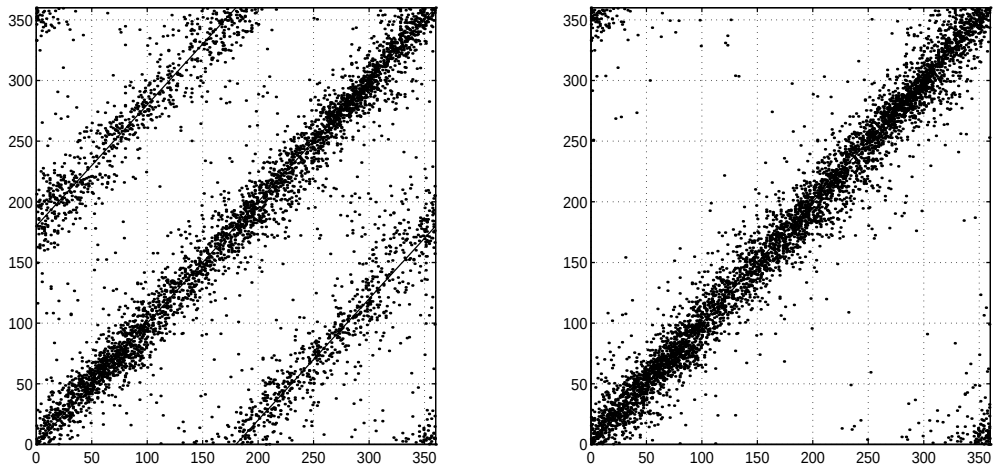
(a) (b)

Figure 17: Model performance, comparing networks which take incidence angle as an input, by the number of kernels,



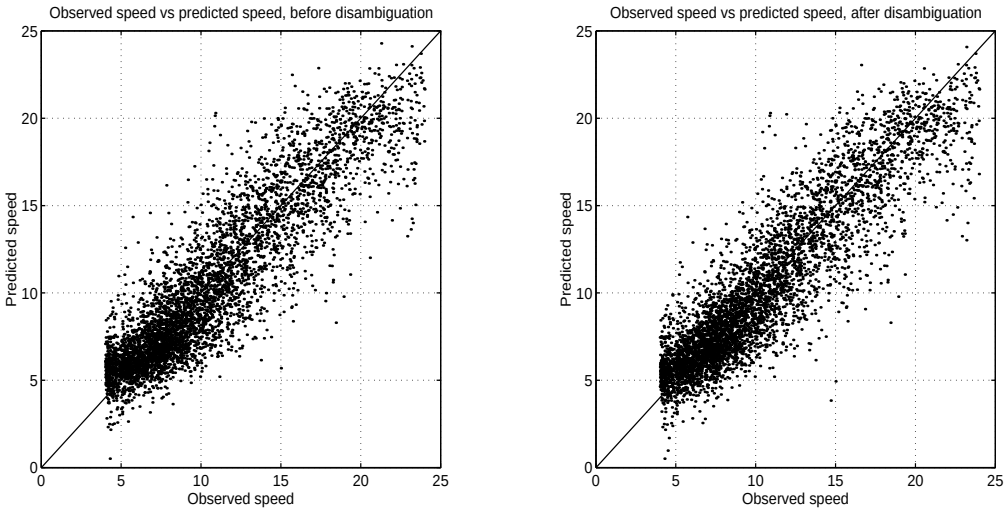
(a) (b)

Figure 18: An example of overfitting in $P(u, v \mid \boldsymbol{\sigma}^o)$ space.



(a) Direction predictions before de-aliasing (b) Direction predictions after de-aliasing

Figure 19: Scatter plots of observed vs predicted wind directions



(a) Speed predictions before de-aliasing

(b) Speed predictions after de-aliasing

Figure 20: Scatter plots of observed vs predicted wind speed

4 Conclusions and future work

The preceding two sections have demonstrated that MDN*s can be applied to directly model wind vector components from scatterometer data. In this concluding section, the aims of subsection 1.5 are re-visited, the findings of this project reviewed, and on-going and potential future work is described.*

4.1 Conclusions

The overall aim of this project is to assess the feasibility of directly modelling the wind vectors (u, v) from scatterometer data $\boldsymbol{\sigma^o}$. The results presented in Section 2 clearly show that this method is feasible. Further questions were posed concerning the complexity of the mapping $\boldsymbol{\sigma^o} \rightarrow (u, v)$ and the complexity of $P(u, v \mid \boldsymbol{\sigma^o})$.

- Investigating the mapping $\boldsymbol{\sigma^o} \rightarrow (u, v)$, it has been shown that a MDN that has a MLP with twenty units in the hidden layer, and a GMM with four kernels successfully maps this relationship. A more complicated model describing the mapping $(\boldsymbol{\sigma^o}, \theta) \rightarrow (u, v)$ was considered. This mapping is modelled using a MDN with a MLP with thirty five units in the hidden layer, and five kernels in the GMM.
- Considering $P(u, v \mid \boldsymbol{\sigma^o})$, the hybrid MDN configuration yields similar results to a MDN with two free centres, and shows that the conditional probability distribution is generally bimodal with the two modes positioned diametrically opposite one another. The distribution of the noise on the wind vectors has been shown to be more complex than the spherically symmetrical Gaussian noise model originally assumed. This is shown by the MDNs which have the ability to model more complex (yet still dominantly bimodal) distributions than the one assumed by having four, five and twelve kernels. While the distribution is still generally bimodal, it is heavier tailed than the Gaussian distribution assumed, and is not always spherically symmetric.
- Other work in the field solves the inverse problem by directly modelling wind speed and wind direction with two separate models. The models of this project are similar in performance to other local methods (Cornford *et al.*, 1997). These methods however do not compare as favourably with those methods which take spatial information surrounding the cell of interest as part of their inputs (Richaume *et al.*, 1998).
- The large number of MDNs trained in this project was made possible by developing *fast mdn* that moved the computation of the error gradient into parameter space (the outputs of the MLP). This improved training time from a few days to a few hours for MDNs trained by trace, and allowed MDNs with large data sets to be trained, such as those that take incidence angle as an input, in a realistic time frame.

The final conclusions of this project are that mixture density networks provide a principled framework within which to model wind vectors directly from satellite scatterometer data, and the quality of the results provide an encouraging path of investigation for novel disambiguation techniques.

4.2 On-going work

The ultimate aim is to build models that provide autonomous ambiguity removal from satellite scatterometer data, that is without reference to winds derived from numerical weather prediction

models. On-going work at Aston is investigating autonomous ambiguity removal within a Bayesian framework. Within this framework, we can use either the local inverse models developed in this project, or local forward models (such as those developed by Guillaume Ramage during his MSc project). Continuing the work of this project, an implementation is currently being developed using the local inverse models.

The Bayesian method is fundamentally different to that of Richaume *et al.* (1998). They explicitly incorporate spatial information in the model construction, taking into account correlation between neighbouring cells. The Bayesian method has two stages. First the local models are trained. When training it is assumed that there is no spatial correlation between the local models. The second stage is to apply Bayes' theorem and combine the local models with global wind prior models to impose the spatial physical constraints of wind fields. Bayes' theorem can only be applied if it is assumed that for the forward model the probabilities, of the scatterometer measurements, $\boldsymbol{\sigma_i^o}$, are independent, conditional on the wind vectors, (u_i, v_i). This assumption further implies that the models of Richaume *et al.* (1998) cannot be used in the Bayesian context presented here.

The wind field, U,V, is represented by a density probability over a wind field (U,V) conditional on the scatterometer measurements $\boldsymbol{\Sigma^o}$:

$$P(U,V \mid \boldsymbol{\Sigma^o}). \tag{34}$$

Bayes' theorem is applied to (34) to express the posterior probability in terms of a global forward model:

$$P(U,V \mid \boldsymbol{\Sigma^o}) \propto P(\boldsymbol{\Sigma^o} \mid U,V)P(U,V). \tag{35}$$

The global forward model is expressed as a product of probabilities given by a local forward model, assuming that they are conditionally independent:

$$P(U,V \mid \boldsymbol{\Sigma^o}) \propto \prod_i P(\boldsymbol{\sigma_i^o} \mid u_i, v_i)P(U,V). \tag{36}$$

Bayes' theorem is applied again to express the local forward model in terms of the inverse model, this is called the scalled likelihood method (Williams, 1997):

$$P(U,V \mid \boldsymbol{\Sigma^o}) \propto \left(\prod_i \frac{P(u_i, v_i \mid \boldsymbol{\sigma_i^o})P(\boldsymbol{\sigma_i^o})}{P(u_i, v_i)} \right) P(U,V). \tag{37}$$

Finally, the the local scatterometer measurements $P(\boldsymbol{\sigma_i^o})$ are constant for a given scene and the posterior probability distribution is expressed as:

$$P(U,V \mid \boldsymbol{\Sigma^o}) \propto \left(\prod_i \frac{P(u_i, v_i \mid \boldsymbol{\sigma_i^o})}{P(u_i, v_i)} \right) P(U,V). \tag{38}$$

Equation (38) defines a probability density which has a dimension given by the number of wind vectors in the wind field. The posterior is described by a combination of three probability models: the local conditional inverse model $P(u_i, v_i \mid \boldsymbol{\sigma_i^o})$, the local unconditional model $P(u_i, v_i)$ and the global wind prior $P(U,V)$. These three models are implemented in MATLAB using the local inverse models developed in this project, local unconditional models and the global wind prior of Cornford (1998). The parameter space of the posterior distribution, the wind vectors (U,V), is explored using Markov Chain Monte Carlo techniques. These techniques use stochastic methods to sample from the posterior distributions such as (38). Once the stationary distribution is found inference is made on the model parameters (U,V).

4.3 Potential future work

The results of this project have raised as many questions as they have answered, and there is still much to learn about the structure of the inverse mapping and the behaviour of MDNs. To finalise this report the remainder of this subsection suggests possible avenues of work.

4.3.1 Further investigation on the $\boldsymbol{\sigma^o} \rightarrow (u, v)$ mapping

There are many possible changes to the MDN structure that might help improve the performance of the model, two of the most promising are:

- For MDNs with two free kernels and trained by trace. Training MDNs with an increasing number of units in the hidden layer of the MLP until optimal performance is achieved. Once the optimal configuration is established, a comparison can be made with the current results from MDNs with four kernels to establish how the assumption that the noise in the targets is Gaussian and spherically symmetric affects model performance.
- Further investigate the mapping $(\boldsymbol{\sigma^o}, \theta) \rightarrow (u, v)$ by reducing the number of hidden units in the MLP for MDNs taking incidence angle as an input, and find the point where model performance significantly reduces. The extra number of hidden units in the MLP will give an indication of the complexity of the relationship between incidence angle and the measurement manifold for each trace.

4.3.2 Further investigation of the structure of the probability distribution $P(u, v \mid \boldsymbol{\sigma^o})$

The results have shown that the noise on the targets in the current data set appears non-Gaussian, and dominantly bimodal. Improvements to the model structure by modifying the architecture of the MDN to model this distribution may be of benefit:

- Build a hybrid MDN with two free kernels and two mirroring centres. This will be a less complex model than a full model with four kernels and should be able to model the non-Gaussian modes more efficiently than the hybrid MDN with two kernels.
- The noise distribution on the targets appears to be heavier tailed than the Gaussian distribution originally assumed. This assumption could be modified by replacing the kernels with a heavier tailed distribution (such as a t-distribution), and retrain MDNs with two kernels.

4.3.3 Further work to improve generalisation

There is also potential work in improving the generalisation performance of the MDNs with respect to the quality of the training data and the training methods employed:

- Outliers in the training set will affect the ability of the MDN to generatlise. By carefully removing outliers from the training set (either manually or otherwise) and retraing the MDNs we expect to see an increase in generalisation performance.

- Cornford *et al.* (1997) showed that using committees of MDNs (with kernels of circular normal densities) to predicted wind direction improved their results by roughly 5%. Given that there are several MDNs trained in this project, this method is a simple way of improving the results without the retraining the MDNs.
- MDNs can get stuck in local minima on the error surface. By changing the starting position on the error surface (by choosing a different seed in the random number generators) the MDNs may find a better location on the error surface with respect to generalisation.
- The MDNs trained in this project are unregularised. Regularisation controls the complexity of neural networks during training, and making the generalisation performance less sensitive to the initial model complexity. Lars Hjorth, a fellow MSc student, is developing a regularised MDN framework. When he has completed his work, his framework could be applied to this problem.
- Little is known about the learning dynamics of the MDN. Investigation into the evolution of the parameter vector may well provide an insight into the way MDNs learn, and lead to improvements that increase the generalisation properties of these methods.

In this final section the conclusions of this project have been drawn: It has been shown that MDN*s offer a feasible framework in which to directly extract wind vector components from satellite scatterometer data. There is on-going work, which has been described, putting the local inverse models of this project into the larger context of autonomous disambiguation methods. Finally, there are still many more questioned to be answered, a few have been proposed here, with a hope to inspire other researchers, and anyone who reads this thesis, to continue on this path of discovery.*

Acknowledgements

Thanks to Dr. Dan Cornford for his guidance during the first year of this project and Dr. Ian Nabney for patiently reading and commenting on the draft versons of this report.

References

Bishop, C. M. 1994. Mixture Density Networks. Technical Report NCRG/94/004, Department of Computer Science and Applied Mathematics, Aston University, Birmingham, B4 7ET, UK.

Bishop, C. M. 1995. *Neural Networks for Pattern Recognition*. Oxford University Press.

Bishop, C. M. and I. T. Nabney 1996. Modelling Conditional Probability Distributions for Periodic Variables. *Neural Computation* **8**, 1123–1133.

Cornford, D. 1998. Non-Zero Mean Gaussian Prior Wind Field Models. Technical Report Unpublished, Department of Computer Science and Applied Mathematics, Aston University, Birmingham, B4 7ET, UK.

Cornford, D. and I. T. Nabney 1998. NEUROSAT: An Overview. Technical Report NCRG/98/011, Department of Computer Science and Applied Mathematics, Aston University, Birmingham, B4 7ET, UK.

Cornford, D., I. T. Nabney, and C. M. Bishop 1997. Neural Network based Wind Vector Retrieval from Satellite Scatterometer Data. (Submitted (Neural Computing and Applications)).

Evans, D. J. 1998. Mixture Density Network Training by Computation in Parameter Space. Technical Report NCRG/98/016, Department of Computer Science and Applied Mathematics, Aston University, Birmingham, B4 7ET, UK.

Long, D. and J. M. Mendel 1991. Identifiability in Wind Estimation From Scatterometer Measurements. *IEEE Transactions on Geoscience and Remote Sensing* **29**, 268–276.

McLachlan, G. J. and K. E. Bashford 1988. *Mixture Models: Inference and Applications to Clustering*. New York: Marcel Dekker.

Offiler, D. 1994. The Calibration of ERS-1 Satellite Scatterometer Winds. *Journal of Atmospheric and Oceanic Technology* **11**, 1002–1017.

Ramage, G. 1998, September. Neural Networks for Modelling Wind Vectors.

Richaume, P., F. Badran, M. Crepon, C. Mejia, H. Roquet, and S. Thiria 1998. Neural Network Wind Retrieval from ERS-1 Scatterometer Data. *Journal of Geophysical Research*.

Robinson, I. S. 1985. *Satellite Oceanography, An Introduction for Oceanographers and Remotesensing Scientists*. Ellis Horwood Limited.

Stoffelen, A. and D. Anderson 1997a. Ambiguity Removal and Assimilation of Scatterometer Data. *Quarterly Journal of the Royal Meteorological Society* **123**, 491–518.

Stoffelen, A. and D. Anderson 1997b. Scatterometer Data Interpretation: Estimation and Validation. *Journal of Geophysical Research* **102**, 5767–5780.

Stoffelen, A. and D. Anderson 1997c. Scatterometer Data Interpretation: Measurement Space and Inversion. *Journal of Atmospheric and Oceanic Technology* **14**, 1298–1313.

Thiria, S., C. Mejia, and F. Badran 1993. A Neural Network Approach for Modelling Nonlinear Transfer Functions: Application for Wind Retrieval From Spaceborne Scatterometer Data. *Journal of Geophysical Research* **98**, 22827–22841.

Williams, C. K. I. 1997. Combining Spatially Distributed Predictions From Neural Networks. Technical Report Unpublished, NCRG, Department of Computer Science and Applied Mathematics, Aston University, Birmingham, B4 7ET, UK.

A The gradient of the error function of a MDN

This appendix is provided to show the reader the detail of the derivation of the gradient of the negative log likelihood of a mixture density network (the error function) with respect to the outputs of the feed forward network (which is usually a multi-layer perceptron).

A.1 The Error function and its partial derivatives

The negative log likelihood of MDN for the n^{th} training pattern, where $\boldsymbol{x_n}$ represents the n^{th} input pattern and $\boldsymbol{t_n}$ represents the n^{th} target pattern, is defined as:

$$E_n = -\ln\left\{\sum_{j=1}^{m} \alpha_j(\mathbf{x}_n)\phi_j(\mathbf{t}_n|\mathbf{x}_n)\right\}. \tag{39}$$

The j^{th} kernel, ϕ_j, for the n^{th} pattern, is defined as a spherical Gaussian of the form:

$$\phi_j = \frac{1}{(2\pi)^{\frac{c}{2}}\sigma_j^c(\boldsymbol{x}_n)} \exp\left(-\frac{\|\boldsymbol{t}_n - \boldsymbol{\mu}_j(\boldsymbol{x}_n)\|^2}{2\sigma_j^2(\boldsymbol{x}_n)}\right). \tag{40}$$

The total error is the summation of the error of each pattern:

$$E = \sum_{n=1}^{N} E_n. \tag{41}$$

Because of (41) the following analysis is on a per-pattern basis. For typographical clarity references to the target and training data sets are removed where possible from (39) and (40) and are represented in the following form:

$$E_n = -\ln\left\{\sum_{j=1}^{m} \alpha_j\phi_j\right\}. \tag{42}$$

and

$$\phi_j = \frac{1}{(2\pi)^{\frac{c}{2}}\sigma_j^c} \exp\left(-\frac{\|\mathbf{t}_n - \boldsymbol{\mu}_j\|^2}{2\sigma_j^2}\right). \tag{43}$$

The objective is to compute the derivatives of E_n at the outputs of the MLP network. Backpropagation is used to compute the errors at the inputs of the MLP (Bishop, 1994; Bishop, 1995). The derivatives of interest (using the terminology of Bishop (1994)) are,

- The partial derivative with respect to the outputs corresponding to the mixing coefficients z^α:

$$\frac{\partial E_n}{\partial z_j^\alpha}. \tag{44}$$

- The partial derivative with respect to the outputs corresponding to the variances or widths z^{σ}:

$$\frac{\partial E_n}{\partial z_j^{\sigma}}. \tag{45}$$

- The partial derivative with respect to the outputs corresponding to the centres or positions in target space z_{jk}^{μ}:

$$\frac{\partial E_n}{\partial z_{jk}^{\mu}}. \tag{46}$$

In order to simplify the analysis and notation, the posterior probability of a point is defined. Using Bayes theorem:

$$\pi_j = \frac{\alpha_j \phi_j}{\sum_{l=1}^{m} \alpha_l \phi_l}, \tag{47}$$

where,

$$0 \leqslant \pi_j \leqslant 1 \quad \forall j, \tag{48}$$

$$\sum_{j=1}^{M} \pi_j = 1. \tag{49}$$

A.1.1 Computing the derivatives of the mixing coefficients

The mapping constraints from the output of the MLP to the parameters of the GMM are considered when computing the partial derivatives. Using the chain rule:

$$\frac{\partial E_n}{\partial z_j^{\alpha}} = \sum_k \frac{\partial E_n}{\partial \alpha_k} \frac{\partial \alpha_k}{\partial z_j^{\alpha}}, \tag{50}$$

then from (42):

$$\begin{aligned} \frac{\partial E_n}{\partial \alpha_k} &= -\left[\frac{1}{\sum_{j=1}^{m} \alpha_j \phi_j} . \phi_k \right] \frac{\alpha_k}{\alpha_k} \\ &= -\frac{\alpha_k \phi_k}{\sum_{j=1}^{m} \alpha_j \phi_j} \frac{1}{\alpha_k}, \end{aligned} \tag{51}$$

and substituting (47):

$$\frac{\partial E_n}{\partial \alpha_k} = -\frac{\pi_k}{\alpha_k}. \tag{52}$$

Some care is needed when deriving $\frac{\partial \alpha_k}{\partial z_j^{\alpha}}$. Each α_k represents a mixing coefficient for each Gaussian in the mixture model. To ensure that the mixing coefficients represent probabilities they must always sum to one, that is $\sum_{j=1}^{M} \alpha_j = 1$. This is achieved by using the 'softmax' function on the output of the network such that:

$$\alpha_j = \frac{\exp(z_j^{\alpha})}{\sum_{l=1}^{m} \exp(z_l^{\alpha})}. \tag{53}$$

Using the quotient rule for differentiation and considering the two cases for $j = k$ and $j \neq k$ we have:

- for the case when $j \neq k$:

$$\begin{aligned}\frac{\partial \alpha_k}{\partial z_j^\alpha} &= \frac{\sum_{l=1}^m \exp(z_l^\alpha).0 - \exp(z_j^\alpha)\exp(z_k^\alpha)}{(\sum_{l=1}^m \exp(z_l^\alpha))^2}, \\ &= -\alpha_j \alpha_k, \qquad j \neq k.\end{aligned} \tag{54}$$

- for the case when $j = k$,

$$\begin{aligned}\frac{\partial \alpha_k}{\partial z_j^\alpha} &= \frac{\sum_{l=1}^m \exp(z_l^\alpha).\exp(z_j^\alpha) - \exp(z_j^\alpha)\exp(z_k^\alpha)}{(\sum_{l=1}^m \exp(z_l^\alpha))^2} \\ &= \frac{\exp(z_j^\alpha)}{\sum_{l=1}^m \exp(z_l^\alpha)} - \left[\frac{\exp(z_j^\alpha)}{\sum_{l=1}^m \exp(z_l^\alpha)}\right]^2, \\ &= \alpha_k - \alpha_k^2, \qquad j = k.\end{aligned} \tag{55}$$

We can summarise (54) and (55) by

$$\frac{\partial \alpha_k}{\partial z_j^\alpha} = \delta_{jk}\alpha_k - \alpha_k \alpha_j \qquad \begin{cases} j = 1, 2, ..., m \\ k = 1, 2, ..., m \\ \delta_{jk} \text{ is the Kronecker delta function.} \end{cases} \tag{56}$$

To compute the final derivative, substituting (52) and (56) into (50) yields

$$\begin{aligned}\frac{\partial \alpha_k}{\partial z_j^\alpha} &= \sum_k -\frac{\pi_k}{\alpha_k}\left(\delta_{jk} - \alpha_k \alpha_j\right) \\ &= \sum_k -\frac{\pi_k}{\alpha_k}\alpha_k \delta_{jk} + \sum_k \frac{\pi_k}{\alpha_k}\alpha_k \alpha_j \\ &= -\pi_j + \alpha_j.\end{aligned} \tag{57}$$

because $\sum_k \pi_k = 1$ and $\sum_k \pi_k \delta_{jk} = \pi_j$.
Then the final result is

$$\frac{\partial E_n}{\partial z_j^\alpha} = \alpha_j - \pi_j. \tag{58}$$

A.1.2 Computing the derivatives of the variances

The term z_j^σ refers to the *variance* of the Gaussian. When differentiating, we must be aware that we are differentiating with respect to the variance, σ_j^2. Again, considering the mapping constraints between the outputs of the MLP and the model parameters, the chain rule is used to expand the partial derivative:

$$\frac{\partial E_n}{\partial z_j^\sigma} = \frac{\partial E_n}{\partial \sigma_j^2}\frac{\partial \sigma_j^2}{\partial z_j^\sigma}. \tag{59}$$

Differentiating (42) with respect to σ_j^2 yields:

$$\frac{\partial E_n}{\partial \sigma_j^2} = -\left[\frac{1}{\sum_{l=1}^m \alpha_j \phi_j}\frac{\partial(\alpha_j \phi_j)}{\partial \sigma_j^2}\right], \tag{60}$$

the kernel, ϕ_j, is defined as a Spherical Gaussian (43), expanding (60) gives:

$$\alpha_j \phi_j = \alpha_j \frac{1}{(2\pi)^{\frac{c}{2}} \sigma_j^c} \exp\left(-\frac{\|\mathbf{t}_n - \boldsymbol{\mu}_j\|^2}{2\sigma_j^2}\right). \tag{61}$$

Completing the differentiation:

$$\begin{aligned}
\frac{\partial(\alpha_j \phi_j)}{\partial \sigma_j^2} &= \alpha_j \left\{ -c \frac{1}{2\sigma_j^2} \frac{1}{(2\pi)^{\frac{c}{2}} \sigma_j^c} \exp\left(-\frac{\|\mathbf{t}_n - \boldsymbol{\mu}_j\|^2}{2\sigma_j^2}\right) \right. \\
&\qquad \left. + \frac{\|\mathbf{t}_n - \boldsymbol{\mu}_j\|^2}{2\sigma_j^4} \frac{1}{(2\pi)^{\frac{c}{2}} \sigma_j^c} \exp\left(-\frac{\|\mathbf{t}_n - \boldsymbol{\mu}_j\|^2}{2\sigma_j^2}\right) \right\} \\
&= \underbrace{\alpha_j \frac{1}{(2\pi)^{\frac{c}{2}} \sigma_j^c} \exp\left(-\frac{\|\mathbf{t}_n - \boldsymbol{\mu}_j\|^2}{2\sigma_j^2}\right)}_{equation\ (61)} \left\{ -\frac{c}{2\sigma_j^2} + \frac{\|\mathbf{t}_n - \boldsymbol{\mu}_j\|^2}{2\sigma_j^4} \right\} \\
&= \frac{\alpha_j \phi_j}{2} \left\{ -\frac{c}{\sigma_j^2} + \frac{\|\mathbf{t}_n - \boldsymbol{\mu}_j\|^2}{\sigma_j^4}. \right\}
\end{aligned} \tag{62}$$

Combining (60) and (62):

$$\begin{aligned}
\frac{\partial E_n}{\partial \sigma_j^2} &= -\left[\frac{\alpha_j \phi_j}{2 \sum_{l=1}^m \alpha_l \phi_l} \left\{ -\frac{c}{\sigma_j^2} + \frac{\|\mathbf{t}_n - \boldsymbol{\mu}_j\|^2}{\sigma_j^4} \right\} \right] \\
&= -\frac{\pi_j}{2} \left\{ \frac{\|\mathbf{t}_n - \boldsymbol{\mu}_j\|^2}{\sigma_j^4} - \frac{c}{\sigma_j^2} \right\}.
\end{aligned} \tag{63}$$

The second term in expression (59) is easily computed:

$$\begin{aligned}
\frac{\partial \sigma_j^2}{\partial z_j^\sigma} &= \exp(z_j^\sigma) \\
&= \sigma_j^2.
\end{aligned} \tag{64}$$

Then substituting (63) and (64) into (59) the final derivative becomes

$$\frac{\partial E_n}{\partial z_j^\sigma} = -\frac{\pi_j}{2} \left\{ \frac{\|\mathbf{t}_n - \boldsymbol{\mu}_j\|^2}{\sigma_j^2} - c \right\}. \tag{65}$$

A.1.3 Computing the partial derivative with respect to the kernel centres

For this derivative there is no constraint (that is to say $\mu_{jk} = z_{jk}^\mu$) applied on the output of the MLP as there is in the previous two cases. Therefore $\frac{\partial E_n}{\partial z_{jk}^\mu}$ is computed directly from (42):

$$\frac{\partial E_n}{\partial z_{jk}^\mu} = -\left[\frac{1}{\sum_{l=1}^m \alpha_l \phi_l} \frac{\partial(\alpha_j \phi_j)}{\partial z_{jk}^\mu} \right]. \tag{66}$$

Then differentiating ϕ_j (43) with respect to each z_{jk}^μ yields:

$$\begin{aligned}\frac{\partial \phi_j}{\partial z_{jk}^{\mu}} &= \left(\frac{t_k - \mu_{jk}}{\sigma_j^2}\right) \frac{1}{(2\pi)^{\frac{c}{2}} \sigma_j^c} \exp\left(-\frac{\|\mathbf{t}_n - \boldsymbol{\mu}_j\|^2}{2\sigma_j^2}\right) \\ &= \frac{t_k - \mu_{jk}}{\sigma_j^2} \phi_j.\end{aligned} \tag{67}$$

Then substituting (67) into (66) yields the final result:

$$\begin{aligned}\frac{\partial E_n}{\partial z_{jk}^{\mu}} &= -\left[\frac{\alpha_j \phi_j}{\sum_{j'=1}^{m} \alpha_{j'} \phi_{j'}} \frac{t_k - \mu_{jk}}{\sigma_j^2}\right] \\ &= \pi_j \left\{\frac{\mu_{jk} - t_k}{\sigma_j^2}\right\}.\end{aligned} \tag{68}$$

A.2 Summary of results

The partial derivatives computed with respect to the feed forward network outputs are summarised below:

$$\begin{aligned}\frac{\partial E_n}{\partial z_j^{\alpha}} &= \alpha_j - \pi_j, \\ \frac{\partial E_n}{\partial z_j^{\sigma}} &= -\frac{\pi_j}{2}\left\{\frac{\|\mathbf{t}_n - \boldsymbol{\mu}_j\|^2}{\sigma_j^2} - c\right\}, \\ \frac{\partial E_n}{\partial z_{jk}^{\mu}} &= \pi_j \left\{\frac{\mu_{jk} - t_k}{\sigma_j^2}\right\}.\end{aligned}$$

B The gradient of the error function of the hybrid MDN

In this appendix the derivation of the hybrid MDN *framework is analysed in detail. The* MDN *framework is modified to encode the ambiguous directions that exist in the inverse mapping*

B.1 The Error function and its partial derivatives

The error function of a hybrid MDN is some what simpler than the full MDN:

$$E_n = -\ln\Big\{\alpha(\boldsymbol{x_n})\phi(\boldsymbol{t_n}|\boldsymbol{x_n}) + (1-\alpha(\boldsymbol{x_n}))\psi(\boldsymbol{t_n}|\boldsymbol{x_n})\Big\}, \tag{69}$$

$$\phi(\boldsymbol{t_n}|\boldsymbol{x_n}) = \frac{1}{2\pi\sigma^2(\boldsymbol{x_n})}\exp\left(-\frac{\|\boldsymbol{t_n}-\boldsymbol{\mu}(\boldsymbol{x_n})\|^2}{2\sigma^2(\boldsymbol{x_n})}\right), \tag{70}$$

$$\psi(\boldsymbol{t_n}|\boldsymbol{x_n}) = \frac{1}{2\pi\sigma^2(\boldsymbol{x_n})}\exp\left(-\frac{\|\boldsymbol{t_n}+\boldsymbol{\mu}(\boldsymbol{x_n})\|^2}{2\sigma^2(\boldsymbol{x_n})}\right), \tag{71}$$

simplified thus:

$$E_n = -\ln\Big\{\alpha\phi + (1-\alpha)\psi\Big\}. \tag{72}$$

Define the two posterior probabilities for each point:
The free centre:

$$\pi = \frac{\alpha\phi}{\alpha\phi + (1-\alpha)\psi}. \tag{73}$$

and the hybrid centre:

$$\gamma = \frac{(1-\alpha)\psi}{\alpha\phi + (1-\alpha)\psi}. \tag{74}$$

B.1.1 Computing the derivatives of the mixing coefficients

Using the chain rule:

$$\frac{\partial E_n}{\partial z^\alpha} = \frac{\partial E_n}{\partial \alpha}\frac{\partial \alpha}{\partial z^\alpha}, \tag{75}$$

taking the first term from (72):

$$\begin{aligned}\frac{\partial E_n}{\partial \alpha} &= -\left[\frac{1}{\alpha\phi + (1-\alpha)\psi}(\phi-\psi)\right] \\ &= -\left[\frac{\phi}{\alpha\phi + (1-\alpha)\psi} - \frac{\psi}{\alpha\phi + (1-\alpha)\psi}\right].\end{aligned} \tag{76}$$

Then simplifying by using posterior distributions

$$\begin{aligned} &= -\left[\frac{\phi}{\alpha\phi + (1-\alpha)\psi}\frac{\alpha}{\alpha} - \frac{\psi}{\alpha\phi + (1-\alpha)\psi}\frac{(1-\alpha)}{(1-\alpha)}\right] \\ &= -\left[\frac{\phi}{\alpha} - \frac{\gamma}{(1-\alpha)}\right] \\ &= -\left[\frac{\phi}{\alpha} - \frac{\gamma}{(1-\alpha)}\right] \\ &= -\left[\frac{(1-\alpha)\pi - \alpha\gamma}{\alpha(1-\alpha)}\right], \end{aligned} \tag{77}$$

but as $\gamma = 1 - \pi$, the final solution is:

$$\frac{\partial E_n}{\partial \alpha} = \frac{\pi - \alpha}{\alpha(1-\alpha)}. \tag{78}$$

The mixing coefficient α is a probability, and therefore is constrained by $0 \leqslant \alpha \leqslant 1$. The logistic function on the output of the MLP achives this:

$$\alpha = \frac{1}{1 + \exp(-z^\alpha)}. \tag{79}$$

Calculating the second term of (75),

$$\begin{aligned} \frac{\partial \alpha}{\partial z^\alpha} &= \frac{\exp(-z^\alpha)}{(1+\exp(-z^\alpha))^2} \\ &= \frac{1}{(1+\exp(-z^\alpha))}\left(\frac{\exp(-z^\alpha)}{1+\exp(-z^\alpha)}\right) \\ &= \alpha\left(1 - \frac{1}{(1+\exp(-z^\alpha))}\right) \\ &= \alpha(1-\alpha). \end{aligned} \tag{80}$$

Combining (78) and (80) the result for the derivative with respect to network outputs for the mixing coefficients gives,

$$\frac{\partial E_n}{\partial z^\alpha} = \pi - \alpha. \tag{81}$$

B.1.2 Computing the derivatives of the kernel variances (widths)

Using the chain rule:

$$\frac{\partial E_n}{\partial z^\sigma} = \frac{\partial E_n}{\partial \sigma^2}\frac{\partial \sigma^2}{\partial z^\sigma}, \tag{82}$$

and differentiating (72) with respect to σ^2 yields:

$$\begin{aligned}\frac{\partial E_n}{\partial \sigma^2} &= -\left[\frac{1}{\alpha\phi + (1-\alpha)\psi}\frac{\partial(\alpha\phi + (1-\alpha)\psi)}{\partial\sigma^2}\right] \\ &= -\left[\alpha\phi\left\{-c\frac{1}{2\sigma^2} + \frac{\|\boldsymbol{t_n} - \boldsymbol{\mu}\|^2}{2\sigma^4}\right\}\right. \\ &\quad \left. + (1-\alpha)\psi\left\{-c\frac{1}{2\sigma^2} + \frac{\|\boldsymbol{t_n} + \boldsymbol{\mu}\|^2}{2\sigma^4}\right\}\right]\frac{1}{\alpha\phi + (1-\alpha)\psi} \\ &= -\left[\frac{\pi}{2}\left\{-\frac{c}{\sigma^2} + \frac{\|\boldsymbol{t_n} - \boldsymbol{\mu}\|^2}{\sigma^4}\right\} + \frac{\gamma}{2}\left\{-\frac{c}{\sigma^2} + \frac{\|\boldsymbol{t_n} + \boldsymbol{\mu}\|^2}{\sigma^4}\right\}\right].\end{aligned} \tag{83}$$

The second term in expression (82) is easily computed:

$$\begin{aligned}\frac{\partial\sigma^2}{\partial z^\sigma} &= \exp(z^\sigma) \\ &= \sigma^2,\end{aligned} \tag{84}$$

and combining (83) and (84) equation (82) is complete:

$$\frac{\partial E_n}{\partial z^\sigma} = -\left[\frac{\pi}{2}\left\{\frac{\|\boldsymbol{t_n} - \boldsymbol{\mu}\|^2}{\sigma^2} - c\right\} + \frac{\gamma}{2}\left\{\frac{\|\boldsymbol{t_n} + \boldsymbol{\mu}\|^2}{\sigma^2} - c\right\}\right]. \tag{85}$$

B.1.3 Computing the derivatives of the kernel centres (means)

For this derivative there is no constraint (that is to say $\mu_{\mathbf{k}} = z_k^\mu$) applied on the output of the mlp as there is in the previous two cases. Therefore $\frac{\partial E_n}{\partial z_{jk}^\mu}$ is computed directly from (42):

$$\frac{\partial E_n}{\partial z_k^\mu} = -\left[\frac{1}{\alpha\phi + (1-\alpha)\psi}\frac{\partial(\alpha\phi + (1-\alpha)\psi)}{\partial z_k^\mu}\right], \tag{86}$$

and,

$$\frac{\partial(\alpha\phi + (1-\alpha)\psi)}{\partial z_k^\mu} = \alpha\phi\left(\frac{t_k - \mu_k}{\sigma^2}\right) - (1-\alpha)\psi\left(\frac{t_k + \mu_k}{\sigma^2}\right). \tag{87}$$

Combining (86) and (87) yields the final result:

$$\frac{\partial E_n}{\partial z_k^\mu} = -\left[\pi\left(\frac{t_k - \mu_k}{\sigma^2}\right) - \gamma\left(\frac{t_k + \mu_k}{\sigma^2}\right)\right]. \tag{88}$$

B.2 Summary of results

The partial derivatives computed with respect to the feed forward network outputs are summarised below:

$$\begin{aligned}\frac{\partial E_n}{\partial z^\alpha} &= \pi - \alpha, \\ \frac{\partial E_n}{\partial z^\sigma} &= -\left[\frac{\pi}{2}\left\{\frac{\|\boldsymbol{t_n} - \boldsymbol{\mu}\|^2}{\sigma^2} - c\right\} + \frac{\gamma}{2}\left\{\frac{\|\boldsymbol{t_n} + \boldsymbol{\mu}\|^2}{\sigma^2} - c\right\}\right], \\ \frac{\partial E_n}{\partial z_k^\mu} &= -\left[\pi\left(\frac{t_k - \mu_k}{\sigma^2}\right) - \gamma\left(\frac{t_k + \mu_k}{\sigma^2}\right)\right].\end{aligned}$$

C Definitions of the summary measures used in the results

This appendix gives the details of the tools used to analyse the performance of the inverse model, FoM and vector root mean square error. These statistics are computed after applying a simple disambiguation procedure which is detailed first

C.1 Disambiguation for model appraisal

The following method of disambiguation permits the comparison of inverse model performance in terms of the quality of retrieved wind vectors. The predicted direction, D_{pred}, and predicted wind speed, U_{pred}, are chosen using a simple de-aliasing algorithm. The observed wind vector, V_{obs} (derived from the numerical weather prediction model, and gives the best estimate of a 'true' wind vector available), is compared with the two most probable wind vectors inferred from the model by measuring the Euclidean distance between each inferred wind vector and the observed wind vector. The predicted wind vector, V_{pred}, is chosen as the wind vector with a minimum Euclidean distance from the observed wind vector. The predicted wind vector is then resolved to compute the predicted direction D_{pred} and the predicted wind speed U_{pred}.

C.2 Figure of Merit

This measure was proposed by David Offiler of the UK Meteorological Office and is becoming a more widely used statistic for comparing the performance of models within this field (Cornford *et al.*, 1997; Richaume *et al.*, 1998).

$$FoM = \frac{(F1 + F2 + F3)}{3}, \tag{89}$$

where:

$$F1 = \frac{40}{|U_{bias}| + 10U_{sd} + |D_{bias}| + D_{sd}}, \tag{90}$$

$$F2 = \frac{1}{2}\left(\frac{2}{U_{rms}} + \frac{20}{D_{rms}}\right), \tag{91}$$

$$F3 = \frac{4}{V_{rms}}. \tag{92}$$

where U represents wind speed, D the wind direction and V the wind vector (u, v). Where the parameters are defined:

$$U_{bias} = \frac{1}{N}\sum_{i=1}^{N} U_{res(i)}, \tag{93}$$

$$U_{res} = U_{pred} - U_{obs}, \tag{94}$$

$$U_{sd} = \sqrt{\left(\frac{1}{N}\sum_{i=1}^{N}(U_{res(i)})^2\right) - (U_{bias})^2}), \tag{95}$$

$$U_{rms} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(U_{res(i)})^2}, \tag{96}$$

$$V_{res} = \sqrt{U_{obs}^2 + U_{pred}^2 - 2U_{obs}U_{pred}\cos(D_{res})}, \tag{97}$$

and

$$D_{res} = D_{pred} - D_{obs}. \tag{98}$$

C.3 Predicted wind vector root-mean-square error

The predicted wind vector root-mean-square error is defined as

$$V_{rms} = \sqrt{\sum_{i}^{N}(u_{res(i)}^2 + v_{res(i)}^2)}, \tag{99}$$

where the residuals of u_i, v_i are:

$$u_{res(i)}^2 = (u_{pred(i)} - u_{obs(i)})^2 \tag{100}$$

and

$$v_{res(i)}^2 = (v_{pred(i)} - v_{obs(i)})^2, \tag{101}$$

where the predicted wind vectors are obtained using the method detailed in Subsection C.1

C.4 *Performance at* 20°

Performance at 20° (denoted *perf.* @ 20°) is a statistic that measures the percentage predicted wind directions that are within 20° of the target wind direction. This statistic is used in work by Thiria *et al.* (1993), Cornford *et al.* (1997) and Richaume *et al.* (1998).