NEURAL COMPUTING
NCRG
RESEARCH GROUP

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/

A Neural Network Sensor Model with Input Noise

Dan Cornford and Guillaume Ramage and Ian T. Nabney
d.cornford@aston.ac.uk

Technical Report NCRG/98/021 October 22, 1998

Abstract

The ERS-1 satellite carries a scatterometer which measures the amount of radiation scattered back toward the satellite by the ocean's surface. These measurements can be used to infer wind vectors. The implementation of a neural network based forward model which maps wind vectors to radar backscatter is addressed. Input noise cannot be neglected. To account for this noise, a Bayesian framework is adopted. However, Markov Chain Monte Carlo sampling is too computationally expensive. Instead, gradient information is used with a non-linear optimisation algorithm to find the maximum *a posteriori* probability values of the unknown variables. The resulting models are shown to compare well with the current operational model when visualised in the target space.

Keywords: Non-linear regression; input uncertainty; wind retrieval; scatterometer

1 Introduction

The ERS-1 satellite was launched in 1991 by the European Space Agency. It carries a scatterometer which measures the return radar power from three antennae that form a swathe to the right side of the satellite ground track. Some necessary technical background and notation are given in (Nabney *et al.*, 1998, this issue). As the three antennae sweep a 500 km wide swathe, the incidence angle, θ, with which the cells are illuminated varies.

In order to infer a wind field from scatterometer measurements, we need a probabilistic forward model for $P(\boldsymbol{\sigma^o} \mid u, v, \theta)$, where $\boldsymbol{\sigma^o}$ is the backscatter triplet and (u, v) are the wind vector components. Several algorithms for wind retrieval use deterministic forward models, such as the empirical model, CMOD4 (Offiler, 1994), used operationally. CMOD4 assumes the 3 antennae are equivalent and has a functional form defined by:

$$\sigma^o_{\text{lin}} = B_0(1 + B_1 \cos(\chi) + B_3 \tanh(B_2) \cos(2\chi))^{1.6} \tag{1}$$

where B_0, B_1, B_2, B_3 are complicated functions of the wind speed, s, and the beam incidence angle, θ. χ is the wind direction relative to the beam look angle and σ^o_{lin} is the backscatter measured on a linear scale (Stoffelen and Anderson, 1997). A forward model based on neural networks is presented in (Mejia *et al.*, 1998), but, it fits the observations poorly in $\boldsymbol{\sigma^o}$ space (Ramage, 1998) as it was trained without accounting for input noise.

2 Input uncertainty

2.1 Evidence for input uncertainty

In this problem the targets, $\boldsymbol{\sigma^o}$, are multi-dimensional and they can be plotted in 3D space where σ^o is the logarithm of σ^o_{lin}, rendering the noise distribution additive. Using such a representation, we show that input uncertainty cannot be neglected with respect to noise in the target variables.

In Fig. 1(a), a large number (10,000) of $\boldsymbol{\sigma^o}$ triplets are represented in target space, for all wind speeds and directions, and for a fixed incidence angle, $\theta = 34.9°$. These points are projected on the plane $\sigma^o = 0$ for the mid beam. As the measurements depend predominantly on two geophysical variables (wind speed and direction), they lie on a well defined manifold. Their distance from the manifold is small (around 0.2 dB) and is mostly due to instrumental noise.

These satellite measurements are labelled with wind vectors obtained from Numerical Weather Prediction (NWP) models. In Fig. 1(b), the measurements are selected so that the speed, given by the NWP model, lies in the range $9 \leq s \leq 10$ ms^{-1}. The solid surface corresponds to the operational forward model, CMOD4, plotted over the same range of wind speeds. This surface shows where the points should lie in the absence of input noise. The spread of the points is very large (around 5 dB) compared to instrumental noise. Therefore, input noise cannot be neglected.

2.2 Effects of input uncertainty

The test error of a dataset with noisy inputs is a poor measure of model accuracy. Using an independent test set is the most common way for assessing the quality of a regression model (Bishop, 1995) in the absence of input noise. If the input noise is not accounted for during training of a *nonlinear* model, model bias is likely to result (Wright, 1998). Thus, the computed test error

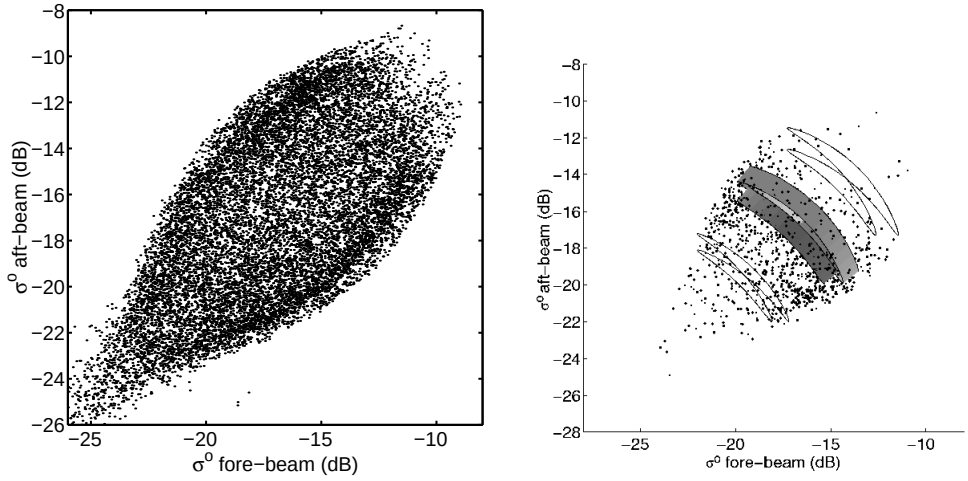

Figure 1: The manifold in $\boldsymbol{\sigma^o}$ space at an incidence angle of 34.9°. The points on the right plot are sub-sampled from the points on the left plot with the criterion $s \in [0, 9\ ms^{-1}]$. The surface is drawn for the same range of speeds. For reference, the other lines represent wind speeds of 6 (bottom left) and 13 ms^{-1} (top right).

is significantly affected by input noise and is not a true measure of model accuracy. Such a test error only tells us how good the model is at predicting targets from noisy inputs, but we want to infer the regression over noiseless inputs.

As explained above, the targets should lie on a manifold, for a fixed incidence angle. The surface defined by a model trained without accounting for input noise does not fit this manifold (see Section 6). Thus, one fundamental test for a model is its ability to fit this theoretical surface. In order to improve the accuracy of retrieval of high wind speeds, which are of most interest to meteorologists, the training dataset is sub-sampled from the available data.

2.3 Modelling the noise

In order to train a model while accounting for input noise, this noise must be modelled. The noise is difficult to describe in terms of wind speed and direction (Stoffelen and Anderson, 1997). Indeed, the noise in the speed component has a complicated skewed distribution at low wind speeds. However, in terms of Cartesian wind components, the noise distribution can be described by a spherical Gaussian distribution (Stoffelen and Anderson, 1997). The noise in target space is also assumed to have a spherical Gaussian distribution with a much smaller variance. The noise variances are set using results from (Stoffelen and Anderson, 1997).

3 Bayesian learning framework

A Bayesian approach to neural network modelling with input uncertainty is proposed in (Wright, 1998). The posterior probability of a new target is:

$$P(\boldsymbol{t}^* \mid \boldsymbol{x}^*, D') \propto \int_{\boldsymbol{w}} P(\boldsymbol{t}^* \mid \boldsymbol{x}^*, \boldsymbol{w})\, P(\boldsymbol{w} \mid D')\, d\boldsymbol{w}, \tag{2}$$

where D' is the noisy training set, $\boldsymbol{t}^*$ is the target and $P(\boldsymbol{t}^* | \boldsymbol{x}^*, \boldsymbol{w})$ is the target density conditional on a noiseless input $\boldsymbol{x}^*$ modelled by a neural network with weights $\boldsymbol{w}$. $P(\boldsymbol{w} | D')$ can be expanded:

$$P(\boldsymbol{t}^* \mid \boldsymbol{x}^*, D') \propto \int_{\boldsymbol{w}} P(\boldsymbol{t}^* \mid \boldsymbol{x}^*, \boldsymbol{w}) \times \int_{\boldsymbol{x_n}} [\prod_n \underbrace{P(\boldsymbol{t_n} \mid \boldsymbol{x_n}, \boldsymbol{w})}_{P_1} \underbrace{P(\boldsymbol{z_n} \mid \boldsymbol{x_n})}_{P_2} \underbrace{P(\boldsymbol{x_n})}_{P_3}] \underbrace{P(\boldsymbol{w})}_{P_4} \, d\boldsymbol{x_n} d\boldsymbol{w}, \tag{3}$$

where $\boldsymbol{t_n}$ are the targets in the training data, $\boldsymbol{x_n}$ are the corresponding noiseless (latent) inputs and $\boldsymbol{z_n}$ are the associated noisy inputs. Training the network consists of determining $P(\boldsymbol{w} \mid D')$. This can be done using Markov Chain Monte Carlo sampling. From (3), it can be seen that the Markov chain samples from $\{\boldsymbol{x_n}, \boldsymbol{w}\}$. Thus the size of the data set has to be reduced as much as possible in order to keep the Markov chain at a reasonable size. However, to ensure a highly accurate model for all possible inputs a large data set is required. A practical alternative to sampling is to compute the required derivatives and determine the maximum *a posteriori* probability values of $\{\boldsymbol{x_n}, \boldsymbol{w}\}$ using, for instance, a scaled conjugate gradient algorithm. This is the approach we adopt.

4 Chosen architecture

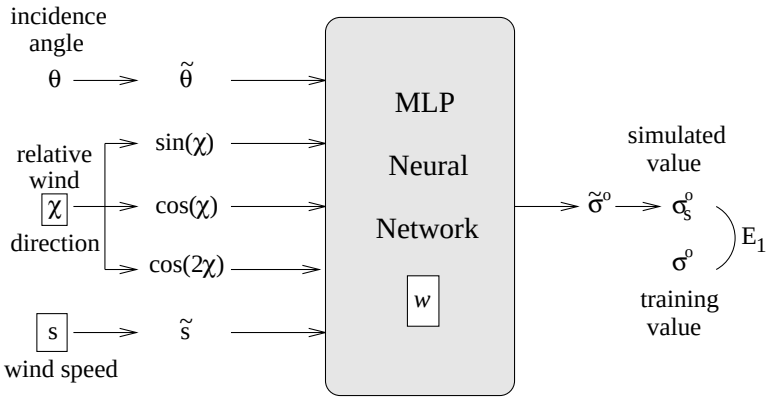

Figure 2: Architecture of the neural network model. Boxed variables are determined by non-linear optimisation.

Although we need $P(\boldsymbol{\sigma^o} \mid u, v, \theta)$, wind vectors should not be presented to the network as vector components. At constant speed and fixed incidence angle, σ^o varies roughly as $\cos(2\chi)$ (see (Mejia *et al.*, 1998)). Thus $\cos(2\chi)$ forms an input to the network. Both $\cos(\chi)$ and $\sin(\chi)$ are also used as inputs to respect the continuity of χ. All boxed variables in Fig. 2 are variables which will be optimised. All variables are normalised to zero mean with a common standard deviation around 0.5–0.7. As normalisation functions are part of the model, we want to keep them simple. The wind speed is related to the normalised wind speed by an exponential function to ensure it is positive.

5 Practical implementation

In order to compute the maximum *a posteriori* of $P(\boldsymbol{w} \mid D')$, we compute the four errors $E_i = -\ln(P_i)$, see equation (3). These terms are:

- $E_1 = -\ln(\prod_n P(\boldsymbol{\sigma^o} \mid s_s, \chi_s, \theta, \boldsymbol{w}))$ is the error of the model, calculated for the observed satellite measurements and for sampled wind vectors (s_s, χ_s) which tend to the noise free values during training.
- $E_2 = -\ln(\prod_n P(s_s, \chi_s \mid s, \chi))$ is the error due to the sampled wind vectors differing from their associated noisy wind vectors.
- $E_3 = -\ln(P(s, \chi)) = -\ln(P(s))$ is the prior distribution of true wind speeds in the training set. This is uniform in relative direction and so depends only on speed.
- $E_4 = -\ln(P(\boldsymbol{w}))$ is the prior over the weights which regularises the neural network (Bishop, 1995).

P_1 is assumed to be spherically Gaussian in target space, thus:

$$E_1 = \sum (\sigma_s^o - \sigma^o)^2 / (2\sigma_{\sigma^o}^2), \tag{4}$$

where the sum is over the three σ^o values and the patterns in the training set, σ_{σ^o} is the standard deviation of the errors in the $\boldsymbol{\sigma^o}$ measurements and σ_s^o is the output obtained propagating the sampled inputs (s_s, χ_s) through a Multi-Layer Perceptron (MLP) with M hidden units (Fig. 2). This can be written:

$$\begin{aligned} \tilde{\sigma}^o = \sum_{j,k=1}^{M} w_{k,j} \tanh(w_{j,1}\tilde{\theta} + w_{j,2}\sin(\chi) + w_{j,3}\cos(\chi) + \\ w_{j,4}\cos(2\chi) + w_{j,5}\tilde{s} + w_{j,0}) + w_{k,0}, \end{aligned} \tag{5}$$

where $\tilde{\cdot}$ denotes the associated normalised quantities. The output is then transformed into real σ^o space by inverting the normalisation.

P_2 is assumed to be spherically Gaussian in vector components of the wind with standard deviation σ_u, so that:

$$E_1 = \sum \left((u_s - u)^2 + (v_s - v)^2 \right) / (2\sigma_u^2). \tag{6}$$

The wind speed distribution, P_3, is represented by a uniform distribution between 4 and 28 ms^{-1}, similar to that in the dataset, with smooth Gaussian decrease at the ends:

$$\begin{aligned} E3 &= (4 - s_s)^2 / (2\sigma_u^2) \qquad s_s < 4\ ms^{-1} \\ E3 &= (28 - s_s)^2 / (2\sigma_u^2) \qquad s_s > 28\ ms^{-1} \end{aligned} \tag{7}$$

Finally, P_4 corresponds to the weight decay prior:

$$E_4 = \sum_{\boldsymbol{w}} \boldsymbol{w}^2 / (2\sigma_{\boldsymbol{w}}^2). \tag{8}$$

As noted previously, in order to compute the integral (2) using Monte Carlo integration, we need to build an independent set, $\{s, \chi, \boldsymbol{w}\}$, drawn from the expansion of $P(\boldsymbol{w}|D')$ in (3). We cannot obtain this in a reasonable time so a non-linear optimisation is performed using gradient information. The following derivatives are computed analytically:

$$\frac{\partial E_i}{\partial \tilde{\chi}}, \quad \frac{\partial E_i}{\partial \tilde{s}}, \quad \frac{\partial E_i}{\partial \boldsymbol{w}}, \qquad i = 1...4 \tag{9}$$

A training set of 10,000 patterns is used and thus we compute more than 20,000 derivatives at each step in the optimisation. In conventional training of neural networks we only need $\partial E_1 / \partial \boldsymbol{w}$.

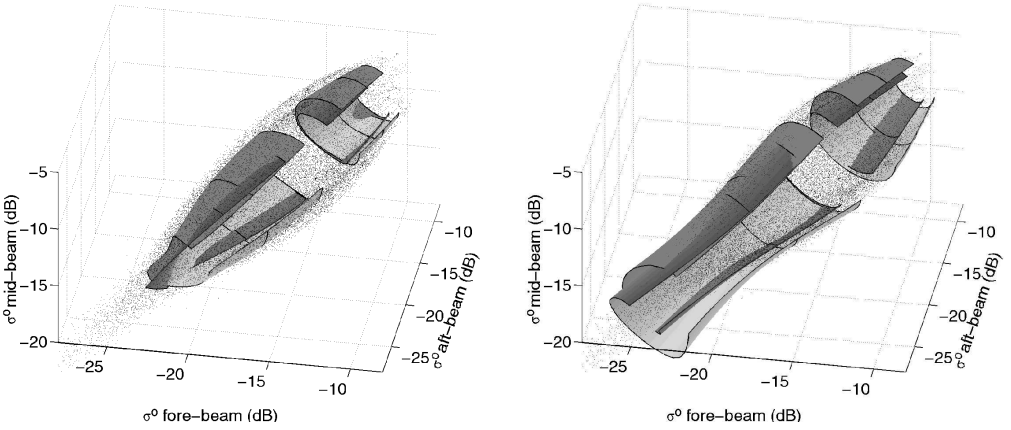

Figure 3: In target space, the surface defined by the model lies inside the target points (left plot) when we fit a model ignoring input noise. This is corrected if we account for input noise (right plot). Some parts of the manifold are removed to allow visualisation. Shading represents wind direction, lines on the surfaces represent constant wind speeds of 4, 8, 12, 16, 20 and 24 ms^{-1}. The surface is not drawn for 12–16 ms^{-1}.

6 Results and Discussion

Assessing the quality of a model is difficult; we use graphical representations in this study. Fig.3 shows the neural network based model trained without (left) and with (right) accounting for input noise. The surface defined by the model accounting for input noise fits the target data well in $\boldsymbol{\sigma^o}$ space, while the model trained without accounting for input noise lies largely within the interior of the manifold defined by the observations.

The model trained accounting for input noise can be seen to fit the observed $\boldsymbol{\sigma^o}$ values poorly at low wind speeds. This is believed to be a result of data selection. All *noisy* wind speeds below 4 ms^{-1} are discarded[1]. However, $\boldsymbol{\sigma^o}$ measurements corresponding to *true* wind speeds below 4 ms^{-1} are still present, but all of them are labelled with over-estimated speeds above 4 ms^{-1}. This sub-sampling on the basis of noisy data introduces bias in the data (Ramage, 1998) which will be removed by careful data selection in $\boldsymbol{\sigma^o}$ space in future work.

The models presented here are different from CMOD4 (Fig. 4 (left)) which has a strongly restrictive functional form, and fits the observations poorly at both high and low wind speeds.

In Fig. 4 (right) we can see the evolution of the wind vectors during optimisation of the model accounting for input noise ($\sigma^2_{\boldsymbol{w}} = 10$, $\sigma^2_u = 1.5\ ms^{-1}$, $\sigma^2_{\sigma^o} = 0.2\ dB$, 2500 iterations of the scaled conjugate gradient algorithm). The lines appear organised, which means the vector adjustments are correlated. This should not happen for good models as the wind vectors in the training set are selected so that their errors are uncorrelated.

The change in direction is slightly direction dependent. Absolute wind direction has no simple geophysical meaning as it is relative to each satellite beam. Therefore, this dependency is probably due to some misfit in the model, rather than systematic errors in the NWP wind directions. The change in speed appears speed dependent, and is due to the uniform selection of data from a fixed speed range as noted earlier. Finally, the change in speed is larger than the change in direction.

[1] At wind speeds below 4 ms^{-1} the $\boldsymbol{\sigma^o}$ measurements become unreliable (Offiler, 1994).

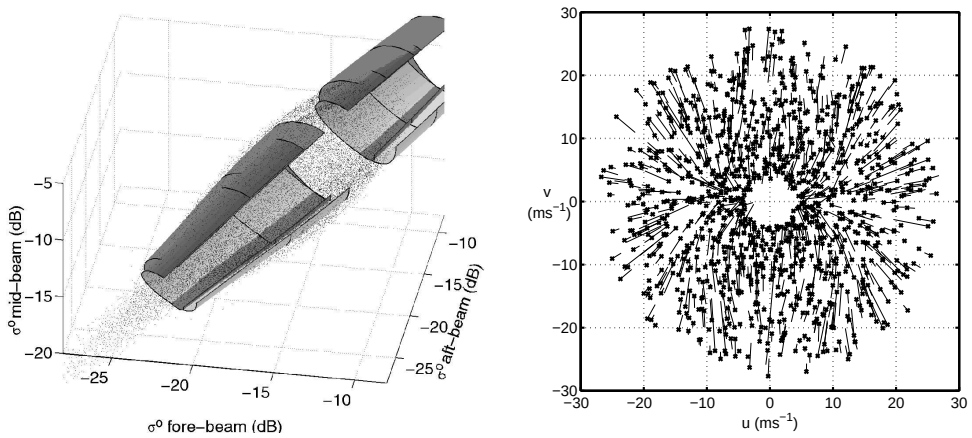

Figure 4: Left plot: The currently operational model, CMOD4, plotted as in Fig. 3. Right plot: crosses represent NWP winds (used as starting points). Straight lines link them to wind vectors at the end of optimisation. All incidence angles are represented and the number of points is reduced for clarity.

Describing the noise on the winds in terms of (χ, s) components might improve the model, although this effect may be due to the shape of the manifold in $\boldsymbol{\sigma^o}$ space.

7 Conclusions

The design of a non-linear regression model with input uncertainty is a difficult task, especially with multidimensional inputs and outputs. Even if the neural network has enough degrees of freedom to model the mapping, noise on the variables should be modelled accurately to obtain good results. The method we propose could be enhanced to determine the noise variances σ_u^2 (or σ_χ^2, σ_s^2) and $\sigma_{\sigma^o}^2$ as part of the modelling procedure by introducing a prior over these variances.

Due to the large number of derivatives that must be computed, training takes roughly twice as long as conventional training of neural networks in our implementation. Unlike a fully Bayesian treatment, only the maximum *a posteriori* probability value of $\boldsymbol{w}$ is computed so forward-propagation through the network is fast when we have noise free inputs. If we have noisy inputs then it is necessary to consider an additional integral over the inputs, to retrieve the output given a noisy input (Wright, 1998).

A proper data selection, based on sub-sampling using target ($\boldsymbol{\sigma^o}$) information, should minimise the errors associated with data selection based on noisy variables. Future work will also investigate the retrieval of wind fields using the forward model (Nabney *et al.*, 1998). This will enable a quantitative comparison of models.

Acknowledgements

This work is supported by the European Union funded NEUROSAT programme (grant number ENV4 CT96-0314). We thank Andy Wright for useful discussions.

References

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

Mejia, C., S. Thiria, N. Tran, M. Crepon, and F. Badran 1998. Determination of the Geophysical Model Function of the ERS-1 Scatterometer by the Use of Neural Networks. *Journal of Geophysical Research* **103**, 12853–12866.

Nabney, I. T., D. Cornford, and C. K. I. Williams 1998. Bayesian Inference for Wind Field Retrieval. *Neurocomputing Letters*. submitted.

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. Msc thesis, Aston University.

Stoffelen, A. and D. Anderson 1997. Scatterometer Data Interpretation: Estimation and Validation of the Transfer Function CMOD4. *Journal of Geophysical Research* **102**, 5767–5780.

Wright, W. A. 1998. Neural Network Regression With Input Uncertainty. In *Neural Networks for signal processing*, Volume 8, pp. 284–293.