# Estimating Conditional Volatility with Neural Networks

Ian T Nabney* H W Cheng†

## 1 Introduction

It is well known that one of the obstacles to effective forecasting of exchange rates is heteroscedasticity (input dependent conditional variance). The autoregressive conditional heteroscedastic (ARCH) model and its variants have been used to estimate a time dependent variance for many financial time series. However, such models are essentially linear in form and we can ask whether a non-linear model for variance can improve forecasting results just as non-linear models (such as neural networks) for the mean have done.

In this paper we consider two neural network models for variance estimation. Mixture Density Networks [1, 15] combine a Multi-Layer Perceptron (MLP) and a mixture model to estimate the conditional data density. They are trained using a maximum likelihood approach. However, it is known that maximum likelihood estimates are biased and lead to a systematic under-estimate of variance. More recently, a Bayesian approach approach to parameter estimation in such models has been developed [3] that shows promise in removing the maximum likelihood bias. However, up to now, this model has not been used for time series prediction.

Here we compare these algorithms with two other models to provide benchmark results: a linear ARIMA model and a conventional neural network trained with a sum-of-squares error function. In both these cases, the model estimates the conditional mean of the time series with a constant variance noise model. This comparison is carried out on daily exchange rate data for five currencies.

In this paper we are concerned with models that predict the conditional variance for the next time step. The conditional variance can be used to provide 'error bars' (also known as 'prediction intervals' in the regression literature) around the conditional mean. When the size of the error bars increases, then the value of the next forecast is less certain. This is less useful for options pricing than longer term variance forecasts, but the information can be incorporated into trading rules. For example, the size of error bars is a measure of how likely the predicted price movement is likely to be accurate. We are interested in comparing the generalisation performance of different models and use log likelihood on out of sample data with one step ahead prediction to compare results.

The rest of this paper is organised as follows. In section 2 the various different models that we employ are described and contrasted. Section 3 describes the methodology that

*Neural Computing Research Group, Aston University, Birmingham, B4 7ET, UK. Correspondence to first author at this address or by email at `i.t.nabney@aston.ac.uk`

†Faculty of Business Administration, University of Macau, Macau

is used in the empirical trials and then discusses the results. In the final section we draw together the main conclusions of this study and suggest future avenues of research.

## 2 Models

In this section we shall describe the main features of the models that we are comparing. Throughout this paper we shall use $z_t$ to denote the target values (the actual time series values) and $y_t$ to denote predictions made by models. For simplicity, we shall assume that the time series is univariate, although all the methods can be extended to multivariate time series.

### 2.1 ARIMA model

The autoregressive-integrated-moving average scheme (ARIMA) is a linear model that expresses an output $r_t$ at time $t$ in terms of previous outputs and random effects (or 'noise') $\epsilon_t$, which are the residuals (i.e. $y_t - z_t$, the difference between predicted and actual values at time $t$) of the model at earlier time steps.

$$y_t = \delta + \sum_{i=1}^{p} \phi_i y_{t-i} + \epsilon_t - \sum_{j=1}^{q} \theta_j \epsilon_{t-j} \tag{1}$$

Often a time series may be differenced to remove trends. A model of the form given by equation (1) that is applied to a time series that has been differenced $d$ times is said to be of orders $p$, $d$ and $q$, written ARIMA$(p, d, q)$. The $\delta$ term is a constant drift term.

The parameters $\delta$, $\phi_i$, $\theta_j$ are estimated from a training dataset. We used the following method to determine the model structure (defined by the integers $p$, $d$, and $q$).

- If the partial autocorrelation function (PACF) of the differenced series displays a sharp cutoff and/or the lag $-\,1$ autocorrelation is positive — i.e., if the series appears slightly 'underdifferenced' — then we tried adding an AR term to the model. The lag at which the PACF cuts off is the indicated number of AR terms.

- If the autocorrelation function (ACF) of the differenced series displays a sharp cutoff and/or the lag $-$ 1 autocorrelation is negative — i.e., if the series appears slightly 'overdifferenced' — then we tried adding an MA term to the model. The lag at which the ACF cuts off is the indicated number of MA terms.

- It is possible for an AR term and an MA term to cancel each other's effects, so if a mixed ARMA model seems to fit the data, we also tried a model with one fewer AR term and one fewer MA term, particularly if the parameter estimates in the original model require more than 10 iterations to converge.

If we assume that the random effects $\epsilon_t$ have a Gaussian distribution $N(0, \sigma^2)$ with zero mean and constant variance, then we can compute the log likelihood of the actual target value once we have estimated $\sigma^2$. This can be done by calculating the sample average of the residuals on the training set. It is then straightforward to estimate the parameters using a maximum likelihood approach. Thus the ARIMA scheme is a linear model for the conditional mean with a constant noise variance.

## 2.2 MLP with constant variance

The multi-layer perceptron (MLP) is a neural network model that can be used for regression (as here) or classification. In the case of regression, each network output is the linear combination of the activations of $n$ so called *hidden units*, each of which is a nonlinear function applied to a linear combination of the inputs. If, for the sake of simplicity, we assume that the output is one dimensional, we can write this model in the form

$$y = f(\mathbf{x}; \mathbf{w}) = \sum_{i=1}^{n} a_i \phi(\mathbf{u}_i^T \mathbf{x} + b_i) \tag{2}$$

where $\mathbf{x}$ is the input vector, $\mathbf{w}$ denotes the set of parameters (or *weights*) in the model, $\mathbf{u}_i$ are the weights from the inputs to hidden unit $i$, $b_i$ is the bias for the $i$th hidden unit, and $a_i$ are the hidden to output weights. The function $\phi$ is the *activation* function, and is chosen to be nonlinear (for example tanh). When applied to time series forecasting, the input vector $\mathbf{x}$ is typically a vector of previous values from the time series, which makes the network a non-linear auto-regressive model. Some work has been done on incorporating past residuals as inputs for financial time series forecasting [4] with some success, but we will not pursue this approach here (partly because we are interested in using the residuals to model the conditional variance of the time series).

The ARIMA model is *parametric* in that a specific functional form (linear in this case) is assumed and the parameters are then fitted from the data. In contrast, the MLP can be viewed as a nonlinear (due to the activation function) *semi-parametric* data model. This is because the MLP allows a very general class of functional forms (in fact, the MLP approximates any continuous function of its inputs to an arbitrary accuracy: see [5, 7, 6]) in which the number of adaptive parameters (which is governed by $n$ in equation 2) can be varied in a systematic way to build ever more flexible models, and where this number is independent of the training data set size.

In the usual approach to regression, the sum of squares error function is used:

$$E = \frac{1}{2} \sum_{k=1}^{N} [f(\mathbf{x}_k; \mathbf{w}) - z_k]^2 \tag{3}$$

where the index $k$ runs over the $N$ training patterns. Then it is well know (see [2]) that the optimal function (in the sense of minimising the error) is

$$f(\mathbf{x}; \mathbf{w}^* = \langle z | \mathbf{x} \rangle \tag{4}$$

the *conditional mean* of the target $z$ given $\mathbf{x}$. It can also be shown that at the global minimum of the error function, its residual value is the average variance of the target value around its conditional average. We can represent the conditional distribution of the target data by a Gaussian function with centre (depending on the input $\mathbf{x}$) given by $f(\mathbf{x}; \mathbf{w}^*)$ and a constant variance determined by the residual error.

The use of a least squares error function does not require the conditional distribution of the target data to be Gaussian, but it cannot distinguish between a Gaussian distribution and any other distribution with the same conditional mean and constant variance. If we do assume that the target has a Gaussian conditional distribution, then the sum of squares error function arises naturally through a maximum likelihood approach, assuming that the data is drawn independently from some fixed distribution. The error function in equation 3 is given by $E = -\ln \mathcal{L} + c$ where $c$ is a constant (which can be ignored when minimizing $E$) and $\mathcal{L}$, the data likelihood, is given by

$$\mathcal{L} = \prod_{k=1}^{N} p(z_k|\mathbf{x}_k)p(\mathbf{x}_k) \tag{5}$$

Time series data is *not* an independent sample, but Williams has shown in [18] how a similar decomposition can be achieved under the assumption that the conditional density at each time step depends only on a fixed number of previous values from the time series.

$$p(x_t|x_{t-1}, \dots, x_0) = p(x_t|x_{t-1}, \dots, x_{t-T}) \tag{6}$$

Despite the constant variance constraint, MLPs have been used with great success for a number of forecasting problems.

To train a neural network, it is necessary to minimise the value of $E$ by adjusting the parameter vector $\mathbf{w}$. This can be done with a number of different non-linear optimisation algorithms: however, most of these require the partial derivatives

$$\frac{\partial E}{\partial \mathbf{w}} \tag{7}$$

to speed up the search in high dimensional parameter space. One of the reasons for choosing a model of the form 2 is that these partial derivatives can be computed efficiently using the back-propagation algorithm. In our experiments we used quasi-Newton methods with the BFGS update formula (see [16] for an implementation), or scaled conjugate gradient [12].

Of course, a maximum likelihood approach with no regularisation to penalise overly-complex solutions is prone to over-fitting, where the noise in the finite training dataset is fitted, rather than the underlying generator of the data (the true conditional mean). In this study, rather than use a Bayesian regularisation method to solve this [9], we simply used *early stopping* (as in [13]). This method is based on the fact that during a typical training session, the training set error decreases monotonically. However, the error measured with respect to independent data, the *validation set*, often shows a decrease at first followed by an increase as the network starts to over-fit. Training can therefore be stopped at the points where the validation set error increases: this network is expected to have good generalisation performance.

There are two drawbacks of this approach with financial time series. Firstly, the amount of noise (and indeed, the likely non-stationarity of the underlying data generator) mean that early stopping may stop too early, with an under-trained network. Secondly, the validation set has to be independent from the training data, and so if we select contiguous blocks of data (to minimise the correlation between datasets), this means that the test dataset is separated by a longer interval of time from the training set (assuming that the order is training: validation: test). This increases the likelihood of poor generalisation caused by non-stationarity.

## 2.3 Mixture Density Networks

The MLP provides a very flexible model for predicting the conditional mean of an unknown function. However, the constant variance assumption is often unrealistic. In financial data, for example, many time series are known to exhibit heteroscedasticity, and it is therefore logical to extend the simple MLP framework to estimate the conditional variance of the target data in addition to the conditional mean. This variance can be used to give a more accurate estimate of the noise model.

A maximum likelihood approach to this problem is quite straightforward. For each target value, the neural network has two outputs, each of which is connected to all the hidden units: one represents the target value (the conditional mean, in fact), while the other represents the conditional variance. The conditional mean is a linear combination of the hidden units as in equation 2. However, the variance must be a non-negative quantity. It is convenient to use an exponential function to constrain the network output to the correct values.

$$\sigma^2(\mathbf{x}) = \exp\left[\sum_{i=1}^{n} a_i \phi(\mathbf{u}_i^T \mathbf{x} + b_i)\right] \tag{8}$$

It is a straightforward exercise in calculus to calculate the relevant partial derivatives of the negative log likelihood of the data, and then optimisation algorithms can be used to train the network parameters. In our experiments, we used a quasi-Newton algorithm with the BFGS update method for training and early stopping to regularise the network. Similar models have been applied to predicting time series before. In [15] a pair of networks were trained with a more complicated procedure.

If the output is multi-dimensional then this approach can be generalised to a multi-variate Gaussian noise model, where the network predicts the conditional mean and the covariance matrix. In [18] this approach is used to model the correlations between multiple currency markets.

This model simply extends the MLP by allowing the variance of a Gaussian representing the conditional density to be input dependent. However, by using more complex conditional densities (for example, mixture models) with parameters estimated by the network, it is possible to model arbitrary conditional distributions [1]. The probability density of the target data is represented by a linear combination of kernel functions of the form

$$p(t|\mathbf{x}) = \sum_{j=1}^{m} \alpha_j(\mathbf{x}) \phi_i(t|\mathbf{x}) \tag{9}$$

where the *mixing coefficients* $\alpha_j$ satisfy the following constraints

$$\alpha_j(\mathbf{x}) \geq 0 \qquad \text{and} \qquad \sum_{j=1}^{m} \alpha_j(\mathbf{x}) = 1 \qquad \forall \mathbf{x} \tag{10}$$

Various choices for the kernel functions $\phi$ are possible. In this paper we have chosen Gaussians with input dependent means and variances. For a good model, the relationship of the mixing coefficients and the kernel parameters on the input vector $\mathbf{x}$ may be non-linear. It therefore makes sense to use a neural network to model this relationship.

## 2.4 Bayesian Inference of Noise Levels

Instead of using a maximum likelihood approach to estimating the model coefficients (or weights), which attempts to find a single optimal set of values, the Bayesian approach generates a probability distribution function in parameter space representing the relative degrees of belief in different values for the parameter vector. This function is initially set to some prior distribution $p(\mathbf{w})$. Once the training data $D$ has been observed, the prior is converted into a posterior distribution $p(\mathbf{w}|D)$ through the use of Bayes' theorem and the data likelihood $p(D|\mathbf{w})$.

In principle we make predictions and estimate $\sigma^2(\mathbf{x})$ by averaging the predictions made by all possible networks weighted by their corresponding posterior posterior probability. However, as this posterior distribution tends to be very complex, this procedure requires computationally intensive methods such as Markov Chain Monte Carlo. A more practical approach is to select a single network given by the *mode* of the posterior distribution (i.e. the parameter vector $\mathbf{w_{MP}}$ that maximises $p(D|\mathbf{w})$. Because we estimate the probability distribution of the parameter estimates, we can also give error bars on our forecasts that take into account the uncertainty in the weight vector.

It is well known that the maximum likelihood estimate of variance is biased (it tends to underestimate variance). The regularisation methods that we have described above have drawbacks for financial time series, and so it is of interest to apply a Bayesian approach to learning as this should, in theory, give rise to unbiased estimates.

Although the Bayesian framework is very attractive from a theoretical point of view, it can be difficult to apply in practice. MLP networks give rise to posterior weight distributions that are difficult to evaluate [9]. Instead we use a generalised linear regression model (which is basically equivalent to a radial basis function (RBF) network) as in [3]. The regression output is given by

$$y(\mathbf{x};\mathbf{w}) = \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}) \tag{11}$$

where $\boldsymbol{\phi}$ represents a vector of basis functions (one of which is a constant $\phi_0 = 1$ and is the bias term). It turns out to be convenient to use two separate networks: one for regression and one for the variance. The inverse variance model is given by

$$\beta(\mathbf{x};\mathbf{u}) = \exp(\mathbf{u}^T\boldsymbol{\psi}(\mathbf{x})). \tag{12}$$

The basis functions $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ are chosen to be Gaussians (in this instance; other choices are possible, see [8]) and are parameterised so that they model the unconditional probability density of the input data using the EM algorithm to train a mixture model with the same number of centres [2].

The algorithm involves a hierarchical approach to modelling with 'hyperparameters' to control the prior distributions that are estimated from the data. Between each re-estimation of the hyperparameters the most probable value of the weight vectors $\mathbf{w}$ and $\mathbf{u}$ is found. The optimisation of $\mathbf{w}$ turns out to be straightforward, as the error for this network (based on penalised negative log likelihood) is quadratic in the weights, and so can be solved by standard techniques from linear algebra. (This is another reason for choosing an RBF network in place of an MLP). The error function for $\mathbf{u}$ is not quadratic, so we use a standard

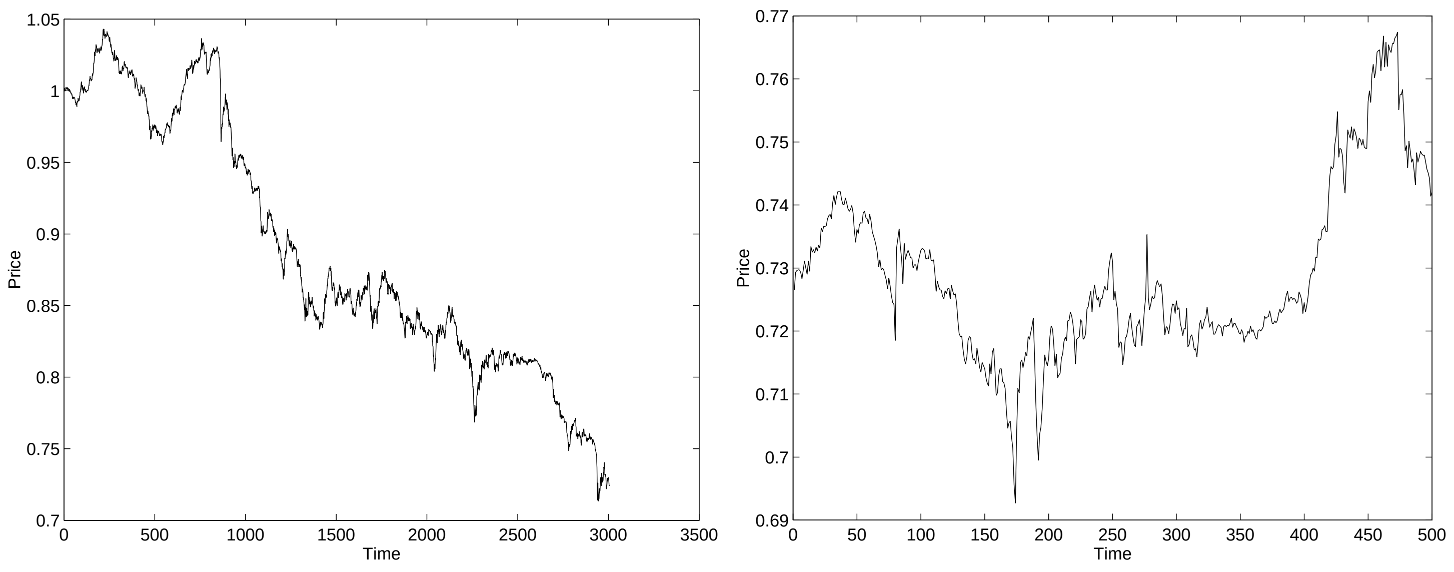

Figure 1: Datasets for US dollar/Canadian dollar: training/validation (left) and test (right).

non-linear optimisation algorithm. In the work reported here, the scaled conjugate gradient algorithm was used [12].

The algorithm described involves computing the Hessian matrix (the matrix of second order partial derivatives). Some of the intermediate steps of this computation require the storage of matrices of size $O(N^2)$, where $N$ is the number of training points. Even on workstations, this put an upper limit on the size of dataset that can be used for training. In our experiments, we used training sets of size at most 600. Because the Bayesian approach to training provides regularisation (and thus controls network complexity), there is no need for a validation set, so we sub-sampled the combined training and validation sets.

## 3 Experiments

### 3.1 Methodology

We used data from five currency markets: US dollar/Canadian dollar (CAD); US dollar/sterling (GBP); US dollar/Deutsche Mark (DEM); US dollar/Swiss Franc (CHF); US dollar/Japanese Yen (JPY). The daily closing prices in the period June 1, 1973 to May 21, 1987 were used, giving 3505 time periods in total. For the neural network models, each input pattern consisted of the five previous prices, and the price for the next time step and the conditional variance were the outputs. The structure of the ARIMA models was determined using the method described in section 2.1. We used the first 2505 patterns as a training set, the next 500 for a validation set (where relevant for early stopping) and the last 500 as the test set. Where early stopping was used, the validation set performance was evaluated every 50 cycles of the training algorithm.

We are interested in the generalisation performance of the different models, so they were compared on the basis of the negative log likelihood of the test set (i.e. 'out of sample'

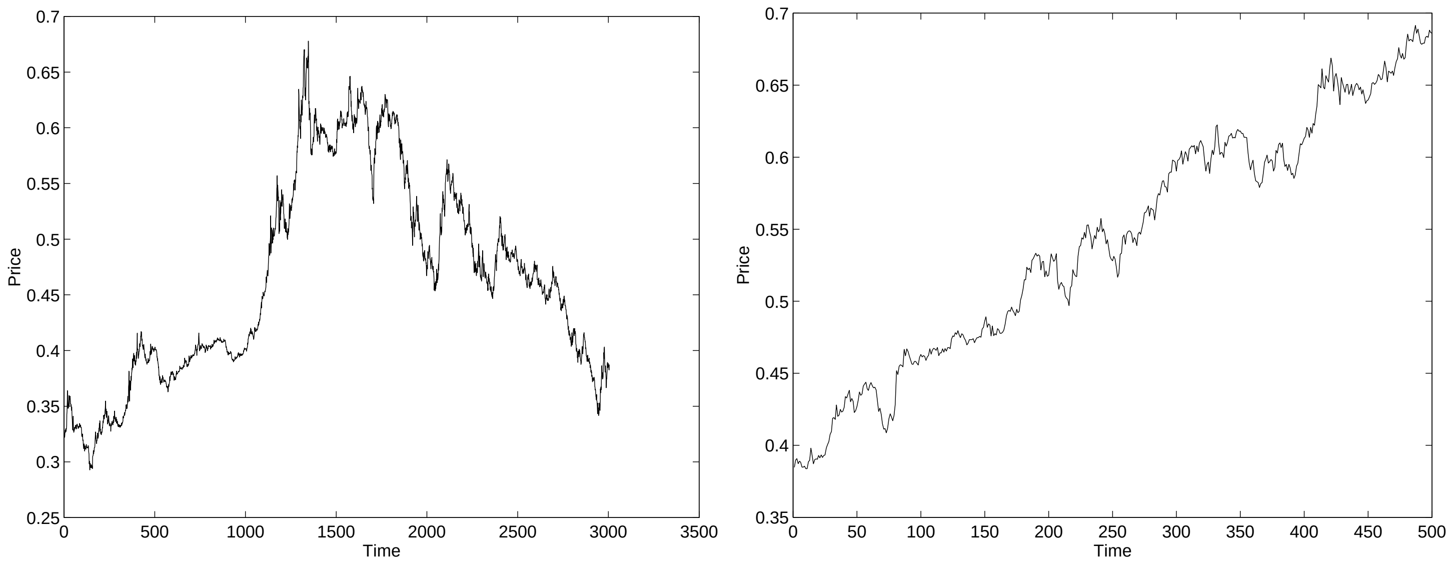

Figure 2: Datasets for US dollar/Swiss Franc: training/validation (left) and test (right).

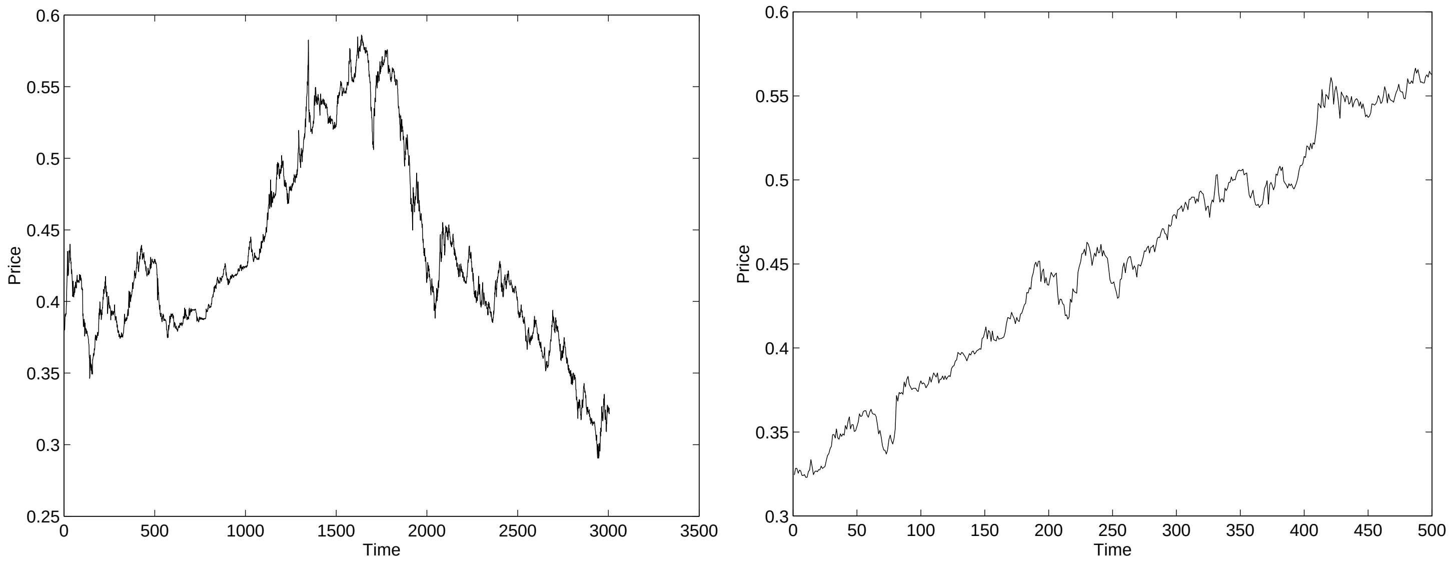

Figure 3: Datasets for US dollar/Deutsche Mark: training/validation (left) and test (right).

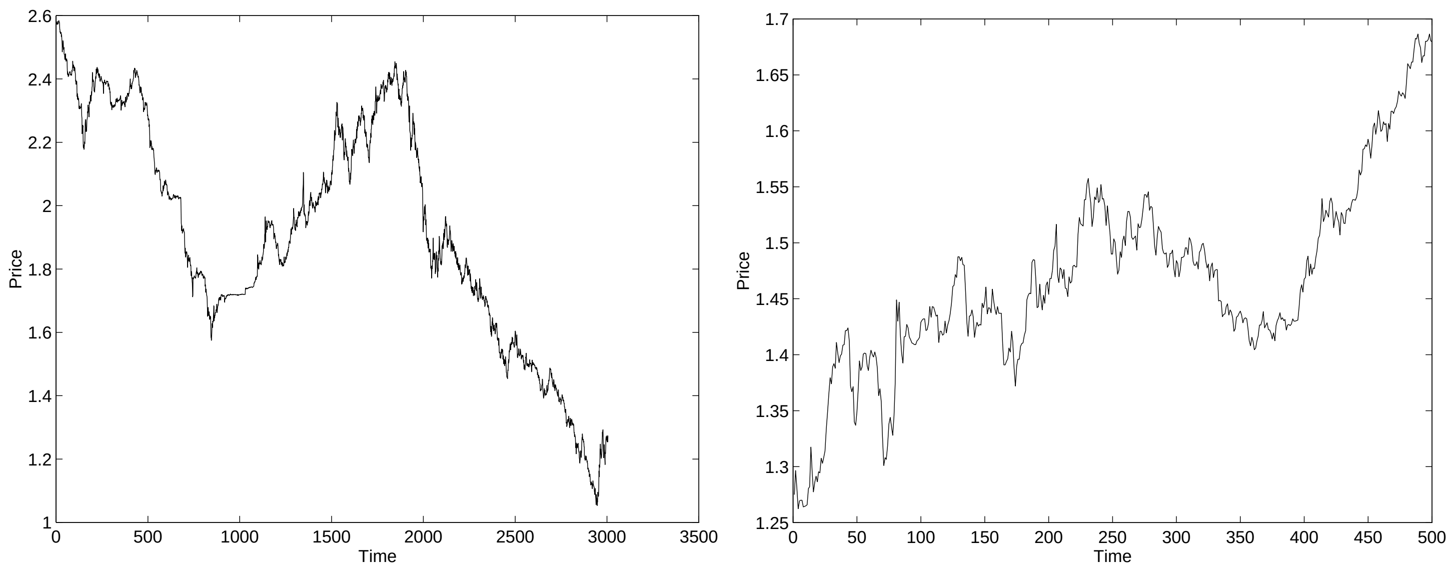

Figure 4: Datasets for US dollar/British pound: training/validation (left) and test (right).

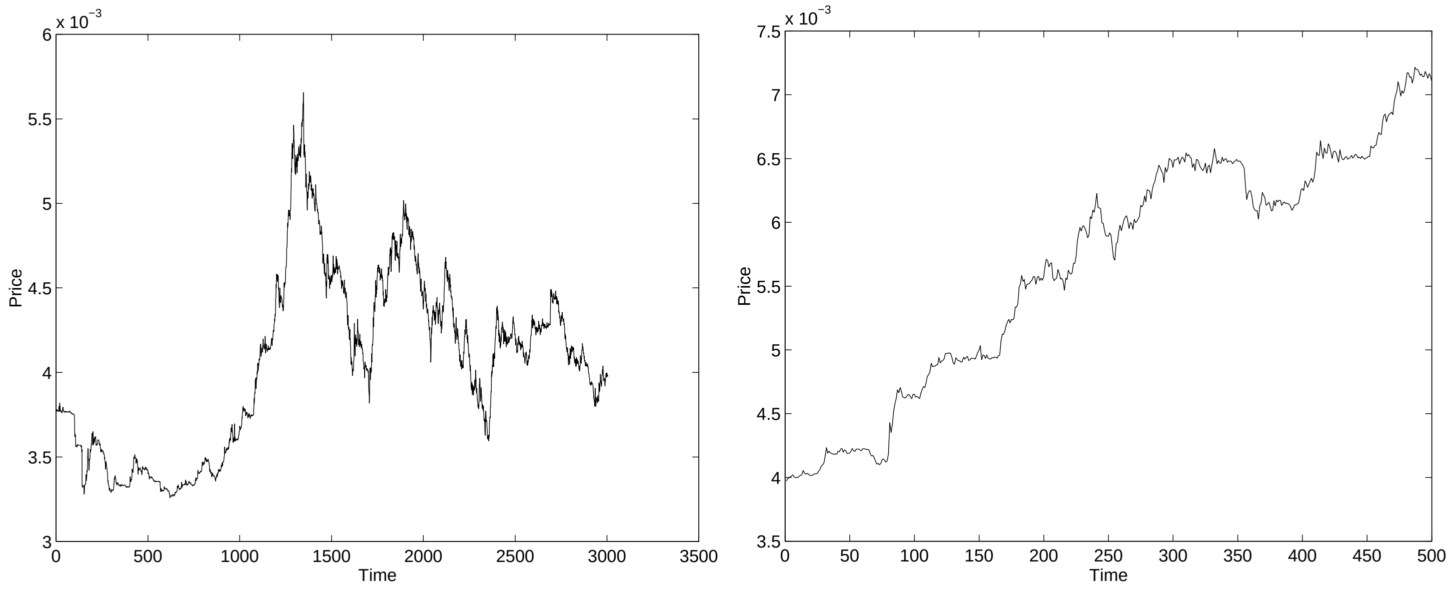

Figure 5: Datasets for US dollar/Japanese Yen: training/validation (left) and test (right).

testing). Of course, as there is no on-line adjustment of parameters on the test set, all the models are vulnerable to the effects of non-stationarity in the test sets.

Following [17] we can identify the following sources of variation when evaluating the generalisation performance of different algorithms:

1. Random selection of test cases.
2. Random selection of training set.
3. Random initialisation of learning method.
4. Stochastic elements in the training algorithm.
5. Stochastic elements in the predictions from a trained method (e.g. Monte Carlo estimates from the posterior predictive distribution).

Rasmussen goes on to describe procedures to estimate the effects of these causes of variation in order to measure statistically the true significance of differences in generalisation performance. Unfortunately, these rely on being able to select training and test data from the same distribution independently, something which clearly breaks down for time series data. Hence we will not be able to give results of significance tests for the differences between the generalisation performance of different algorithms. This is something that we intend to investigate in the future.

## 3.2 Results

We used very simple data pre-processing: the training data was normalised to zero mean and unit variance for the neural network models. The test data was normalised with the same linear transformation. Although most work in this field models log returns (i.e. $\log z_t - \log z_{t-1}$), we found that this gave worse results for the ARIMA models (some of which failed to converge) so we modelled raw prices throughout. Table 1 contains the generalisation performance of each model tested.

We had little difficulty with training any of the models with the exception of the Bayesian treatment of input dependent noise. These networks often converged to local minima, and when the size of the regression network was increased to 40 hidden units, the Hessian became singular and the weight vector overflowed. The results in table1 were obtained for a regression model with 30 hidden units, and a noise model with 10 hidden units.

The generalisation results demonstrate the the Mixture Density Network method performs best on all markets. There is a slight improvement in performance for a model with a mixture of three Gaussians at the output. Early stopping had very little effect on generalisation performance.

The results on the Japanese Yen data were much more varied than for the other currencies. This was particularly so in the case of the Bayesian treatment, where the log likelihood for the test set was usually in the order of $10^5$. The figure given in table 1 is very much an outlier. This was because the regression network performed very poorly towards the end of the test set, where the range of inputs lies well outside that in the training data. This is a particular problem for the RBF network when local basis functions (like Gaussians) are used,

|  | CAD | CHF | DEM | GBP | JPY |
|---|---|---|---|---|---|
| ARIMA | −2271.2 | −1910.4 | −1947.8 | −1525.8 | −4114.0 |
| MLP | −2118.6 | −1251.6 | −1889.4 | −1251.6 | −4321.0 |
| MDN 1 centre | −2307.0 | −1928.3 | −2034.66 | −1527.8 | 4780.1 |
| MDN 3 centres | −2342.0 | −1937.8 | −2050.0 | −1518.6 | −4207.2 |
| MDN 5 centres | −2360.4 | −1945.5 | −1816.4 | −1519.3 | −3180.3 |
| Bayesian model | −1553.0 | −1127.5 | −1119.0 | −1440.0 | −67.9 |

Table 1: Negative log likelihood of test data

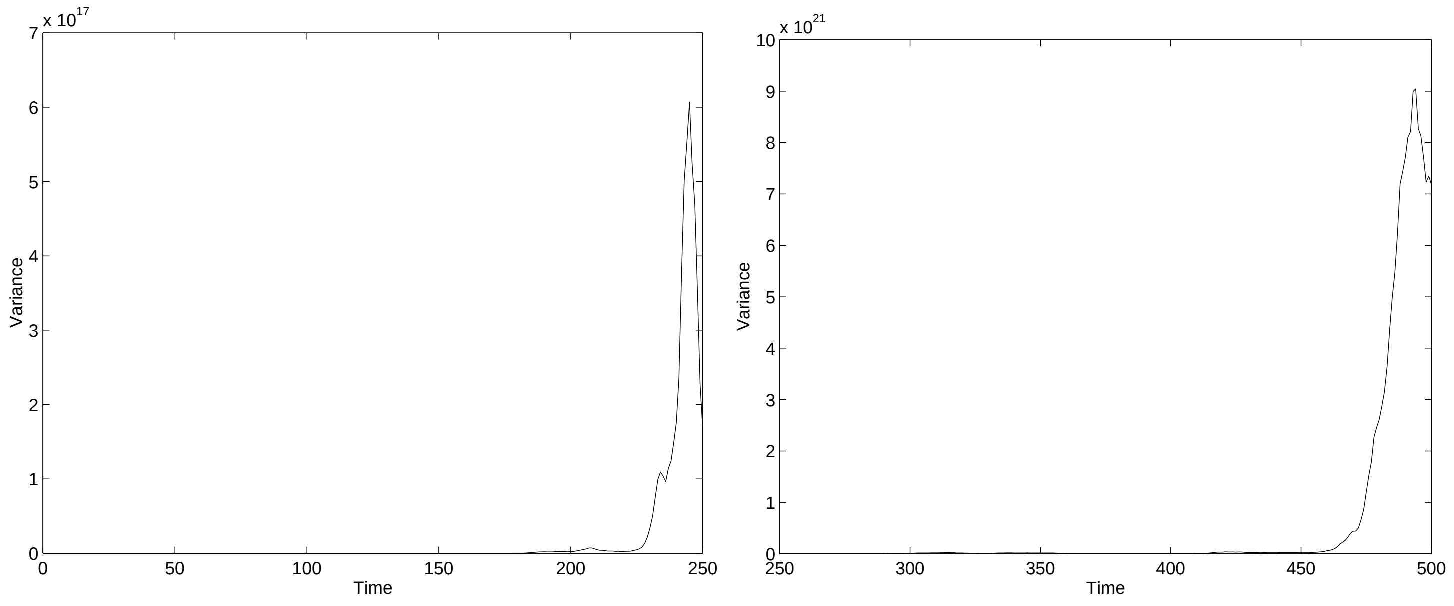

Figure 6: Predicted test data variance for US dollar/Japanese Yen: first 250 points (left) and second 250 points (right).

since these will extrapolate extremely poorly outside the range of data they were trained on as their response will be zero. An MLP, which uses linear combinations of its inputs, will extrapolate in a somewhat more predictable and reasonable fashion. The best answer to this problem is to detect novel data, and re-train (or adjust) the parameters in the model. Even for the best model, the predicted test data variance contained some extremely large values, as can be seen in figure 6. The variance results for the other currencies (figures 7 and 8 are more in line with expectations.

The structure of the ARIMA models, and the variance parameter for the ARIMA and MLP models are given in table 2. It is rather surprising to see that the variance for the ARIMA model (which is the average training set residual) is less than that for the MLP. It seems likely that this is due to the moving average terms making a significant contribution to the accuracy of the conditional mean prediction.

Because all the ARIMA models used differencing, we experimented with pre-processing the data for the neural network models by taking the difference (i.e. $z_t - z_{t-1}$), but it did

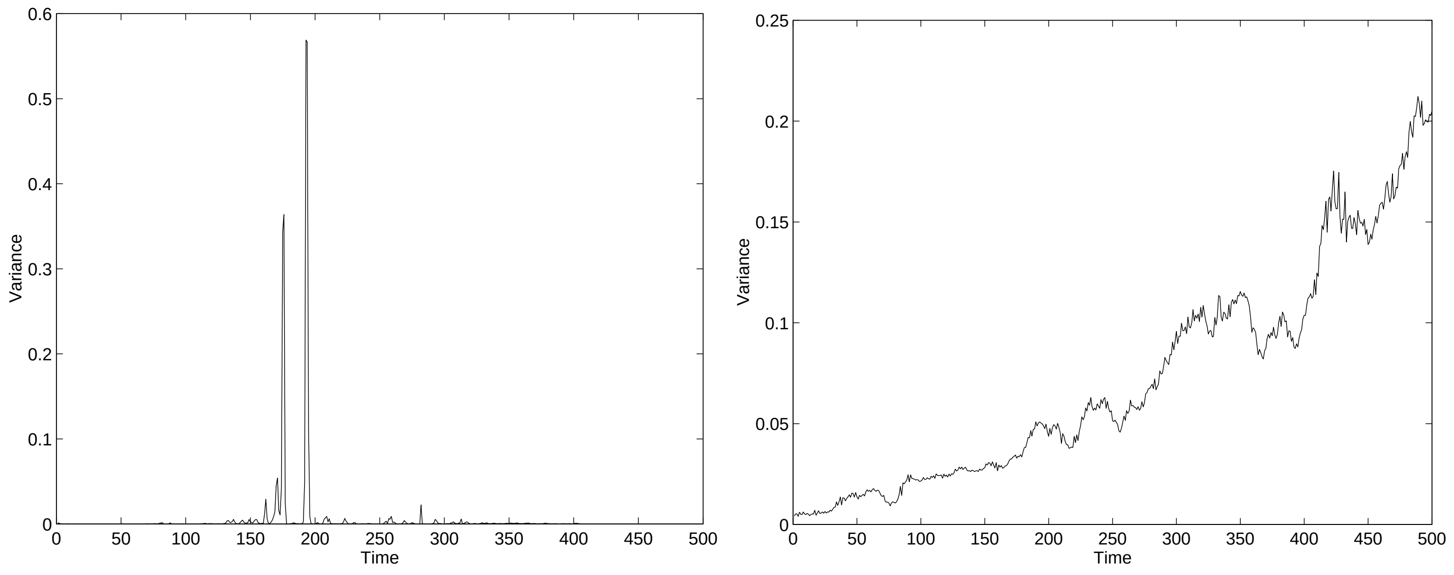

Figure 7: Predicted test data variance for US dollar/Canadian dollar (left) and US dollar/Swiss Franc (right).

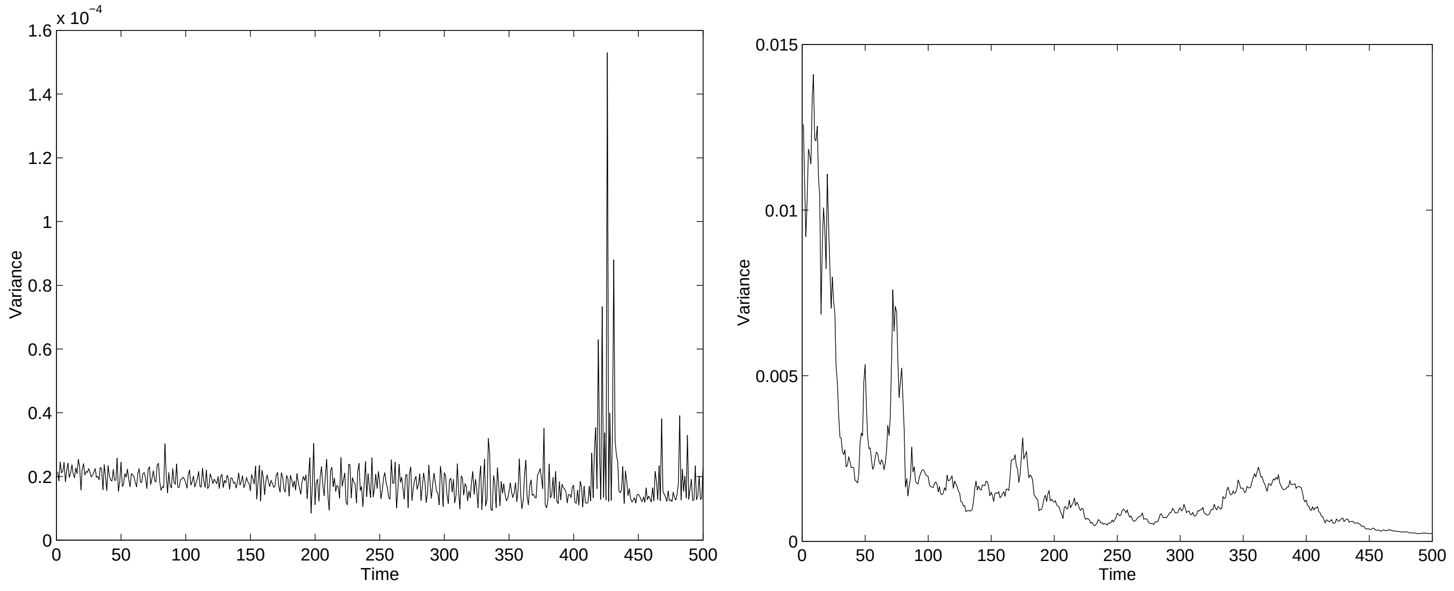

Figure 8: Predicted test data variance for US dollar/Deutsche Mark (left) and US dollar/British pound (right).

|  | CAD | CHF | DEM | GBP | JPY |
|---|---|---|---|---|---|
| ARIMA order | $(5,1,2)$ | $(6,1,1)$ | $(6,1,2)$ | $(9,1,0)$ | $(7,1,2)$ |
| ARIMA variance | $4.90 \times 10^{-6}$ | $1.57 \times 10^{-5}$ | $1.39 \times 10^{-5}$ | $1.32 \times 10^{-4}$ | $9.59 \times 10^{-10}$ |
| MLP variance | $2.76 \times 10^{-5}$ | $9.20 \times 10^{-4}$ | $6.66 \times 10^{-5}$ | $9.20 \times 10^{-4}$ | $4.96 \times 10^{-9}$ |

Table 2: Model structure

not change generalisation performance to any noticeable degree.

## 4 Conclusions

This paper has demonstrated that more complex models for the conditional variance for currency markets can improve generalisation performance. Mixture Density Networks, which in their most general form can model non-Gaussian conditional probability distributions gave the most accurate results. Early stopping had little effect on generalisation performance, which suggests that more principled forms of regularisation may be required. This was why a Bayesian approach, which has been used successfully on other regression problems, was tried on this data, but the results proved to be disappointing. This seemed to be mainly because of the difficulty of fitting the RBF networks: it is likely that the use of non-local basis functions (as in [8]) would improve this.

There is still scope for improving the models that we use for this problem. Some issues that we intend to address are:

- Including moving average terms in the neural network models. The only technical difficulty with this is calculating the relevant partial derivatives efficiently, as the network structure becomes recursive.
- On-line estimation of variance to cope with non-stationary data. In [14, 11] constructive on-line algorithms based on RBFs were used to predict the next price in the Deutsche Mark/French Franc market. These models were able to correct their forecasts after major shocks much better than a range of alternatives. Generalising this to estimating the conditional variance as well would give a better assessment of risk shortly after major changes in market conditions, and would also cope with more gradual shifts in behaviour.
- Developing methods for deciding a good structure for the variance model. When predicting the conditional mean, we can use the ACF and PACF (as for ARIMA modelling) to give some clues to the optimal model structure. We know of no such methods for conditional variance. It is possible that Automatic Relevance Determination (ARD), which is a Bayesian approach that has been used for regression problems [10].

It is also likely that high frequency data would exhibit more 'interesting' (i.e. less Gaussian, with a skew or even multi-modal distribution) conditional densities.

## References

[1] C. M. Bishop. Mixture density networks. Technical Report NCRG/4288, Neural Computing Research Group, Aston University, U.K., 1994.

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

[3] C. M. Bishop and C. S. Qazaz. Bayesian inference of noise levels in regression. In C. von der Malsburg, W. von Seelen, J. C. Vorbruggen, and B. Sendhoff, editors, *ICANN*, volume LNCS 1112, pages 59–64. Springer-Verlag, 1996.

[4] A. N. Burgess and A. N. Refenes. The use of error feedback terms in neural network modelling of financial time series. In C. Dunis, editor, *Forecasting Financial Markets*, chapter 12, pages 261–274. John Wiley, 1996.

[5] G. Cybenko. Approximation by superposition of a sigmoidal function. *Math. Control, Signals and Systems*, 2:303–314, 1989.

[6] K. Funahashi. On the approximate realization of continuous mapping by neural networks. *Neural Networks*, 2:183–192, 1989.

[7] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–355, 1989.

[8] D. Lowe. On the use of nonlocal and non positive definite basis functions in radial basis function networks. In *IEE ANN 1995*, pages 206–211, 1995.

[9] D. J. C. Mackay. A practical Bayesian framework for back-propagation networks. *Neural Computation*, 4:448–472, 1992.

[10] D. J. C. Mackay. Bayesian methods for backpropagation networks. In E. Domany, J. L. van Hemmen, and K. Schulten, editors, *Models of Neural Networks III*, chapter 6. Springer-Verlag, 1994.

[11] Alan McLachlan. Online modelling of time series with resource allocating neural networks. To appear in Proceedings of the 4th IMA Conference on Mathematics in Signal Processing, 1996.

[12] M. Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6:525–533, 1993.

[13] I. T. Nabney, C. Dunis, R. Dallaway, S. Leong, and W. Redshaw. Leading edge forecasting techniques for exchange rate prediction. In C. Dunis, editor, *Forecasting Financial Markets*, chapter 10, pages 227–244. John Wiley, 1996.

[14] I. T. Nabney, A. McLachlan, and D. Lowe. Practical methods of tracking non-stationary time series applied to real world data. In *SPIE Conference on the Applications and Science of Artificial Neural Networks*, pages 152–163, 1996.

[15] D. A. Nix and A. S. Weigend. Learning local error bars for nonlinear regression. In *Advances in Neural Information Processing 7*, pages 486–496. MIT Press, 1995.

[16] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, 2nd edition, 1992.

[17] C. E. Rasmussen. *Evaluation of Gaussian Processes and Other Methods for Non-linear Regression.* PhD thesis, Dept. of Computer Science, University of Toronto, 1996. Available from `http://www.cs.utoronto.ca/~carl/`.

[18] P. M. Williams. Using neural networks to model conditional multivariate densities. *Neural Computation*, 8:843–854, 1996.