Reprint from: BNP Working Paper in Financial Economics Series (3) September 1997

A Trading System for FTSE-100 Futures Using Neural Networks and Wavelets

D L Toulson
S P Toulson

Intelligent Financial Systems Limited
ifs@if5.com
www.if5.com

ABSTRACT

In this paper, we shall examine the combined use of the Discrete Wavelet Transform [7] and regularised neural networks to predict intra-day returns of the LIFFE FTSE-100 index future.

The Discrete Wavelet Transform (DWT) has recently been used extensively in a number of signal processing applications [5, 6]. In this work, we shall propose the use of a specialised neural network architecture (WEAPON) that includes within it a layer of *wavelet neurons*. These wavelet neurons serve to implement an initial wavelet transformation of the input signal, which in this case, will be a set of lagged returns from the FTSE-100 future. We derive a learning rule for the WEAPON architecture that allows the dilations and positions of the wavelet nodes to be determined as part of the standard back-propagation of error algorithm. This ensures that the child wavelets used in the transform are optimal in terms of providing the best discriminatory information for the prediction task.

We then examine how the predictions obtained from *committees* of WEAPON networks may be exploited to establish trading rules for adopting positions in the FTSE-100 Index Future using a Signal Thresholded Trading System (STTS). The STTS operates by combining predictions of the future return estimates of a financial time series over a variety of different prediction horizons. A set of trading rules is then determined that act to optimise the risk adjusted performance (Sharpe Ratio) of the trading strategy using realistic assumptions for bid/ask spread, slippage and transaction costs.

1. INTRODUCTION

Over the past decade, the use of neural networks for financial and econometric applications has been widely researched. In particular, neural networks have been applied to the task of providing forecasts for various financial markets ranging from spot currencies to equity indexes. The implied use of these forecasts is often to develop systems to provide profitable trading recommendations.

However, in practice, the success of neural network trading systems has been somewhat poor. This may be attributed to a number of factors. In particular, we can identify the following weaknesses in many approaches:

1. **Data Pre-processing** – Inputs to the neural network are often simple lagged returns (or even prices!). The dimension of this input information is often much too high in the light of the number of training samples likely to be available. Techniques such as Principal Components Analysis (PCA) and

Discriminant Analysis can often help to reduce the dimension of the input data [11,12]. In this paper, we present an alternate approach using the Discrete Wavelet Transform (DWT).

2. **Model Complexity** – Neural networks are often trained for financial forecasting applications without suitable regularisation techniques. Techniques such as Bayesian Regularisation [2,3,10] or simple weight decay help control the complexity of the mapping performed by the neural network and reduce the effect of *over-fitting* of the training data. This is particularly important in the context of financial forecasting due to the high level of noise present within the data.

3. **Confusion Of Prediction And Trading Performance** – Often researchers present results for financial forecasting in terms of root mean square prediction error or number of accurately forecast *turning points*. Whilst these values contain useful information about the performance of the predictor they do not necessarily imply that a successful trading system may be based upon them. The performance of a trading system is usually dependent on the performance of the predictions at *key points* in the time series. This performance is not usually adequately reflected in the overall performance of the predictor averaged over all points of a large testing period.

We shall present a practical trading model in this paper that attempts to address each of these points.

2. THE PREDICTION MODEL

In this paper, we shall examine the use of committees of neural networks to predict future returns of the FTSE-100 Index Future over 15, 30, 60 and 90 minute prediction horizons. We shall then combine these predictions and determine from them a set of trading rules that will optimise risk adjusted performance (Sharpe Ratio).

We shall use as input to each of the neural network predictors, the previous 240 lagged minutely returns of the FTSE-100 Future. The required output shall be the predicted future return for the appropriate prediction horizon. This process is illustrated in Figure 1.

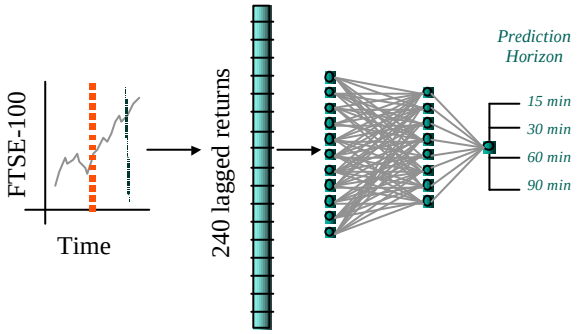

Figure 1: Predicting FTSE-100 Index Futures: 240 lagged returns are extracted form the FTSE-100 future time series. These returns are used as input to (WEAPON) MLPs. Different MLPs are trained to predict the return of the FTSE-100 future 15, 30, 60 and 90 minutes ahead.

A key consideration concerning this type of prediction strategy is how to encode the 240 available lagged returns as a neural network input vector. One possibility would be to simply use all 240 raw inputs. The problem with this approach is the high dimensionality of the input vectors. This will require us to use an extremely large set of training examples to ensure that the parameters of the model (the weights of the neural network) may be properly determined. Due to computational complexities and the non-stationarity of financial time series, using extremely large training sets is seldom practical. A preferable strategy is to attempt to reduce the dimension of the input information.

A popular approach to reducing the dimension of inputs to neural networks is to use a Principal Components Analysis (PCA) transform to reduce information redundancy in the input vectors due to inter-component correlations. However, as we are working with lagged returns from a single financial time series we know, in advance, that there is little (auto) correlation in the lagged returns. In other work [11, 12], we have approached the problem of dimension reduction through the use of Discriminant Analysis techniques. These techniques were shown to lead to significantly improved performance in terms of prediction ability of the trained networks.

However, such techniques do not, in general, take any advantage of our knowledge of the temporal structure of the input components, which in this case will be sequential lagged returns. Such techniques are also implicitly linear in their assumptions of separability, which may not be generally appropriate when considering obtaining an optimal set of inputs to (non-linear) neural networks.

We shall consider, as an alternative means of reducing the dimension of the input vectors, the use of the Discrete Wavelet Transform.

3. THE DISCRETE WAVELET TRANSFORM (DWT)

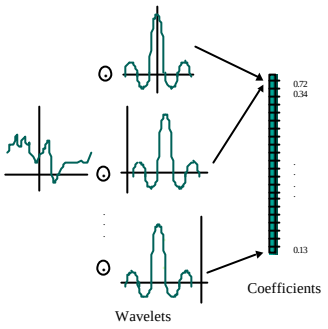

Figure 2: The discrete wavelet transform

The Discrete Wavelet Transform [4, 15] has recently received much attention as a technique for the pre-processing of data in applications involving both the compact representation of the original data (i.e. data compression or factor analysis) or as a discriminatory basis for pattern recognition and regression problems. The transform functions by projecting the original signal onto a sub-space spanned by a set of child wavelets derived from a particular *Mother* wavelet.

For example, let us select the Mother wavelet to be the Mexican Hat function

$$y(t) = \frac{2}{\sqrt{3}} p^{\frac{1}{4}} \left(1 - t^2\right) e^{-\frac{t^2}{2}}. \tag{1}$$

The wavelet *children* are then dilated and translated forms of (1), i.e.

$$f^{t,z}(t) = \frac{1}{\sqrt{z}} f\left(\frac{t - t}{z}\right) \tag{2}$$

Now, let us select a finite subset C from the infinite set of possible child wavelets. Let the members of the subset be identified by the discrete values of position t_i and scale z_i, $i = 1, \ldots, K$, i.e.

$$C = \left\{ t_i, z_i \quad i = 1, \ldots, K \right\} \tag{3}$$

The j^{th} component of the projection of the original signal $\boldsymbol{x}$ onto the K dimensional space spanned by the child wavelets is then

$$y_j = \sum_{i=1}^{N} x_i f^{t_j, z_j}(i) \tag{4}$$

The significant questions to be answered with respect to using the DWT to reduce the dimension of the input vectors to the neural network are firstly how many child wavelets should be used and given that, what values of shift and dilation, t_i and $z_{i,}$, should be chosen?

In this paper, we shall present a method of choosing a suitable set of child wavelets such that the transformation of the original data (the 240 lagged returns) will enhance the non-linear separability of different classes of signal (i.e. future positive and negative returns) whilst significantly reducing its dimension. We show how this may be achieved naturally by implementing the wavelet transform as a set of neurons contained in the first layer of a multi-layer perceptron.

4. THE WAVELET ENCODING A PRIORI ORTHOGONAL NETWORK (WEAPON)

The WEAPON architecture is shown below in Figure 3. The architecture is essentially a Multi Layer Perceptron (MLP) with an additional layer of special *wavelet* nodes. Each of these nodes represents a single child wavelet and its output response is simply the projection of the input vector onto that particular wavelet i.e.

$$y_j = \sum_{i=1}^{N} x_i f^{t_j, z_j}(i) \tag{5}$$

where in this case, y_j is the output of the j^{th} wavelet node, x_i is the i^{th} component of the input vector and t_j and x_j are respectively the shifts and dilations associated with the j^{th} wavelet node.

The scales and shifts for each wavelet node are optimised by including them as parameters within the usual *backprop* training algorithm. We can thus determine the optimal set of scales and shifts appropriate for the prediction task we wish to solve. Details of the derivation of backprop for the wavelet neurons are given in Appendix A.

In addition to this training rule to optimise the shifts and scales of the wavelet nodes, we have also devised mechanisms to control both the orthogonality of the individual wavelets and the regularisation of the complexity of the network mapping as a whole. Details of this are again included in Appendix A.

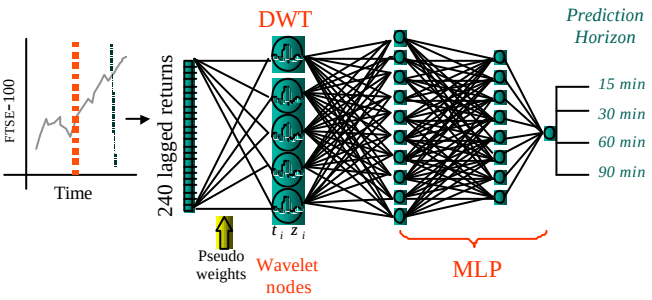

Figure 3: The WEAPON architecture

5. PREDICTING FTSE-100 FUTURES USING WEAPON NETWORKS

5.1. The Data

We shall apply the network architecture and training rules described in the previous section to the task of predicting future returns of the FTSE-100 index futures quoted on LIFFE. The historical price data used was tick-by-tick quotes of actual trades supplied by LIFFE. The data was pre-processed to a 1-minutely format by taking the average volume adjusted traded price during each minute. Missing values were filled in by interpolation but were marked *un-tradable*.

Prices were obtained in this manner for the whole of January 1995- June 1996 to yield approximately 200,000 distinct prices. The entire data set was then divided into three distinct subsets, training/validation, optimisation and test. We trained and validated the neural network models on the first six months of 1995 data. The prediction performance results, quoted in this section, are the results of applying the neural networks to the second six months of the 1995 data. The STTS trading model parameters (described in the next section) were also optimised using this period. We reserved the whole of 1996 for out-of-sample trading performance ***test*** purposes.

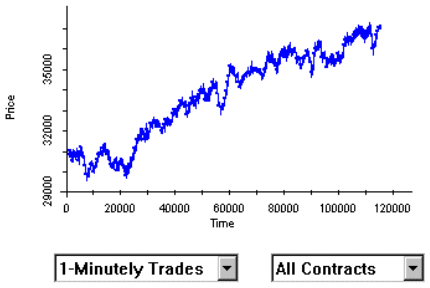

Figure 4: The FTSE-100 future January 1995 to January 1996

5.2. Individual Predictor Performances

Table 1 to 4 show the performances of four different neural network predictors for the four prediction horizons (15, 30, 60 and 90 minute ahead predictions). The predictors used were

1. A simple *early-stopping* MLP trained using all 240 lagged return inputs with an optimised number of hidden nodes found by exhaustive search (2-32 nodes).
2. A standard weight decay MLP trained using all 240 lagged returns with the value of weight decay, lambda, optimised by cross validation.
3. An MLP trained with Laplacian weight decay and weight/node elimination (as in Williams [14])
4. A WEAPON architecture using wavelet nodes, soft orthogonalisation constraints and Laplacian weight decay for weight/node elimination.

The performances of the architectures are shown in terms of

1. RMSE prediction error in terms of desired and actual network outputs
2. Turning point accuracy: This is the number of times the network correctly predicts the *sign* of the future return.
3. *Large* turning point accuracy: This is the number of times that the network correctly predicts the sign of returns whose magnitude is greater than one standard deviation from zero (this measure is relevant in terms of expected trading system performance).

Prediction horizon	% Accuracy	Large % Accuracy	RMSE
15	51.07%	54.77%	0.020231
30	52.04%	59.55%	0.039379
60	51.69%	54.61%	0.074023
90	51.12%	50.82%	0.085858

Table 1: Results for MLP using early stopping

Prediction horizon	% Accuracy	Large % Accuracy	RMSE
15	50.75%	52.70%	0.022533
30	51.00%	56.08%	0.034591
60	53.35%	54.09%	0.060929
90	54.82%	57.24%	0.128560

Table 2: Results for weight decay MLP

Prediction horizon	% Accuracy	Large % Accuracy	RMSE
15	51.25%	48.55%	0.020467
30	54.16%	54.34%	0.035261
60	46.14%	43.48%	0.064493
90	50.39%	50.82%	0.090002

Table 3: Results for Laplacian weight decay MLP

Prediction horizon	% Accuracy	Large % Accuracy	RMSE
15	53.27%	57.43%	0.01879
30	52.79%	56.19%	0.03414
60	54.62%	57.94%	0.06044
90	55.11%	58.28%	0.08118

Table 4: Results for WEAPON

We conclude that the WEAPON architecture and the simple weight decay architecture appear significantly better than the other two techniques. The WEAPON architecture appears to be particularly good at predicting the sign of large market movements.

5.3. Use of Committees for Prediction

In the previous section, we presented prediction performance results using a single WEAPON architecture applied to the four required prediction horizons.

A number of authors have suggested the use of linear combinations of neural networks as a means of improving the robustness of neural networks for forecasting and other tasks. The basic idea of a committee is to independently train a number of neural networks and to then combine their outputs. Suppose we have N trained neural networks and that the output of the i^{th} net is given by $y_i(\underline{x})$. The committee response is given by

$$y(\underline{x}) = \sum_{i=1}^{10} a_i \, y_i(\underline{x}) \quad + \quad a_0 \tag{6}$$

where a_i is the *weighting* for the i^{th} network and a_0 is the *bias* of the committee.

The weightings, a_i may either be simple averages (Basic Ensemble Method) or may be optimised using an OLS procedure (Generalised Ensemble Method). Specifically, the OLS weightings may be determined by

$$\underline{a} = \Xi^{-1}\Gamma \tag{7}$$

where Ξ and Γ are defined in terms of the outputs of the individual trained networks and the training examples, i.e.

$$\Xi = \left[x_{i,j}\right] = \frac{1}{T}\sum_{t=1}^{T} y_i(\underline{x}_t) y_j(\underline{x}_t)$$
$$\underline{\Gamma} = \left[g_i\right] = \frac{1}{T}\sum_{t=1}^{T} y_i(\underline{x}_t) t_t \qquad (8)$$

where $\underline{x}_i$ is the i^{th} input vector, t_i is the corresponding i^{th} target response and T is the number of training examples. Below, we show the prediction performances of committees composed of five independently trained WEAPON architectures, for each of the prediction horizons. We conclude that the performances (in terms of RMSE) are superior to those obtained using a single WEAPON architecture. Turning point detection accuracy however, is broadly similar.

Prediction horizon	% Accuracy	Large % Accuracy	RMSE
15	53.25%	57.27%	0.01734
30	53.14%	56.98%	0.03216
60	54.47%	57.71%	0.05592
90	55.19%	58.69%	0.08091

Table 5: Results for Committees of five independently trained WEAPON architectures.

6. THE SIGNAL THRESHOLDED TRADING SYSTEM

6.1. Background

One might think that if we have a neural network or other prediction model correctly predicting the future direction of a market 60 percent of the time, then it would be relatively straightforward to devise a profitable trading strategy, In fact, this is not necessarily the case. In particular one must consider the following:

- What are the effective transaction costs that are incurred each time we execute a round-trip trade?
- Over what *horizon* are we making the predictions? If the horizon is particularly short term (i.e. 5-minute ahead predictions on intra-day futures markets) is it really possible to get in and out of the market quickly enough and more importantly to get the quoted prices? In terms of building profitable trading systems it may be more effective to have a lower accuracy but longer prediction horizons.
- What level of risk is being assumed by taking the indicated positions? We may, for instance, want to optimise not just pure profit but perhaps some risk-adjusted measure of performance such as Sharpe Ratio or Sterling Ratio.

An acceptable trading system has to take account of some or all of the above considerations.

6.2. The Basic STTS Model

Assume we have *P* predictors making predictions about the expected FTSE-100 Futures returns. Each of the predictors makes predictions for t_i time steps ahead. Let the prediction of the i^{th} predictor at time *t* be denoted by $p_i(t)$. We shall define the *normalised trading signal* $S(t)$ at time *t* to be:

$$S(t) = \sum_{i=1}^{P} \frac{w_i p_i(t)}{t_i} \tag{9}$$

where w_i is the *weighting* given to the i^{th} predictor. An illustration of this is given in Figure 5.

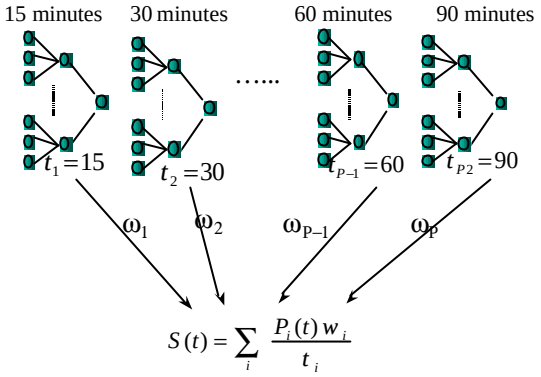

Figure 5: Weighted summation of predictions from four WEAPON committee predictors to give a single trading signal

We shall base the trading strategy on the strength of the combined trading signal $S(t)$ at any given time *t.* At time *t* we compare the trading signal $S(t)$ with two thresholds, denoted by a and b. These two thresholds are used for the following decisions:

- a is the threshold that controls when to open a long or short trade.
- b is the threshold used to decide when to close out an open long or short trade.

At any given time *t*, the trading signal will be compared with the appropriate threshold using the current trading position. In particular, details of the actions defined for each trading position are found in Table 6:

Current position	Test	Action: Go...
Flat	if S(t) > α	Long
Flat	if S(t) < -α	Short
Long	if S(t) < -β	Flat
Short	if S(t) > β	Flat

Table 6: Using the trading thresholds to decide which action to take.

Figure 6 demonstrates the concept of using the two thresholds for trading. The two graphs shown Figure 6 are the trading signals $S(t)$ for each time *t* (top graph) and the associated prices $p_i(t)$ displayed in the bottom graph. The price graph is colour coded for the different trading position that are recommended, blue for a long recommendation, red for a short trading recommendation and grey otherwise. At the beginning of trading we are in a flat position. We shall open a trade if the trading signal exceeds the absolute value of a. At the time marked ❶ this is the case since the trading signal is greater than a. We shall open a long trade. Unless the trading signal falls below $-b$, this long trade will stay open. This condition is fulfilled at the time marked ❷, when we shall close out the long trade. We are now again in a flat position. At time ❸ the trading signal falls below $-a$, so we open a short trading position. This position is not closed out until the trading signal exceeds b, which occurs at time ❹ when the short trade is closed out.

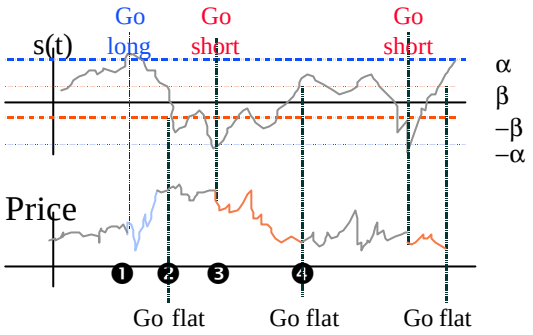

Figure 6: Trading signals and prices

7. RESULTS

An STTS trading system, as described above, was formed using as input 4 WEAPON committee predictors. Each committee contained five independently trained WEAPON networks and was trained to produce 15, 30, 60 and 90-minute ahead predictions, respectively. A screen-shot from the software used to perform this simulation (Amber) is shown below in Figure 7.

The optimal values for the STTS thresholds a and b and the four STTS predictor weightings, w_i to use were found by assessing the performance of the STTS model on the optimisation data (last 6 months of 1995) using particular values for the parameters. The parameters were then optimised using simulated annealing with the objective function being the trading performance on this period measured in terms of Sharpe Ratio.

In terms of trading conditions, it was assumed that there would be a three minute delay in opening or closing any trade and that the combined bid-ask spread / transaction charge for each round trip trade would be 8 points. Both are considered conservative estimates.

After the optimal parameters for the STTS system were determined, the trading system was applied to the **previously unseen** data of the first half of 1996.

Table 7 summarises the trading performance over the six-month test period in terms of over-all profitability, trading frequency and Sharpe Ratio.

Monthly net profitability in ticks	53
Average monthly trading frequency (round-trip)	18
Sharpe ratio daily (monthly)	0.136 (0.481)

Table 7: Results of trading system on the unseen test period.

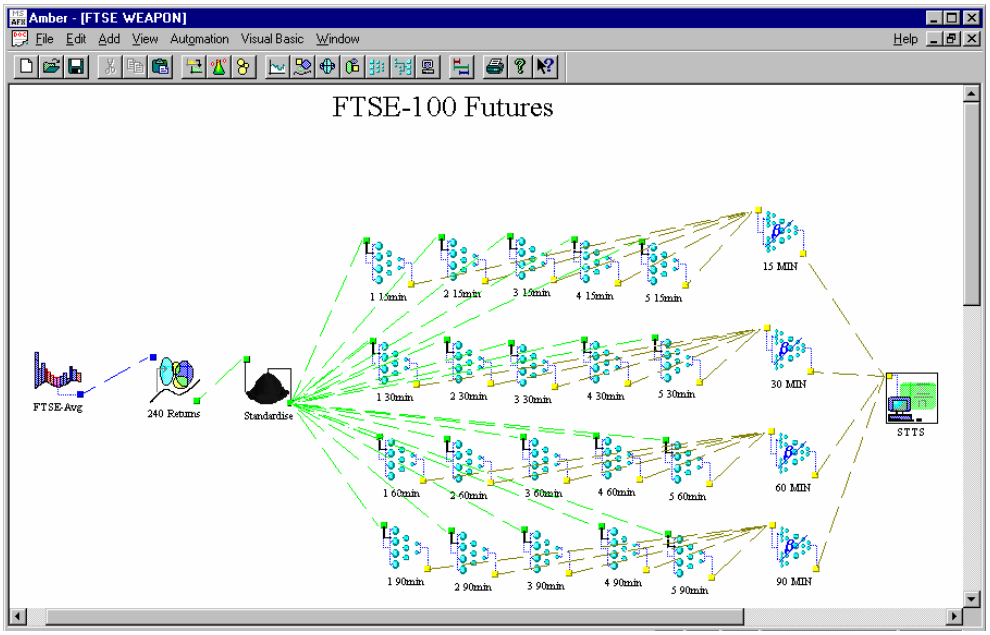

Figure 7: The Trading System for FTSE-100 futures. 240 lagged returns are extracted from the FTSE-100 future time series and after standardisation, input to the 20 WEAPON predictors, arranged in four committes. Each committee is responsible for a particular prediction horizon. The predictions are then combined for each committee and passed onto the STTS trading module.

8. CONCLUSION

We have presented a complete trading model for adopting position in the LIFFE FTSE-100 Future. In particular, we have developed a system that avoids the three weaknesses that we identified in the introduction, namely

1. **Data Pre-Processing** – We have constrained the effective dimension of the 240 lagged returns by imposing a Discrete Wavelet Transform on the input data via the WEAPON architecture. We have also, within the WEAPON architecture devised a method for automatically discovering the optimal number of wavelets to use in the transform and also which scales and dilations should be used.

2. **Regularisation** – We have applied Bayesian regularisation techniques to constrain the complexity of the prediction models. We have demonstrated the requirement for this by comparing the prediction performances of regularised and unregularised (early-stopping) neural network models.

3. **STTS Trading Model** – The STTS model is designed to transform predictions into actual trading strategies. Its objective criterion is therefore not RMS prediction error but the **risk adjusted profit** of trading strategy.

The model has been shown to provide relatively consistent profits in simulated out-of-sample high frequency trading over a 6-month period.

9. BIBLIOGRAPHY

[1] D.E. Rummelhart, G.E. Hinton, R.J. Williams **Learning Internal Representations By Error Propagation.** In **Parallel Distributed Processing.** Chapter 8. MIT Press. 1986

[2] D.J.C MacKay **Bayesian Interpolation.** Neural Comput. 4(3), 415-447, 1992.

[3] D.J.C MacKay **A Practical Bayesian Framework For Backprop Networks.** Neural Comput. 4(3), 448-472,1992.

[4] B.A. Telfer, H. Szu G.J. Dobeck **Time-Frequency, Multiple Aspect Acoustic Classification** World Congress on Neural Networks, Vol.2 pp II-134 – II-139, July 1995.

[5] D.P Casasent J.S Smokelin **Neural Net Design of Macro Gabor Wavelet Filters for Distortion-Invariant Object Detection In Clutter.** Optical Engineering, Vol. 33, No.7, pp 2264-2270 July 1994.

[6] H. Szu B. Telfer **Neural Network Adaptive Filters For Signal Representation.** Optical Engineering 31, 1907-1916, 1992.

[7] I. Debauchies **Orthonormal Bases of Compactly Supported Wavelets** Communications in Pure and Applied Mathematics, 1988 Vol. 61, No. 7, pp 909-996.

[8] K Fukunaga, **Statistical Pattern Recognition** (2nd Edition), Academic Press, 1990.

[9] M.F.Moller **A Scaled Conjugate Gradient Method For Fast Supervised Learning.**

[10] W.L.Buntine A.S.Weigend **Bayesian Back-Propagation** Complex Systems. **5,** 603-643.

[11] D L Toulson, S P Toulson, **Use of Neural Network Ensembles for Portfolio selection and Risk Management**, Proc. Forecasting Financial Markets, Third International conference, London, 1996.

[12] D L Toulson, S P Toulson, **Use of Neural Network Mixture Models for Forecasting and Application to Portfolio Management**, Sixth International Symposium on forecasting, Istanbul, 1996.

[13] S.E. Fahlman. **Faster Learning Variations On Back-Propagation: An Empirical Study.** Proceedings Of The 1988 Connectionist Models Summer School, pp. 38-51. Morgan Kaufmann.

[14] P.M. Williams **Bayesian Regularisation and Pruning Using A Laplace Prior.** Neural Computation, Vol **5,** 1993.

[15] Y Meyer **Wavelets and Operators** Cambridge University Press, 1995.

APPENDIX A– DERIVING BACKPROP FOR WEAPON

The MLP is usually trained using error backpropagation. Backprop requires the calculation of the partial derivatives of the data error E_D with respect to each of the free parameters of the network (usually the weights and biases of the nodes). For the case of wavelet neurons, the weights between the neuron and the input pattern are not free but are constrained to assume discrete values of a particular child wavelet.

The free parameters for the wavelet nodes are therefore not the weights, but the values of translation and dilation t and z. To optimise these parameters during training, we must obtain expressions for the partial derivatives of the error function with respect to these two wavelet parameters.

The usual form of the backpropagation algorithm is:

$$\frac{\partial E}{\partial w_{i,j}} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial w_{i,j}} \tag{10}$$

The term $\frac{\partial E}{\partial y}$, often referred to as d_j, is the standard backpropagation of error term, which may be found in the usual way for the case of the wavelet nodes. The partial derivative $\frac{\partial y}{\partial w_{i,j}}$ must be substituted with the partial derivatives of the node output *y* with respect to the wavelet parameters. For a given mother wavelet $f(x)$, consider the output of the wavelet node, given in Equation (4). Taking partial derivatives with respect to the translation and dilation yields:

$$\begin{aligned}
\frac{\partial y_j}{\partial t_j} &= \frac{\partial}{\partial t_j}\left(\sum_{i=1}^{N} x_i \frac{1}{\sqrt{z_j}} f\left(\frac{i-t_j}{z_j}\right)\right) \\
&= -\sum_{i=1}^{N} x_i \frac{1}{\sqrt{z_j}} f'\left(\frac{i-t_j}{z_j}\right) \\
\frac{\partial y_j}{\partial z_j} &= \frac{\partial}{\partial z_j}\left(\sum_{i=1}^{N} x_i \frac{1}{\sqrt{z_j}} f\left(\frac{i-t_j}{z_j}\right)\right) \\
&= -\frac{1}{2}\sum_{i=1}^{N} x_i \frac{1}{z_j^{\frac{3}{2}}} f\left(\frac{i-t_j}{z_j}\right) - \sum_{i=1}^{N} x_i \frac{(i-t_j)}{z_j^{\frac{5}{2}}} f'\left(\frac{i-t_j}{z_j}\right)
\end{aligned} \tag{11}$$

Orthogonalisation of the Wavelet Nodes

A potential problem with using wavelet nodes is that duplication in the parameters of some of the wavelet nodes may occur.

One way of avoiding this type of duplication would be to apply a soft constraint of orthogonality on the wavelets of the hidden layer. This could be done through use of the addition of the error function

$$E_W^f = \sum_{i=1,\, j\geq i}^{N} \left\langle f^{t_i z_i}, f^{t_{ji} z_{ji}} \right\rangle \tag{12}$$

where $\langle\ \rangle$ denotes the projection

$$\langle f, g \rangle = \sum_{i=-\infty}^{\infty} f(i) g(i) \tag{13}$$

In the previous section, backprop was derived in terms of the unregularised sum of squares data error term, E_D. We now add in an additional term for the orthogonality constraint to yield a combined error function *M(W)*, given by

$$M(W) = aE_D + gE_W^f \tag{14}$$

Weight and Node Elimination

A number of techniques have been suggested in the literature for node and/or weight elimination in neural networks. We shall adopt the technique proposed by Williams [14, 2, 3] and use a Laplacian prior as a natural method of eliminating redundant nodes. The Laplacian Prior on the weights implies an additional term in the error function, i.e.

$$M(W) = aE_D + gE_W^f + bE_W \tag{15}$$

where E_W is defined as

$$E_W = \sum_{i,j} \left| w_{i,j} \right| \tag{16}$$

A consequence of this prior is that during training, weights are forced to adopt one of two positions. A weight can either adopt equal data error sensitivity as all the other weights or is forced to zero. This leads to *skeletonisation* of a network. During this process, weights, hidden nodes or input components may be removed from the architecture. As the weights emerging from redundant wavelet nodes will have negligible data error sensitivity, this will cause them to be eliminated.