Confidence in Data Mining Model Predictions: a Financial Engineering Application

Jerome V. Healy, Maurice Dixon, Brian J. Read and Fang F. Cai, *Member, IEEE*

Abstract— **This paper describes a generally applicable robust method for determining prediction intervals for models derived by non-linear regression. Hypothesis tests for bias are applied. The concept is demonstrated by application to a standard synthetic example, and is then applied to prediction intervals for a financial engineering example viz. option pricing using data from LIFFE for 'ESX' European style options on the FTSE 100 index. Unbiased estimates of the standard error are obtained. The method uses standard regression procedures to determine local error bars and avoids programming special architectures. It is appropriate for target data with non-constant variance.**

Index Terms— **Data mining, Decision support, Financial Engineering, Neural nets.**

I. Introduction

Trading in financial derivatives is an immensely important aspect of financial markets in terms of value and of volume [1], [2]. A key factor affecting the deployment in financial engineering of data mining approaches is the confidence we can place in the reliability of the predictions that are made. Surprisingly, there has been little reported for the confidence factors associated with the modelling of option prices using neural nets, denoted here by NN. Even more surprising is the absence from the literature of standard statistical hypothesis testing [3], comparisons between models are made on measures such as mean-squared-error and R^2 [4], [5]. These are rather insensitive since they consist of a single figure of merit over the full data set. Amilon's Technical Report is an exception to this [6].

In our studies of financial systems we observed that a particular model could be statistically acceptable for a set of data but that if the inputs were partitioned then, for some partitions, it failed the statistical tests. This led us to the conclusion that we needed to consider error estimates on a point by point basis. A prediction interval is a measure of the confidence that we can have in the output value from an unseen input row from a model trained on similar data; it necessarily spans the model error. Non-constant variance is characteristic of the data we consider. Tibshirani [7] has compared some error estimations for neural network models; however these are unsuitable for our work because they involved either fixed variance, or detailed knowledge of the network parameters, or complex bootstrapping. Nix and Weigend [8] developed a special network architecture from which we drew the inspiration for this work.

We present a robust method for determining prediction intervals for models derived by non-linear regression. It is based on simultaneously modelling the conditional mean and variance of the target and uses a standard NN-architecture. We demonstrate proof of concept by successfully applying the method to a synthetic problem defined in [8]. We then create a synthetic option pricing problem with realistic noise. We show that the method works well for this synthetic option pricing problem. We can recover both the option prices and the prediction intervals to a good level of accuracy; indeed the final Phase of refinement is seen to be unnecessary. There is a performance limit in the case of very small residuals.

II. Background

Multivariate linear regression is well characterised. It is well known that under the classical assumptions for least squares regression [9], the variance of the target (dependent) variable is equal to the variance of the noise term for the (normally unknown) true regression function, and the squared residuals for the estimated regression function can be used to estimate the variance of the noise term. Neural nets are a generalization of multivariate non-linear regression which is not so well characterised. Nix and Weigend [8] considered regression using neural nets. They proposed a model for situations where the variance of the error terms is not constant, termed heteroskedasticity in the literature. The model assumed a two parameter distribution of the true errors. They proposed a neural network with two outputs that simultaneously estimated the conditional mean of the specific target and its input dependant variance. The architecture used differed from a standard NN with fully connected hidden layers. It had a special hidden layer for the variance which was connected to both the input layer and the hidden layer for

Manuscript received 15 May, 2003.

F. F. Cai is with Department of Computing, Communications Technology and Mathematics, London Metropolitan University, 100 Minories, London, EC3N 1JY, UK; e-mail: ff.cai@londonmet.ac.uk.

M.Dixon is with Department of Computing, Communications Technology and Mathematics, 31 Jewry Street, London EC3N 2EY, UK. phone: (+44) 207- 320-1386; fax: (+44) 207-320-3009; e-mail: M.Dixon@londonmet.ac.uk

Visitor: BITD, CLRC Rutherford Appleton Laboratory, Chilton, DIDCOT, Oxon. OX11 0QX, UK.

J.V. Healy is with Department of Computing, Communications Technology and Mathematics, London Metropolitan University, 31 Jewry Street, London EC3N 2EY, UK; e-mail: J.Healy@londonmet.ac.uk

B. J. Read is a Visiting Scientist with Department of Computing, Communications Technology and Mathematics, 31 Jewry Street, London EC3N 2EY, UK; e-mail: Brian.Read@bcs.org.uk.

0-7803-7906-3/03/$17.00 ©2003 IEEE.

the conditional mean of the target. They minimised the negative log likelihood cost function. During the iterative search for a minimum of the cost function, the weight update equations (delta terms) are multiplied by the inverse variance in what is effectively a weighted least squares regression. They used the following three Phase training algorithm.

Phase I: Split data into equal sets A and B. Determine conditional mean d^* from A assuming $\sigma^2(\boldsymbol{x})$ is constant. Uses B for validation to determine minimum without over-fitting.

Phase II: Uses Phase I model with frozen net-weights to derive estimates of $\sigma^{*2}(\boldsymbol{x})$ from set B; deploys a special architecture.

Phase III: Resplit the data into two equal sets A' and B'. Train both d^* and estimates of $\sigma^{*2}(\boldsymbol{x})$ on A' until cost function is minimised on B'.

N.B. Set A is used as a training set. Set B is used as a validation set, (i.e. training is terminated when the cost function is minimised on B)

The outcome is a model for both $y(\boldsymbol{x})$, the underlying true regression, and $\sigma^2(\boldsymbol{x})$ the true squared error. Nix and Weigend applied it to a synthetic data example and demonstrated good recovery of the target function and the variance against the criteria of 1) a normalised mean squared error and a mean cost 2) visual inspection of diagrams. Hypothesis testing results were not reported. They then applied the method to laser data. However we have seen no report of this technique being applied to the financial option domain.

Heskes [10] considered prediction intervals for non-linear regression with small datasets and proposed a noise model for which the variance was input dependent. He derived confidence intervals for the error in the estimated regression function using a bootstrap procedure[11] in which a set of regressions is averaged. The data samples are drawn from the training set by random sampling with replacement. He derived prediction intervals for targets corresponding to patterns used for validation but not used in training. A weighting device was used to avoid wasting data. He assumes the regression noise and the residual noise are independent. A negative least likelihood model for residual noise is applied.

Heskes notes that in practice, neural nets produce biased estimates because finite samples tend to over-smooth sharp peaks. He suggests that since we do not know how to handle the resulting bias then attempting confidence limits of better than of order 1/(number of input rows) is too ambitious. (second order terms are of order 1/(number of input rows) $^{3/2}$). Heskes applied the method to a synthetic data problem very similar to [8]. A very small training set of 50 points with 25 bootstrap samples was used. Using 1000 test points, the aim was for 68% within prediction interval while 65% was achieved. Assessment was by commentary on the graphs, no hypothesis testing was done and it was not applied to another case.

LeBaron and Weigend [12] considered regression by bootstrapping neural nets on the very noisy data constituting the volume of transactions on the New York Stock Exchange. They normalised drift by dividing the data by a moving average. The value of their results is that they suggest errors due to differences in architecture are dominated by errors due to use of different training sets (sampling error) for this type of data. The implication is that optimising architectures (i.e. by pruning, adding additional inputs, extra hidden layers etc.) on a single training set is unlikely to improve generalisation performance, and that ensemble techniques (i.e. bootstrapping) which allow quantification and reduction of the sampling error are to be preferred for this type of data.

Prediction intervals are defined as confidence intervals for a prediction $d^*(\boldsymbol{x})$ made from an unseen set of inputs $\boldsymbol{x}$, where there is no corresponding target $d(\boldsymbol{x})$. Consider the following expression

$$[d(\boldsymbol{x}) - d^*(\boldsymbol{x})] = [y(\boldsymbol{x}) - d^*(\boldsymbol{x})] + n(\boldsymbol{x})$$

Prediction intervals are concerned with the left hand side, the difference between the target and the prediction produced by the estimated regression function. Confidence intervals are concerned with the first right hand term, the difference between the true and estimated regression functions. The second right hand term is the (sample dependent) random noise term. Normally, the noise and the true regression function cannot be observed. However, if the variance of the noise is estimated as $\sigma^{*2}(\boldsymbol{x}_i) = s^2{}_i$, using the squared residuals $[d_i - d^*{}_i]^2$, then a (heteroskedasticity consistent) estimate of the prediction interval is given by

$$P \approx d^*(\boldsymbol{x}_i) \pm t_{0.05,(n-k-1)} \sqrt{\frac{n\sigma^{*2}(\boldsymbol{x}_i)}{(n-k-1)}}$$

In this case, k is the applicable degrees of freedom and t is the critical value from Student's t-distribution.

III. Model / Algorithm

A. Desired Properties for Approach to Prediction intervals

There are several properties that we aim for in an approach to prediction intervals. a) The method should be applicable to a range of multivariate non-linear regression techniques. b) The method should be able to achieve a defined level of accuracy; Heskes [10] has indicated that 1/(number of input rows) is the relevant limit. c)The method should be straightforward to implement and should not require a special architecture since it is intended for use in a data-centric framework using commercial neural net tools. d) The method should not be complex or computationally costly e) The method must yield prediction intervals consistent with non-constant noise variance (heteroskedasticity).

The Nix-Weigend [8] method uses a special non-standard architecture requiring special programming. Methods using the inverted Hessian matrix of partial second derivatives of parameters can be problematic because a) many neural net packages do not provide access to the network weights b) inversion can fail if over-fitting occurs c) accuracy of these

methods is relatively poor [7]. The 'naïve' bootstrap does not produce estimated standard errors consistent with non-constant noise variance. Moreover, it is computationally costly and supported by few commercial packages.

B. Modelling Prediction Intervals and Cost Function

For option market participants, prediction intervals and confidence in the predicted values from unseen input rows are of greater practical interest than confidence intervals for the true regression. We therefore introduce a model that calculates the prediction interval directly; we avoid the bootstrapping that is a practical necessity in obtaining the confidence intervals for the true regression. Our model has two outputs; one output is the target variable and the other is the squared error. It differs from the Nix and Weigend [8] model because a) It deploys a standard NN architecture and b) It uses a sum of squares cost function and does not assume a Gaussian noise distribution c) It uses independent training and validation sets, rather than interchanging validation sets.

We follow Heskes [10] who suggests that it is desirable that the training set for fitting the squared errors is independent of the set used for training and validating the model for the target variable. The reason is, since training is stopped on the training set when the sum of squared errors is minimised on the validation set, the two are associated and thus not truly independent.

C. Training Algorithm

1) Phase I:

Randomly split the training data into two data sets, Set A and Set B. Using Set A, train a NN model on the target variable $d(\boldsymbol{x})$. Run the trained NN model on Set B, to obtain a set of squared residuals. By using squared residuals on a test set (Set B) as the second target for Phase II, over fitting and consequent underestimation of the standard error is avoided.

2) Phase II:

Using Set B for training, train a second NN with two output nodes. The target for the first output node is the variable $d(\boldsymbol{x})$; the target for the second output node is the squared residuals obtained in Phase I from set B using the model trained on Set A.

3) Phase III (optional):

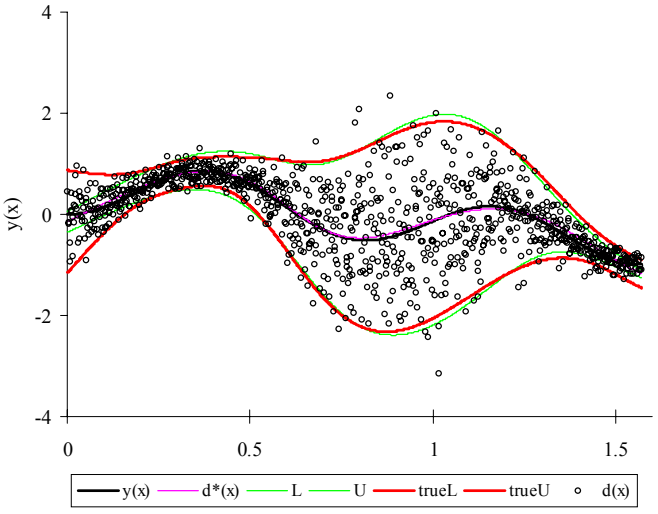
Using Set A for training, train a further NN with two output nodes. The target for the first output node is the variable $d(\boldsymbol{x})$; the target for the second output node is squared residuals for the estimate $d^*(\boldsymbol{x})$ obtained in Phase II using the model trained on Set B. Training is now complete.

Set A is itself randomly split into a training and a validation portion; as is Set B. Testing of each Phase is performed on an independent test set, Set C. Note: their [8] Phase II differs from ours as does their cost function. which involves $\sigma^{*2}(\boldsymbol{x})$ terms.

IV. Synthetic Example

Nix and Weigend [8] defined a univariate synthetic example to demonstrate the effectiveness of their model. For comparison purposes our proposed training algorithm and network is applied to the same univariate synthetic example, called Example #1 in their paper and here. This example uses a one-dimensional data set where $y(\boldsymbol{x})$ the true regression, and $\sigma^2(\boldsymbol{x})$, the variance of the noise, are known. The true regression $y(\boldsymbol{x})$, is given by the equation $y(\boldsymbol{x}) = Sin(3x)Sin(5x)$, where x is a uniformly distributed random number from the interval [0, π/2]. Heskes [10] used similar trigonometric functions for the true regression and the noise variance. The noise $n(\boldsymbol{x})$ consists of numbers from the normal distribution $N[0, \sigma^2(\boldsymbol{x})]$, where $\sigma^2(\boldsymbol{x}) = 0.02 + 0.25[1 - Sin(5x)]^2$. The target value for training is $d(\boldsymbol{x}) = y(\boldsymbol{x}) + n(\boldsymbol{x})$.

We then adopted the following procedure. We used the same number of hidden nodes as Nix and Weigend for regression and for noise variance. For Phase I, a network with a single input node, 10 hidden layer nodes and a single output node is used. Phases II and III use a network with a single input node, 20 hidden layer nodes and 2 output nodes, one for $d(\boldsymbol{x}) = y(\boldsymbol{x}) + n(\boldsymbol{x})$ and one for $\sigma^2(\boldsymbol{x})$. Fig. 1 shows a plot of the data points, the true regression $y(\boldsymbol{x})$, and the approximate prediction band obtained on a test set, for our Phase III model. The effect of Phase III was to tighten the prediction curves into the true model.



Prediction Bands for Example#1

Fig. 1 $y(\boldsymbol{x})$ is the true regression, $d(\boldsymbol{x})$ are the target data points, $d^*(\boldsymbol{x})$ is the estimate of the true regression, L and U are the true upper and lower prediction intervals and L* and U* are the estimated prediction intervals obtained using the network estimate $\sigma^{*2}(\boldsymbol{x})$for the noise variance.

Table I shows the results of statistical tests for the performance of the proposed network in predicting the true regression, the true noise variance function, and the target data points $d(\boldsymbol{x})$ and actual squared errors for Example #1; $d^*(\boldsymbol{x})$ is the estimate of the target. The results show that the network produces unbiased estimates of the mean and the variance of

TABLE I PERFORMANCE OF PROPOSED NETWORK (EXAMPLE#1)

Model	Layers	In	Output	R^2	F-stat		t-stat		t-Test
	[no. of nodes]				F_{crit}(1 tail)	F_{calc}(0.05)	t_{crit}(2 tail)	T_{calc}(2,0.05)	[H0:O=O*(x)]
Estimated v. True Regression Function & Noise Variance Function									
PhaseIII	1-10-2	x	$\mu_y^*(x)$	0.99	1.03	1.01	1.96	-1.12	Unbiased
PhaseIII	1-20-2	x	$\sigma^{*2}(x)$	0.99	0.97	0.86	1.96	-1.96	Unbiased
Estimated v. Actual Target (d) & Squared Errors									
PhaseIII	1-20-2	x	$\mu_y^*(x)$	0.42	1.03	2.40	1.96	-0.91	Unbiased
PhaseIII	1-20-2	x	$\sigma^{*2}(x)$	0.34	1.03	4.38	1.96	0.32	Unbiased

Table I Shows the proposed network produces an unbiased estimate of both the mean and the variance for its two outputs. The estimate of the actual target $d(\boldsymbol{x})$ and the squared errors is also unbiased, and the hypothesis test for the means shows there is no difference at the 95% significance level.

both the true regression function $y(\boldsymbol{x})$, and the noise variance function $\sigma^2(\boldsymbol{x})$.

The approximate upper and lower prediction intervals calculated from the estimated noise variance function are also unbiased estimates of the true upper and lower prediction intervals. For comparison of the proposed network performance with that of the Nix-Weigend network we have computed, and show in Table II, the statistics used by the authors in [8].

TABLE II RESULTS FOR EXAMPLE#1

		This Work		Nix-Weigend	
row		Test Set	($n=10^4$)	Test Set	($n=10^5$)
	Target d	E_{NMS}	Our Mean Cost	E_{NMS}	NW Mean Cost
1	Phase I	0.764	0.454	0.593	0.882
2	Phase II	0.577	0.344	0.593	0.566
3	Phase III	0.578	0.344	0.570	0.462
4	n(x) (exact additive noise)	0.575	0.343	0.563	0.441
	Target e^2	ρ (PIII)	ρ (PII)	ρ (PIII)	ρ (PII)
5	ρ(σ*(x), residual errors)	0.571	0.569	0.548	N/a
6	ρ(σ(x), residual errors)	0.586	0.585	0.584	N/a
	Distribution P(III)	1 std.	2 std.	1 std.	2 std.
7	% of errors < σ*(x); 2σ*(x)	67.4	93.1	67.0	94.6
8	% of errors < σ(x); 2σ(x)	66.9	95.0	68.4	95.4
9	(exact Gaussian)	68.3	95.4	68.3	95.4

In Table II E_{NMS} is the mean squared error normalised by the (global) variance of the target d. Our Mean Cost is the mean of the cost function $(d-d^*)^2$. Row 4 gives these figures for $[(d-y(x))^2 = n(x)^2]$ and represents the best performance attainable. Row 5 gives the correlations between the absolute errors and the network estimate for the standard deviation of the errors. Row 6 gives the correlations between the absolute errors and the true noise standard deviation. Row 7 gives the percentage of absolute errors that are less than 1 and 2 times the corresponding network estimate for the standard deviation of the error. Row 8 gives the percentage of absolute errors that are less than 1 and 2 times the corresponding true noise standard deviation. Row 9, which is included for comparison purposes, gives the percentage of observations that are less than 1 and 2 standard deviations in a Gaussian distribution.

Table II shows that compared to the Nix-Weigend network there is little improvement in the fit to the target d between our Phases II and III. However the Phase III fit (row 3) for the proposed network is close to the best attainable, deviating only by 0.6%. The Nix-Weigend network on the other hand does not approach the best attainable figure quite so closely, deviating by 1.4%. The proposed network figures for correlation of the actual absolute errors with the network prediction and the true values (rows 5 and 6), is slightly better than the corresponding figures for the Nix-Weigend network, even in Phase II. The distributions of errors reported in rows 7 and 8 differ only slightly from those for the Nix-Weigend method.

Overall, on the basis of the results in Table II the proposed network performs comparably with a Nix-Weigend network and actually outperforms it slightly on the errors and correlations. The results in Table II however, are not based on formal statistical tests. For this reason we prefer to rely on the results presented in Table 1. The results of F and t tests presented in Table I show that the proposed network can provide unbiased estimates of an underlying true regression function, an associated noise variance function, and also the actual target and squared errors in the univariate case.

V. Option Pricing Example

A. The Data

The option market data were from LIFFE [1]. They consist of daily closing prices for the FTSE-100 index call option for all trading dates from 13 March 1992 to 1 April 1997. The data set contains 119,413 records. The data were cleaned to exclude options with invalid or missing parameters. Only options which had actually been traded, as indicated by positive values of bid, offer, spread, trading volume, and open interest (number of unclosed transactions) were used. To exclude mis-priced options only those with moneyness between 0.8 and 1.4, and at-the-money implied volatilities (as tabulated by LIFFE) less than 40% were included. The cleaned data set comprised 14,254 records. This data set was randomly split into a training set and a test set. The resulting training sets contained 7,083 records with 3629 in Set A and 3454 in Set B. 50% of these were randomly sampled and used for validation. The test set contained 7,171 records.

B. Tests with Synthetic Prices and Noise

In this section the proposed network is applied in the more realistic multivariate setting of option pricing. Our approach was tested using synthetic option prices and synthetic noise, labelled Example#2. For this purpose we follow Hutchinson et al [13] and omit the volatility and risk-free interest rate from the standard Black-Scholes inputs. The synthetic option prices were created using a trained NN option pricing model as the

underlying known regression function. The NN was trained using observed market prices as the target and the variables moneyness, ($m = S/X$), and time to maturity, t, as inputs; S represents the price of the asset in index points and X is the strike price for the option. Analysis of squared residual errors for NN option pricing models indicated that $\sigma^2(t,m) = 510t^4 + 361m^{17}$ was a suitable choice for approximating the underlying residuals to give a known noise variance function; it is important to emphasise that this function has no significance other than as a residual model. Using this function, a closely similar shaped distribution, as indicated by variance, standard deviation, skewness, and kurtosis, is obtained for the absolute values of the synthetic noise generated, and for those of real residual errors for NN option pricing models. Moreover, t-tests indicate that the obtained target $d(t,m)$ is not significantly different from observed market prices. Artificial noise from the normal distribution N was drawn as $N(0,\sigma^2(t,m))$ and added to the outputs of the trained NN to generate a synthetic target option price $d(t,m)$. As in Example#1 the aim is to determine whether the method can successfully recover an underlying known regression function and noise variance function. The results for Example#2 are presented in Table III.

TABLE III: RESULTS FOR EXAMPLE#2

row		Test Set	
	Target d = C_{NN} + noise	E_{NMS}	Our Mean Cost
1	Phase I	0.059	564.660
2	Phase II	0.062	599.406
3	Phase III	0.059	566.805
4	n(x) (exact additive noise)	0.058	556.988
	Target e^2	ρ (PIII)	ρ (PII)
5	ρ(σ*(x), residual errors)	0.537	0.536
6	ρ(σ(x), residual errors)	0.562	0.591
	Distribution (PII)	1 std.	2 std.
7	% of errors < σ*(x); 2σ*(x)	51.3%	81.4%
8	% of errors < σ(x); 2σ(x)	70.0%	96.1%
	Distribution (PIII)		
9	% of errors < σ*(x); 2σ*(x)	32.5%	60.2%
10	% of errors < σ(x); 2σ(x)	71.4%	96.2%
11	(exact Gaussian)	68.3%	95.4%

In Table III E_{NMS} is the mean squared error normalised by the (global) variance of the target d. Our Mean Cost is the mean of cost function $(d-d^*)^2$. Row 4 gives these figures for $[(d-y(\mathbf{x}))^2 = n(\mathbf{x})^2]$ and represents the best performance attainable. Row 5 gives the correlations between the absolute errors and the network estimate for the standard deviation of the errors. Row 6 gives the correlations between the absolute errors and the true noise standard deviation. Row 7 gives the percentage of absolute errors that are less than 1 and 2 times the corresponding network estimate for the standard deviation of the error. Row 8 gives the percentage of absolute errors that are less than 1 and 2 times the corresponding true noise standard deviation. Rows 9 and 10 report these results for PhaseIII. Row 11, which is included for comparison purposes, gives the percentage of observations that are less than 1 and 2 standard deviations in a Gaussian distribution. Here (**x**) represents the two input variables (t,m)

In Table III The Phase III fit for the target $d(t,m)$, (Row 3) is only 0.001 (1.7%) more than the lowest attainable value (Row 4). The Phase II fit (Row 2), is not quite as good but is still only 0.004 (7%) more than the lowest attainable value. Although the difference is not statistically significant at the 95% level as shown by the t-tests, the Phase II fit to E_{NMS} is slightly poorer than the Phase I fit (Row 1). This is because, in contrast to [8], our Phase II involves training a new model constrained to fit both the target $d(t,m)$ and the squared residuals from Phase I. Like [8] we use 10 hidden layer nodes per output node, but in our method these are in a single hidden layer of 20 nodes with full connectivity to all input and output nodes. Pruning runs, not reported here, indicate fewer nodes can achieve the relevant accuracy.

As in Example#1 the correlation of the absolute errors with the estimated absolute errors and with the true noise standard deviation (Rows 5 and 6) improves slightly from Phase II to Phase III. The correlation results in Table III are of a similar order to those for Example#1 in Table II. Row 7 and Row 9 distribution results show the dispersion of the actually occurring absolute errors is greater than the estimated and true noise standard deviation results given in Row 8 and Row 10 indicate. The decreased correlation of absolute values of residual errors with the known noise standard deviation (Row 6) suggests that Phase III training should be omitted in the more realistic multivariate setting for this data.

This conclusion is supported by Table IV which gives the results of hypothesis tests for Example#2. They show that the (proposed network) estimate for the mean of the target $d(\mathbf{x})$ is unbiased for all three training phases. The Phase II estimate of the mean of the known noise variance function and the actual squared errors is unbiased as indicated by the hypothesis test results. However Phase III estimates of both are biased, unlike the Example#1 results. The F statistics indicate there is no difference between the estimated and actual variance of the known noise variance function for both PhaseII and PhaseIII.

The Table III & IV results for PhaseII training in Example#2 show the proposed network produces an unbiased estimate of the input dependent noise variance function $\sigma^2(t,m)$, which is a smooth function of time to maturity t, and moneyness m. Moreover, this is an unbiased estimate of the actually occurring squared errors for the estimate of the target. These results suggest that given a set of unseen input variables for which there is no corresponding targets, the proposed network is capable of producing an unbiased estimate of a target $d(t,m)$, and a corresponding (unknown) noise variance function. The unbiased estimate of both the mean of the target and the true noise variance function suggest that prediction intervals upon them will be a good estimate of the 95% prediction intervals.

VI. Conclusions

This work has shown that it is possible to implement a robust method for computing statistical prediction intervals generalised for non-linear regression. The method has been applied successfully to a standard synthetic set of data and gave statistically acceptable results. The method avoids the need for a special neural net architecture or the calculation of

TABLE IV PERFORMANCE OF PROPOSED NETWORK (EXAMPLE#2)

Phase	Layers	Input	Output	R^2	Adj.R^2	F_Statistic	t-statistic			t-test
	[no. nodes]					F_{crit}(1 tail)	F_{calc}(0.05)	t_{crit}(2 tail)	t_{calc}(0.05)	[H0:O=O*]
Estimated Regression Function										
Phase I	2-10-1	t,m	μ_y*(d)	0.942	0.967	1.04	1.08	1.96	-0.02	Unbiased
Estimated v. Actual Target (d) & True Noise Variance Function										
Phase II	2-20-2	t,m	μ_y*(d)	0.939	0.964	0.96	0.98	1.96	-1.62	Unbiased
Phase II	2-20-2	t,m	σ^{*2}(d)	0.845	0.557	0.96	0.45	1.96	1.52	Unbiased
Phase III	2-20-2	t,m	μ_y*(d)	0.942	0.966	1.04	1.08	1.96	-0.61	Unbiased
Phase III	2-20-2	t,m	σ^{*2}(d)	0.900	0.855	0.96	0.75	1.96	6.20	Biased
Estimated v. Actual Squared Errors										
Phase II	2-20-2	t,m	σ^{*2}(d)	0.246	-0.139	0.96	0.77	1.96	-1.16	Unbiased
Phase III	2-20-2	t,m	σ^{*2}(d)	0.219	0.118	1.04	1.51	1.96	4.39	Biased

Table IV shows the proposed network produces an unbiased estimate, μ_y*(d), of the mean of the target d in all three training phases. The estimated variance σ^{*2}(d) is also unbiased estimate of both the true noise variance and the actual squared error for PhaseII; the corresponding Phase III estimates are biased.

gradients of net weights so is suitable for use in standard developments.

A synthetic option pricing test was constructed and the true noise variance function recovered. These tests show that the method is appropriate for determining prediction intervals for target data with non-constant variance. The use of standard hypothesis tests allows us to detect and reject apparent improvements achieved as a consequence of distorting the variance. In work not reported here we find there appears to be a lower limit to what we can expect to recover when there is a very good fit as indicated by the adjusted R^2 and small mean square residuals

REFERENCES

[1] London International Finanical Futures and Options Exchange http://www.liffe.com/liffedata/contracts/month_onmonth.xls

[2] J. Benell and C. Sutcliffe, "Black-Scholes Versus Artificial Neural Networks in Pricing FTSE 100 Options", *Discussion Paper 00-156*, School of Management, Southampton University, 2000

[3] C Sutclife, Private Communication. January 2003. We are grateful to Professor Sutcliffe for confirming this observation.

[4] R. Herrmann and A. Narr, "Risk Neurality." *Risk, 10, Technology Supplement*, 1997, pp. 23-29.

[5] J.V. Healy, M. Dixon, B.J. Read, and F.F. Cai,. "A Data-centric Approach to Understanding the Pricing of Financial Options". *European Physics Journal B 27*, 2002 pp. 219-227.

[6] H. Amilon . "A Neural Network Versus Black-Scholes: A Comparison of Pricing and Hedging Performances" *Working Paper* Department of Economics, Lund University, Lund, Sweden, 2001.

[7] R. Tibshirani, "A comparison of some error estimates for neural network models". *Neural Computation*, 8, 1996, pp. 152-163.

[8] D.A. Nix, and A.S. Weigend,. "Learning Local Error Bars for Non-Linear Regression", In: *Proceedings of NIPS 7*, 1995, pp. 489-496

[9] R.L. Thomas, *Modern Econometrics*, Addison Wesley, 1997.

[10] T. Heskes. "Practical Confidence and Prediction Intervals", "*Advances in Neural Information Processing Systems 9*", eds : M.Mozer, M.Jordan & T.Petsche, MIT Press, Cambridge, MA, 1997, pp. 176-182.

[11] B. Efron and R.J. Tibshirani, "*An Introduction to the Bootstrap",* Chapman & Hall, New York, 1993.

[12] A. LeBaron and A. Weigend,. "Evaluating Neural Network Predictors by Bootstrapping", In: *Proceedings of the International Conference on Neural Information Processing (ICONIP'94)*, Seoul,1994.

[13] J. Hutchinson, A. Lo, and T. Poggio. "A Non-parametric Approach to Pricing and Hedging Derivative Securities via Learning Networks." *Journal of Finance, 49, No. 3*, 1994, pp. 851-889.

ACKNOWLEDGMENT

This work has benefited from collaboration with the Data Mining programme in the Business Information Technology Department at the Rutherford Appleton Laboratory.