Alternative Neural Network Approach for Option Pricing and Hedging

Andrew Carverhill

School of Business, The University of Hong Kong, Pokfulam Road, Hong Kong

Terry H.F. Cheuk

School of Business, The University of Hong Kong, Pokfulam Road, Hong Kong

Abstract

Since its introduction in 1973, the Black-Scholes model has found increasingly more resistance in application. In order to use Black-Scholes to price any option, one needs to know the implied volatility surface. The existence of such surface is an evidence of misspecification of the model. In this case, the problem is with the assumption of a geometric Brownian motion for the stock price process. There is strong empirical evidence that stocks do not follow such process. However, no agreement has been reached on what is the best distribution to use.

Neural Network approaches the problem very differently. Essentially, a Neural Network is a non-parametric estimation technique. It does not make any distributional assumption regarding the underlying variable. Instead, it puts up a formula with a set of unknown parameters and let the optimization routine search for the parameters best fitted to the desired results. Hutchinson-Lo-Poggio (1994) showed that it is indeed possible to use a Neural Network to price S&P futures options. In this paper, we will continue with this line of research. Specifically, we will examine the best way to set up and train a Neural Network for option pricing and hedging. We will also investigate if a Neural Network could produce better hedging parameters than the standard option pricing model. We use S&P futures options data covering the period 1990–2000.

Key words: Derivatives, Options, Hedging, Neural Network, Non-Parametric Estimation

Email addresses: `carverhill@business.hku.hk` (Andrew Carverhill), `terrycheuk@business.hku.hk` (Terry H.F. Cheuk).

Preliminary version: December 2003

1 Introduction

Neural Network has proved to be useful in a large variety of financial applications. For instance, Altman-Marco-Varetto (1994) discussed an experiment by a large Italian commercial bank using Neural Network to assign credit ratings to their corporate customers, while Trippi-Turban (1996) described other financial applications of Neural Networks. There is also interest in applying this technique to derivative pricing and hedging. Hutchinson-Lo-Poggio (1994) showed Neural Networks can be used to reproduce S&P futures options prices and confirm it can produce better hedge than Black-Scholes, Garcia-Gençay (2000) exploited the homogeneity property of option models to improve the performance of Neural Networks, while Meissner-Kawano (2001) found replacing the implied volatility with GARCH volatility estimate improves the pricing performance of Neural Networks.

Since its introduction in 1973, the Black-Scholes model has found increasingly more resistance in application. In order to use Black-Scholes to price any option, one needs to know what implied volatility to use. Unfortunately, the implied volatility is not unique per stock as assumed by the model, but it is a function of option expiry and moneyness. That is, there is an implied volatility surface per stock. The existence of such surface is an evidence of misspecification of the model. In this case, the problem is with the assumption of a geometric Brownian motion for the stock price process. There is strong empirical evidence that stocks do not follow such process. However, no agreement has been reached on what is the best distribution to use.

Neural Network approaches the option pricing and hedging problem very differently. It does not make any assumption regarding the stochastic process governing the underlying stock movements. Instead, it puts up a framework with a set of unknown parameters and let the optimization routine search for the parameters best fitted to the desired results. This process is often called "learning" or "training". In this paper, we will examine the best way to set up and train a Neural Network for option pricing and hedging. A draw back of such approach is that the resulting framework is hard to interpret, and therefore Neural Network is often criticized as a black box method.

In reality, a Neural Network approach is not different than any other non-parametric approach to statistical data modeling.[1] In both cases, you are trying to limit the number of assumptions you made to the absolute minimum. Then you apply an optimization procedure to find the best parameters to reproduce the results. Hutchinson-Lo-Poggio (1994) examined three types of Neural Networks for option pricing and hedging, including a Multilayer

[1] See Ripley (1996) for a general discussion of Neural Networks and its relationship to the mainstream non-parametric estimation methods.

Perceptron. They trained their Networks with option prices, while the hedge parameters were derived from the resulting Neural Network pricing formula by taking partial derivatives. [2]

Our paper is an extension of the line of research followed by Hutchinson-Lo-Poggio (1994). We will show in this paper that training a Neural Network to observed option prices and subsequently deriving the hedge ratios from the resulting pricing equation is not the best strategy. Instead, it is better to train a Network with observed option price “changes”.

The rest of the paper is organized as follows. In section 2, we provide a brief review of Neural Network and optimization method, and develop a Neural Network of option price, similar to the one examined in Hutchinson-Lo-Poggio (1994). An alternative Neural Network based on option price “changes” is proposed in section 3, which produces the desired hedge parameters directly without going through an option pricing formula. We will put these two Neural Networks to a test using S&P futures options data for the period January 1990 to December 2000. Results are discussed in section 4. Section 5 contains the conclusion.

2 Neural Network of price

As Multilayer Perceptron is by far the most popular type of Neural Network, we will limit ourselves to such type of Network in this paper. However, results from this research is not unique to this particular type of Network. It should also apply for other Networks.

Neural Network and other types of Neural Networks can be seen as methods for non-linear multiple regression analysis. A number of inputs are fed into the system and one output is produced. A Neural Network could produce more than one output, just like a system of simultaneous regression equations could. There are various ways to set up a Neural Network. Our discussion is centered on one-hidden-layer Multilayer Perceptron (MLP), as it is by far the most applied in practice and in research. MLP is also known as Backpropagation Network. Actually this name is linked to the optimization technique used in the very first Neural Network. Backpropagation is basically a gradient decent method. But one is free to choose another optimization technique in a MLP. Hornik-Stinchcombe-White (1989) showed that that a one-hidden-layer MLP could approximate a large class of linear and nonlinear functions with arbitrary precision. Therefore, our set up does not constitute a restriction.

[2] Choi-Marcozzi (2001), Garcia-Gençay (2000), and Meissner-Kawano (2001) also trained their Neural Networks with option prices.

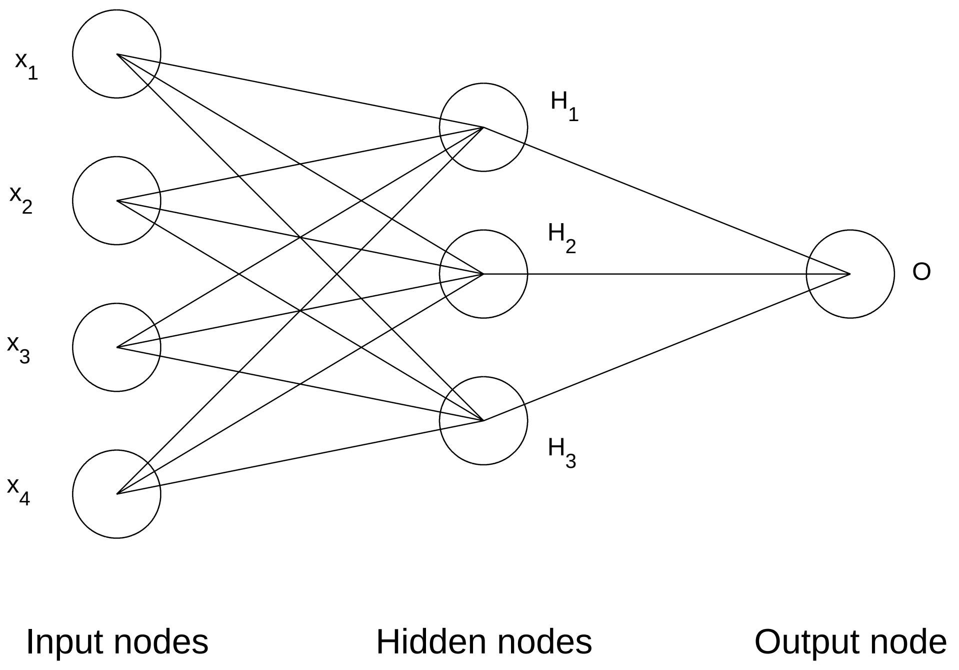

Fig. 1. Multilayer Perceptron of price

Figure 1 describes the MLP model with the option price as output. There are four inputs. They are strike/price, interest rate, implied volatility, and expiry. Note, we do not use both the strike and the underlying price as inputs, but use the ratio strike/price instead. As option price is homogenous of degree one with respect to the underlying asset price and the strike, we can scale the model by the price and save one input parameter. This improves estimation performance. We have tried different number of hidden nodes in our research. From our experience, three hidden nodes are performing quite well already. More hidden nodes will slow down the optimization without significant error reduction. In this Network, there is only one output node, which is the option price.

Input nodes do not perform any processing. All the inputs will be fed into the hidden nodes, which are processed by the so-called sigmoid function f(x):

$$f(x) = \frac{1}{1 + \exp(g(x))} \tag{1}$$

with

$$g(x) = \sum_{i=1}^{n} w_i x_i + w_0 \tag{2}$$

Here, x is the vector of x_i, with x_i denoting the input variables, while w_i the weights. The sigmoid function in the hidden nodes and the output node are the same. This sigmoid function is standard in Neural Network models, although alternative sigmoid functions are occasionally being used.[3] For the hidden nodes, we have $n = 4$, as there are four inputs coming from the input nodes. The sigmoid function at the output node has three inputs only, that is $n = 3$, as there are three hidden nodes feeding into the output node.

Training a Neural Network is to minimize the regression error, by choosing a set of weights in the hidden nodes and the output node. We define the regression error as the sum of squared error between model option prices produced by the Network and the observed option prices. Historically, Multilayer Perceptron are solved using gradient decent as this is the optimization being used in the very first Neural Network. Many of today's Neural Network systems are still using this method. We decided to use a more advanced optimization technique called Levenberg-Marquardt, which is developed by Marquardt (1963). This method is similar to gradient decent when it is far away from the optimum, but it utilizes the inverse of the Hessian whenever the optimizer came near to the optimum, which results in a faster convergence.[4] A Neural Network has many parameters, so that the Hessian matrix does not necessarily has the full rang. In our implementation we used Singular Value Decomposition (SVD) to invert the Hessian matrix. SVD is known to work with ill-conditioned matrices.[5] Once the model has been trained with actual option prices, hedge parameters can be derived from the sigmoid function at the output node, by taking the partial derivatives, as the output node is the Neural Network option pricing equation.

The formulas of this Network and the derivation of delta and vega are given below.

We number all the weights in all the hidden and output nodes consecutively, so that it fit into a vector, it will facilitate the implementation of the optimization routine. We now only need to optimize over one vector, in stead of many different vectors or variables. $p0 - p4$ are the weights in the first hidden node, H_1, $p5 - p9$ belongs to second hidden node, H_2, $p10 - p14$ belongs to the third hidden node, H_3, and $p15 - p18$ are from the output node, O. $x1 - x4$ are the four inputs strike/futures, interest rate, implied volatility, and expiry.

The value of $H1$ to $H3$ is given by the sigmoid function:

$$H_1 = 1/(1 + A) \tag{3}$$
$$H_2 = 1/(1 + B) \tag{4}$$

[3] A sigmoid function increases monotonically from 0 to 1, over entire real line.

[4] The Hessian is the matrix of second order partial derivatives.

[5] See Press et al. (2002) for an implementation of Levenberg-Marquardt and Singular Value Decomposition.

$$H_3 = 1/(1+C) \tag{5}$$

with,

$$A = \exp(-p0 - p1\,x1 - p2\,x2 - p3\,x3 - p4\,x4) \tag{6}$$
$$B = \exp(-p5 - p6\,x1 - p7\,x2 - p8\,x3 - p9\,x4) \tag{7}$$
$$C = \exp(-p10 - p11\,x1 - p12\,x2 - p13\,x3 - p14\,x4) \tag{8}$$
$$\tag{9}$$

The value of output node O s given by the sigmoid function applied on the values at from H_1, H_2, and H_3:

$$O = 1/(1+D) \tag{10}$$

with,

$$D = \exp\left(-p15 - p16\,H_1 - p17\,H_2 - p18\,H_3\right) \tag{11}$$

The option price $\hat{P}$ is given by the output node O:

$$\hat{P} = OX \tag{12}$$

with X the strike.

Taking the partial derivative of the model option price function with respect to the futures price, we have equation for delta:

$$\hat{D} = \frac{D\left(\frac{A\,p1\,p16}{(1+A)^2} + \frac{B\,p6\,p17}{(1+B)^2} + \frac{C\,p11\,p18}{(1+C)^2}\right)}{(1+D)^2} \tag{13}$$

with F being the futures price.

Similarly, taking the partial derivative with respect to implied volatility gives the equation for vega:

$$\hat{V} = \frac{D\,X\left(\frac{A\,p3\,p16}{(1+A)^2} + \frac{B\,p8\,p17}{(1+B)^2} + \frac{C\,p13\,p18}{(1+C)^2}\right)}{(1+D)^2} \tag{14}$$

There are 19 parameters for this model, $p0 - p18$.

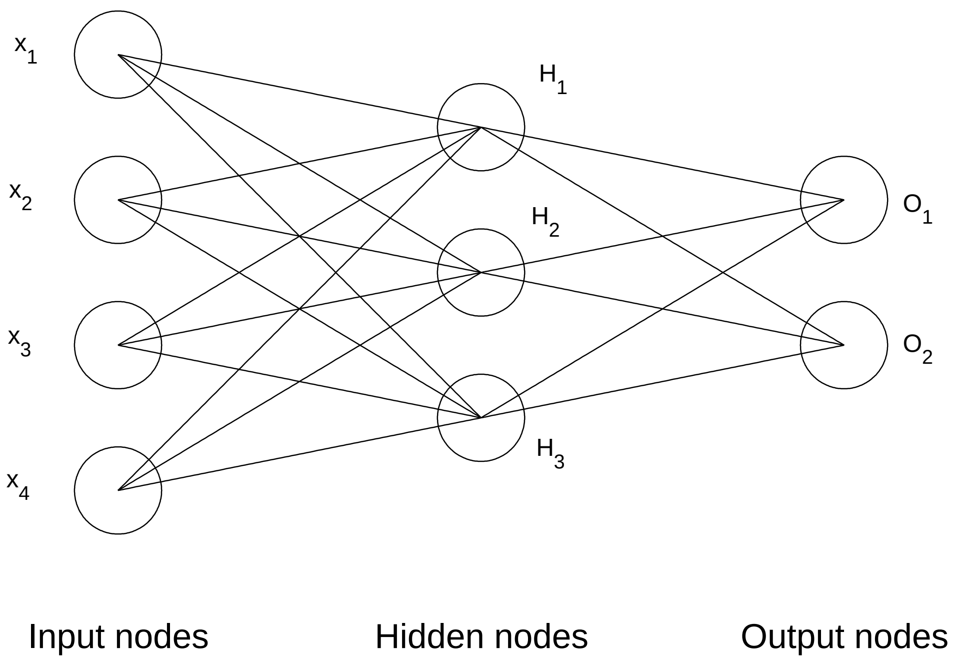

Fig. 2. Multilayer Perceptron of delta/vega

3 Neural Network of delta and vega

Although it is possible to derive the hedge parameters analytically from the sigmoid function at the output node from the previous setup, there is no guarantee that the resulting hedge ratios are reasonable. In stead of having one output node for the price, we are proposing to have two nodes, one for the delta and one for the vega. Delta and vega are the most important hedge ratios for options. In practice, traders are monitoring these two hedging parameters for their option books closely.

Figure 2 described the new setup for the Neural Network. Everything is the same as in the previous case, except there are now two output nodes (delta and vega), in stead of one (price). The training of this Network is a bit more involved as we can not minimize the model delta and vega with the observed delta and vega — hedge parameters are not observable. However, as hedge ratios predict price changes, they can be derived from option price movements.

$$\Delta P_t = \hat{D}_t \ \Delta S_t + \hat{V}_t \ \Delta IV_t + \varepsilon_t \tag{15}$$

where $\Delta P_t = P_{t+1} - P_t$ is the change in option price, $\Delta S_t = S_{t+1} - S_t$ is the futures price change, and $\Delta IV_t = IV_{t+1} - IV_t$ is the change in implied volatility for the at-the-money option, at time t. Implied volatility are can be calculated using Black-Scholes model for European Options or Cox-Ross-Rubinstein (1979) model for American options. $\hat{D}_t$ and $\hat{V}_t$ are the delta and vega hedge parameter estimated using data available at time t. ε_t is residual term, which captures all the factors not included in the model. The model is always using information observable at any point in time to forecast the price movement in the next period.

The formulas for the Neural Network of delta and vega is presented below.

Again, we number all the weights consecutively, so that it fit into a vector, which facilitates the implementation of the optimization routine. $p0 - p4$ are the weights in the first hidden node, H_1, $p5 - p9$ belongs to second hidden node, H_2, $p10 - p14$ belongs to the third hidden node, H_3, while $p15 - p18$ are from first output node, O_1, and $p19 - p22$ from second output node, O_2. $x1 - x4$ are the four inputs strike/futures, interest rate, implied volatility, and expiry.

The value of H_1 to H_3 is the same as in the previous model:

$$
\begin{aligned}
H_1 &= 1/(1+A) && (16)\\
H_2 &= 1/(1+B) && (17)\\
H_3 &= 1/(1+C) && (18)
\end{aligned}
$$

with,

$$
\begin{aligned}
A &= \exp(-p0 - p1\,x1 - p2\,x2 - p3\,x3 - p4\,x4) && (19)\\
B &= \exp(-p5 - p6\,x1 - p7\,x2 - p8\,x3 - p9\,x4) && (20)\\
C &= \exp(-p10 - p11\,x1 - p12\,x2 - p13\,x3 - p14\,x4) && (21)\\
& && (22)
\end{aligned}
$$

There are two output nodes. The value of output nodes O_1 and O_2 are given by the sigmoid function applied on the values at from H_1, H_2, and H_3:

$$
\begin{aligned}
O_1 &= S/(1+D) && (23)\\
O_2 &= X/(1+E) && (24)
\end{aligned}
$$

with,

$$
\begin{aligned}
D &= \exp\left(-p15 - p16\,H_1 - p17\,H_2 - p18\,H_3\right) && (25)\\
E &= \exp\left(-p19 - p20\,H_1 - p21\,H_2 - p22\,H_3\right) && (26)
\end{aligned}
$$

O_1 is the delta, while O_2 is the vega:

$$\hat{D} = O_1 \tag{27}$$
$$\hat{V} = O_2 \tag{28}$$

S is the sign for the delta function, it is 1 for a call, and -1 for a put. X is the strike level. It is added here for the scaling. As the model uses futures/strike as an input, the resulting vega need to be re-scaled back to the strike level. There are 23 parameters, $p0 - p22$. That is, four more parameters than the previous model, as there are two output functions.

4 Applied to S&P futures options data

We use the CME futures and associated options on the S&P500 index. These futures contracts trade on a cycle with maturities in March, June, September and December, and each contract matures on the third Friday of the month. Each futures contract is associated with an option, which matures on the same day as the future, and two "serial" options, which mature one and two months earlier than the futures contract. We will restrict our attention to the non-serial options, for simplicity, and because these options are much more heavily traded, and at longer maturities, than the serial options. The options are paid for when they are purchased, and the position is marked to market daily. On exercise, the final mark-to-market is performed and the underlying futures contract is delivered. The delivered futures has no value on delivery as it is also marked to the market. That is, its strike equals the futures settlement price. To value these options, one should replace the dividend yield by the interest rate. Both puts and calls are sometimes optimally exercised early.

Our data is purchased from the Futures Industry Institute, in Washington, DC. It covers an eleven year period of daily data from January 1990 to December 2000. Daily data are known to exhibit the day-of-the-week effect. Monthly data might be a better choice, but then there are only 132 months in eleven years. Also, one month is much too long a period for any option hedging program. Although the day-of-week effect on futures prices might be small, the effect on options could be very large, as near-the-money options are very sensitive to changes in the underlying futures price. And near-the-money options are the most actively traded ones. To balance between seasonality effects and power of the test, we decided to use only the weekly data. There are 573 weeks in the period covered. We will work with settlement prices, which are based on the option prices during the closing period of the day's trading. These prices are likely to be very reliable, because the daily margining is based on them, and so they are scrutinized closely by the market participants.

We also need US dollar interest rates. We use BBA-LIBOR data for maturity one week, one month and three months. We use settlement prices of the Eurodollar futures for interest rate maturing longer than three months. Eurodollar futures are also on a maturity cycle of March, June, September, and December. Eurodollar contracts is the most actively exchange traded product, with contracts trading up to an expiry of 10 years. BBA-LIBOR data are sourced from Datastream, while Eurodollar futures data are from Futures Industries Institures. With these data, a interest rate curve is constructed each day, then the rate for the required maturity is interpolated. Convexity correction for the Eurodollar data is ignored. This is not a restriction, as we are examining the two nearest to maturity option contracts, which could be at most six months away. Any convexity adjustment will be too small to have an effect. During the interest rate curve construction, care is taken with regard to the relevant day count convention.

We are estimating the model with the two option series simultaneously – The nearest to maturity option series, and the next nearest. Each of these series could have a different implied vol. As calls and puts do not necessarily have the same dynamics, we will construct one model for the calls and one for the puts. In total, we will estimate four set of parameters, for the nearest maturing call, nearest maturing put, the next-nearest maturing call, and the next-nearest maturing put. Results of these estimation will be used at the end of this section to construct a hedged portfolio to compare with that constructed using standard option pricing formula.

In the optimization for the parameters, we do not only take the current option prices (or price changes) into consideration. We use the full history of weekly data, exponentially weighted. The weight λ is chosen to reflect a half-time of thirteen weeks, or three months. That is, $\lambda^{13} = 1/2$, or $\lambda = 0.9481$. Therefore, recent data has a larger impact that older data, and data from thirteen weeks ago has half the impact as this week's data. The model is estimated using data for the period January 1990 to December 2000, while the hedge return is calculated over 1991 - 2000. So that, the estimation would at least cover one full year of weekly data.

The objective function for the Neural Network of option price is:

$$\min_{w_i} \sum_{s,m,k} \lambda^{t-s} \left(P_s(m,k) - \hat{P}_s \right)^2 \tag{29}$$

Current time is t, which is an index for the week. $P_s(m,k)$ is the observed option price with maturity m, and moneyness k, at time s. The minimization is performed over all the weights, w_i, in all the hidden and output nodes. $\hat{P}_s$ is the model option price, which is a function of all the weights in the hidden and output nodes and the x_i value in the four input nodes.

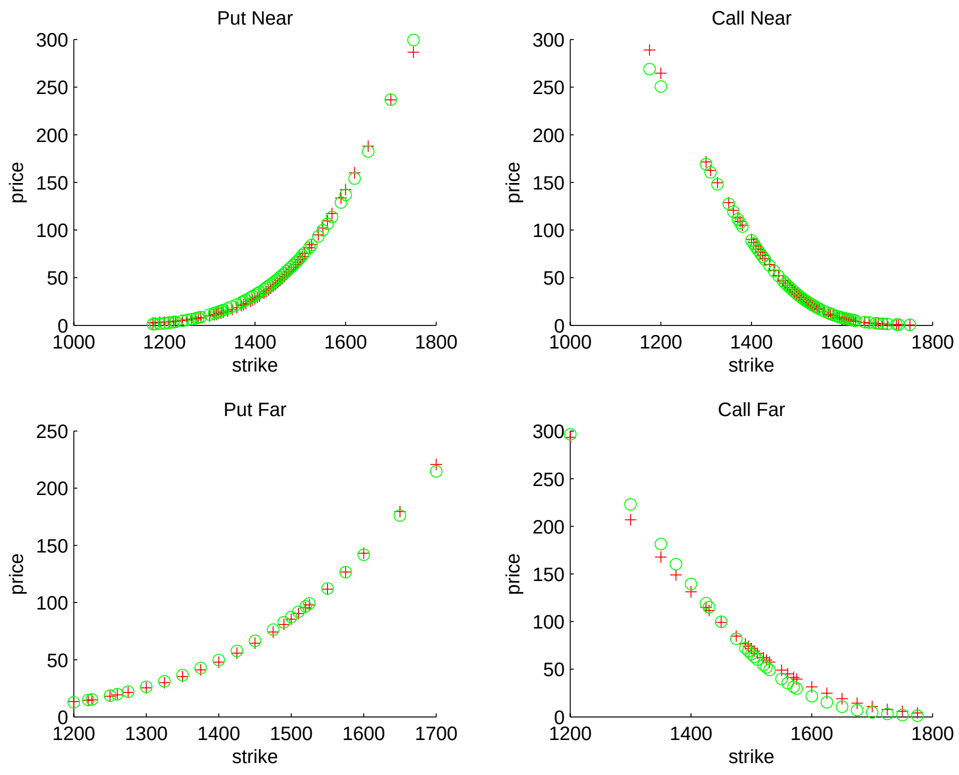

Fig. 3. Multilayer Perceptron of price - observed option prices (+) versus model option prices (o). “Put Near” is the nearest to maturity put option, while “Put Far” is the next nearest to maturity put option. Similar for calls.

As delta and vega are not observable, the objective function for the Neural Network of delta and vega is more involved:

$$\min_{w_i} \sum_{s,m,k} \lambda^{t-s} \Big(\Delta P_s(m,k) - \hat{D}_s \; \Delta F_s(m) - \hat{V}_s \; \Delta IV_s(m) \Big)^2 \tag{30}$$

Current time is t, which is an index for the week. $\Delta P_s(m,k) = P_{s+1}(m,k) - P_s(m,k)$ is the option price change with maturity m, and moneyness k, at time s. $\Delta F_s(m)_s = F_{s+1}(m) - F_s(m)$ is the price change for the corresponding futures contract at time s, which also has maturity m. $\Delta IV_s(m) = IV_{s+1}(m) - IV_s(m)$ is the change in the implied volatility for the at-the-money options, with maturity m, at time s. In practice, call and put options do not have exactly the same implied volatility, but they are very close. Therefore, $IV_s(m)$ is the average implied volatility for the at-the-money call and put options. The objective function is optimized over all the weights, w_i, in the hidden and output nodes. $\hat{D}_s$ and $\hat{V}_s$ are functions of the all the weights in the hidden and output nodes and the x_i value in the four input nodes.

Figure 3 showed the fitted option prices versus observed option prices on July 5, 2000, calculated by the Neural Network of option price. Recall that the

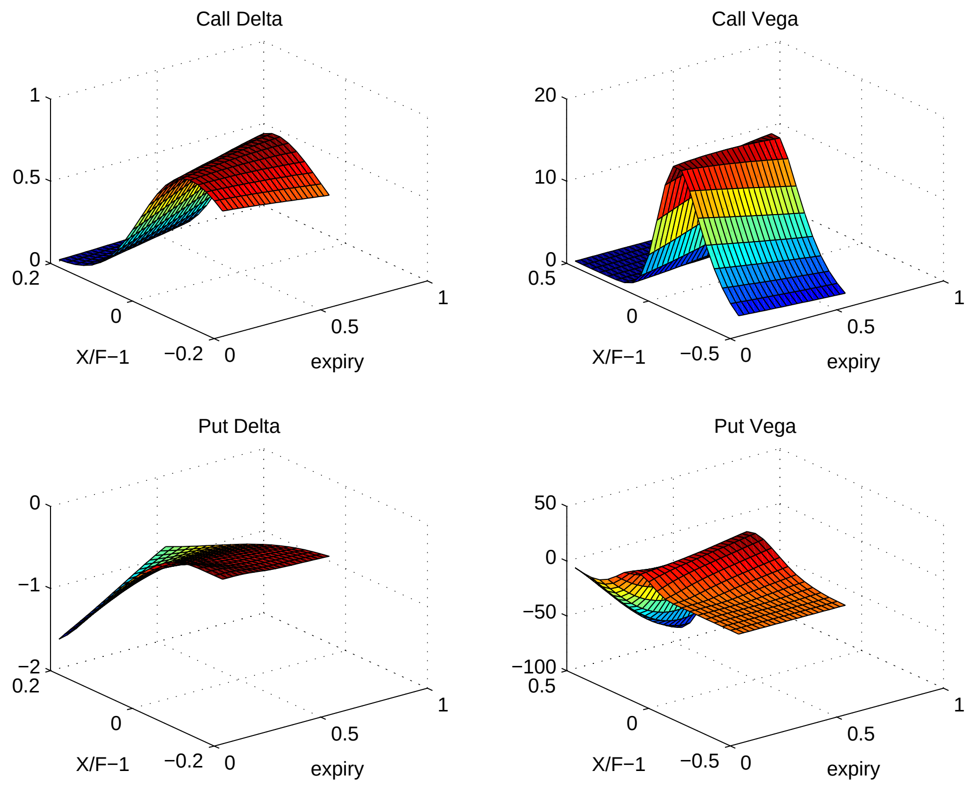

Fig. 4. Multilayer Perceptron of price - hedge parameters

optimization is performed over one year of weekly data, while there are 19 parameters in this model, and 23 in the Network model of Delta and Vega. As there are more data than parameters, the fit on any one day might not be exact. However, this also prevents model overfitting. The fit is actually quite good.

Figure 4 and 5 are hedge ratios derived from the two different Neural Networks. Both figures are drawn for the estimated parameter as of July 5, 2000. This is a typical day. Figure 4 shows the hedge ratios derived from the Neural Network of option price. Delta for deep out-of-the-money options look rather strange and it is sloping downward with decreasing moneyness, in stead of flattening out. Similar pattern for the Call Delta has been reported in Hutchinson-Lo-Poggio (1994). On the other hand, vega does not seem to increase noticeably with increasing expiry. This is in sharp contrast to Figure 5 for the ratios derived from the Neural Network of delta and vega. Here, Delta surface looks a lot more familiar, and it is more steep for near expiry options than longer lived ones, while vega shows a significant increase with expiry, as expected.

Using the Network model, the formula for the predicted option price changes as a function of changes in futures price and implied volatility is:

$$\hat{D}_s \; \Delta F_s(m) + \hat{V}_s \; \Delta IV_s(m) \tag{31}$$

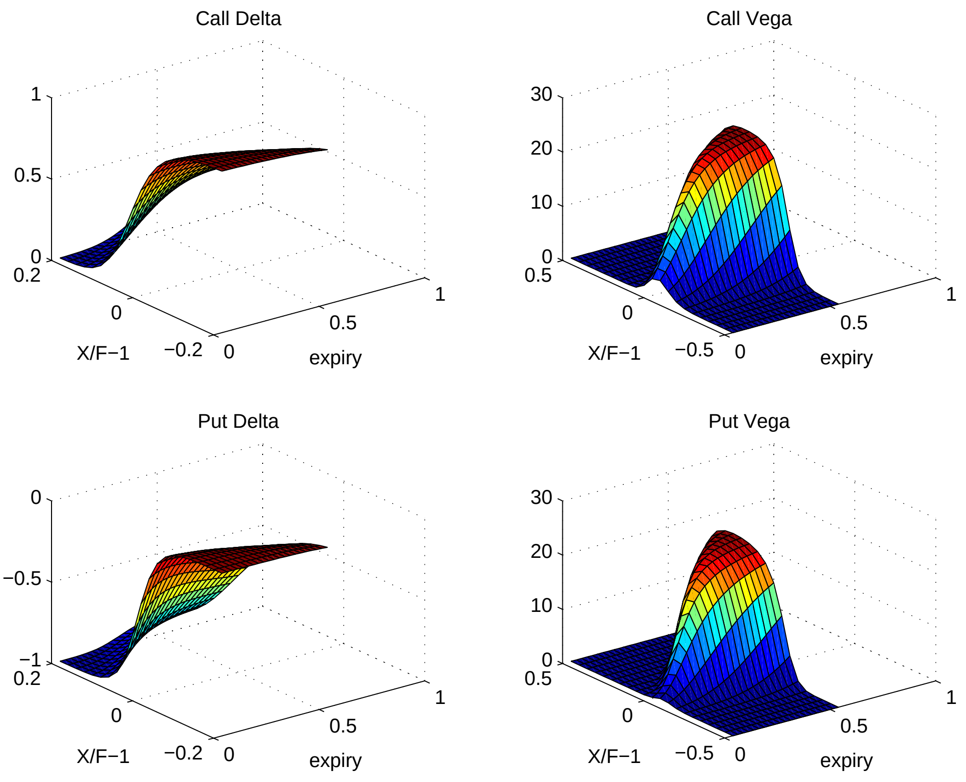

Fig. 5. Multilayer Perceptron of delta/vega - hedge parameters

with $\hat{D}_s$ being the Network delta function at time s, and $\hat{V}_s$ the Network vega function, ΔF_s is the changes of underlying futures price, and ΔIV_s is the changes in the at-the-money implied volatility as calculated by a 200 steps Cox-Ross-Rubinstein model for American options.

Figure 6 show the actual option price changes versus the option price changes predicted by the Network of price. Compare Figure 6 with Figure 7 of the actual option price changes versus the option price changes predicted by the Network of Delta and Vega. Some difference in performance is to be expected, as the latter model is explicitly optimized over the delta and vega, while the former is optimized over option prices. However, the extent of the difference is rather large.

As a last test, we calculate the cumulative hedge return with both networks, and also for that with Black-Scholes hedge parameters. As S&P experienced a very sharp increase over the period under consideration, we need to take proper care in the construction of our test portfolio. Otherwise, the more recent years' hedge results will overshadow the earlier years' results. In the delta neutral portfolio, we have a combination of at-the-money put (or call) option and one dollar notional value of the underlying futures contract. The amount of options is so chosen to neutralize the delta of the futures, while it would have an exposure in vega. Similarly, for the delta-vega neural portfolio,

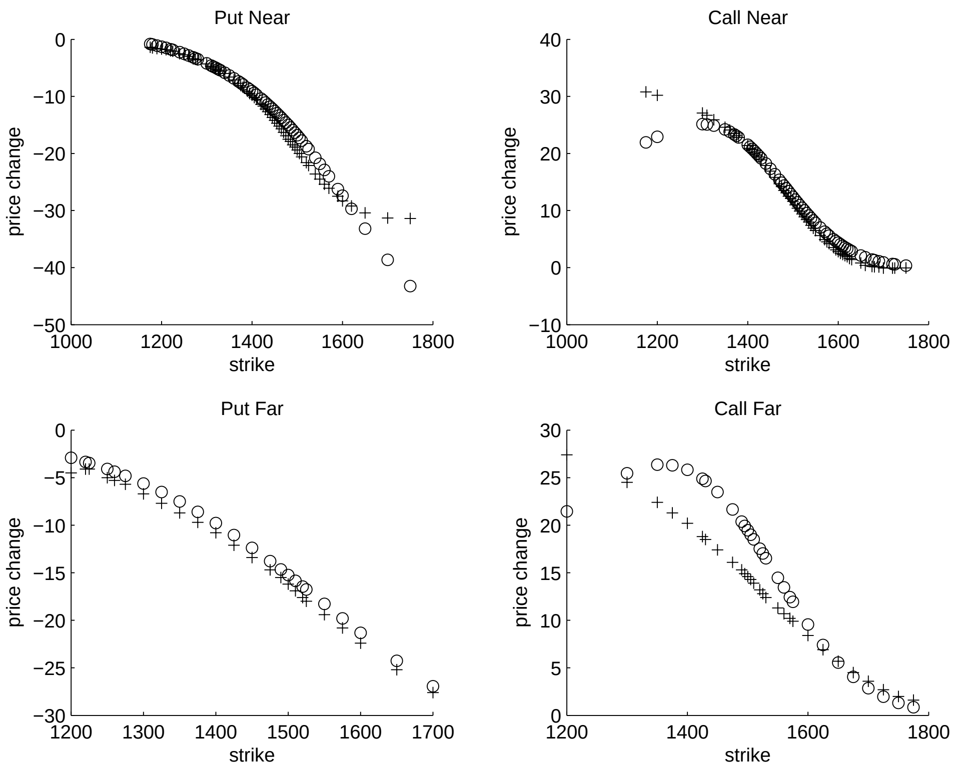

Fig. 6. Multilayer Perceptron of price - observed option price changes (+) versus hedged returns (o). "Put Near" is the nearest to maturity put option, while "Put Far" is the next nearest to maturity put option. Similar for calls.

we combined a one dollar notional value of underlying futures contract both at-the-money put and call options such that delta and vega of the total portfolio are neuralized. Such construction ensures that each day's hedging result has an equal impact. Every time we will use the nearest to maturity futures contract and its corresponding options. We bought the portfolio on day one, and sell it one week later. Buy another portfolio and sell it the week after. When the nearest maturity is less than 14 calendar days, we will switch to the next maturity, in order to maintain market liquidity. Funding costs/revenues are included in the results. Transaction costs are ignored.

We only use at-the-money options in this test, as away-from-the-money options exhibit "smirk", also known as "smile", which is stochastic. Doing delta and vega hedging is not sufficient for away-the-money options, as this strategy does not take changes in the smirk into consideration. The smirk in S&P 500 futures options is examined in Carverhill-Cheuk-Dyrting (2002).

Results from the cumulative hedging test are listed in Table 1. Cox-Ross-Rubinstein is losing money for all three portfolios, while both Networks are making money in Delta-Vega Neural, and the Delta-Neural Call portfolios. Of course, both gain and losses indicate hedging error. However, the cumulative hedging gains are statistically insignificant, as the corresponding standard

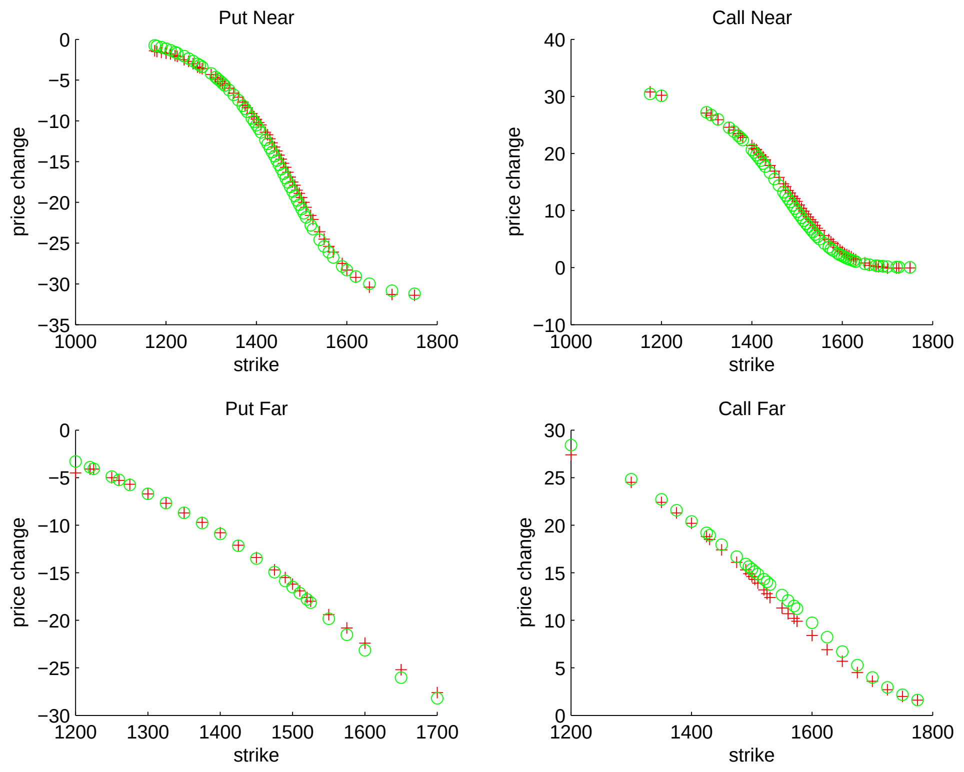

Fig. 7. Multilayer Perceptron of delta/vega - observed option price changes (+) versus hedged returns (o). "Put Near" is the nearest to maturity put option, while "Put Far" is the next nearest to maturity put option. Similar for calls.

	Delta-Vega Neutral		Delta Neutral Call		Delta Neutral Put	
	mean	std dev	mean	std dev	mean	std dev
Cox-Ross-Rubinstein	-.07	1.8	-.08	2.5	-.07	3.7
MLP (price)	.12	2.1	.59	5.1	-.81	8.5
MLP (delta/vega)	.02	1.4	.56	4.6	-.84	7.8

Table 1
Cumulative Returns from Hedged Portfolios.

deviations are large. Now, let's turn our attention to the standard deviations. If the hedging gains and losses are negligible, the strategy with the lowest standard deviation should be a better hedging tool.

For both delta neutral call and put portfolios, using Cox-Ross-Rubinstein delta estimate results in the smallest standard deviation. That is, Cox-Ross-Rubinstein gives a better hedge when you only care about delta. However, it is not a fair comparison for the Neural Network of delta/vega, because it is constructed to minimize the hedging error resulting from both delta and vega. For the delta-vega neutral portfolio, we see that the Neural Network of delta/vega has the lowest standard deviation for hedging error. It is also lower than standard deviation of two delta neural portfolios, as delta neutral portfolio is far from riskless — it has vega risk. The Neural Network of option price is always inferior to the Network of delta/vega for all three test portfolios. Also, the Neural Network of price is performing worse than Cox-Ross-Rubinstein.[6]

5 Conclusion

We have examined two different methods to set up a Neural Network for option pricing and hedging. As the accuracy of hedging parameters are of paramount importance to traders in practice, it is advisable to set up the system to give delta and vega directly, in stead of deriving from the sigmoid function for the price. However, a Neural Network of delta/vega does not produce option price as an output. For practical use, we suggest to construct a Neural Network model with three output nodes: option price, delta and vega, with option price calibrated to observed option prices, while delta and vega calibrated to observed option price changes. Our results also showed that a Neural Network of delta/vega produces better hedging parameters than the standard Cox-Ross-Rubinstein model for American options.

References

Altman, E.I., G. Marco, and F. Varetto, 1994, "Corporate distress diagnosis: Comparisons using linear discriminant analysis and neural networks", Journal of Banking and Finance 18, 505–529.

Black, F., and M. Scholes, 1973, "The pricing of options and corporate liabilities", Journal of Political Economy 81, 637–659.

Carverhill, A., T.H.F. Cheuk, and S. Dyrting, 2002, "The Price of the Smirk: Returns to Delta and Vega Neutral Portfolios of S&P500 Futures Options", Working paper, University of Hong Kong.

Cox, J.C., S.A. Ross, and M. Rubinstein, 1979, "Option Pricing: A Simplified Approach", Journal of Financial Economics 7, 229–263.

Garcia, R., and R. Gençay, 2000, "Pricing and Hedging Derivative Securities with Neural Networks and a Homogeneity Hint", Journal of Econometrics 94, 93-115.

Hornik, K., M. Stinchcombe, and H. White, 1989, "Multi-Layer Feedforward Networks are Universal Approximators", Neural Networks 2, pp. 359–366.

[6] This is consistent with Hutchinson-Lo-Poggio (1994), where Neural Network is reported to perform worse than Black-Scholes in terms of Delta Hedging for short-term near-the-money options, in table XVI.

Hutchinson, J.M., A.W. Lo, and T. Poggio, 1994, “A Nonparametric Approach to Pricing and Hedging Derivatives Securities Via Learning Networks”, Journal of Finance 49, 851–889.

Marquardt, D.W., 1963, “An algorithm for least-squares estimation of nonlinear parameters”, Journal of the Society for Industrial and Applied Mathematcs 11, 431-441.

Meissner, G., and N. Kawano, 2001, “Capturing the Volatility Smile of Options on High-Tech Stocks–A Combined GARCH-Neural Network Approach”, Journal of Economics and Finance 25, pp. 276-292.

Press, W.H., S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, 2002, “Numerical Recipes in C++”, 2nd Edition, Cambridge University Press.

Ripley, B.D., 1996, “Pattern Recognition and Neural Networks”, Cambridge University Press.

Trippi, R.R., and E. Turban, 1996, “Neural Networks in Finance and Investing”, 2nd Edition, Irwin.