Pattern Recognition Applications in Securities Trading

Richard Pospesel

Abstract
In this paper, I give a brief introduction to the important topics in securities trading and the general problem to be solved. I provide a broad overview of the issues surrounding pattern recognition with applications in securities trading and I discuss the types of potential problems unique to financial applications and how they differ from traditional pattern recognition tasks. Finally, I describe and compare the effectiveness of various published classification and regression techniques as applied to securities price forecasting and speculate on how they can be improved.

1. Introduction
A security is any kind of financial instrument which can be bought and sold on the open market such as company stocks, commodities contracts, mutual funds or currency. The efficient market hypothesis is a widely held belief among economists that all the important information related to a security's underlying value is perfectly represented by its current price. Furthermore, it is impossible to forecast a security's price at any time in the future because it will be based on information which does not exist yet and cannot be known ahead of time (barring things like insider trading). If the efficient market hypothesis is accurate then it should be impossible to build statistical models which can forecast security prices beyond 50% accuracy. That is, tomorrow's price should have no correlation whatsoever with any past information and each day's price should follow a so-called "random walk" [1].

Despite the supposed impossibility of statistically significant security trading profitability, day traders, stock brokers and mutual fund managers seem to somehow exist and make a living by determining which securities will go up or down with some degree of accuracy. In the investment industry, there are two broad categories of data analysis used to determine whether or not a security is a good buy or sell opportunity: fundamental analysis and technical analysis.

Fundamental analysis is the process of determining whether or not a security is overvalued or undervalued based on "fundamental" information. If it is found to be currently undervalued, then the investor ought to buy shares. If it is found to be currently overvalued, then the investor ought to sell shares. In the case of trading company stocks, such fundamental information would include information about the company which the stock represents. Various statistics about the company such as profitability, earnings, growth, etc would be used to determine the company stock's so called intrinsic value which would then be compared to the stock's actual price [1] [2].

On the other hand, technical analysis uses so called "technical" information to forecast whether or not a security will go up or down. A technical analyst does not care what the security they are trading actually represents. All that matters to the technical analyst is the technical information: the open, high, low, close and volume of a security over time[1]. Technical indicators, such as moving averages, are derived from this technical information and are used to determine good buying or selling opportunities. The main idea behind technical analysis based trading is that security prices have a certain amount of momentum at times due to human psychology which an investor should be able to take advantage of this by following the inherent upwards or downwards price trend.

Unlike other pattern recognition problems such as handwritten digit classification, there is not one overarching problem trying to be solved in the domain of financial forecasting. In a broad sense, there are two categories of problems to be solved: classification and regression. I present different approaches used to solve each problem.

With the regression approach, the goal is to accurately forecast the price of a security at some point in the future based on current and past information. Typical systems try to forecast the security price one day in advance. This approach primarily makes use of technical information and indicators [3] [4] [5] [6].

The classification approach is a bit more varied. Instead of trying to forecast exactly what the price will be at a future time, the classification approach generally tries to pigeonhole the security into discrete growth classes where each class represents a growth range that security will fall into. The time period we are forecasting over varies from one day ahead to a year ahead with these methods. Classifiers make use of both technical and fundamental information depending on the time frame [6] [7] [8] [9] [10] [11].

The rest of the paper will be organized as follows: Section 2 will cover the different classes of features used as input to the various regression and classification systems. Section 3 will discuss an interesting heuristic for determining the quality of training data for classifier creation. Section 4 will discuss the various methods for evaluating classification and regression systems while Section 5 will describe some of the potential issues associated with using pattern recognition techniques to develop expert trading systems. Section 6 will describe a survey of techniques of classification and Section 7 will describe various techniques which use regression.

2. Feature Selection

The majority of research seems to be focused on short term forecasting, so generally only technical indicators and information are used as inputs to the classification and regression models, though there are exceptions.

The most common class of features are those based on the vector of past closes. The time frame ranges from only 1 day [3] to 4 weeks [9]. Various methods are used to preprocess these features. The simplest approach is to simply use the raw data as input [3]. Others linearly rescale the raw data to fit a specified range that is appropriate for the classification or regression method used [9]. The most

[1] The open is the price of the security at the start of a trading period; the close is the price of the security at the end of a trading period; the low and high are the lowest and highest price of the security over the extent of the time period; the volume is the number of shares which were bought and sold during that time period.

common method is to use the previous few days' daily return as features [2] [6]. Zhu et al. added volume measurements to their vector of daily returns [7]. Huang et al. used the previous day's log returns as input to their classifier [8]. He et al. used the 1-day, 5-day, 10-day, 20-day and 60-day moving average[2] as features in their forecasting system [5].

Instead of using variations on the daily close as features, both Kim et al. and Man-Chung et al. used multiple technical indicators as the feature vector for their models [4] [11]. Kim et al. performed a one-way ANOVA analysis on various technical indicators applied to their dataset and used the ones which were significant to at least the 5% level [10]. These indicators are summarized in Table 1. Quah et al. used a set of seemingly arbitrarily picked fundamental indicators.

Technical Indicator (Significance Level)	Formula
6 Day Moving Average (5%)	$\frac{sum\ of\ closing\ price\ over\ 6\ days}{6}$
Relative Strength Index (1%)	$\frac{sum\ of\ positive\ closing\ price\ over\ 25\ days}{sum\ of\ closing\ price\ over\ 25\ days} \times 100$
Momentum (1%)	$most\ recent\ closing\ price - closing\ price\ from\ 6\ days\ ago$
Stochastic Oscillator (1%)	$\frac{most\ recent\ closing\ price - lowest\ close\ from\ past\ 6\ days}{highest\ close\ from\ past\ 6\ days - lowest\ close\ from\ past\ 6\ day}$
Volume Ratio (1%)	$\frac{positive\ growth\ volume\ from\ past\ 6\ days - negative\ growth\ volume\ from\ past\ 6\ days}{total\ volume\ from\ past\ 6\ days} \times 100$
Disparity (1%)	$\frac{most\ recent\ closing\ price}{6\ Day\ Moving\ Average} \times 100$
Rate of Change (1%)	$\frac{most\ recent\ closing\ price}{closing\ price\ from\ 6\ days\ ago} \times 100$

Table 1 definition of technical indicators used by Kim et al. as features for their stock price classifier [11]

I have found no research which examines using all of the available raw technical information for either classification or regression forecasting financial time series. All of the forecasting methods used only features which relate to the daily closes without taking into account the daily lows, highs, opens or volume (with the exception of Zhu et al. which evaluated the usefulness of including volume statistics [7]). It would be interesting to see what effect including this other information would have on forecasting accuracy considering that day traders often utilize all of this information.

3. Data Selection

If the efficient market hypothesis is correct, than it should be impossible accurately and consistently forecast future stock prices and returns because all relevant information relating to the stock is present in its price [1]. That is, future prices should only depend on new information which is fundamentally

[2] A n-day moving average is just the mean value of the past n daily closes.

unknowable ahead of time and therefore a model of the future price action of an equity based on past information should be impossible to create. Mathematically speaking, future prices should appear to follow a random walk and have no correlation with past information. That is to say, tomorrow's equity price should have no correlation with today's price or yesterday's price. In reality, the behavior of equity prices, in terms of how "random" they are, varies over time.

The randomness of a time series can be evaluated by estimating its Hurst exponent [13]. For financial time series, the Hurst exponent is estimated using a series of percent changes between closes. A Hurst exponent value of 0.5 implies a pure random walk or no correlation between past and future values in the series. A Hurst exponent less than 0.5 indicates an anti-persistent series while a value more than 0.5 indicates a persistent series. An anti-persistent series has the property such that greater values are more likely to be followed by lesser values and vice versa. A persistent series has the opposite property such that greater values are more likely to be followed by greater values and lesser values are more likely to be followed by lesser values. The direction of the trend of the series has no bearing on whether it is persistent or anti-persistent. A time series which is persistent can persist in decreasing or increasing. As the Hurst exponent moves to 0.0, the time series should become more chaotic and change directions more frequently. As it tends to 1.0 the time series should become more deterministic and follow a stronger trend. Thus, a Hurst exponent of 0.5 indicates that greater and lesser values are equally likely to occur regardless of whether or not the previous value is greater or lesser than the value before it. So, a time series with a Hurst exponent of 0.5 should appear to be random.

A market which is truly "efficient" should follow a random walk and therefore have a Hurst exponent of about 0.5. That is, an equities change in price today should have no correlation with what it does tomorrow. However, it has been shown that the Hurst exponent for equity prices actually varies over time. For instance, from Jan 2, 1930 to May 14, 2004 the Dow Jones Industrial Average's (DJIA) Hurst exponent has existed on the interval between 0.4503 and 0.6405 [6]. Figure 1 shows how the Hurst Exponent for the DJIA has changed over time.

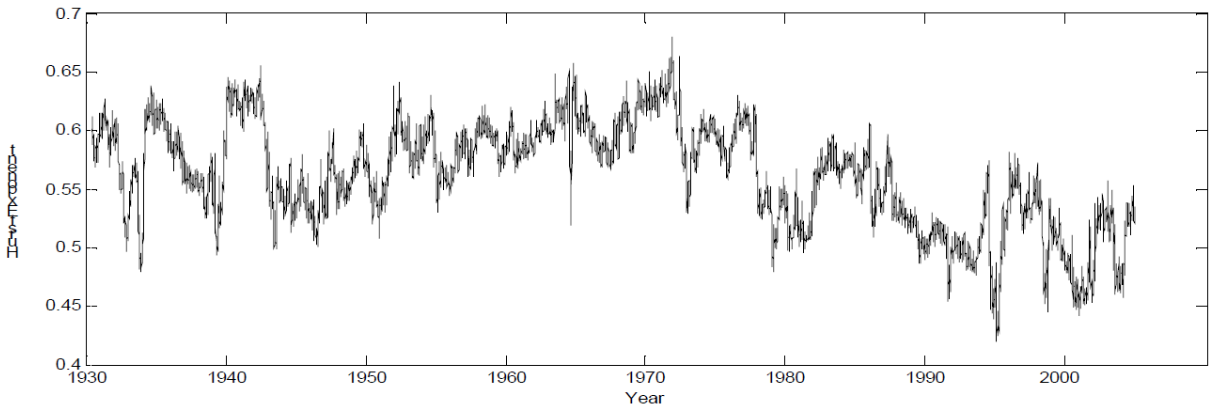

Figure 1. Value of estimated Hurst exponent for DJIA over time [6]

As we can see in Figure 1, there seem to be times when the Hurst exponent is large at which point the DJIA should be more predictable than at other times when the Hurst exponent is smaller. To confirm this hypothesis, Qian and Rasheed used a Multilayer Perceptron (MLP) to determine how predictable the price time series for the DJIA was for period where the Hurst exponent is large versus when it is smaller. They determined that MLPs trained on data from time period with a large (H>0.65) Hurst exponent had statistically significantly lower error on a test set with a similar Hurst exponent as compared to a training set taken from a time period with a lower (0.55 > H > 0.54) Hurst exponent [6].

That is, an MLP trained on data taken from predictable time periods have better forecasting performance when used to forecast during time periods of high predictability (a large estimated Hurst exponent). Alternatively, an MLP trained on data from time periods with low predictability and subsequently tested on data with low predictability should have higher error. This interesting fact can enable better data selection when building a training set for a classifier designed for stock selection or equity forecasting. Using the Hurst exponent as a heuristic, we can only select training data which has meaningful information. The Hurst exponent can also be used as a filter in an expert system to reject equities to forecast when determining portfolio allocation. Equities with a higher Hurst exponent should statistically be more likely to be accurately forecasted by a given classification model than those with a lower Hurst exponent.

The value of the estimated Hurst exponent at a point in time for a given equity would seem to be a significant indicator of how useful the data associated with that time period would be for training a classification or regression model. The Hurst exponent could also be used in an expert equity trading system as a filter to determine whether the classification or regression model should be used or whether the system should take a pass and stay out of the market.

4. System Evaluation

All of the classifiers used were evaluated using an accuracy measurement, though none but one [11] of the papers gave additional statistics such as precision or recall [8] [10] [9] [8] [2]. This is especially an issue for evaluating a financial forecasting model because we need to know explicitly what sort of errors it makes. For instance, if a classifier is designed to forecast which direction a security will go (either up or down in price) then just saying that it got 65% accuracy means nothing because there are certainly time frames during which 65% of all stocks could have gone up (such as during the .com bubble of the late 1990s) or during which 65% of all stocks could have gone down (such as during the housing market collapse of 2008). During these time periods, an expert system could easily get 65% accuracy by always saying that the security is always going up or is always going down. Such a model would be useless.

In addition to accuracy, Quah also evaluated the average annual appreciation of the stocks picked using his classification system [10].

All of the regression models used the Mean Square Error (MSE) for evaluation [5] [7] [4] [3]. In addition to MSE, Man-Chung et al. also evaluated the directional accuracy of their forecasting model [4].

5. Potential Expert System Pitfalls

When collecting data for training a financial classifier, there are some special biases which need to be considered that are not typically associated with other types of classifiers. When collecting financial data such as company stock data, it may be important to also include data from companies which are no longer listed on the exchange because they went bankrupt or otherwise delisted. Otherwise, your data may be skewed toward companies which are successful based on the fact that they currently exist. This type of bias is known as the survivorship bias [12].

A second potential problem to look out for is the look-ahead bias. This problem occurs when you use data to train a model which was not available at the time period for which it is forecasting [12]. That is to say, the model is using figures to forecast day t that were not actually available until day $t + 1$. Alternatively, it is possible that historical data that is available may not have been available then or it has been since changed or revised. For example, if we want to create a classifier which estimates yearly growth based on technical indicators, then we would need the technical data which was available at the time which we are trying to forecast.

Another potential problem with creating a classification or regression model which accurately forecasts price movements based on pattern recognition is that the methods must be kept secret or else it will stop working. This problem is known as the "January effect." The original January effect was a seemingly reliable pattern such that daily returns seemed to inexplicably be greater in the first week of January. After this effect was discovered, every trader in the market quickly took advantage of this fact and as a result simply did not work anymore [1]. Similarly, a profitable trading system would need to remain secret or else it will cease to work.

For example, assume that we have an expert system which says stock X is going to go up today. Ideally, we would want to buy it before it started to go up and sell it when it reaches its peak. However, if everyone knows that stock X is going to go up, than everyone will attempt to buy it at its lowest point and sell it at its peak. As soon as everyone attempts to buy shares, the liquidity goes down to zero (since nobody is going to want to sell shares they own because the price is going to go up) and the price will sky rocket. Once this occurs, everybody is going to try to dump their shares at the same time and the price will plummet. A lucky few may end up on the right side of both of these transactions, but most will only take minor profits or major losses because they were too late in buying or too late in selling or both. This sort of behavior happens all the time for smaller stocks on the market due to rumors and the like [1].

6. Classification

Neural Network

Zorin compared using a multilayer perceptron and a Kohonen self-organizing map to forecast the Ventspils Nafta (A Latvian holdings company) stock [9]. For his input features, Zorin used the 20 daily closes previous to the day being forecasted. For outputs, he divided his data into 7 classes. Each class coincided with a range of daily returns for the next day:

Class	0	1	2	3	4	5	6
Range	(-∞, -4.5%]	(-4.5%, -2.5%]	(-2.5%, -0.5%]	(-0.5%, 0.5%]	(0.5%, 2.5%]	(2.5%, 4.5%]	(4.5%, ∞)

Using these daily return ranges, Zorin used classification to forecast one day into the future.

To build his input/output pairs, Zorin used roughly a year's worth of data which ranged from December 1, 2000 to December 28, 2001. All of the closes were between 0.85 and 0.58 so no preprocessing on the data was performed. He divided the data into a training set and a testing set, though he did not explicitly say how.

An MLP with a 20-10-7 architecture performed best (amongst other MLP architectures) while a 20-7 Kohonen network performed best overall in terms of classification accuracy. Unfortunately, Zorin failed to provide much useful information such as the prior probabilities for each class for the training and testing set or a confusion matrix for either the MLP or Kohonen network classifiers. It would be interesting to see how accurate the Kohonen net compared to the MLP for a larger dataset (such as the DJIA dataset used by Qian and Rasheed [2] and to see how correlated classification accuracy is to the Hurst exponent [6].

Quah used an MLP to pick companies in the DJIA index[3] for the purpose of annual portfolio allocation [10]. For his input features, Quah used 11 seemingly arbitrarily picked fundamental indicators. Recall that a fundamental indicator relates to information about the entity behind the security. In this case, the entities are companies so the fundamental indicators relate to things like company profitability, debt, assets, etc. For outputs, Quah divided the data into two sets of classes. The first class was defined as any stock which appreciates more than or equal to 80% in a year. All other stocks are defined as being class two.

To build his input/output pairs, Quah used ten years worth of data ranging from 1995 to 2004. The first 80% of the data is used for training while the last 20% is used for testing. He used an 11-22-1 MLP architecture for classification and hyperbolic tangent activation function for both hidden and output units. When the output unit was evaluated as positive, the input was classified as class 1 and when the output unit was negative, the input was classified as class 2. On the test set, Quah's MLP had 71.35% classification accuracy and an estimated 13% annual average appreciation of the selected stocks, compared to 11.22% annual average appreciation of the entire market.

It would seem that this sort of long term forecasting has serious potential for long term low-risk investments, but probably does not have the same kind profit potential that shorter term day-trading style of investment has. I think it would be a good idea to try to reduce the forecasting window by using quarterly data instead of annual data. For instance, Toyota may have seemed to be a good solid investment at the start of 2010, but only recently a number of design flaws and defects have been discovered in their cars and sales have since decreased. The first quarter numbers for this year will probably reflect this and a quarter-based MLP classifier may notice the change and reallocate your

[3] A stock index is a collection of companies. An indices' price at a given time is simply the sum of the prices of the companies represented in the index, weighted by each size companies relative to the others in terms of number of shares.

portfolio accordingly. However, an annual portfolio allocating MLP would not take notice until the next year rolls around after which point your holding sin Toyota will have plummeted.

Ensemble

Huang et al. forecasted the weekly price movement of the NIKKEI 225 Index using a SVM [8]. They used the previous day's log daily change of the S&P 500 index and the previous day's log daily change of the Japanese Yen to US Dollar currency conversion rate as the features to classify on. They reasoned that the Japanese economy is highly dependent on the US economy, so it may be possible to forecast how well the Japanese stock market does tomorrow based on how well the US stock market and economy as a whole did today. The outputs for each input were one of two classes: class 1 if the NIKKEI 225 went up in value and class 2 if it went down in value.

They created their input/output pairs from data ranging from January 1, 1990 to December 21, 2002 for a total of 676 weekly observations. The first 640 pairs were used to generate the parameters for the SVM while the remaining 36 were used for evaluation of the model.

The radial kernel was used:

$$K\left(x_i, x_j\right) = e^{-\frac{1}{10}\|x_i - x_j\|^2}$$

The SVM on its own had an accuracy of 73%. Huang et al. also compared the SVM classifier to LDA, QDA and an Elman backpropagation neural network (EBNN) classifier. The architecture for the EBNN was not defined. The accuracy results are summarized in Table 2:

Method	Accuracy
LDA	55%
QDA	69%
EBNN	69%
SVM	73%

Table 2 Accuracy on test data for 4 different classifiers used by Quah et al. [8]

After evaluating each classifier individually, Quah et al. combined them all by weighting each classifier's vote by its relative accuracy:

$$w_j = \frac{a_j}{\sum a_k}$$

The total accuracy for this combined classifier was 75%.

It would seem that the NIKKEI 225's behavior is highly predictable considering it only requires 2 features to get such a high accuracy rate. I suspect the accuracy may be improved by including more fundamental and technical indicators as features. Boosting may also help improve accuracy a bit using the classifiers already created.

Similar to Huang et al. [8], Qian and Rasheed forecasted the direction of the DJIA [2]. They used the four previous day's returns as their feature vector for an MLP, K-nearest neighbor (KNN) classifier and decision tree (DT) classifier. The desired outputs for each input vector were one of two classes: class 1 if the DJIA goes up and class 2 if it goes down.

They experimented with MLP architectures with 4 input units and 1 output units witch hidden units ranging from 1 to 10. The hyperbolic tangent activation function was used in all hidden and output units. During training, an output value of 1 represented class 1 and an output value of -1 represented class 2. During evaluation, positive outputs were evaluated as class 1 and negative values were evaluated as class 2.

They created their input/output pairs from data ranging from June 4, 1969 to June 4, 1973. The first 80% of the data was used for classifier creation and the last 20% was used for evaluation. This four year time period was selected because it has a rather high Hurst exponent and thus easier to forecast [6].

The best MLP architecture was the 4-3-1 MLP with an average accuracy of 61.09% directional accuracy. KNN performed best with k=5 with an accuracy of 56.64% directional accuracy. The decision tree which allowed at most 1 impure case in each leaf performed best with 56.38% accuracy. Combining the classifiers using simple majority voting gave a directional accuracy of 60.50%. When combined using unanimous voting, the directional accuracy increased to 65.36%, though the reject rate was not given.

Kim et al. used a MLP (architecture not specified), a human expert (7 years experience) and a human amateur (less than a year experience) combined using a genetic algorithm to classify the weekly change in the KOSPI Korean stock price index [11]. They used the seven technical indicators discussed in Section 2 as input to their MLP. The data was divided into 4 classes by weekly return in a fashion similar to Zorin [9]:

Class	0	1	2	3
Range	(-∞, -3%]	(-3%, -0%]	(0%, 3%]	(3%, ∞)

They created their input/output pairs from the 624 weeks from January 1990 to December 2001. The first 312 weeks were used for training the MLP, the second 156 weeks were used for validation while the last 156 were used for testing.

To combine the classifications from the trained MLP, the expert and the amateur, multiple voting methods were used: Borda count, weighted Borda count, Bayesian formalism, BKS and Dempster-Shafer.   They also used a genetic algorithm (GA) to evolve weights 12 weights for a weighted voting method, one weight for each classifier/class pair. The GA was evolved based on the data in the training set.

Among the individual classifiers, the MLP was the best with 66.0% overall accuracy. The amateur broker did barely better than random with overall accuracy of 51.9% while the expert had overall accuracy of 59.0%. Dempster-Shafer was the best voting method with overall accuracy of 76.3% while the GA had 79.5% classification accuracy.

7. Regression

Neural Network

Zhu et al. forecasted daily, weekly and monthly prices of the NASDAQ, DJIA and STI indices using an MLP [7]. For each experiment, they used the previous returns over the given time step for the 5 stocks whose price is most highly correlated with the price of the index they are a member of as features. In addition to these five features, they also experimented with various volume measurements:

1. No volume
2. The total volume of the index during the past time step
3. The volume over the last time step for the 3 stocks most highly correlated with the index
4. Both 2 and 3

The desired output value is the return for the index over the next time step.

To build their input/output pairs, Zhu et al. used data from the following ranges:

1. November 5th 1997 to October 18th, 2005 for the daily dataset
2. June 12th 1990 to October 18th, 2005 for the weekly dataset
3. March 1st 1989 to October 18th, 2005 for the monthly dataset

For each dataset, the first 90% of input/output pairs were used for training while the last 10% were used for testing.

Both the MSE and the sign prediction rate were evaluated for MLPs with 5, 10, 15, and 20 hidden neurons for each index, volume feature type, and time frame for a total 4 * 3 * 4 * 3 = 144 experiment types.

In general, Zhu et al. found that including trading volume as a feature does not help under a short-term next-day horizon, though it can help improve directional accuracy under medium-term weekly and long-term monthly windows. Interestingly, the NASDAQ index seemed to consistently be the most predictable in terms of directional accuracy for all experiments.

Man-Chung et al. used a 10-5-1 MLP to forecast 11 individual stocks listed on the Shanghai Stock Exchange [4]. For input features, they used the previous 5 day's exponential moving average along with 5 other technical indicators for the day prior to the day they wished to forecast. Each input feature was normalized between 0.05 and 0.. The desired output value of the output neuron given a feature vector was the exponential moving average price of the stock on the next day. After some algebra, this can be used to estimate the forecasted price for the next day.

To build their input/output pairs, Man-Chung et al. collected the daily closes for the 11 companies from 1994 to 1996. The first 500 days of data were used for training while the remaining 150 were used for testing.

Man-Chung et al. describe a weight initialization method which uses multiple-linear regression to estimate the last layer of weights after randomly initializing the first layer. This is done by calculating the outputs of the hidden units and using a first order Taylor series expansion of the output sigmoid unit. After a little algebra, standard least squares techniques can be used to find a good starting space for the last layer of weights.

On the testing data, conjugate gradient descent learning coupled with multiple-linear regression achieved 73.545% directional accuracy. The weight initialization technique coupled with standard backpropagation only achieved 69.303% directional accuracy compared to standard backpropagation without weight initialization which achieved 72.564% directional accuracy. This is probably the case because conjugate gradient descent performs a line search to identify the best next weight space to minimize error. This allows it to more easily escape local minima while standard backpropagation only moves with a fixed time step in the direction of least error. The weight initialization technique probably forces the weights into a poor local minima which standard backpropagation has trouble escaping.

Ensemble

Pai and Lin use a hybrid autoregressive integrated moving average (ARIMA) model and SVM model to forecast the following 10 stocks [3]:

Kodak	General Motors	JP Morgan Chase	Philip Morris	SBC Communications
Citigroup	General Electric	Southwest Water	ATP Oil & Gas	American National Insurance

They used an ARIMA(0,1,0) method to model the linear portion of the price change over time. Since the price of stocks fluctuates in a chaotic non-linear fashion, there is a non-linear error associated with each predication made by the ARIMA model. Pai and Lin use the SVM to model this error using regression. Instead of using the SVM's hyper-plane to divide two classes which have been blown up to a higher feature space, the hyper-plane is instead fit to these points. They used the Gaussian kernel function:

$$K(x_i, x_j) = e^{-\frac{1}{(2\sigma^2)}\|x_i - x_j\|^2}$$

Pai and Lin create their input/output pairs for each company from data taken from October 21, 2002 to February 28, 2003. The points from October to the end of December were used for creating the SVM and ARIMA models. The data from January was used for validation, while the data from February were used in testing.

They compared their hybrid model to both the ARIMA and the SVM model on their own, and the hybrid model performed much better in terms of MSE than either of the models alone. Unfortunately, directional accuracy for the models was not provided, so it is difficult to determine how effective this forecasting method is.

He and Shen use a bootstrapped collection of 5-2-1 MLPs to forecast daily exchange rates [5]. They forecasted the exchange rate of (in terms of United States Dollars) the Australian Dollar, British Pound, Canadian Dollar, European Euro, Japanese Yen and the Swiss Franc. For input features, they used the previous day's close, the 5 day moving average, the 10 day moving average, the 20 day moving average and the 60 day moving average. The desired value of the output neuron is the exchange rate for the next day.

To build the input/output pairs, daily close data was acquired for each currency over the time period from January 2, 2003 to December 29, 2006. The first 251 data points January 2, 2003 to December 31, 2003 were used for training, while the remaining 752 data point were used for testing. They used significantly more data for testing than they did for training, which is a departure from the methods of the previous researchers.

The training data was divided into 10 roughly equal periods and for each period an MLP was trained just on that data. To evaluate on the test data, the input vector is pushed through each of the 10 MLP's and the final forecast value is simply the average of all 10 outputs. The MSE on the test data using this ensemble approach is significantly less than that for a single MLP trained on all of the training data for each of the six currencies. Unfortunately, He and Shen did not give any information relating to the directional accuracy of their forecasting system.

8. Conclusion

Pattern recognition in finance seems to be an interesting field of study, though I suspect most of it is done behind closed doors due to the necessity of secrecy for such techniques to work. The general theme that seems to hold true is that combinations of classifiers or repressors tend to out-perform any one individual technique.

It would be interesting to see what a collection of properly trained neural nets could do when given a lot of technical data which spanned decades. The best place for active research in my view would be the currency markets; simply because they are more resistant to the so called January effect because of their very high liquidity (around 1.5 trillion dollars are transferred every day in the currency market).

9. References

[1] B. Malkiel. "The efficient market hypothesis and its critics," *The Journal of economic Perspectives*, vol. 17, no. 1, pp. 59-82, 2003

[2] B. Qian and K. Rasheed. "Stock market prediction with multiple classifiers," *Applied Intelligence*, vol. 26, pp. 25-33, 2007

[3] P. F. Pai and C.S. Lin. "A hybrid ARIMA and support vector machines model in stock price forecasting," *Omega*, vol. 33, pp. 497-505, 2005

[4] C. Man-Chung, W. Chi-Cheong and L. Chi-Chung. "Financial time series forecasting by neural network using conjugate gradient learning algorithm and multiple linear regression weight initialization," *Computing in Economics and Finance*, vol. 61, 2000

[5] H. He and X. Shen. "Bootstrap methods for foreign currency exchange rates prediction," in *Proceedings of the International Joint Conference on Neural Networks*, 2007

[6] B. Qian and K. Rasheed. “Hurst exponent and financial market predictability,” in *Proceedings of the Second IASTED International Conference on Financial Engineering and Applications*, 2004, pp. 203-222

[7] X. Zhu, H. Wang, L. Xu and H. Li. “Predicting stock index increments by neural networks: the role of trading volume under different horizons,” *Expert Systems with Applications*, vol. 34, pp. 3043-3054, 2008

[8] W. Huang, Y. Nakamori and S.Y. Wang. “Forecasting stock market movement direction with support vector machine,” *Computers & Operations Research*, vol. 32, pp. 2513-2522, 2005

[9] A. Zorin. “Stock price prediction: Kohonen versus backpropagation,” *Modeling and Simulation of Business Systems,* pp. 115-119, 2003

[10] T. S. Quah. “DJIA stock selection assisted by neural network,” *Expert Systems with Applications*, vol. 35, pp. 50-58, 2008

[11] M.J. Kim, S.H. Min and I. Han. “An evolutionary approach to the combination of multiple classifiers to predict a stock price index,” *Expert Systems with Applications*, vol. 31, pp. 241-247, 2006

[12] S. Mahfoud and G. Mani. “Financial forecasting using genetic algorithms,” *Applied Artificial Intelligence*, vol. 10, no. 6, pp. 543-566, 1996

[13] H. E. Hurst. “Long-term storage of reservoirs: an experimental study,” *Transactions of the American Society of Civil Engineers*, pp. 770-799, 1951