Using Neural Networks for Classification Problems in Finance

A Senior Project

presented to

the Faculty of the Statistics Department

California Polytechnic State University, San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Bachelor of Science

by

Joe Rolle

June, 2010

Table of Contents

Executive Summary

I used a Feed-Forward Neural Network for two different classification problems in the field of finance. The first was to explore Sector Classifications of the S&P 500. The Neural Network correctly classified assets intro their respective sectors about 30% of the time. The second classification problem dealt with Asset Jumps. The classification rate for the Neural Network fluctuated between 35% and 50% depending on the network architecture and inputs into the network.

What is a Neural Network?

Neural Networks (NN) are a branch of the broad field known as Artificial Intelligence (AI). John McCarthy, who in 1956 coined the term Artificial Intelligence, defined AI as "the science and engineering of making intelligent machines". Neural Networks are modeled after the human brain, and mimics it in two main ways. The first way is that knowledge is acquired through a learning process. Secondly, it stores that knowledge by adjusting the interneuron connection strengths (known as synaptic weights).

The specific type of Neural Network that I will use for this project is called a Feed-Forward Neural Network with a single hidden layer. It consists of an input layer, hidden layer and an output layer (Figure 1), where each unit in each layer is connected to each unit (via weights) in the following layer.

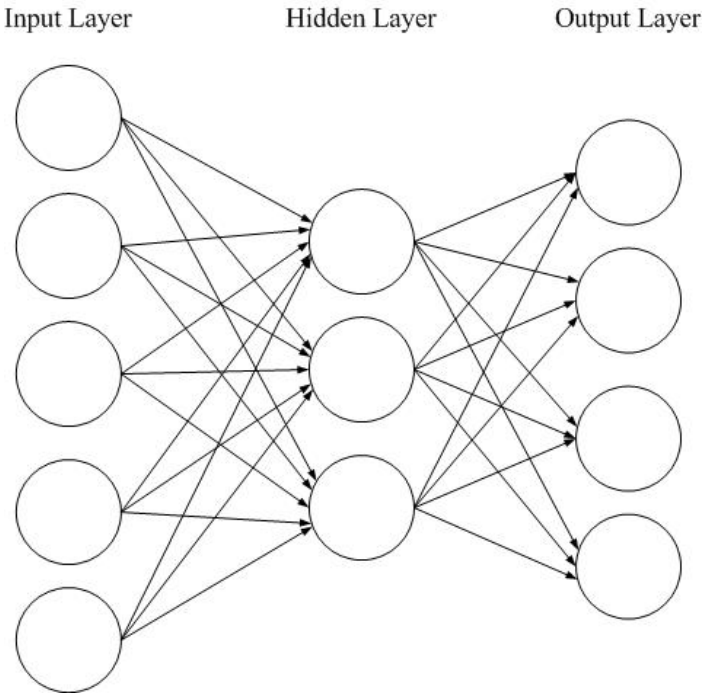

Figure 1: Diagram of a Feed-Forward NN with a single hidden layer.

Sector Classification

Financial Traders use sector classifications in their portfolio optimizations all the time, but do assets really fit into just one of ten disjoint categories? This was the first question that I wanted to explore while using a Neural Network. After acquiring a list of all the companies included in the S&P500 (and what sector they were classified as), I scraped financial data on each company from Yahoo Finance. The time frame of the data I slurped was from January 4th, 2010 to April 1st, 2010.

Network Inputs and Outputs

The input vector for the NN consisted of the following: Mean Open, Mean Close, Mean High, Mean Low, log(Mean Volume), Mean Return. Where the return, at time t, is defined as:

$$r_t = \log(\text{Adj.Close}_t - \text{Adj.Close}_{t-1})$$

The output from the Neural Network is a vector of approximate *a posteriori* class probabilities for each of the ten different sectors.

Investigating Network Architecture

One design choice you have to make when using a NN is the architecture of the network, specifically the number of neurons in the hidden layer. There are a couple of prevailing theories about how to make this choice. The rule of thumb is that you should have five training examples for each synaptic weight in the network. It is also said that the fewer neurons in the hidden layer lead to better generalization whereas more neurons will lead to better memorization (over fitting).

Using the aforementioned *rule of thumb* as a rough guideline, I decided to test the effect of varying the number of hidden neurons from 1 to 10. For all ten of those different values, I trained the NN using *all* of the data and then tested it also using *all* of the data. The results of this could be seen below (Table 1):

Table 1: Classification Rate with Varying Number of Hidden Neurons

# Hidden Neurons	Classification Rate
1	27.5%
2	32.1%
3	36.1%
4	36.3%
5	36.9%
6	41.8%
7	40.4%
8	16.3%
9	41.8%
10	30.5%

The NN with 6-neurons and 9-neurons in the hidden layer performed the best with a 41.8% classification rate, greatly outperforming random-guessing (a classification rate of

10% would be expected using random guessing). However, this classification rate was achieved using the same data to both train and test the network, which could bias the results. A better technique in evaluating a model's predictive ability would be to use a cross-validation technique called LOOM. Another motivation for using cross-validation was to test the theory that the less number of neurons in the hidden layer will actually lead to better generalization from the NN. The result is below (Table 2).

Table 2: Classification Rate with Varying Number of Hidden Neurons (LOOM)

# Hidden Neurons	Classification Rate
3	30.1%
6	29.5%
9	29.1%

The classification rates got worse in each of the three cases (but still performed better than random guessing). The classification rates for the 6-neuron network and 9-neuron network dropped over 10% while the 3-neuron network only dropped 5%. These drops in classification rates were to be expected since "new data" is being input to the NN. The theory that the smaller number of neurons in the hidden layer leads to better network generalization when predicting new data seems plausible.

Results

The model architecture that I decided to use was the one that had three neurons in the hidden layer. I chose this architecture because it had the highest overall classification rate (30.1%) using cross-validation. The classification rate fluctuated substantially between the different sectors. The NN had the highest classification rates when predicting the following sectors: Utilities (63.6%), Information Technology (42.5%), Consumer Staples (38.1%) and Industrials (35.1%). Lowest on the spectrum of classification rates by the NN were: Telecommunication Services (0.0%) and Materials (3.2%). Telecommunication Services were commonly miss-classified as belonging to the Utilities sector (50.0%) or the Information Technology sector (20.0%). There wasn't as strong of a pattern in the miss-classifications of the Materials sector.

However the goal of this classification analysis was to see if individual assets might belong in a different sector or maybe belong to more than one sector. So it would make sense to explore the miss-classifications at the asset level. I was provided with a list of several assets (that didn't belong in the S&P500) to classify. Included in these assets was a SPDR for nine of the ten sectors. The NN correctly predicted 3 of the 9 SPDRs (Table: 3).

Table 3: Asset Level Probabilities for Selected Assets

Asset	Actual Sector (Prob.*)	Predicted Sector (Prob.*)
XLY	Consumer Discretionary (13.8%)	Consumer Staples (31.5%)
XLP	Consumer Staples (44.0%)	Consumer Staples (44.0%)
XLE	Energy (4.31%)	Health Care (22.0%)
XLF	Financials (16.5%)	Materials (46.1%)
XLV	Health Care (6.55%)	Utilities (50.2%)
XLI	Industrials (6.40%)	Consumer Staples (50.7%)
XLB	Materials (31.2%)	Materials (31.2%)
XLK	Information Technology (0.05%)	Consumer Staples (43.3%)
XLU	Utilities (65.1%)	Utilities (65.1%)

* - The approximate a posteriori class probabilities

The NN gave a high probability of the XLU belonging to the Utilities sector, while having a probability below 1% of XLK belonging to the Information Technology sector. The most egregious miss-classification out of these assets was the prediction for Treehouse Foods Inc (THS). The NN gave a class probability of belonging to its actual sector, Consumer Staples, of 0.03%. Instead the NN classified THS as belonging to the Health Care sector (25.5%).

Asset Jump Classification

I wanted to use a NN for was to predict if an asset was going to go up or down in price, using just historical, publicly available data for the asset. To accomplish this, I chose four assets and scraped data on each asset from October 20th, 2004 to April 28th, 2010. The four assets I chose to investigate on were: Materials Select Sector SPDR (XLB), Industrial Select Sector SPDR (XLI), Iconix Brand Group, Inc. (ICON), and Ormat Technologies Inc. (ORA).

Network Inputs and Outputs

I tested a couple of different models but they all consisted of the same basic ingredients. The first of such is a moving average of return (lagged *d* days) which is defined as:

$$\bar{r}_{t,d} = \frac{\sum_{t-d}^{t} r_t}{d}$$

Similarly, moving average of floating volume (lagged *d* days) is defined as,

$$\bar{f}_{t,d} = \frac{\sum_{t-d}^{t} f_t}{d}$$

where floating volume (at time t) is defined as:

$$f_t = \frac{v_t}{V}$$

The output is a vector of approximate *a posteriori* class probabilities for each of the three different Jump Signals, where Jump Signal is defined as:

$$y_t = \begin{cases} 1\ , r_t \geq U \\ -1,\ r_t \leq D \\ 0\ ,\ \ else \end{cases}$$

where U is the Up-Jump boundary and D is the Down-Jump boundary. These boundary values were provided to me. (Frame & Ramezani (2010), 'Bayesian Estimation of the Jump-Diffusion Processes' to be submitted to Annals of Finance)

Exploring Training Window Size

Besides predicting the jump signal, I wanted to investigate the effect on the results of changing the size of the training window. So I decided to test three different training window sizes (200, 400, 600), while keeping the size of the validation set at 25% the length of the training set. Figure 2 below gives a visual representation of the training process for each asset.

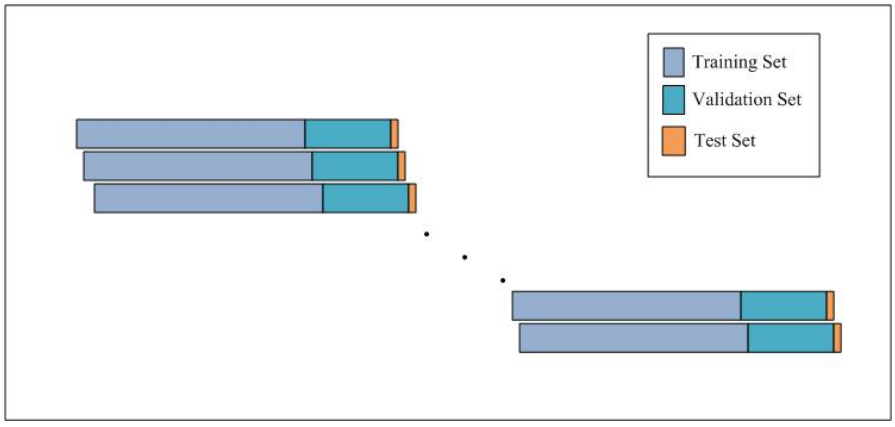

Figure 2: The Training Process of the NN for Each Asset

Basically the NN is trained on a training set using several different network architectures. Using the *rule of thumb* as a guideline, I would vary the number of hidden neurons from two below the suggested number and two above it. So I would be testing five different networks each time. Then whichever network performs the best on the validation set is used to predict the single data point of the Test Set. Once that prediction is made, everything slides down one day's time, and the process is repeated. Since this was a computational heavy process, I only repeated that process 500 times per asset (for each size of the training window).

Table 4 shows that for three of the four assets, a training window of size 400 performed the best (using Model A, see *Results* below for model specification). The NN classified XLI substantially worse than the other three assets.

Table 4: Classification Rates with Varying Training Window Sizes Using Model A

Asset	200	400	600
XLB	44.8%	48.8%	44.8%
XLI	36.0%	36.8%	35.0%
ICON	47.2%	49.8%	47.4%
ORA	45.0%	45.4%	47.2%

Results

With a Training Set size of 400 performing the best on three of the four assets, that was the size I used when I searched for the optimal combination of inputs into the NN. The first model I tested was the one I used when varying the training window sizes. It consisted of 8 inputs: moving-average of return (lagged 15, 30 and 200 days), moving-averages of floating volume (lagged 15, 30 and 200 days), the previous days' return and previous days' floating volume. I considered this model my base-line model and compared the performance of the other models I tested against it.

Model B consisted of the following 6 inputs: moving-average of return (lagged 5, 15, and 30 days), moving-averages of floating volume (lagged 5, 15, and 30days). Table 5 below shows that for all four assets, this model performed worse than Model A.

Table 5: Classification Rates for Model B

	Classification Rate	Difference from Model A
XLB	45.8%	-3.00%
XLI	35.8%	-1.00%
ICON	48.0%	-1.80%
ORA	45.0%	-0.40%

Model C included just the variables using return data. These inputs were: Moving-averages of return (lagged 5, 15, 30, and 200 days) and the previous day's return. Overall this model performed better than Model A as seen below in Table 6.

Table 6: Classification Rates for Model C

	Classification Rate	Difference from Model A
XLB	48.4%	-0.4%
XLI	41.0%	4.2%
ICON	47.8%	-2.0%
ORA	48.4%	3.0%

Model D included just the variables using floating volume data. These inputs were: Moving-averages of floating volume (lagged 5, 15, 30, and 200 days) and the previous day's floating. This model didn't perform as well as Model C did but it had the highest classification rate for any individual asset with a 50.2% classification rate (Table 7).

Table 7: Classification Rates for Model D

	Classification Rate	Difference from Model A
XLB	48.0%	-0.80%
XLI	35.6%	-1.20%
ICON	50.2%	0.40%
ORA	45.4%	0.00%

Overall, Model C performed the best across the board with classification rates ranging between 41% and 48%. Random guessing would be expected to yield a classification rate of 33%, so Model C is better than random guessing.

Conclusion

Using a Neural Network as a basis of modeling performed well in both of the classification problems. Even though some assets might not fit nicely into one of ten disjoint sectors, the NN correctly classified assets at about a rate of 30% (three times the rate of random guessing). It also provided alternative classifications for individual assets which could be used to better optimize portfolios. For Asset Jump Prediction, the results were mixed. The NN had a classification rate ranging between 35.6% (barely better than random guessing) and 50.2% depending on what set of inputs were used.

Improvements for Future Studies

The first thing that I would improve is to increase the number of assets used. Due to the computing time, I had to cut back on the number of assets I tried to predict. In this project, I just used publicly available price and volume data. Utilizing data from other sources such as blogs or financial statements might be beneficial.

References

Harvey, Robert L. *Neural Network Principles*. Englewood Cliffs, NJ: Prentice Hall, 1994.

Haykin, Simon S. *Neural Networks: a Comprehensive Foundation*. New York: Macmillan, 1994.