

Statistical Arbitrage Stock Trading using Time Delay Neural Networks

Chris Pennock

Final Project, Machine Learning, Fall 2004

Instructor: Yann Le Cun

Introduction:

Can a TDNN be used to profit from the stock market? That question is the motivation for this project. More specifically, this project will explore whether there is some TDNN architecture that, having trained on a stock's past data, can accurately predict when to buy a stock.

The Efficient Markets Hypothesis states that "In an informationally efficient market, price changes must be unforecastable if they are properly anticipated, i.e., if they fully incorporate the expectations and information of all market participants" [1]. If true, success in this project will involve finding inefficiency in the market – patterns that the human players in the market do not capitalize on.

With that challenge in mind, the aim of this project is to experiment with a number of TDNN architectures and with a number of segments of data from a number of kinds of stocks, to determine if some combination thereof is learnable, and hence profitable.

Two broad categories of TDNN architectures were tested. The first was a 2-layer network with one convolutional layer and one fully-connected layer. The second was a 3-layer network, with two convolutional layers and one fully-connected layer.

The trading strategy learned by the networks was as follows: If at the end of a sequence of 50 daily closing prices, the price will go up by 2% or more over the next 20 days, buy it. This strategy was chosen as a balance between two opposing forces. On one hand, it is easier to predict the movement of a time series a short period into the future. On the other hand, there are costs associated with making a trade, so trading too frequently is undesirable. This is described in [2].

The Dataset:

Thirteen different segments of stock data were used. They were chosen because they varied in length, age, and industry, and because it was thought that these factors might affect the types of patterns contained in them and hence the performance of a learning machine. The length of each segment was either 500 or 1220 trading days. It was thought that a longer segment might simply provide more training data, or conversely that a shorter segment might contain patterns more relevant to the near future movement of the stock. The age of the segments was varied from 1975 to present, since perhaps the methods of trading had changed over time, perhaps becoming more efficient, which would change the nature of the patterns that had not been capitalized upon. Finally, both tech stocks and non-tech stocks were used, in case tech stocks proved to be more volatile, which might affect learning.

The raw closing price data was split into 50-day windows. For example, if the raw data was [1 2 3 4 5], and the window size 3, the samples generated would be [1 2 3], [2 3

4], [3 4 5]. The data was then split into training and test sets, with approximately 100 trading days in the test set, and the remainder in the training set. This split was done chronologically, rather than as a random sampling of the data, since the intended application of this learning machine would be to learn from recent contiguous stock data and predict the trading action for the near future.

The data was split into windows because the alternative to having windows was to simply let the machine learn the entire training set as one sample, essentially using the kernel size as the window size. This was undesirable for two reasons. First, only by using windows could the training data could be shuffled, likely resulting in better learning. Second, based on the trading strategy, the machine would need to learn how a sequence of daily stock prices predicted the movement of the stock beyond that sequence, specifically whether or not the stock would move up by 2% within a 20 day period after the end of the 50 day window. By using separate window and kernel sizes, the machine could learn patterns within a window, and these patterns could be anywhere within the window. This seemed appropriate, given that the 2% increase could be anywhere within the 20 days following the sequence.

Finally, labels were then generated for each sample. A 1 was assigned to a sample if the stock would move up by 2% during the 20 days after the end of the sample. Otherwise, it was assigned a 0.

Architecture of the Machine:

Two separate machines were used for this project: a TDNN with one convolutional layer and one fully-connected layer, and a TDNN with two convolutional layers and one fully-connected layer. Both machines used a simple Euclidean cost module, and learned via gradient-descent backpropagation through the cost module and each of the layers.

The single-convolutional-layer TDNN consists of two major subcomponents: a convolutional layer and a fully connected layer, as shown in Figure 1. The convolutional layer had a kernel size of 20 days, and 8 feature detectors. The kernel size was chosen to be 20 because of its similarity to the period of the trading strategy, and because this was small enough to be meaningful in relation to the window size of 50. The fully-connected layer is straightforward, with a connection from each hidden unit in each feature detector in the convolutional layer to each output of the machine. There were two outputs, representing the two labels to which the samples would be assigned.

In the TDNN with two convolutional layers, an additional convolutional layer was inserted between the two layers of the 2-layer TDNN. The intermediate convolutional layer took as input the feature detectors of the first convolutional layer. Its outputs were the inputs of the fully-connected layer, as described above. Figure 2 contains a graphical representation of this. Two variations of the 3-layer machine were used, which differed only in the kernel size of the first convolutional layer.

One variation used a kernel size of 10 in the first layer and 20 in the second layer. The goal in using two different kernel sizes was to capture patterns at multiple periodicities. The kernel sizes were chosen to differ by a factor of two because a factor much larger or smaller would be undesirable. A larger factor would make one kernel either too small to capture any meaningful patterns, or too large relative to the size of the window. A smaller factor would make the kernels too similar, and cause redundancy.

The other variation of 3-layer TDNN used a kernel size of 20 in both convolutional

layers. These kernel sizes were chosen because they would provide a nice comparison to both the 2-layer network, which also had a single kernel size of 20, and to the 3-layer network, which had one kernel of size 20 and one half that size.

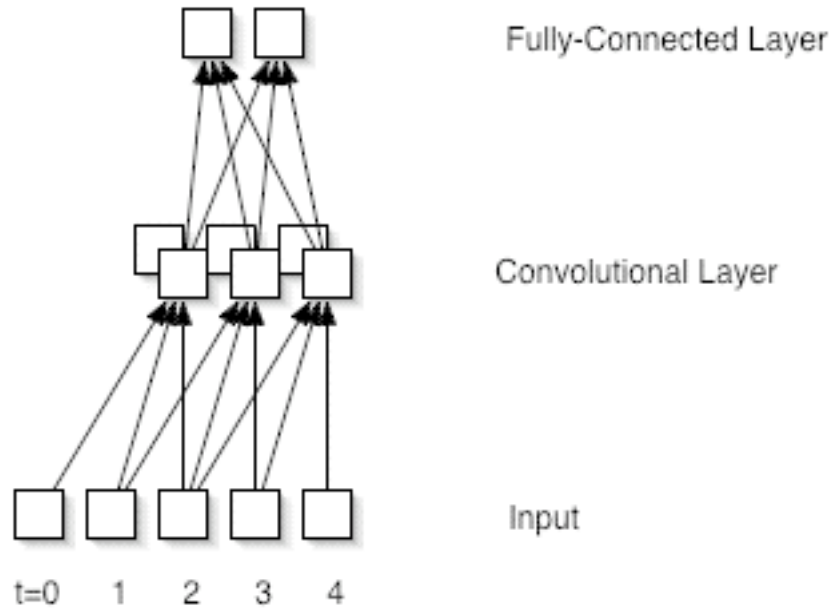


Figure 1

The architecture of a 2-layer TDNN. The second row of hidden units in the convolutional layer represents a second feature detector. Although its connections have been removed for clarity, all feature detectors would have the same connection pattern, as shown in the front feature detector.

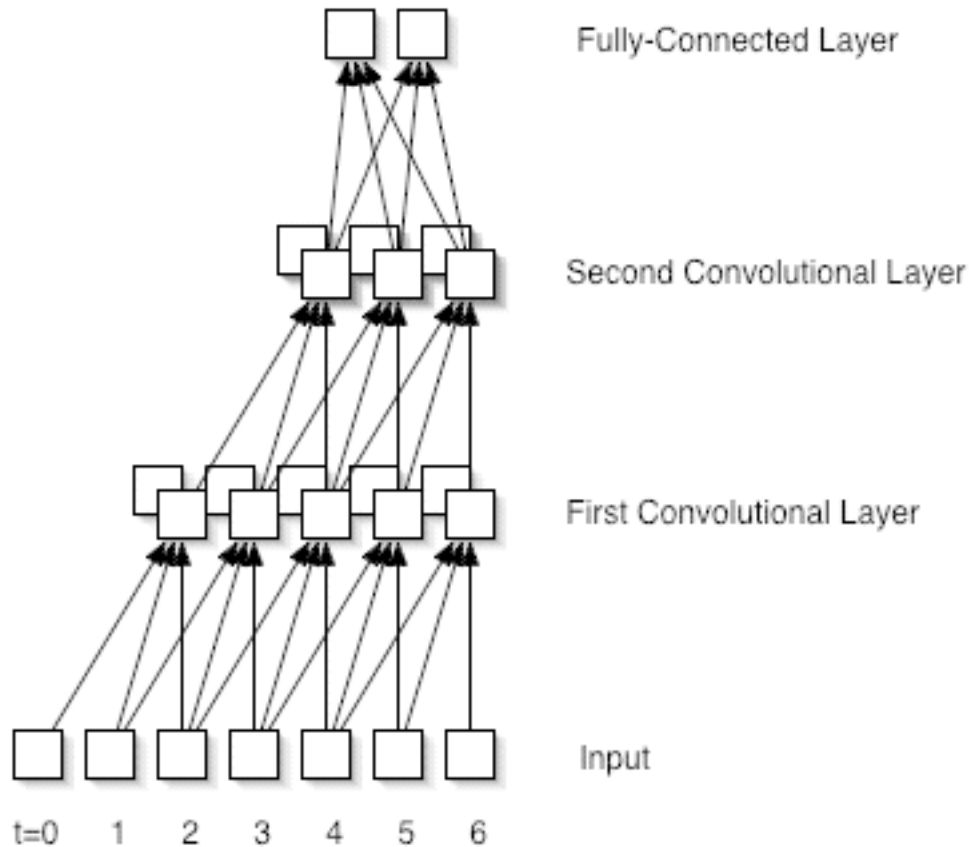


Figure 2

The architecture of a 3-layer TDNN. The second row of hidden units in each convolutional layer represents a second feature detector. Although their connections have been removed for clarity, all feature detectors would have the same connection pattern, as shown in the front feature detectors.

Experiments Performed:

For each combination of stock and learning machine, the machine was trained on the training set of the stock data. The machine was then tested on the test set, and the percent of samples correctly predicted was recorded. The results can be seen in Table 1.

A second test measured the profit that would be generated by running the trained machine on the test set and using the trading strategy to simulate buying and selling the stock. The purpose of this test was to tie the performance of the learning machine more closely to its intended environment, stock trading, and to estimate its usefulness in the real world. The results of this test can also be seen in Table 1.

The profit was calculated by making an algorithm around the trading strategy that was used to generate the labels, running the machine on the test samples, and running the output of the machine through this algorithm. The trading strategy for this test was as follows:

- For a given sample, if the machine outputs a buy, buy the stock.
- If the stock increases by 2% during the 20 day holding period, sell it at that time.
- If the stock does not go up by the required amount, sell it at the end of the holding period.

- Sell the stock at the end of the test samples, if the stock is owned.

The profit was then measured as the percent of the initial capital that existed at the end of the test samples. This was calculated by multiplying the each of the percent gains from each sale of stock. Also, this percentage was extrapolated to a yearly percentage, by exponentiating the percent profit by the ratio of trading days in a year divided by the trading days in the test set. These percentages were then compared to the actual movement of the stock over that period, and to the movement of the stock extrapolated to a yearly figure.

Table 1

| Stock/Machine | 2-layer (20 8) | 3-Layer (20 8 20 8) | 3-Layer (10 8 20 8) | Actual Profit |
|-------------------|----------------|---------------------|---------------------|---------------|
| NOK 02-04 | | | | |
| Train % | 70.94 | 79.58 | 73.82 | |
| Test % | 54.17 | 55.21 | 56.25 | |
| Test Profit | 14.35 | 20.02 | 14.35 | 28.74 |
| Annual Profit | 32.24 | 46.26 | 32.24 | 69.28 |
| AAPL 92-94 | | | | |
| Train % | 50.39 | 66.75 | 65.45 | |
| Test % | 11.34 | 89.69 | 51.55 | |
| Test Profit | 0 | 55.56 | 33.81 | 35.72 |
| Annual Profit | 0 | 148.70 | 82.32 | 87.71 |
| AAPL 94-96 | | | | |
| Train % | 69.01 | 74.74 | 74.74 | |
| Test % | 61.86 | 62.89 | 60.82 | |
| Test Profit | -26.08 | -23.06 | -23.06 | -18.62 |
| Annual Profit | -46.37 | -41.77 | -41.77 | -34.62 |
| AAPL 00-02 | | | | |
| Train % | 67.53 | 76.62 | 62.86 | |
| Test % | 19.59 | 87.63 | 14.43 | |
| Test Profit | 2.03 | 35.05 | 3.33 | 33.55 |
| Annual Profit | 4.24 | 85.82 | 6.98 | 81.59 |
| IBM 03-05 | | | | |
| Train % | 60.96 | 69.38 | 60.67 | |
| Test % | 73.33 | 70.00 | 85.56 | |
| Test Profit | 14.76 | 14.76 | 17.59 | 17.24 |
| Annual Profit | 35.81 | 35.81 | 43.36 | 42.41 |
| IBM 85-90 | | | | |
| Train % | 68.13 | 69.52 | 69.80 | |
| Test % | 71.67 | 69.17 | 62.50 | |
| Test Profit | 7.50 | -7.29 | -3.63 | -1.48 |
| Annual Profit | 12.81 | -11.86 | -5.97 | -2.45 |
| IBM 75-80 | | | | |
| Train % | 64.86 | 65.61 | 66.64 | |
| Test % | 58.33 | 41.67 | 70.83 | |
| Test Profit | -0.49 | 0 | 5.25 | -6.29 |
| Annual Profit | -0.83 | 0 | 8.91 | -10.27 |
| GE 75-80 | | | | |
| Train % | 58.25 | 68.41 | 68.87 | |
| Test % | 80.83 | 76.67 | 80.83 | |
| Test Profit | 2.96 | -6.58 | 2.16 | -4.54 |
| Annual Profit | 4.98 | -10.73 | 3.63 | -7.46 |

Results for learning a number of segments of stock data on a number of learning machines. The numbers following the stock symbol indicate the time period of the stock segment. Train % is the percent of training samples that the learning machine predicted correctly, and Test % is the same for the test samples. Test Profit is the percent profit that the trained learning machine would generate during the test samples, and Annual Profit extrapolates that percentage to an annual return rate.

The machine 2-Layer (20 8) is a learning machine with one convolutional layer and one fully-connected layer, with a kernel size of 20 and 8 feature detectors. The machines 3-Layer (20 8 20 8) and 3-Layer (10 8 20 8) have 2 convolutional layers and one fully connected layer, with kernel sizes 20 and 10, respectively, in their first layers, and 20 in their second layers.

Table 1, continued

| Stock/Machine | 2-layer (20 8) | 3-Layer (20 8 20 8) | 3-Layer (10 8 20 8) | Actual Profit |
|-------------------|----------------|---------------------|---------------------|---------------|
| AAPL 90-95 | | | | |
| Train % | 74.75 | 74.57 | 74.57 | |
| Test % | 84.00 | 84.00 | 84.00 | |
| Test Profit | 27.16 | 27.16 | 27.16 | 20.54 |
| Annual Profit | 46.89 | 46.89 | 46.89 | 34.84 |
| GE 00-02 | | | | |
| Train % | 56.32 | 59.21 | 57.37 | |
| Test % | 30.21 | 47.92 | 43.75 | |
| Test Profit | 0 | 4.95 | -4.08 | 22.81 |
| Annual Profit | 0 | 10.59 | -8.31 | 53.43 |
| GE 90-95 | | | | |
| Train % | 72.69 | 72.69 | 73.50 | |
| Test % | 81.60 | 81.60 | 80.80 | |
| Test Profit | 13.32 | 13.32 | 23.34 | 20.04 |
| Annual Profit | 22.16 | 22.16 | 39.89 | 39.95 |
| IBM 72-74 | | | | |
| Train % | 60.42 | 64.58 | 61.72 | |
| Test % | 43.75 | 41.67 | 43.75 | |
| Test Profit | 0 | -4.16 | 0 | 8.75 |
| Annual Profit | 0 | -8.49 | 0 | 13.36 |
| GE 70-72 | | | | |
| Train % | 73.71 | 77.58 | 76.55 | |
| Test % | 49.48 | 53.61 | 32.99 | |
| Test Profit | -3.21 | -4.07 | -8.04 | -3.43 |
| Annual Profit | -6.52 | -8.21 | -15.88 | -6.94 |

Results and Conclusion:

The experimental results indicate that a TDNN can learn to predict patterns in a series of stock closing prices. Of the 39 combinations of stock segment and learning machine, only 12 test sets were predicted at less than 50% correct. Even where the test set prediction rate was just above chance, this is of interest, since every percent above chance can be potentially be translated into profit. On average, though, the profit produced by these learning machines on the test data was not high enough to warrant using this system for actual arbitrage trading. However, in the performance of these learning machines, a number of overall trends were observed, which perhaps could be used to profit more consistently.

First, the longer training sets produced better results. Among the stock data segments of 1220 days, no machine produced a catastrophically low percent correct (<20%) on the test set. Also, 10 of 15 combinations of machine and long segment produced a profit better than the movement of the stock itself, vs. 4 of 24 for the shorter segments.

Second, the 3-layer networks performed slightly better than that of the 2-layer network. The 3-layer networks had less likelihood of catastrophically low test set performance (1 out of 26 vs. 3 out of 13 for the 2-layer networks, all of which occurred in the shorter 500 day segments.). Aside from these cases, though, the performance of the 2- and 3-layer networks was roughly equivalent.

Other differences between trials, such as industry and age of stock segment, had no discernable effect.

These trends are promising, and would seem to indicate that by training a machine on the proper amount of data and by using the right type of learning machine, one might be able to beat the performance of an individual stock, on average.

So, in response to the question put forth by this project, “Can a TDNN be used to profit in the stock market?”, the answer is a qualified Yes. The machines produced by this project would be a risky investment vehicle, to be sure. But these very preliminary results indicate that they could be profitable on average.

References:

- [1] Farmer, J. and Lo, A. Frontiers of finance: Evolution and efficient markets. In *Proceedings of the National Academy of Sciences*. Vol. 96, August 1999.
- [2] Saad, E. et al. Comparative Study of Stock Trend Prediction Using Time Delay, Recurrent and Probabilistic Neural Networks. In *IEEE Transactions on Neural Networks*, Vol. 9, No. 6, November 1998.