A Survey of Deep Learning Techniques Applied to Trading

Published on July 31, 2016

by Greg Harris

http://gregharris.info/a-survey-of-deep-learning-techniques-applied-to-trading/

Deep learning has been getting a lot of attention lately with breakthroughs in image classification and speech recognition. However, its application to finance doesn't yet seem to be commonplace. This survey covers what I've found so far that is relevant to systematic trading. Please tell me if you know of some research I've missed.

Acronyms:

DBN = Deep Belief Network

LSTM = Long Short-Term Memory

MLP = Multi-layer Perceptron

RBM = Restricted Boltzmann Machine

ReLU = Rectified Linear Units

CNN = Convolutional Neural Network

Limit Order Book Modeling

Sirignano (2016) predicts changes in limit order books. He has developed a "spatial neural network" that can take advantage of local spatial structure, is more interpretable, and more computationally efficient than a standard neural network for this purpose. He models the joint distribution of the best bid and ask at the time of the next state change. Also, he models the joint

distribution of the best bid and ask prices upon the change in either of them.

Architecture – Each neural network has 4 layers. The standard neural network has 250 neurons per hidden layer, and the spatial neural network has 50. He uses the tanh activation function on the hidden layer neurons.

Training – He trained and tested on order books from 489 stocks from 2014 to 2015 (a separate model for each stock). He uses Level III limit order book data from the NASDAQ with event times having nanosecond decimal precision. Training involved 50TB of data and used a cluster with 50 GPUs. He includes 200 features: the price and size of the limit order book across the first 50 non-zero bid and ask levels. He uses dropout to prevent overfitting. He uses batch normalization between each hidden layer to prevent internal covariate shift. Training is done with the RMSProp algorithm. RMSProp is similar to stochastic gradient descent with momentum but it normalizes the gradient by a running average of the past gradients. He uses an adaptive learning rate where the learning rate is decreased by a constant factor whenever the training error increases over a training epoch. He uses early stopping imposed via a validation set to reduce overfitting. He also includes an l^2 penalty when training in order to reduce overfitting.

Results – He shows that limit order books exhibit some degree of local spatial structure. He predicts the order book 1 second ahead and also at the time of the next bid/ask change. The spatial neural network outperforms the standard neural network and logistic regression with non-linear features. Both neural networks have 10% lower error than logistic regression.

Price-based Classification Models

Dixon et al. (2016) use a deep neural network to predict the sign of the price change over the next 5 minutes for 43 commodity and forex futures.

Architecture – Their input layer has 9,896 neurons for input features made up of lagged price differences and co-movements between contracts. There are 5 learned fully-connected layers. The first of the four hidden layers contains 1,000 neurons, and each subsequent

layer tapers by 100 neurons. The output layer has 135 neurons (3 for each class {-1, 0, 1} times 43 contracts).

Training – They used the standard back-propagation with stochastic gradient descent. They speed up training by using mini-batching (computing the gradient on several training examples at once rather than individual examples). Rather than an nVidia GPU, they used an Intel Xeon Phi co-processor.

Results – They report 42% accuracy, overall, for three-class classification. They do some walk-forward training instead of a traditional backtest. Their boxplot shows some generally positive Sharpe ratios from the mini-backtests for each contract. They did not include transaction costs or crossing the bid-ask spread. All their predictions and features were based on the mid-price at the end of each 5-minute time period.

Takeuchi and Lee (2013) look to enhance the momentum effect by predicting which stocks will have higher or lower monthly returns than the median.

Architecture – They use an auto-encoder composed of stacked RBMs to extract features from stock prices which they then pass to a feed-forward neural network classifier. Each RBM consists of one layer of visible units and one layer of hidden units connected by symmetric links. The first layer has 33 units for input features from one stock at a time. For every month t, the features include the 12 monthly returns for month t-2 through t-13 and the 20 daily returns approximately corresponding to month t. They normalize each of the return features by calculating the z-score relative to the cross-section of all stocks for each month or day. The number of hidden units in the final layer of the encoder is sharply reduced, forcing dimensionality reduction. The output layer has 2 units, corresponding to whether the stock ended up above or below the median return for the month. Final layer sizes are 33-40-4-50-2.

Training – During pre-training, they split the dataset into smaller, non-overlapping mini-batches. Afterwards, they un-roll the RBMs to form an encoder-decoder, which is fine-tuned using back-propagation. They consider all stocks trading on the NYSE, AMEX, or NASDAQ with a price greater than $5. They train on data from 1965 to 1989 (848,000 stock-month samples) and test on data from 1990 to 2009 (924,300 stock-month samples). Some training data held-out for validation for the number of layers and the number of units per layer.

Results – Their overall accuracy is around 53%. When they consider the difference between the top decile and the bottom decile predictions, they get 3.35% per month, or 45.93% annualized return.

Batres-Estrada (2015) predicts which S&P 500 stocks will have above-median returns for each given day, and his work appears to be heavily influenced by Takeuchi and Lee (2013).

Architecture – He uses a 3-layer DBN coupled to an MLP. He uses 400 neurons in each hidden layer, and he uses a sigmoid activation function. The output layer is a softmax layer with two output neurons for binary classification (above median or below). The DBN is composed of stacked RBMs, each trained sequentially.

Training – He first pre-trains the DBN module, then fine-tunes the entire DBN-MLP using back-propagation. The input includes 33 features: monthly log-returns for months t-2 to t-13, 20 daily log-returns for each stock at month t, and an indicator variable for the January effect. The features are normalized using the Z-score for each time period. He uses S&P 500 constituent data from 1985 to 2006 with a 70-15-15 split for training-validataion-test. He uses the validation data to choose the number of layers, the number of neurons, and the regularization parameters. He uses early-stopping to prevent over-fitting.

Results – His model has 53% accuracy, which outperforms regularized logistic regression and a few MLP baselines.

Sharang and Rao (2015) use a DBN trained on technical indicators to trade a portfolio of US Treasury note futures.

Architecture – They use a DBN consisting of 2 stacked RBMs. The first RBM is Gaussian-Bernoulli (15 nodes), and the second RBM is Bernoulli (20 nodes). The DBN produces latent features which they try feeding into three different classifiers: regularized logistic regression, support vector machines, and a neural network with 2 hidden layers. They predict 1 if portfolio goes up over 5 days, and -1 otherwise.

Training – They train the DBN using a contrastive divergence algorithm. They calculate signals based on open, high, low, close, open interest, and volume data, beginning in 1985, with some points removed during the 2008 financial crisis. They use 20 features: the “daily trend” calculated over different time frames, and then normalized. All parameters are chosen using a validation dataset. When training the neural net classifier, they mention using a

momentum parameter during mini-batch gradient descent training to shrink the coefficients by half during every update.

Results – The portfolio is constructed using PCA to be neutral to the first principal component. The portfolio is an artificial spread of instruments, so actually trading it is done with a spread between the ZF and ZN contracts. All input prices are mid-prices, meaning the bid-ask spread is ignored. The results look profitable, with all three classification models performing 5-10% more accurately than a random predictor.

Zhu et al. (2016) make trade decisions using oscillation box theory based on DBNs. Oscillation box theory says that a stock price will oscillate within a certain range in a period of time. If the price moves outside the range, then it enters into a new box. The authors try to predict the boundaries of the box. Their trading strategy is to buy the stock when it breaks through the top boundary or sell it when it breaks through the bottom boundary.

Architecture – They use a DBN made up of stacked RBMs and a final back-propagation layer.

Training – They used block Gibbs sampling to greedily train each layer from lowest to highest in an unsupervised way. They then train the back-propagation layer in a supervised way, which fine-tunes the whole model. They chose 400 stocks out of the S&P 500 for testing, and the test set covers 400 days from 2004 to 2005. They use open, high, low, close prices as well as technical analysis indicators, for a total of 14 model inputs. Some indicators are given more influence in the prediction through the use of “gray relation analysis” or “gray correlation degree.”

Results – In their trading strategy, they charge 0.5% transaction costs per trade and add a couple of parameters for stop-loss and “transaction rate.” I don’t fully understand the result tables, but they seem to be reporting significant profits.

Text-based Classification Models

Rönnqvist and Sarlin (2016) predict bank distress using news articles. Specifically, they create a classifier to judge whether a given sentence indicates distress or tranquility.

Architecture – They use two neural networks in this paper. The first is for semantic pre-training to reduce dimensionality. For this, they run a sliding window over text, taking a sequence of 5 words and learning to predict the next word. They use a feed-forward topology where a projection layer in the middle provides the semantic vectors once the connection weights have been learned. They also include the sentence ID as an input to the model, to provide context and inform the prediction of the next word. They use binary Huffman coding to map sentence IDs and word to activation patterns in the input layer, which organizes the words roughly by frequency. They say feed-forward topologies with fixed context sizes are more efficient than recurrent neural networks for modeling text sequences. The second neural network is for classification. Instead of a million inputs (one for each word), they use 600 inputs from the learned semantic model. The first layer has 600 nodes, the middle layer has 50 rectified linear hidden nodes, and the output layer has 2 nodes (distress/tranquil).

Training – They train it with 243 distress events over 101 banks observed during the financial crisis of 2007-2009. They use 716k sentences mentioning the banks, taken from 6.6m Reuters news articles published during and after the crisis.

Results – They evaluate their classification model using a custom "Usefulness" measure. The evaluation is done using cross-validation, leaving N banks out in each fold. They aggregate the distress counts into various timeseries but don't go so far as to consider creating a trading strategy.

Fehrer and Feuerriegel (2015) train a model to predict German stock returns based on headlines.

Architecture – They use a recursive autoencoder with an additional softmax layer in each autoencoder for estimating probabilities. They perform three-class prediction {-1, 0, 1} for the following day's return of the stock associated with the headline.

Training – They initialize the weights with Gaussian noise, and then update through back-propagation. They use an English ad-hoc news announcement dataset (8,359 headlines) for the German market covering 2004 to 2011.

Results – Their recursive autoencoder has 56% accuracy, which in an improvement over a more traditional random forest modeling approach with 53% accuracy. They do not develop a trading strategy.

They have made a Java implementation of their code publicly available.

Ding et al. (2015) use structured information extracted from headlines to predict daily S&P 500 moves. Headlines are processed with Open IE to obtain structured event representations (actor, action, object, time). A neural tensor network learns the semantic compositionality over event arguments by combining them multiplicatively instead of only implicitly, as with standard neural networks.

Architecture – They combine short-term and long-term effects of events, using a CNN to perform semantic composition over the input event sequence. They use a max pooling layer on top of the convolutional layer, which makes the network retain only the most useful features produced by the convolutional layer. They have separate convolutional layers for long-term events and mid-term events. Both of these layers, along with an input layer for short-term events, feed into a hidden layer which then feeds into two output nodes.

Training – They extracted 10 million events from Reuters and Bloomberg news. For training, they corrupt events by replacing one event argument with a random argument. During training, they assume that the actual event should be given a higher score than the corrupted event. When it isn’t, model parameters get updated.

Results – They find that structured events are better features than words for stock market prediction. Their approach outperforms baseline methods by 6%. They make predictions for the S&P 500 index and 15 individual stocks, and a table appears to show that they can predict the S&P 500 with 65% accuracy.

Volatility Prediction

Xiong et al. (2015) predict the daily volatility of the S&P 500, as estimated from open, high, low, close prices.

Architecture – They use a single LSTM hidden layer consisting of one LSTM block. For inputs they use daily S&P 500 returns and volatilities. They also include 25 domestic Google trends, covering sectors and major areas of the economy.

Training – They used the “Adam” method with 32 samples per batch and mean absolute percent error (MAPE) as the objective loss function. They set the maximum lag of the LSTM to include 10 successive observations.

Results – They show their LSTM method outperforms GARCH, Ridge, and LASSO techniques.

Portfolio Optimization

Heaton et al. (2016) attempt to create a portfolio that outperforms the biotech index IBB. They have the goal of tracking the index with few stocks and low validation error. They also try to beat the index by being anti-correlated during periods of large drawdowns. They don’t directly model the covariance matrix, rather it is trained in the deep architecture fitting procedure, which allows for nonlinearities.

Architecture – They use auto-encoding with regularization and ReLUs. Their auto-encoder has one hidden layer with 5 neurons.

Training – They use weekly return data for the component stocks of IBB from 2012 to 2016. They auto-encode all stocks in the index and evaluate the difference between each stock and its auto-encoded version. They keep the 10 most “communal” stocks that are most similar to the auto-encoded version. They also keep a varying number of other stocks, where the number is chosen with cross-validation.

Results – They show the tracking error as a function of the number stocks included in the portfolio, but don’t seem to compare against traditional methods. They also replace index drawdowns with positive returns and find portolios that track this modified index.

Related Links

Stock Market Forecasting using deep learning? (reddit.com)

deep learning in finance (reddit.com)

Leveraging Google DeepMind software and Deep Learning to play the stock market(reddit.com)

They might be Robots, Using data for investment management and trading (youtube.com)

Introducing Binatix: A Deep Learning Trading Firm That's Already Profitable (recode.net)

Will AI-Powered Hedge Funds Outsmart the Market? (technologyreview.com)

Adaptive deep learning empowers traders (automatedtrader.net)

Deep Learning in Finance (slideshare.net)

Prediction of Exchange Rate Using Deep Neural Network (slideshare.net)

References

Batres-Estrada, B. (2015). Deep learning for multivariate financial time series. abstract

Ding, X., Zhang, Y., Liu, T., & Duan, J. (2015, June). Deep learning for event-driven stock prediction. In Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (ICJAI) (pp. 2327-2333). abstract

Dixon, M. F., Klabjan, D., & Bang, J. H. (2016). Classification-based Financial Markets Prediction using Deep Neural Networks. Available at SSRN 2756331. abstract

Fehrer, R., & Feuerriegel, S. (2015). Improving Decision Analytics with Deep Learning: The Case of Financial Disclosures. arXiv preprint arXiv:1508.01993. abstract

Heaton, J. B., Polson, N. G., & Witte, J. H. (2016). Deep Portfolio Theory. arXiv preprint arXiv:1605.07230. abstract

Rönnqvist, S., & Sarlin, P. (2016). Bank distress in the news: Describing events through deep learning. arXiv preprint arXiv:1603.05670. abstract

Sharang, A., & Rao, C. (2015). Using machine learning for medium frequency derivative portfolio trading. arXiv preprint arXiv:1512.06228. abstract

Sirignano, J. A. (2016). Deep Learning for Limit Order Books. arXiv preprint arXiv:1601.01987. abstract

Takeuchi, L., Lee, Y. (2013). Applying Deep Learning to Enhance Momentum Trading Strategies in Stocks. abstract

Xiong, R., Nicholas, E. P., & Shen, Y. (2015). Deep Learning Stock Volatilities with Google Domestic Trends. arXiv preprint arXiv:1512.04916. abstract

Zhu, C., Yin, J., & Li, Q. (2014). A stock decision support system based on DBNs. Journal of Computational Information Systems, 10(2),

883-893. abstract·