CS230

Modern DL Algorithms to Systematically Trade a Hedged Portfolio of U.S. Equities

Kevin M. Lalande*
Department of Computer Science
Stanford University
klalande@stanford.edu

Abstract

Use a hybrid DNN-LSTM architecture to predict stock price behavior over the forward 6-week period into 1 of 5 categories for every ticker listed on the NYSE and Nasdaq stock exchanges using a rich set of 4,525 features per ticker, per month. A 22-35% advantage over random guessing is achieved over a 12-year period.

1 Background

Santé Capital ("SC") manages a Long/Short Equity hedge fund using a systematic machine-learning quantitative trading strategy called MindRank to identify mis-priced securities. The performance of SC's algorithms is evaluated with a carefully reconstructed hypothetical track record ("RTR") that spans 155 consecutive months from January 2004 to December 2016. The RTR significantly outperforms equity index benchmarks like the S&P500 over the period with an annualized net return of 22.8%, an annualized monthly volatility of 11.9%, and a Sharpe ratio of 1.90 after fees and expenses versus 7.2%, 13.8% and 0.43, respectively.

SC's current software system, MindRank Gen3.5, is comprised of four components: 1) dataset generator, 2) predictive algorithms, 3) portfolio design and risk management parameters, and 4) trade execution model. The objective of this project is to upgrade SC's predictive algorithms, which were originally implemented in MATLAB circa 2014-2015, to the latest Deep and Recurrent Neural Network architectures ("DNN", "RNN" and together Deep Learning or "DL") using Python and the TensorFlow[1] and Keras[3] DL frameworks. The other three MindRank components – datasets, portfolio parameters, and trade model – are left untouched in order to produce an valid A/B test of the new predictive algorithms versus a baseline of the current Gen3.5 commercial algorithms.

2 Introduction

MindRank attempts a challenging application: to predict stock price behavior over the forward 6-week period into 1 of 5 categories for every ticker listed on the NYSE and Nasdaq stock exchanges using a rich set of 4,525 features per ticker, per month from a curated dataset with fifteen years of historically accurate data. We exclude tickers with a market capitalization of less than $200 million because these stocks are often too thinly traded to enter and exit large enough Long/Short positions to be commercially relevant. That leaves about 2,500 tickers per month with the exact number fluctuating from one month to the next based on IPO, M&A and delisting activity.

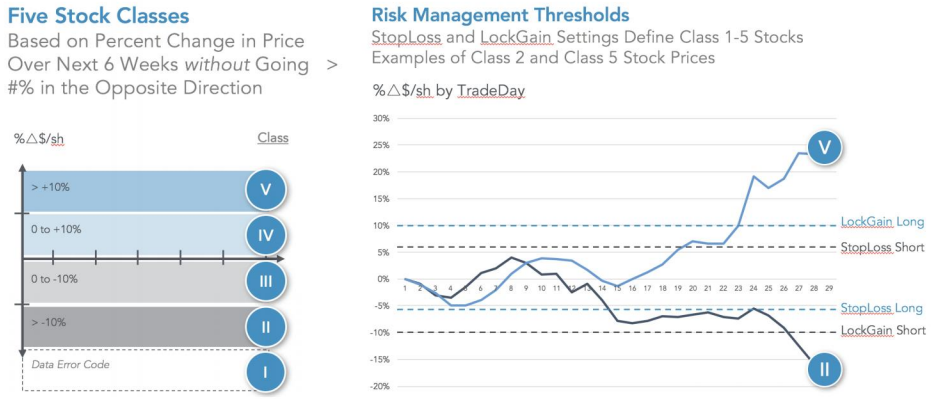
The X input to this algorithm is an (m,n) array of m= 2,500 tickers by n=4,525 features on a given "as-of" date when all of the features reflect the latest publicly available information as of midnight

*Managing Director, Santé Ventures and Santé Capital. Email: kevin.lalande@santeventures.com

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

that day and contain no forward-looking information. Each ticker row represents a single training example. Inputs are presented to the model one month at a time over the 155 months from 200401 to 201612 in order to generate predictions for the following month. The output $\hat{Y}$ of this algorithm is an (m,1) array of categorical Class 1-5 predictions for each of the m= 2,500 tickers in a given month time period.

Figure 1: Class Labeling Scheme



To train the model, ground truth Y labels $\in \mathbb{N}\{1, ..., 5\}$ are supplied based on how each stock price actually behaved over the subsequent 6-week period. The labeling scheme shown in Figure 1 maps tickers into one of five mutually exclusive and collectively exhaustive Class categories. Class 5 tickers, which are candidates for the Long portfolio, increase as a percentage of the closing price on the as-of date by more than a LockGain-Long threshold (e.g., 15%) at any point over the subsequent 6 weeks without first decreasing by more than a StopLoss-Long threshold (e.g., 10%). Class 2 tickers, which are candidates for the Short portfolio, decrease in price by more than a LockGain-Short threshold without first increasing by more than a StopLoss-Short threshold.

Class 4 and 3 tickers are intermediate cases in which the expected value of the increase or decrease relative to starting price is not high enough to justify the risk of buying or selling short the security. Class 1 is a reserved category for tickers that have failed one or more of the quality control tests designed to ensure feature set accuracy and are not considered.

Three specific challenges make this an interesting prediction problem, each of which will be discussed later in the paper:

1. There is no clear proxy for Bayes' error given the way this classification problem is framed and, in a related complication, financial time series price-volume data are noisy.
2. The classification itself is more difficult than simply predicting whether a stock or index price will next move up or down. The models must predict whether a stock will, at any time point over a fixed forward period, move in one direction by more than a certain LockGain threshold without first moving in the opposite direction by more than a certain StopLoss threshold. These additional constraints are necessary to systematically trade the predicted tickers.
3. The prior probabilities for each Class vary widely from one month to the next. For example, the Class 2 incidence rate averages 29% of total tickers over the entire 155 month reconstruction period, but has a standard deviation of 19% in any given month. This makes it difficult for a DNN alone to backpropogate useful gradient updates from one time period to another, and is the primary motivation for the hybrid LSTM – DNN architecture tested herein.

3 Related work

Much of the published research on applying DL techniques in financial market applications is based on attempt to predict forward changes in one-to-a few stock market indexes or stock tickers using

time series price-volume data (open, close, high, low, volume) and a set of 10s-to-100s of easily obtainable technical analysis features like MACD, RSI, Stochastic, Bollinger Bands, etc. Perhaps unsurprisingly, many of these approaches are unable to produce a meaningful or durable advantage over random guessing, particularly after taking into account real-world transaction fees and fund management expenses. I think this is probably due to the use of data sets that lack a sufficiently rich feature set to to provide the predictive models enough of a representation of the underlying phenomena.

That said, several papers make insightful contributions on various pre-processing techniques and network architectures, even if on limited datasets. For example, Wei Bao combines discrete wavelet transforms, stacked autoencoders, and LSTM RNNs to predict several stock indexes.[2] And M'ng combines wavelet transforms with principle component analysis to forecast Asian stock indices.[4].

4 Dataset and Features

SC crawls 10 million URLs to ingest about 1,000 Gb of unstructured text data every day. Various natural language processing techniques are used to extract a proprietary set of features which are then combined with commercially available structured data from several different providers to create a dataset containing 4,525 features for each of the 2,500 publicly traded company on the NYSE and Nasdaq with a market cap greater than $200M in each of the 155 monthly periods from 200401 to 201612. Considerable attention has been paid to quality control systems to ensure that each feature is accurately registered in time to each ticker – that is to say, the right data point at the right time for the right company – and to correct for survivor bias in the commercially purchased datasets.

4.1 Features

The feature set includes macroeconomic indicators, fundamental metrics, technical metrics, investor sentiment, and price-volume data for multiple stock indexes and each of the included company tickers. Feature reporting frequencies vary depending on availability and are either single, daily, weekly, monthly, quarterly, annually. Multi-period features appear in consecutive columns. Figure 2 below shows a few specific examples from the feature set.

Figure 2: Example Features in Dataset

Macroeconomic	Fundamental	News & Sentiment	Technical	Stock	Index
Consumer Price Index	Total Revenues	Press Release	A/D Oscillator	Price Open	S&P 500
Durable Goods Orders	Unlevered Free Cash Flow	Earnings Call	Commodity Channel Index	Price Close	Russell 2000
Employment	Accounts Payable	Aggregate Event Sentiment	Larry William's (R%)	Price High	Nasdaq Composit
Existing Home Sales	Accounts Receivable	Aggregate Event Volume	MACD	Price Low	VIX
Fed Press Conference	Asset Turnover	Abnormal Sentiment	Momentum	Volume	Goldman Commodities Index
FOMC Forecasts	Capital Expenditure	Editorials	ROC Price-rate-of-change	VWAP	Credit Suisse High Yield II Index
FOMC Minutes	Cash and Equivalents	News Volume	RSI	Market Cap	Citigroup World Govt Bond Index
GDP	Cash per Share	Event Novelty	Simple MA 10-day	Enterprise Value	Dow Jones World Index
Producer Price Index	ISS Governance QuickScore	Earnings Evaluations	Stochastic (K%, D%, slow)	Dividend	Dow Jones World Emerging Index

One question considered is whether all of these features are necessary. This is difficult to answer exhaustively given the large number of permutations required to test. However, previous testing has demonstrated that: a) performance improved as features increased from 1,700 to 4,500 so long as regularization techniques were employed, and b) the Top 100 most important features vary significantly from one month to another and in different market environments but the set intersection is 3x higher than would be expected in random chance selection.

4.2 Normalization

All input features are normalized to zero mean and unit variance before training in order to make the contours of the cost function more equally spaced and therefore easier to optimize.

$$Xn^{(i)} = \frac{X^{(i)} - \bar{X}}{S_X}, where: \bar{X} = \frac{1}{m}\sum_{i=0}^{m} X^{(i)} and S_X = \sqrt{\frac{1}{m}\sum_{i=0}^{m}(X - \bar{X})^2}$$

4.3 Wavelet Transform

Price and volume data are denoised using a discrete Wavelet transform. Wavelet transform is an increasingly widespread technique to manage non-stationary financial time series data because it can analyze both frequency and time components simultaneously and is computationally efficient to calculate. [2][4]

Decomposing time series into an orthogonal set of components results in a discrete wavelet transform with father wavelets, $\varphi(t)$ and mother wavelets, $\psi(t)$, which integrate to 1 and 0, respectively. Financial time series can be reconstructed with a sum of projections on the mother and father wavelets indexed by $k \in \{0, 1, 2, ...\}$ and by $j \in \{0, 1, 2, ..., J\}$ where J denotes the number of multi-resolution scales. The brief form of orthogonal wavelet series approximation can be written as:

$$x(t) = S_J(t) + D_J(t) + D_{J-1}(t) + \cdots + D_1(t)$$

where $S_J(t)$ is the coarsest approximation of the input time series x(t). The multi-resolution decomposition of x(t) is the sequence of coefficients $\{S_J(t) + D_J(t) + D_{J-1}(t) + \cdots + D_1(t)\}$[2], which is input to the predictive models below.

4.4 Train/Dev/Test Splits

For the two DNNs, predictions and training updates are made each month based on the subsequent and previous YearMonth's dataset, respectively. Each month has an m= 2,500 tickers which are randomly shuffled and split 90/10 into Train and Dev sets with a batch size of 128. Batch size was selected empirically after testing {1,32,64,128,256 and All}. Over the entire 155 month period, there are 406,950 total examples. The Test set is month ahead prediction.

For the LSTM, data is daily with a batch size of 1 and without shuffling and is pre-trained on m=1,000 prior days split 90/10 Train/Dev set. Batch size was selected to accommodate a day-by-day RNN with seven timesteps (Tx=7) without introducing forward-looking information. The Test set is the day ahead prediction.

4.5 Label Distribution

For the entire 155 month period, Class 1-5 distribution is 8%, 29%, 17%, 24% and 22%, respectively. However, the incidence rate of any given class varies widely from one month to the next. For example, the Class 2 incidence rate is $\mu = 29\%$ and $\sigma = 19\%$. Label data is also pre-processed to support the two binary classification DNNs trained below, one for Class 2 or Not Class 2 and the other for Class 5 or Not Class 5.

5 Methods

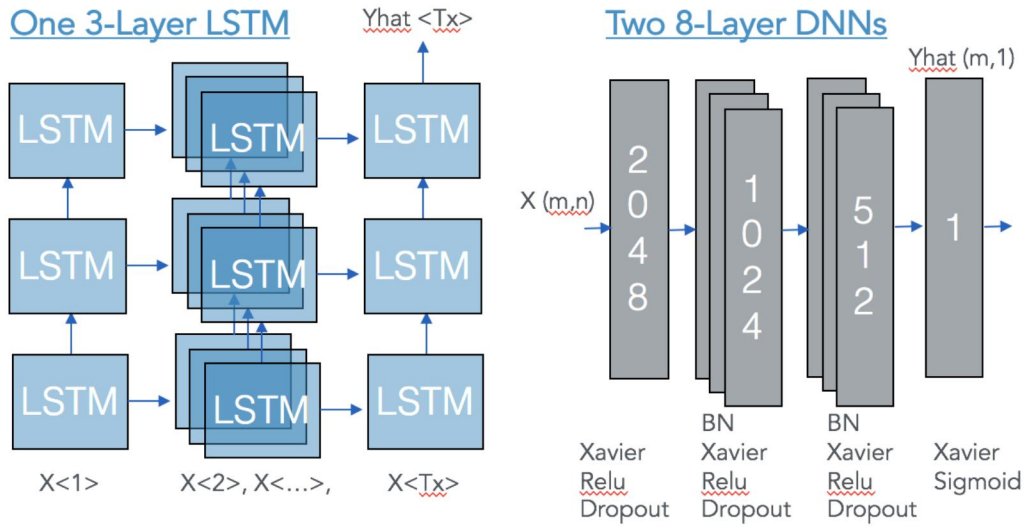
Three separate models were developed to work together to produce an integrated set of predictions for each of the 155 months. Because the Class 1-5 probabilities vary widely from one month to the next, DNNs have difficulty learning useful gradient updates in a period which has, for example, a much lower Class 2 incidence rate than usual. Uncorrected, the number of Class 2 predictions will "tumble" in the subsequent month to near zero as the DNN overreacts to the most recent data. To help address this, I trained a LSTM to predict forward period Rolling Class 2 Arrival Rates and use those predictions to adjust the expected class weights for the DNN each month.

5.1 DNNs

Input X described above is fed into a two different DNNs each with 8 fully-connected layers shown in Figure 3. The first layer has 2048 hidden units, followed by 3 layers with 1024 hidden units each, followed by another 3 layers with 512 hidden units each, followed by an Output layer with 1 hidden unit.

The Output layer uses a Sigmoid activation function to predict a binary classification problem (Class 2 or not) or (Class 5 or not).

Figure 3: Three Predictive Models Trained



Each of the layers uses Xavier weight matrix initialization and batch normalization to help control for vanishing and exploding gradients as the network gets deeper. All layers except the Output use a ReLU activation function and Dropout regularization with a keep probability increasing from 50% to 85% from shallow to deeper layers.

Binary cross-entropy was selected as the Loss function to measure the dissimilarity between the true probability and predicted probability by the following equation:

$$L(\hat{y}, y) = -(ylog(\hat{y}) + (1-y)log(1-\hat{y}))$$

Stochastic gradient descent with momentum was selected as the optimizer with learning rate α=0.05 and β=0.9. ADAM and RMSprop were also considered.

Hyperparameters were tuned using a log-scale search for α and β and a design of experiments for the number of hidden units {256, 512, 1024, 2048, 4096}, mini-batch size {1,32,64,128,256 and All}, number of layers {1...10}, and number of training epochs {1, 2, 5, 10, 20, 40, 80}. In selecting the Best Model architecture, I took into account F1, number of months with fewer than 50 Class 2 predictions and amount of compute time required to train the model.

5.2 LSTM

A three-layer LSTM network with $T_x = 7$ time steps was trained to predict the Rolling Class 2 Arrival Rate 5-Day average.

The input X<t> was the Discrete Wavelet Transform for the time series of prior 90-days for {Class2Percent, Class5Percent, RC2AR5, Russell2000_Close, SP500_Close, VIX_Close} which has a shape (timesteps=7, data_dim=540).

A Sigmoid activation was selected for the Output layer because the predicted variable RC2AR5 is a percentage value that is always between 0 and 1.

Mean Squared Error was selected as the Loss function to measure the distance between the true and predicted value of RC2AR5. RMSprop was selected as the optimizer with a learning rate $\alpha = 0.001$. ADAM and SGD were also considered.

6 Experiments/Results/Discussion

Figure 4 shows summary results for Best Model Classifier Performance of this new hybrid DNN-LSTM architecture versus baseline commercial MindRank Gen3_5 algorithms.

I use Advantage over random guessing as the key performance metric because of the Class imbalance and the practical difference in false positive (likely lose money) versus false negative (lose opportunity). Precision, Recall and F1 are all measured and considered along with IRR, Volatility (risk) and Sharpe Ratio (quality) that are produced by running the classifier predictions through the RTR to calculate financial impact.

The new architecture was able to achieve a substantial advantage over random guessing for both Class 2 (34.7%) and Class 5 (22.3%) tickers. The model appears to be training and generalizing well, with a Train accuracy of 81%, Dev of 70% and Test of 62%. It is difficult to know exactly how much of the Train error is Avoidable Bias because no clear proxy for Bayes' error exists for this problem as framed. Some of the fall off from Train to Dev is the result of over-fitting the training data. And at least some of the fall off from Dev to Test is the result of the class probabilities shifting in a subsequent month in response to a changing market environment.

During development testing, higher performance was achieved with binary classification than with a 5-way Softmax. Advantage over random guessing (discussed below) was 37% vs. 28%. This was an unexpected result as I thought the MECE nature of the class labeling scheme would help the model learn from multiple tasks. But the binary network design clearly performs better, so I train two separate DNNs to predict Class 2 vs. Not Class 2 and Class 5 vs. Not Class 5, and then take the union less the intersection of the two sets to create the final list of predictions.

Figure 4: Classifier Performance - Best Model

	CS230 Project		MindRank 3_5	
	Class 2	Class 5	Class 2	Class 5
# YearMonths	155	155	155	155
Actual	115,826	120,532	118,718	122,582
Predicted	158,506	159,610	139,532	118,995
True Positive	60,766	57,815	58,613	46,989
Precision	38.3%	36.2%	42.0%	39.5%
Recall	52.5%	48.0%	49.4%	38.3%
F1	0.443	0.413	0.454	0.389
Incidence	28.5%	29.6%	28.4%	29.3%
Advantage	34.7%	22.3%	47.9%	34.7%
Train	80.8%	78.6%	na	na
Dev	69.5%	66.5%	na	na
Test	62.3%	59.3%	na	na
IRR	19.3%		23.2%	
Volatility	9.9%		10.9%	
Sharpe Ratio	1.25		1.43	

7 Conclusion/Future Work

Summarize your report and reiterate key points. Which algorithms were the highestperforming? Why do you think that some algorithms worked better than others? For future work, if you had more time, more team members, or more computational resources, what would you explore?

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, pages 265–283, Berkeley, CA, USA, 2016. USENIX Association.

[2] Wei Bao, Jun Yue, and Yulei Rao. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PloS one*, 12(7):e0180944, 2017.

[3] François Chollet et al. Keras. `https://keras.io`, 2015.

[4] Jacinta Chan Phooi M'ng and Mohammadali Mehralizadeh. Forecasting east asian indices futures via a novel hybrid of wavelet-pca denoising and artificial neural network models. *PloS one*, 11(6):e0156338, 2016.

[1] Alexander, J.A. & Mozer, M.C. (1995) Template-based algorithms for connectionist rule extraction. In G. Tesauro, D.S. Touretzky and T.K. Leen (eds.), *Advances in Neural Information Processing Systems 7*, pp. 609–616. Cambridge, MA: MIT Press.

[2] Bower, J.M. & Beeman, D. (1995) *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System.* New York: TELOS/Springer–Verlag.

[3] Hasselmo, M.E., Schnell, E. & Barkai, E. (1995) Dynamics of learning and recall at excitatory recurrent synapses and cholinergic modulation in rat hippocampal region CA3. *Journal of Neuroscience* **15**(7):5249-5262.