4-2019

Using Gleaned Computing Power to Forecast Emerging-market Equity Returns with Machine Learning

Xida Ren

Using Gleaned Computing Power to
Forecast Emerging-market Equity Returns
with Machine Learning

A thesis submitted in partial fulfillment of the requirement
for the degree of Bachelor of Arts / Science in Computer Science from
The College of William and Mary

by

Xida Ren

Accepted for Honors
(Honors, High Honors, Highest Honors)

Type in the name Zhenming Liu

Type in the name Bin Ren

Type in the name Robert Michael Lewis

Type in the name Tyler Frazier

Williamsburg, VA
Apr 30 2019

Using Gleaned Computing Power to Forecast Emerging-market Equity Returns with Machine Learning

Undergraduate Honors Thesis in Computer Science

Xida Ren
Advised by Zhenming Liu

April 2019

Abstract

This paper examines developing machine learning and statistic models to build forecast models for equity returns in an emergent market, with an emphasis on computing. Distributed systems were pared with random search and Bayesian optimization to find good hyperparameters for neural networks. No significant results were found.

Contents

1 Introduction

1.1 Introduction

This paper examines developing machine learning and statistical models to build forecast models for equity returns in an emerging market. In quantitative investment, a major goal is to identify the collection of equities whose returns are stronger than the market return (i.e., maximize the so-called excessive return). Extensive works have been done in this area by both financial professional and academics[1]. There are three major approaches to identify the "winning stocks":

(i) using fundamental data[2] by examining the performance of a company (e.g., whether it had satisfying revenue in the past quarter), we choose the financially robust and growing companies,

(ii) using technical data[3]: by examining the price-volume behaviors of a stock, we interpret how the market "view" the stock in short term and leverage such information to trade,

(iii) using macro data[4]: by examining the "macro economics condition", we determine the sectors that are growing. For example, when a trade deal on oil is made between USA and China, the investors will feel oil-related companies are promising and thus invest on these stocks heavily.

This work examines using technical data to build forecast models. We specifically examine how we may apply recent advances in machine learning and statistical models to extract signal from technical data. Our solution

[1] B. G. Malkiel, "The Efficient Market Hypothesis and Its Critics," *Journal of Economic Perspectives*, 2003, ISSN: 0895-3309. DOI: 10.1257/089533003321164958, G. G. Tian, G. H. Wan, and M. Guo, "Market Efficiency and the Returns to Simple Technical Trading Rules: New Evidence from U.S. Equity Market and Chinese Equity Markets," *Asia-Pacific Financial Markets*, vol. 9, no. 3, pp. 241–258, 2002, ISSN: 1573-6946. DOI: 10.1023/A:1024181515265. [Online]. Available: https://doi.org/10.1023/A:1024181515265, T. Z. Tan, C Quek, and G. S. Ng, "Brain-inspired Genetic Complementary Learning for Stock Market Prediction,"

[2] N. T. Laopodis, "Equity prices and macroeconomic fundamentals: International evidence," *Journal of International Financial Markets, Institutions and Money*, vol. 21, no. 2, pp. 247–276, 2011, ISSN: 10424431. DOI: 10.1016/j.intfin.2010.10.006. [Online]. Available: http://dx.doi.org/10.1016/j.intfin.2010.10.006.

[3] S. Ghoshal and S. J. Roberts, "Thresholded ConvNet Ensembles: Neural Networks for Technical Forecasting," 2018. arXiv: 1807.03192, G. E. Pinches, "The Random Walk Hypothesis and Technical Analysis," *Financial Analysts Journal*, vol. 26, no. 2, pp. 104–110, 1970, ISSN: 0015198X. [Online]. Available: http://www.jstor.org/stable/4470663.

[4] Y. F. Wang, "Predicting stock price using fuzzy grey prediction system," *Expert Systems with Applications*, vol. 22, no. 1, pp. 33–38, 2002, ISSN: 09574174. DOI: 10.1016/S0957-4174(01)00047-1, K. J. Oh and K. J. Kim, "Analyzing stock market tick data using piecewise nonlinear model," *Expert Systems with Applications*, vol. 22, no. 3, pp. 249–255, 2002, ISSN: 09574174. DOI: 10.1016/S0957-4174(01)00058-6.

aims to address the cross-cutting constraints between system and statistical models. Modern statistical model requires astonishingly heavy compute resources. Usually large systems do not automatically result in better models), but since stock forecasting has a competitive component to it, larger, more costly systems are likely to result in more profitable forecasts (see the section on Fama and his Efficient Market Hypothesis). So a key challenge we tackle is co-optimize model performance and system scalability. Our research project consists of the following key components:

1. Data processing and objective: how we pre-processed the data and set up objectives of our forecasting model.
2. Determination of models
3. Compute demand and massive compute infrastructure
4. Hyperparameter optimization

1.2 Motivation: Financial Markets

1.2.1 Day trading and the efficient market hypothesis

It is easy to imagine profiting from being able to forecast tomorrow's stock prices. When a stock is going to increase in value in the future, a trader who anticipates this change would be able to buy it in the present and profit by selling it when its market price has increased. Strategies like these fall under active investment

The performance of active trading strategies are usually gauged against buy-and-hold strategies, which involve simply buying a selection of stocks whose aggregate performance represent how well the market is doing and hold them as they grow in value. The latter strategy involves less work, less risk, and avoids the potentially overwhelming trading costs associated with the former.

While there has been a constant stream of research on price patterns that could be used to beat buy-and-hold strategies, the idea receives a constant stream of critique. Critics tend to gather around Eugene Fama and the Efficient Market Hypothesis (EMH) which he worked extensively on[5][6]. The hypothesis asserts that, under certain conditions, all available information has already been incorporated into the market price, and hence no statistically significant profit can be gained over buy-and-hold by using public information to forecast and trade on these forecasts. Since since

[5] E. F. Fama, "The Behavior of Stock-Market Prices," *The Journal of Business*, vol. 38, no. 1, pp. 34–105, 1965, ISSN: 00219398, 15375374. [Online]. Available: http://www.jstor.org/stable/2350752.

[6] B. G. Malkiel, "The Efficient Market Hypothesis and Its Critics," *Journal of Economic Perspectives*, 2003, ISSN: 0895-3309. DOI: 10.1257/089533003321164958.

active trading has high transaction costs and management costs, critics like Sharpe go as far as to argue that such practices have no place in the market[7].

One way to understand Fama's claims is to say that any pattern that would allow us to forecast future prices is probably already traded upon, and is hence rendered unprofitable. For example, under the efficient-market hypothesis, if other traders can forecast a price rise of 5% tomorrow in AAPL, they would not sell at any price lower than the anticipated future price, moving the price rise up in time and preventing anyone from collecting a profit off the forecast.

However, the works Fama did on the efficiency of stock markets was based upon a set of assumptions that were not exactly true in real markets. For example, he explains[8] that sufficient conditions for market efficiency include " (i) there are no transactions costs in trading securities, (ii) all available information is costlessly available to all market participants, and (iii) all agree on the implications of current information for the current price and distributions of future prices of each security".

Indeed, Fama acknowledges[9] that on the daily level, the EMH may not be entirely accurate. While he maintains that the market price does incorporate all available information eventually, he finds that the day to day returns of a stock are correlated over time. This means that a model as simple as linear regression of future returns over past daily returns may be able to produce forecasts that can be used to build profitable trading strategies.

In defence of the EMH, Fama stated that the correlations are but marginal, and any strategy based upon these correlations would only show marginal profits in on-paper trading strategies. In the real market, these profits are easily wiped out by varous costs such as transaction costs (exchange fees), cost-of-risk, and the opportunity costs. In essence, the reason why these patterns exist is because there are costs that prevent traders from using them.

1.2.2 Real markets don't follow Fama's conditions

Given this, one can't help wonder if there are patterns out there that breaks one of Fama's other conditions to yield actually profitable trading schemes. For example, on the lines of the linear-model example, there might be other models that have forecasting efficacy because transaction costs prevent traders from exploiting the patterns they detect; models that exist because real markets don't adhere to Fama's condition (i).

[7] W. F. Sharpe, "The Arithmetic of Active Management," *Financial Analysts Journal*, vol. 47, no. 1, pp. 7–9, 1991. DOI: 10.2469/faj.v47.n1.7. [Online]. Available: https://doi.org/10.2469/faj.v47.n1.7.

[8] E. F. Fama, "American Finance Association Efficient Capital Markets: A Review of Theory and Empirical Work," Tech. Rep. 2, pp. 383–417.

[9] E. F. Fama, "American Finance Association Efficient Capital Markets: A Review of Theory and Empirical Work," Tech. Rep. 2, pp. 383–417.

Or there might be forecasting practices that work because they require information not available to the public (in the case of insider trading, which is illegal) or information available at a cost (for example, a neural network may need to be trained extensively on historical data, whose gathering and processing entailes a substantial cost). These models may work because real markets don't adhere to condition (ii).

Or perhaps with the same information some traders are able to reach better conclusions than others. For example, there might be machine learning models that perform better as more computing power is spent to train them. These models violate Fama's condition (iii).

Indeed, in real markets, profitable active investment strategies have been found. A study[10] used probabilistic neural networks to out-perform buy-and-hold in the Taiwanese market, and models such as support vector machines achieved a degree of success[11].

1.2.3 Contribution

This project seeks to find profitable trading strategies where Fama's EMH doesn't hold perfectly; in other words, where there are barriers to entry for the forecasting market.

Our dataset is drawn from an emergent market, where data availability is low and regulations are tough. This helps deviate from conditions (i) and (ii). We choose neural networks as our models, since neural networks are logistically troublesome[12] and computationally intensive to train[13], thus deviating from condition (iii). In this barriered forecasting market, we attempt to establish a competitive advantage by gleaning computer power from idle departmental computers to search massive hyperparameter spaces.

[10] A. S. Chen, M. T. Leung, and H. Daouk, "Application of neural networks to an emerging financial market: Forecasting and trading the Taiwan Stock Index," *Computers and Operations Research*, vol. 30, no. 6, pp. 901–923, 2003, ISSN: 03050548. DOI: 10.1016/S0305-0548(02)00037-0.

[11] H. Ince and T. B. Trafalis, "Short term forecasting with support vector machines and application to stock price prediction," *International Journal of General Systems*, vol. 37, no. 6, pp. 677–687, 2008. DOI: 10.1080/03081070601068595. [Online]. Available: https://doi.org/10.1080/03081070601068595.

[12] J. Bergstra, D. Yamins, and D. D. Cox, "Hyperopt: A Python Library for Optimizing the Hyperparameters of Machine Learning Algorithms," *Proceedings of the 12th Python in Science Conference*, no. Scipy, pp. 13–20, 2013. [Online]. Available: http://www.youtube.com/watch?v=Mp1xnPfE4PY.

[13] J. V. Tu, "Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes," *Journal of Clinical Epidemiology*, vol. 49, no. 11, pp. 1225–1231, 1996, ISSN: 08954356. DOI: 10.1016/S0895-4356(96)00002-9.

2 Data

2.1 Feature Engineering

2.1.1 Data Source

Our data is drawn from an emerging equity market between 2009 and late 2018. The raw data exists as a shape ‘(stock, time)‘ matrix, where the stocks are drawn from an index basket, and the time-points are drawn daily at market open.

2.1.2 Feature types

Over the course of the project, several different feature configurations were tested. We began with forecasting 1-day returns based on 1-day returns gathered over the past 30 days. We then tested a scheme that added 5-day volatility gathered daily over the past 30 days as a feature.

We also tested forecast 5-day future returns, using 1-day returns gathered over the past 100 days.

2.1.3 Weighting

We also tested weighted schemes, where each example (indexed by stock id and time) is weighted according to the trading volume (price of each transaction times number of each stocks in the trade). This weighting has dual purpose.

First, the price of a highly traded stock tends to be more information-rich, since the price is agreed upon by more parties. The price of a "big" or highly traded stock should mean more than the price of a "small" or seldom traded stock.

Second, it is more important to be able to forecast high-volume stocks correctly, since forecasts are easier to turn into profit in these stocks. According to a well-known model for market impact, market impact cost due to buying or selling stocks is proportionate to $\sqrt{\frac{Q}{V}}$, where Q is the number of stocks traded, and V is the trading volume of that stock[14].

Here’s an example to illustrate why volume matters and how market impact works: suppose that my model forecasts that tomorrow AAPL is going to decrease in value. I then proceed to short 1000 shares.

Now if AAPL is small, then perhaps by the time I’ve sold the first 100 shares, the price has already gone down since by selling I’ve increased supply of AAPL in the market, and exhausted some of the higher bids in the market. This results in my remaining shares selling at (progressively) lower

[14] J. P. Bouchaud, “Price Impact,” 2009. arXiv: 0903.2428. [Online]. Available: http://arxiv.org/abs/0903.2428.

prices. And tomorrow, when I buy the shares back, each share bought would increase the price at which I will have to buy the remaining shares, since I'm increasing demand this time and exhausting the lower ask orders in the market. In a large market for AAPL, the impact would be less, and I will be able to profit more off this movement.

Since the market impact model gives a market impact cost proportional to the square root of past trading volume, I decided to use trading volume to weight the costs.

2.1.4 Data "Cuts"

To ensure the generalizability of our models and the validity of our results, we split our data into "cuts" so that we could cross-validate model parameters, cross-validate hyperparameters, and prevent badly performing models from appearing significant by abusing information leaked from the future.

We split our data into 7 cuts. Each cut consists of a testing set and a training set. The last three cuts are "validation cuts" to validate hyperparamaters on, and the first five cuts are used as "hyperparameter selection cuts".

The cuts are constructed as follows. Each testing set is one year in length and defines the cut. Together the testing set covers each year in [2012, 2018] without overlapping. The training set covers the three years before the beginning of the testing set, possibly with a stretch of data deleted towards the end.

Thus cut 0 would have a training set that contains data from the years 2009, 2010, 2011; its training set would be the year 2012. The final cut, cut 6, would have training set covering the years 2015, 2016, 2017, and its testing set would consist of data from 2018.

3 Metrics

For the initial evaluation of each model, we used Pearson's correlation (r) instead of the standard L2 loss used by most other studies, since pearson r is a metric independent of the scaling of the data and allows us to compare closeness of fit over datasets with different standard deviations. Pearson's correlation measures only strength of association (goodness of fit) and does not depend on the sample size. In case of weighted models, we also computed pearson's r based on While it is a good basis to make claims, we also need a certainty metric for the claims we make.

To test our hypothesis that our models were good predictors of future stock returns, we hypothesize that actual stock returns can be modeled as $y = \beta\hat{y}$ where $\hat{y}$ is a vector of forecasts, y is the true returns, and β is a constant scalar. Our null hypothesis is thus $\beta = 0$. Fitting a linear model

'truey = b * predy' conveniently gives us a T-statistic and a p-value for our experimental results under the null hypothesis.

A proprietary, experimental investment simulator was used to simulate investment results based on forecasts. It produces investment metrics such as the Sharpe ratio[15] (average returns divided by standard deviation of returns) and total investment return (from algorithmically investing based on forecasts. The simulator also produces market-adjusted versions of said metrices, and cost-adjusted versions which accounted for the transaction cost associated with buying and shorting stocks.

4 Model

This section describes neural network models used to forecast future returns.

4.1 Neural Networks

We use dense neural networks with with optional convolution-max-pooling layers to forecast stock returns. The model is composed of sub-nets for each feature. A sub-neural-network is created for each feature (two features tested include historical returns and volatility), and the last layer of each subnet is concatenated and then fed through another subnet to produce the final forecast.

Each sub-net consists of a number of convolutional layers (optional) followed by a number of dense layers. Each convolutional layer may be preceded by a max-pool layer (pool-size & kernel size are both hyperparameters). While the size (dense units / convolution filter numbers) and number of layers are specifiable by hyperparameters, for simplicity's sake we fixed the layer size for each type of layer in each sub-net. Regularization (l2, l1, or otherwise) applied to the weights is specifiable via hyperparameters, and so are various other aspects of the model, such as training algorithm.

The entire network is trained with back-propagation. The back-propagation algorithm (SGD, momentum-SGD, SGD with learn-rate annealing, SGD with adaptive learn rate, etc.) is specifiable via hyperparameters.

The section below shows an example of a model, described by a JSon specification of hyperparameters.

4.1.1 Model Specification: an example

Below we have an example hyperparameter specification for a model, and in figure 2 a graph of the model generated from these hyperparameters. The JSon representation is fed through a Python/Tensorflow script to flesh out a

[15] W. F. Sharpe, "The Sharpe Ratio," *The Journal of Portfolio Management*, vol. 21, no. 1, pp. 49–58, 2009, ISSN: 0095-4918. DOI: 10.3905/jpm.1994.409501.

model, train it, and test it to produce metrics. This approach makes it easy to describe candidate models, store specifications of successful models, and search high-dimensional hyperparameter spaces. It enables distributed hyperparameter searching, and allows us to easily view results in aggregation.

```
{
    "batch_size": 46.0,
    "butcher_id": "2019-02-08-csi800",
    "conv_count": 0,
    "conv_filters": 0,
    "conv_kernel_size": 0,
    "dense_count": 0,
    "dense_units": 0,
    "epochs": 7,
    "exp_key": "example_2019_04_12",
    "features": {
        "returns": {
            "conv_count": 2.0,
            "conv_filters": 46.0,
            "conv_kernel_size": 5.0,
            "dense_count": 1,
            "dense_units": 68.0,
            "l2reg": 0.3951370426172572,
            "pool_size": 2.0,
            "shape": [
                100
            ]
        },
        "volatility": {
            "conv_count": 1.0,
            "conv_filters": 33.0,
            "conv_kernel_size": 3.0,
            "dense_count": 3,
            "dense_units": 54.0,
            "l2reg": 0.0,
            "pool_size": 1.0,
            "shape": [
                100
            ]
        }
    },
    "l2reg": 0.5,
    "optimizer": {
        "class_name": "rmsprop",
```

```
40          "config": {
41              "lr": 7.991736568151144e-05
42          }
43      }
44  }
```

Notice there's just a single dense layer condensing the output of the subnets into one. That is because in the top-level object we specified

The outermost object contains specifications for feature input (in line 3, "butcher_id" specifies which directory to pull features from), the way the model is trained (line 2 specifies the batch size, and line 38 through 43 specifies details about the optimizer used to train).

It also includes the specification for the outer subnet that processes the combined output of the inner subnets.

The inner objects under key "features" specify details about each subnet. Each key corresponds to a feature stored in the data directory ("returns" and "volatility"), and the inner objects conntains details on the subnet.

5 Validation

This section explains why each cut has a training and testing dataset, and how we determine whether a neural network model has been trained to have good weights.

Cross validation was done with extreme rigor in this project. This means that for each model we test (a "model" is defined by a set of hyperparameters), we always split our data into training and testing sets, and train our model exclusively on the training set to ensure that our models generate well to previously-unseen data. Otherwise, we may produce models that look like they have low variance (and thus high accuracy) but biased towards the training set instead of the true population[16].

Because our ultimate goal is to produce a model that could be used to trade profitably, we had to pay extra attention in order to prevent future data from being used to build our models; models that require future information are only useful for traders with time-machines, and traders with time-machines require no models.

We never train models in the future and test them in the past. This is done to prevent us from producing models that only work when trained on future data. For example, in the case where we were forecast 5-day future returns using the past 100 daily returns, a model that is tested for forecast accuracy between January 1 2015 and December 31 2015 would only take training

[16] R Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," *Proceedings of the 14th international joint conference on Artificial intelligence - Volume 2*, vol. 2, no. 0, pp. 1137–1143, 1995.

data gathered before July 2014. Note that we did not use features between July and December 2014 because they would overlap with the features of the testing set.

Similarly, we never use information from testing data when de-meaining, scaling, windsorizing, or otherwise transforming our data. Not only does this prevent us from leaking information into the training set, within the testing set it also prevents us from creating artificial patterns that would have been unavailable to a real-market trader. For example, it is common practice in machine learning to scale data such that the standard deviation is 1. However, since it is impossible to know beforhand the standard deviation of a stock's returns over a year (volatility), we avoid scaling the testing set. While this prevents us from reaching false conclusions, these almost-paranoid precautions also prevent us from putting our testing data at mean=0 and variance=1. This presents a slight inconvenience for using the standard metrics for regression, but is easily dealt with.

5.1 Data Cuts

The two following subsections detail the necessity of our scheme of cuts shown in 1

5.2 Parameter Validation

5.2.1 Hyperparameter Validation

Similar to how model parameters may adapt to their training set and not be generalizable to the general returns dataset, the hyperparameters that determine how these parameters are chosen may also adapt to the wrong set of datapoints. Hence we exercise the same precaution when selecting our hyperparameters.

A hyperparameter combination is evaluated based on the performance of the modes they determine. To evaluate a hyperparameter combo, we train a model for each selection cut (cut0 through cut3), and forecast on the testing set of that selection cut. The forecasts are then glued together to produce a large forecast stretching between 2012 and 2015, and the correlation between this forecast and the actual returns in the given time period is used to evaluate the hyperparameter selection. The hyperparameter combination is then selected or adjusted based on the selection-correlation, until we have a satisfactory set of hyperparameter combinations.

We then bring in the validation cuts to test these hyperparameter combinations on. No more tweaking is done, and the correlation over the validation set have final say over whether the experiment is successful or not.

6 Hyperparameter Search

This subsection explains how hyperparameter combinations are chosen.

6.0.1 Grid search

A couple of approaches were attempted during this project. I began with grid search, specifying a few choices for each parameter. With the large amount of hyperparameters, it soon turned out to be computationally unfeasible even with methods detailed in Section 7. With many hyperparameters of unknown relative importance, I manually tuned the search, fixing values of hyperparameters that tend not to significantly affect outcomes.

6.0.2 Random Search

A more principled way to deal with this problem was found in a paper by Bergstra[17]. Adapting my grid-search approach by randomizing the search order and stopping midway, comparable results were achieved at much lower computational cost.

6.0.3 Bayesian Optimization

Towards the end of the research project, Hyperopt[18] was used to choose hyperparameters and run experiments. A Python library for Baysian machine learning on hyperparameters, Hyperopt combined revisiting well-performing hyperparameter values and exploring unvisited portions of hyperparameter-space in search of optimal hyperparameter combinations.

[17] J. Bergstra and Y. Bengio, "Bergstra12a.Dvi," vol. 13, pp. 1–25, 2012. [Online]. Available: papers3://publication/uuid/1190E1AB-0319-40C5-81CD-7207784965DE.

[18] J. Bergstra, D. L. K. Yamins, and D Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," *Proceedings of the 30th International Conference on Machine Learning*, pp. 115–123, 2013. [Online]. Available: http://jmlr.org/proceedings/papers/v28/bergstra13.html.

7 Computing

7.1 Computational Need

Hyperparameter tuning is important to the performance of neural networks[19]. While on some datasets NN models are robust to hyperparameters, training-related hyperparameter such as the loss function are critical to robust performance in the face of outliers[20].

For hyperparameters to which neural networks are typically considered robust, the structure of our data still makes tuning important. An 2008 study concluded that as long as layer sizes are within a certain range, neural networks generalize well to unseen data. However, the study assumed Gaussian noise, an assumption that does not hold with stock market data, where kurtosis can be as high as 6.965339838511829 (the Gaussian distribution has kurtosis 3) and the distribution produces a distinctly non-gaussian histogram, as seen in Figure 3, where all 5-day returns between 2009 and 2018 are plotted.

Counting features for different subnets, depending on the setup we end up with somewhere between 10 and 50 hyperparameters. Allowing only 10 minutes to train and test each hyperparameter combination (as is typical on 2x Nvidia 1080 TI GPUS), a mere thousand hyperparameter combinations takes a week to test. Even with random search and Bayesian optimization, the wait is unreasonably long, not to mention frequent interruptions due to code errors, computer breakdowns, and other researchers competing for the same computing resources.

7.2 Distributed Computing

Fortunately, the nature of our three approaches for hyperparameter searching all lend themselves to easy parallelization. Parallelization is implemented with a job queue and a database to store results (both using MongoDB). An architecture diagram is shown in figure 4.

[19] J. Bergstra, D. Yamins, and D. D. Cox, "Hyperopt: A Python Library for Optimizing the Hyperparameters of Machine Learning Algorithms," *Proceedings of the 12th Python in Science Conference*, no. Scipy, pp. 13–20, 2013. [Online]. Available: http://www.youtube.com/watch?v=Mp1xnPfE4PY, J. Snoek, O. Rippel, K. Swersky, *et al.*, "Scalable Bayesian Optimization Using Deep Neural Networks," 2015. arXiv: 1502.05700. [Online]. Available: http://arxiv.org/abs/1502.05700, A. Daerle and R. V. Zicari, *Neural Networks: Tricks of the Trade Second Edition*. 2010, ISBN: 9783642160912, G. Diaz, A. Fokoue, G. Nannicini, *et al.*, "An effective algorithm for hyperparameter optimization of neural networks," vol. 61, no. 4, pp. 1–20, 2017. arXiv: 1705.08520. [Online]. Available: http://arxiv.org/abs/1705.08520.

[20] K Liano, "Robust error measure for supervised neural network learning with outliers," *IEEE Transactions on Neural Networks*, vol. 7, no. 1, pp. 246–250, 1996, ISSN: 1045-9227. DOI: 10.1109/72.478411.

7.3 Idle Computing Resources

Implementing this scheme still requires computing resources.

I first wrote Python scripts to implement the system, and made a list of the required pip packages for the compute nodes.

Trying to gather all student lab computer IPs in McGlothling 121, I wrote a BASH script to quickly attempt to ssh into all computers in the CS departmental IP range. Successful IPs were recorded in a file.

Not only did this script allow me to collect the IP addresses I wanted, it also revealed to me the BG1~9 servers, where massive amounts of memory, storage, and processing power was available.

An interesting but not very useful find was that the computers in the Professors' offices were also open to ssh connections and allowed login through my student account. They were useless to me because they tend to have different software suits set up, some even running MacOS.

I then wrote another script that performed the following actions in sequence after ssh tunneling into an IP address supplied as a command-line argument

1. install Miniconda on /tmp,
2. install all required pip packages,
3. mount the datadrive via sshfs,
4. set up ssh port-forwarding tunnel to the Job and Result servers,
5. launch a worker thread.

With this script I was able to search around 1200 hyperparameter combinations overnight. Since the lab computers tend to have multi-core CPUs and students tend not to run core-intensive applications, I was able to get away with it without limiting workers to idle workstations. Sadly, when the summer ended, students started using the lab computers and started complaining about the network congestion caused by feeding large amounts of data from the data store to the compute nodes. IT started terminating my processes, and I was reduced to using just the BG servers (whose large disks allowed me to cache my dataset) plus a limited number of student lab computers.

8 Results

8.1 A False Positive

In November 2018 I was able to produce seemingly significant results. I was running random search on the search space described in Figure **??**, and ensembling the top 50 models in hope to produce better results.

```
{
    "use_volatility": [true],
    "volatility_shape": [30],
    "dense_count": [1,2],
    "dense_units": [50,100,150],
    "conv_count": [0],
    "returns": {
        "dense_count": [0],
        "conv_count": [0]
    },
    "volatility": {
        "dense_count": [0],
        "conv_count": [0]
    },
    "optimizer": {
        "optimizer_name": ["FelixOptimizer"],
        "learning_rate": [3e-06,1e-06,3e-07],
        "momentum": [0.3,0.7],
        "m_gamma": [0.0001],
        "m_power": [0.75]
    },
    "l2reg": [1.0,10.0,100.0],
    "train_steps": [50000],
    "save_checkpoints_steps": [5000],
    "keep_checkpoint_max": [1000],
    "host": ["gpubox1"],
    "normalization": ["Felix"],
    "output_dir": ["/home/xren/nn-output/roast_2018-11-12/"]
        ,
    "weighted": [true]
}
```

The forecast-true market correlations are listed as follows:

Listing 1: Out-of-sample Correlations

```
# Unweighted Pearson R (hyperparameter selection)
cut: 1
        0.046622112436332484
cut: 2
        0.04884132100551082
cut: 3
        0.03123235950611614
cut: 4
```

```
        0.048024457544869815
cut: 5
        0.0591479510054449
# Weighted   Pearson R (hyperparameter selection)
cut: 1
        0.06040253565540615
cut: 2
        0.06123432600994524
cut: 3
        0.03367733007635976
cut: 4
        0.05056667387602864
cut: 5
        0.06412042280079279
```

The simulator was able to multiply the initial capital given by 1.8 times over the 3 years from 2015 - 2017, and produced a log-pnl curve as follows.

However, when our industry collaborators updated the data (to include more data from 2018) and the simulator (no reason was given, but the old investment simulator was made unavailable), attempts to replicate these results have failed.

Then, in late December, the simulator went down and never came back up.

8.2 Other things I tried

On the new data, I switched to using Hyperopt to perform hyperparameter search. Despite continuous efforts to tune and around 10 thousand models trained over the course of 13 experiments, no significant results were produced. Different sets of stocks were used (all data available, large index-fund basket, small index-fund basket, combination of the two). Different feature processing schemes were tested (windsorization, rank of data transformed into gaussian, etc.).

One of the experiments came up with a positive correlation of 0.03 over the three validation sets, so I decided to compute the T statistic as a certainty metric. A naiive approach gave me $T = 6.48$, but a more careful examination revealed that the standard error for β was underestimated because of correlated residuals, and using the Newey-West estimator for the covariance matrix to correct the T value yielded a paltry $T = 1.40$ and an insignificant p-value.

8.3 Takeaways

8.3.1 Replicability

The November Results were difficult to replicate, since they relied on old versions of the simulator and data-preprocessor. In the future I should be more careful to maintain working copies of old code and perform regression testing with every change.

8.3.2 Independent Replicability

I relied on the remote, proprietary simulator to evaluate my models. I should have taken the time to write the code myself, so that I wouldn't have to rely on a not-quite-reliable remote server working correctly to evaluate my results.

8.3.3 Pipelining

I used a substantial amount of my time to write code that processed data, searched for good hyperparameters, outputted forecasts, and generated reports with metrics. This was time well spent, as I had to reuse the code many times.

Looking back I wish I had invested even more in scripts that automated the logistics of running experiments. For example, sometimes due to a bug in my code an experiment would stall and I wouldn't be alerted until I check its progress. I could have written a script that sent me an e-mail notification when my experiments terminated.

8.3.4 Not re-writing Code

During the course of the project I did a substantial amount of unnecessary work. I changed my code to accomodate extra features, but forgot to do regression testing to make sure that the code was backwards compatible, and had to search through my Git repository to temporarily roll back the changes whenever I need to revisit earlier experimental results. For grid search and random search, I wrote custom scripts when there are existing Python modules available that would have met all my needs. When old scripts became messy, I attempted to rewrite them, but only to end up with the same mess, and spent a lot of time solving problems that were already solved in the old code.

9 Acknowledgements

I would like to thank:

Dr. Zhenming Liu for his advice and support throughout this research project and my undergraduate studies. His course on "Probability and Computing" showed me things I would never have otherwise learned about computer science and introduced me to machine learning. I would also like to thank him for providing the data, storage, and computing power for this project.

Professors Lewis, Kemper, and Shi, for helping me plan out my 3-year graduation and fit this project into my schedule. I could never have done it without their advice.

Professors Ren, Frazier, and Lewis, for being on my honors examination committee.

10 Bibliography

References

[1] B. G. Malkiel, “The Efficient Market Hypothesis and Its Critics,” *Journal of Economic Perspectives*, 2003, ISSN: 0895-3309. DOI: 10.1257/089533003321164958.

[2] G. G. Tian, G. H. Wan, and M. Guo, “Market Efficiency and the Returns to Simple Technical Trading Rules: New Evidence from U.S. Equity Market and Chinese Equity Markets,” *Asia-Pacific Financial Markets*, vol. 9, no. 3, pp. 241–258, 2002, ISSN: 1573-6946. DOI: 10.1023/A:1024181515265. [Online]. Available: https://doi.org/10.1023/A:1024181515265.

[3] T. Z. Tan, C Quek, and G. S. Ng, “Brain-inspired Genetic Complementary Learning for Stock Market Prediction,”

[4] N. T. Laopodis, “Equity prices and macroeconomic fundamentals: International evidence,” *Journal of International Financial Markets, Institutions and Money*, vol. 21, no. 2, pp. 247–276, 2011, ISSN: 10424431. DOI: 10.1016/j.intfin.2010.10.006. [Online]. Available: http://dx.doi.org/10.1016/j.intfin.2010.10.006.

[5] S. Ghoshal and S. J. Roberts, “Thresholded ConvNet Ensembles: Neural Networks for Technical Forecasting,” 2018. arXiv: 1807.03192.

[6] G. E. Pinches, “The Random Walk Hypothesis and Technical Analysis,” *Financial Analysts Journal*, vol. 26, no. 2, pp. 104–110, 1970, ISSN: 0015198X. [Online]. Available: http://www.jstor.org/stable/4470663.

[7] Y. F. Wang, “Predicting stock price using fuzzy grey prediction system,” *Expert Systems with Applications*, vol. 22, no. 1, pp. 33–38, 2002, ISSN: 09574174. DOI: 10.1016/S0957-4174(01)00047-1.

[8] K. J. Oh and K. J. Kim, “Analyzing stock market tick data using piecewise nonlinear model,” *Expert Systems with Applications*, vol. 22, no. 3, pp. 249–255, 2002, ISSN: 09574174. DOI: 10.1016/S0957-4174(01)00058-6.

[9] E. F. Fama, “The Behavior of Stock-Market Prices,” *The Journal of Business*, vol. 38, no. 1, pp. 34–105, 1965, ISSN: 00219398, 15375374. [Online]. Available: http://www.jstor.org/stable/2350752.

[10] W. F. Sharpe, “The Arithmetic of Active Management,” *Financial Analysts Journal*, vol. 47, no. 1, pp. 7–9, 1991. DOI: 10.2469/faj.v47.n1.7. [Online]. Available: https://doi.org/10.2469/faj.v47.n1.7.

[11] E. F. Fama, “American Finance Association Efficient Capital Markets: A Review of Theory and Empirical Work,” Tech. Rep. 2, pp. 383–417.

[12] A. S. Chen, M. T. Leung, and H. Daouk, “Application of neural networks to an emerging financial market: Forecasting and trading the Taiwan Stock Index,” *Computers and Operations Research*, vol. 30, no. 6, pp. 901–923, 2003, ISSN: 03050548. DOI: 10.1016/S0305-0548(02)00037-0.

[13] H. Ince and T. B. Trafalis, “Short term forecasting with support vector machines and application to stock price prediction,” *International Journal of General Systems*, vol. 37, no. 6, pp. 677–687, 2008. DOI: 10.1080/03081070601068595. [Online]. Available: https://doi.org/10.1080/03081070601068595.

[14] J. Bergstra, D. Yamins, and D. D. Cox, “Hyperopt: A Python Library for Optimizing the Hyperparameters of Machine Learning Algorithms,” *Proceedings of the 12th Python in Science Conference*, no. Scipy, pp. 13–20, 2013. [Online]. Available: http://www.youtube.com/watch?v=Mp1xnPfE4PY.

[15] J. V. Tu, “Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes,” *Journal of Clinical Epidemiology*, vol. 49, no. 11, pp. 1225–1231, 1996, ISSN: 08954356. DOI: 10.1016/S0895-4356(96)00002-9.

[16] J. P. Bouchaud, “Price Impact,” 2009. arXiv: 0903.2428. [Online]. Available: http://arxiv.org/abs/0903.2428.

[17] W. F. Sharpe, “The Sharpe Ratio,” *The Journal of Portfolio Management*, vol. 21, no. 1, pp. 49–58, 2009, ISSN: 0095-4918. DOI: 10.3905/jpm.1994.409501.

[18] R Kohavi, “A study of cross-validation and bootstrap for accuracy estimation and model selection,” *Proceedings of the 14th international joint conference on Artificial intelligence - Volume 2*, vol. 2, no. 0, pp. 1137–1143, 1995.

[19] J. Bergstra and Y. Bengio, “Bergstra12a.Dvi,” vol. 13, pp. 1–25, 2012. [Online]. Available: papers3://publication/uuid/1190E1AB-0319-40C5-81CD-7207784965DE.

[20] J. Bergstra, D. L. K. Yamins, and D Cox, “Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures,” *Proceedings of the 30th International Conference on Machine Learning*, pp. 115–123, 2013. [Online]. Available: http://jmlr.org/proceedings/papers/v28/bergstra13.html.

[21] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. M. A. Patwary, Prabhat, and R. P. Adams, “Scalable Bayesian Optimization Using Deep Neural Networks,” 2015. arXiv: 1502.05700. [Online]. Available: http://arxiv.org/abs/1502.05700.

[22] A. Daerle and R. V. Zicari, *Neural Networks: Tricks of the Trade Second Edition*. 2010, ISBN: 9783642160912.

[23] G. Diaz, A. Fokoue, G. Nannicini, and H. Samulowitz, “An effective algorithm for hyperparameter optimization of neural networks,” vol. 61, no. 4, pp. 1–20, 2017. arXiv: 1705.08520. [Online]. Available: http://arxiv.org/abs/1705.08520.

[24] K Liano, “Robust error measure for supervised neural network learning with outliers,” *IEEE Transactions on Neural Networks*, vol. 7, no. 1, pp. 246–250, 1996, ISSN: 1045-9227. DOI: 10.1109/72.478411.

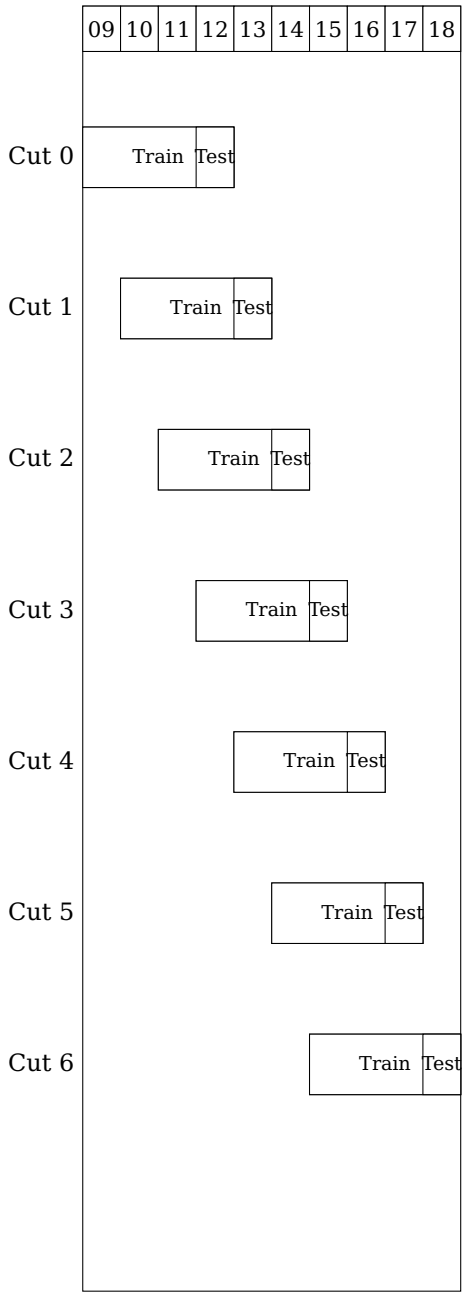

Figure 1: Data Cuts

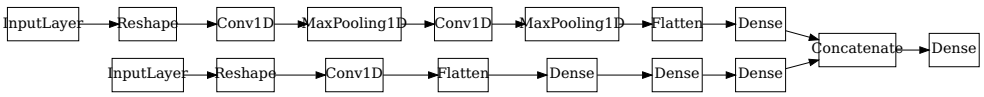

Figure 2: Example model generated from JSon Spec's

```
"conv_count": 0, // line 4
"dense_count":0, // line 7
```

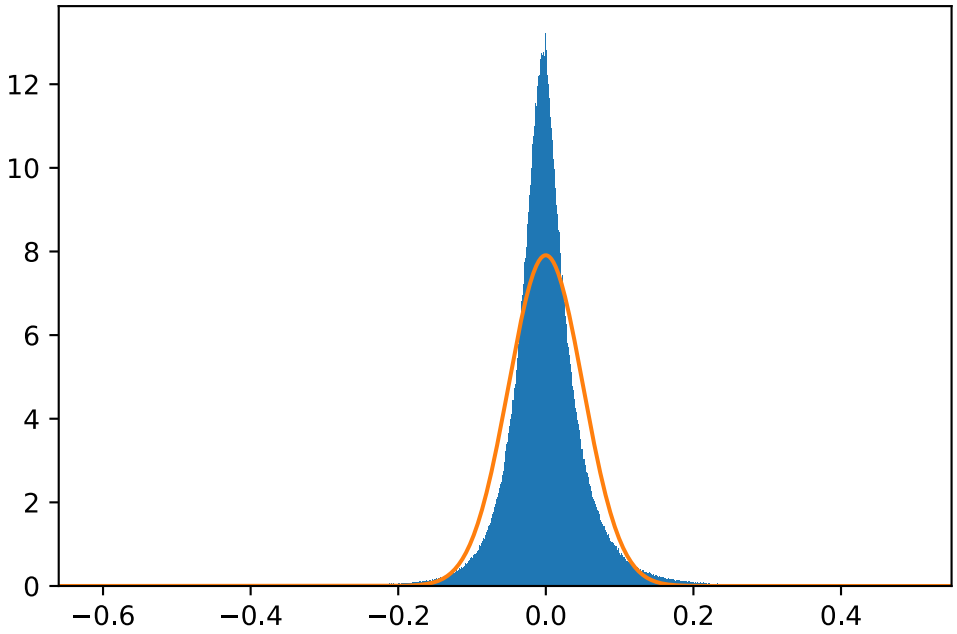

Figure 3: Returns do not have Gaussian distribution

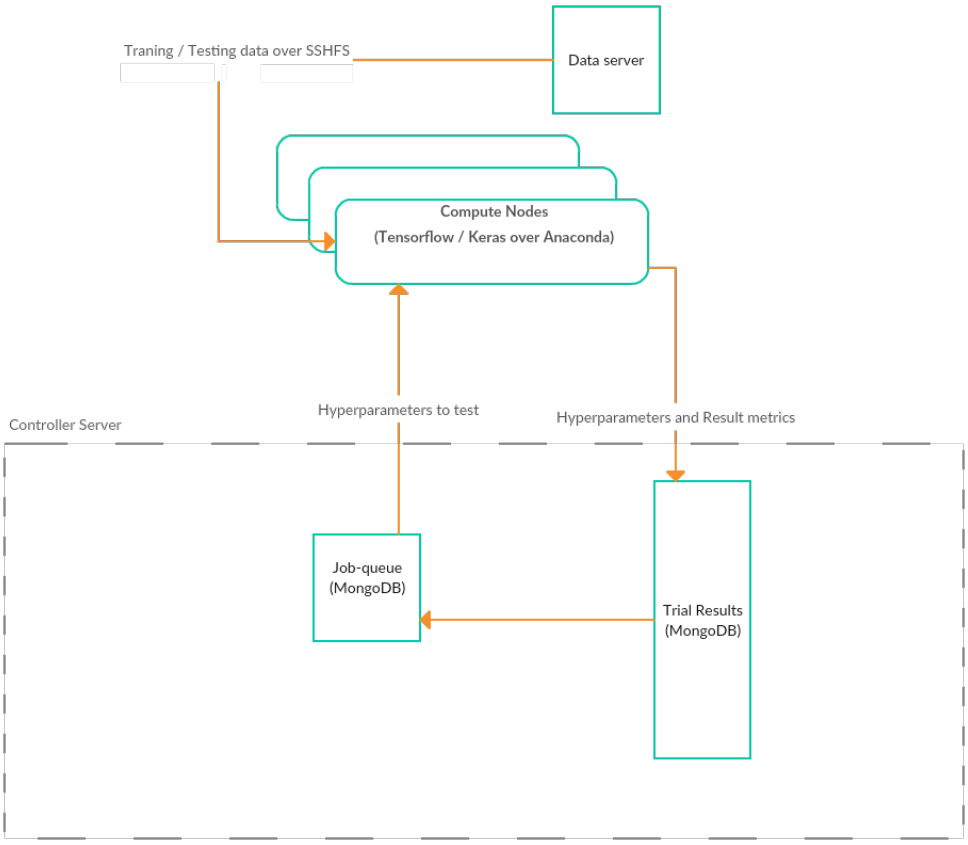

Figure 4: Data flow in distributed training

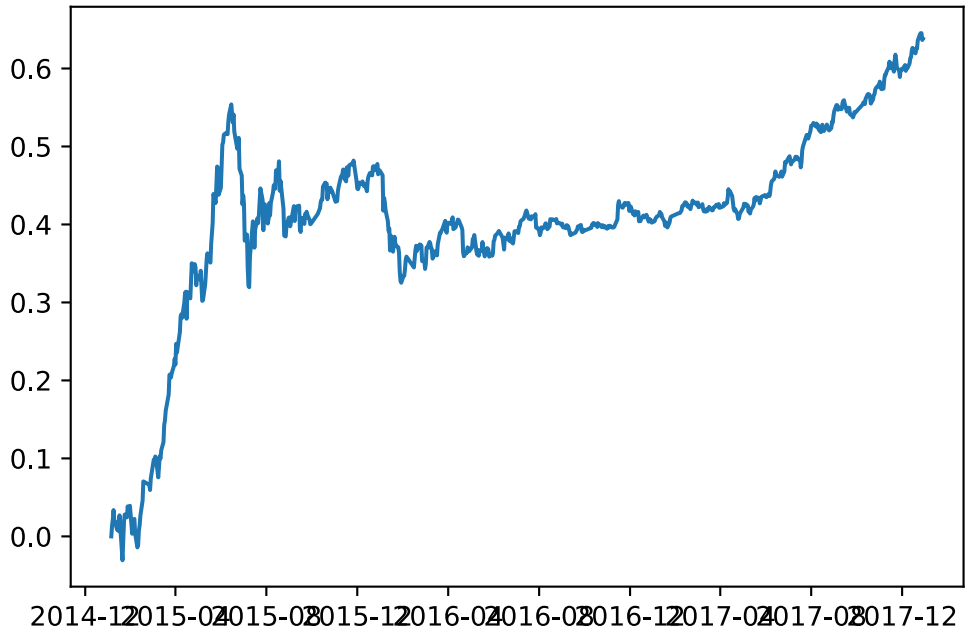

Figure 5: Log PNL over 3 years