Predicting Short Term Stock Returns

Chase Lochmiller
School of Engineering
Stanford University

Yuan Chen
School of Engineering
Stanford University

***Abstract*—As the capital markets evolve and expand, more and more data is being created daily. This explosion of data has made the flow of information much more efficient. As market participants act on this information flow, it drives market prices to more efficient values, . One of the driving forces in this march to efficiency, is the application of various algorithmic learning techniques on both market data and news sources. This paper seeks to examine a number of different forecasting techniques to predict future stock returns based on past returns and numerical news indicators provide by Raven Pack.**

I. Introduction

The capital markets serve as a place for investors to efficiently distribute capital to entrepreneurs and businesses so that they can effectively provide their services to society. In order for this process to work best, markets need to be both liquid and efficient. This means that an investor can buy or sell a reasonable quantity at a fair price. Long-term investors are not the only participants keeping the market working efficiently. There are many other participants that help in the process of price discovery, thus making the market more continuous and efficient. Short-term supply and demand imbalances are what drive stock prices to fluctuate in the short-term, which provides an opportunity for short-term market participants in two ways. First is in the event a short-term supply and demand imbalance pushes a stock out of line with fair-value, its an opportunity to profit by taking a position contrary to the recent price movement and profiting from reversion of the price to true fair-value once an environment of normal supply and demand resumes. In taking a position, it also leads to balancing out the supply and demand, which also helps drive the prices back to efficient values. Strategies such as this are called mean-reversion strategies. The second is for a short-term participant to forecast such supply and demand imbalances before they are fully realized in the marketplace, in which case they would be able to take a position in the direction of the imbalance and benefit from the short-term price appreciation resulting from the imbalance. Strategies such as these are typically referred to as momentum strategies or trend-following strategies. One factor that is effective in forecasting such imbalances is the news. As news comes out, it changes peoples set of information and hence changes the intrinsic value in the securities. This change can often cause one-sided imbalances in the supply and demand for individual securities. A second factor that can help short term investors forecast supply and demand for a particular security is the past return history of that particular security, as well as for other related securities. For instance, if an investor observes the price appreciating in Ford Motor Company, it is likely that many of the causes driving that price appreciation (i.e. consumer spending, new car sales, oil prices, etc) would also benefit General Motors, and so we would expect some related price increase in GM stock as well. This project seeks to use such news factors to predict the price of stocks over a two-hour window, from 2pm ET until 4pm ET when the market closes. An effective abstraction of factors can lead to both a profitable trading strategy, as well as pushing stock prices more in line with fair-value.

December 13, 2013

II. Data

The dataset used for this investigation was from the closed Kaggle Contest entitled *The Big Data Combine Engineered by BattleFin* [1]. Within the dataset we are given 510 days worth of trading data including 200 training days and 310 testing days on 198 different stocks and 244 different news features. The news features are meant to be numerical reflections of sentiment and are provided by RavenPack, a company specializing in providing such sentiment data. Each day contains the stock returns from the previous nights closing price in 5-minute windows up until 2pm, and the corresponding news feature values at each point in time as well. Our target variables are the final returns at the end of the day (4pm) for each stock. These end of day returns are given to us for the training data, but not for the testing data, and solutions must be submitted on the Kaggle website in order to determine efficacy while testing out of sample. Our goal is to build a learning model that incorporates the features and finds the lowest mean absolute error return from 2pm to 4pm compared to the actual return from 2pm to 4pm.

A. Feature Reduction

One of the primary challenges of this forecasting problem is selecting features to incorporate into our model. For each trading day, we are given a time-series of data points across 198 instruments, each of which has 55 measurements in the time-series. Additionally, we are given 244 news factors that also have 55 measurements in each day. If we treat each of our news and return observations independently (note: they are far from being independent), we end up with a training feature space that has 200 observations with 10,890 features for all of the observed returns and 13,420 features for all of the observed news indicator values, giving us a total of

24,310 features for each of our 200 observations. However, given we are working with forecasting within a time-series, the data points from our time series observations are highly correlated, which means if we treat them independently, our problem is highly subject to overfitting. This leaves us with the challenge of feature reduction. For this reason, it is very helpful to model the problem at a higher level before attempting any learning methods.

1) Time-Series Models: Before trying to model and incorporate many different data-points from within the time-series, we will start by looking at our problem from a higher level. Intuitively, if we are trying to predict the return of stock i at 4pm, the most meaningful observation is our most recent one. This effectively describes the AR(1) auto-regressive model. In this model, we have:

$$r_t = \theta_0 + \theta_1 r_{t-1} + \epsilon_t \tag{1}$$

where r_t is the return at time t, r_{t-1} is the return at time $t-1$, θ_0 and θ_1 are constants and ϵ_t represents our noise term with zero mean and constant variance σ_ϵ^2. For our purposes, the θ_0 term represents a systematic drift in our return. Modeling our returns as a random walk, our expected return is zero, thus making our systematic drift zero as well. In our application, we experiment fitting our model above inclusive of the constant drift term as well as the random walk hypothesis, fitting the model:

$$r_t = \theta_1 r_{t-1} \tag{2}$$

However, the AR(1) model only incorporates the last return value for our target stock and we have 24,309 other feature values to consider! For individual returns, we can employ another time-series model to get an aggregate estimate of fair value at 2pm. Stock returns are tricky, as the observed return value does not necessarily equate to fair value. Short term shocks in supply and demand can push the observed return values out of line with fair value. The theory is, that if we can more accurately represent the fair value at 2pm, it will be a more accurate predictor for the return and fair value at 4pm. One such model to help smooth shocks in supply and demand is the method of exponential smoothing. In this model, we can represent our estimate of fair value at time t, FV_t, based upon the returns to be:

$$FV_t = \alpha r_{t-1} + (1-\alpha) FV_{t-1} \tag{3}$$

where α is a smoothing factor, determining how fast we should decay old information in our current estimate of fair value.

While we have been discussing applying these time-series models to our stock return time-series', they can just as easily be applied to the time series' for our news indicators. Applying these methods gives us the most up-to-date representation of our returns at 2pm, and thus we can reduce each of our individual time series into a single value. This reduces our feature space from 55 values per stock or news indicator to 1, and reduces our overall feature space from 24,310 features to 442 features.

2) Clustering Features: One critical observation to make in analyzing financial time-series is that stock returns tend to be very correlated. Similarly, so do news indicators. As such, even with a significant amount of feature reduction, to one feature per stock or news indicator, we are still very much subject to over-fitting because of the intrinsic correlations between our features. To help address this problem, we use a couple of different clustering and grouping techniques to aggregate feature information.

One very tricky aspect to clustering techniques is choosing the number of clusters to reduce your feature set into. One technique we've employed to help us in this process is using PCA to give us a statistical picture of how much variance is explained by the first n eigenvectors. In our application, PCA is a process that performs singular value decomposition on our observation matrix, but before doing so, it normalizes our data set so that our singular values sum up to 1, $\sum_{i=1}^{n} \sigma_i = 1$. In this way, the corresponding eigenvectors represent an orthonormal basis for defining our vector space our singular values represent the percentage of variance in our dataset explained by those eigenvectors. Looking at a plot of this is helpful for us to estimate how many clusters we should be reducing our data to. Plotting variance explained by the principle components of our set of stock returns, we see very clearly, that most of the variance is explained in the first few principle components, so we can group this into a very small group. Namely, the first 4 components explain 82% of the variance, and the first 7 components explain over 98%. For our stock return data, this gives us an idea that we should probably experiment with somewhere between 3 and 7 groupings for our data. For the news indicators, the data was even more skewed, with closet to 95% of the variance being explained by the first principle component and over 99% being explained by the first 3 components. This tells us we should experiment with between 1 and 3 groupings.

Once we have estimated the number of groups we want to reduce our features into, we explored two different approaches for grouping the features. The first is to use linear combinations of the features with the weighting set based upon the first n principle components. This method can at times add noise, as it does not produce sparse solutions, so even if two products are very uncorrelated and they end up mostly being represented in two separate principle components, you still often end up with the products being partially represented in each component.

The second approach we explored was in using K-means to cluster our features into n groups. The K-means algorithm can be described as follows:

```
1. Initialize cluster centroids $\mu_1, \mu_2, ...\mu_n \in \mathbb{R}^k$
2. Repeat until convergence: {
   For every $i$ set
      $c^{(i)} := arg\min_j \|x^(i) - \mu_(j)\|^2$
```

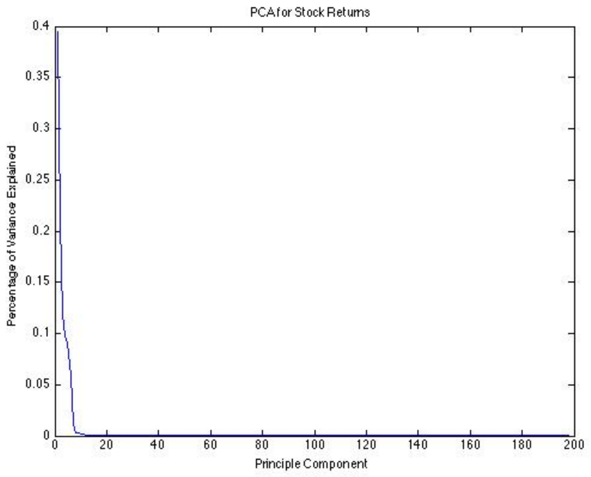

Fig. 1. Plot of the Stock Return Variance Explained by Principle Components

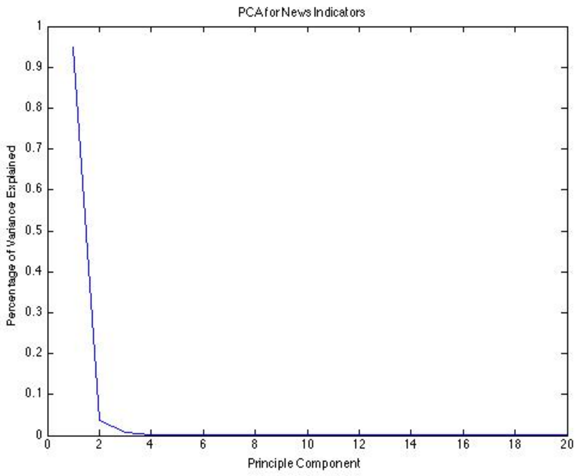

Fig. 2. Plot of the News Indicator Variance Explained by First 20 Principle Components

```
    For each $j$ set
        $\mu_{(j)} := \frac{\sum_{i=1}^{m} 1\{c^{(i)}=j\}x^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)}=j\}}$
}
```

K-Means is a simple algorithm to aggregate features, and it produces sparse solutions, thus if we have two wholly uncorrelated features, each feature will only be represented in one cluster, which helps to filter out noise that can be introduced to our analysis. We will discuss further the application of K-means versus PCA in the results section.

3) Further Feature Reduction: One final method we employed to attempt to reduce our feature space was to take the correlation matrix between all our stocks and news indicators, and set a cutoff for minimum correlation to be included in the regression. This is one blunt way to cut the feature space down.

III. Methods

We employed a number of different forecasting methods to this data set, as part of a trial and error process to find the best forecasting method. The details of the methods are described below and the results are discussed in the next section.

A. Baseline Predictions

To get a couple of baseline predictions, we look at the problem in two ways. The first is to use the random walk hypothesis discussed earlier, which says that our expected return is zero and our MLE for our return at 4pm is simply the same as the return at 2pm. Remember, these are all relative to the previous nights closing price, so our effective predicted return from 2pm to 4pm is zero (i.e. were saying the stock price will not change over the next 2 hours). This gives us training mean absolute error of .4406% and test error of .42596%. The second is to predict that the stock price return will be the average of the returns for the day. This predictor is predicated on the notion that stock prices tend to revert to the average, and is the MLE if we were to assume that the time series values were not serial. Using this baseline prediction our training mean absolute error is .5946% and our test error is .57264%.

B. Linear Regression

Linear regression is one of the most standard tool for predictive modeling because it is stable and simple to implement. We used linear regression in a number of different ways, namely regressing various feature sets to try to predict the future return. With our naive predictor of the last return, we actually found this to be a very difficult benchmark to beat. One effective method was instead of trying to predict the future return, predict the change in return from the last return and then add back the last return value to get your predicted return. So our regression equations become:

$$\delta = r_{4pm} - r_{2pm} = \Theta^T X \tag{4}$$

Linear regression is an effective technique, but it has its shortcomings. One that affects this problem most directly is that if you are solving a linear regression problem using the normal equations, and there are many more features than observations, the observation matrix becomes singular. We have approached this by trying a number of different feature reduction techniques as described above.

C. Poisson Regression

The Poisson regression is another member of the family of generalized linear models with linear regression. It is typically used to predict things counts of random variables, like counts of customers that come into a store. In our case, log of returns has a number of different attractive properties that make this regression particularly interesting. Just like counts of random variables, returns of stocks have a lower bound (i.e. we could never have a return of less than -100%, just as we could never have a negative customer enter our store). Given this however is not true of the log of the return. One common assumption is for prices to be log-normally distributed instead of normally distributed. For the Poisson Regression, regression equation becomes

$$\log(1 + r_{4pm}) = \Theta^T X \quad (5)$$

D. Softmax

Finally, we experimented with implementing a family of classification algorithms in predicting our stock returns, namely we used Softmax. It is very interesting to see the application of classification algorithms performance in the time series domain where regression plays a dominant role.

There are 3 questions need to be answered when applying the classification algorithms. First we must frame the problem properly. To apply classification algorithm to a continuous time series problem, we need to discretize the problem space. To make the prediction tractable, we would like to perform discretization on high probable space. Given our random walk hypothesis, our MLE for the stock return is the most recent return. So we chose to frame the problem as prediction of the stock return at 4:00pm relative to the stock return at 2pm. To apply classification algorithms, we need to generate class labels. Initially we define 1bps as 1 label. For example, if the return at 2p.m. is 2.43, and we are going to have 201 labels, then our prediction of the return at 4:00pm ranges from 2.43 +/- 100bps, which catches most of the return variation in the training data.

Second, we must define the feature space. We picked two feature spaces to study the correlation between the features and the end of the day price. The first feature space choice is the provided by the 244 news features at 2pm. The reason why we only pick the news features at 2pm and dont include the news feature before it is because we believe in efficient market theory in the weak form. First of all, the news features at 2pm. should have included all the information and impact of the news features in the previous time like 1:55pm, so including the past features would be redundant. Secondly, we hope the market efficiency is weak such that the market at 4:00 p.m. still reaction to the news feature at 2pm. The second feature space choice is the return sampled by 5-minute interval ranging from the open to 2pm, giving us a total of 55 features.

Third, and finally, we must prove that classification algorithms are applicable. To test this hypothesis, we performed a control group experiment by applying Multinomial Nave Bayes to classify 201 labels and random number generator to generate 201 labels according to normal distribution with return at 2pm as mean. We tested 198 stocks on 310 days and the results show that classification algorithms has strong prediction capability and proves that classification algorithms are applicable to tackle time series problem. Our results are shown in Table I.

We switched to Softmax as our major classification methods due to its robustness. The cost function of Softmax with weighted decay to tame over fitting is defined as follows:

TABLE I
PREDICTION RESULTS KEY

$PredictionAlgorithm$	Accuracy
Multinomial NB	0.43172
Random Normal distribution	0.47431

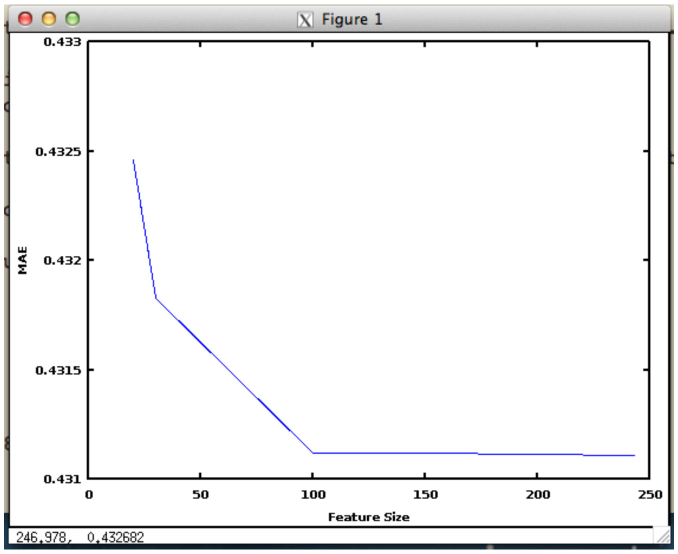

Fig. 3. Plot of MAE from Softmax on the News Indicators with Reduced Feature Space

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{k} 1\left\{y^{(i)} = j\right\} \log \frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}\right] \quad (6)$$

$$+\frac{\lambda}{2}\sum_{i=1}^{k}\sum_{j=0}^{n}\theta_{ij}^2 \quad (7)$$

To minimize this cost function we used a package in Scipy, L-BFGS,to minimize the cost function.

IV. RESULTS

Our general approach in producing results was trying a whole bunch of different combinations of fitting techniques with various sets and subsets of features by employing numerous feature reduction techniques. Implementing softmax on the news indicator space, we tried to find a more efficient solution by reducing the feature space using PCA, but the results actually got worse as seen in Fig. 3. This speaks more broadly to the news indicators having very little predictive power. So in our case, the issue was not with overfitting our news indicators, it was just in the fact that they did not have a lot of information to predict upon. Additionally, using our softmax implementation, we analyzed various numbers of return features to include for each stock. By analyzing, Fig. 4 we can conclude that the prediction of the stock return at 4:00pm can be distorted by including the stock returns in the morning. As using more and more recent stock returns, the MAE consistently reduces.

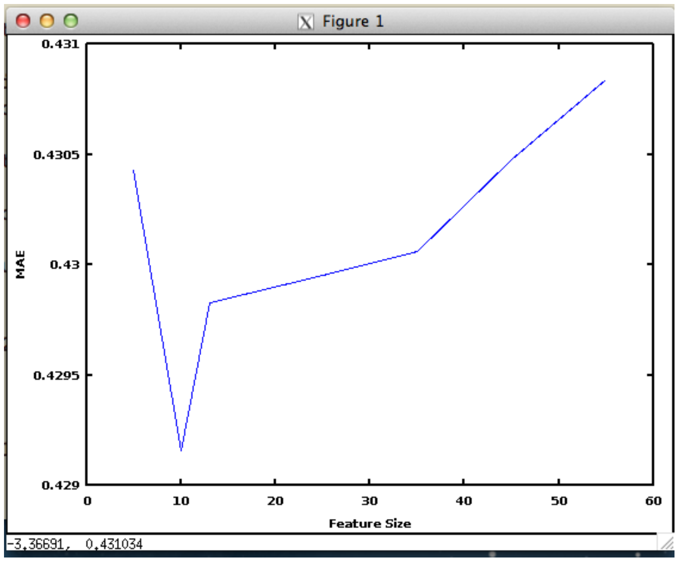

Fig. 4. Plot of MAE from Softmax on the Returns with Reduced Feature Space

For softmax specifically, to further reduce MAE, we investigated another factor that may impact the prediction accuracy. Just as we may be overfitting with the features, we may be discretizing the prediction space with too fine granularity. What we discovered is that if we attach 1 label to 1bps change, we have 201 labels. And with only 200 days training days for each stock, most of the labels are never exercised during the training. It is counterintuitive, but when we increase the descretization granularity, we actually improved our accuracy. In addition, we also decreased our prediction range from +/- 100 bps to 20bps to ensure all our labels get hit during the training and we use 2bps per label with 21 labels to achieved 0.42881, the best result we have for the classification methods to this continuous time series problem.

The table below shows some of our results for various combinations of features and regression methods, and there were many more combinations that were attempted and are not recorded here. There were a few notable results and conclusions that we came to. The news factors as a whole tended to have very little predictive power. In all cases, the results tended to be better without the news factors than with the news factors. As we have no insight into how they are being produced, a likely conclusion is that either the market has already absorbed all of the information being transmitted in the news analytics, or the news information affects the market over a different time horizon compared to our prediction time horizon. A second very notable outcome was that PCA tended to add more noise in the feature reduction process than K-means, and generally speaking K-Means was a more effective factor reduction technique. A third point that became very clear as we experimented with more and more approaches was that predicting the change in return over the final 2 hours of the day, turned out to be a much more effective way to produce our forecasts. One final point that is very notable, is that the market is really efficient and beating our most naive prediction of the last return value turned out to be really challenging. Using all of these advanced data analysis techniques, only a handful were able to outperform this very naive approach. Our most effective approaches were to use linear regression on 5 stock return clusters that were clustered using K-means to estimate the delta between 2pm and 4pm. We used both linear regression and Poisson regression, and the Poisson regression outperformed slightly in test (3bps). As far as a benchmark goes, the linear regression technique would have put us into 12th place in the competition and the Poisson regression technique would have put us into 10th place.

TABLE II
PREDICTION RESULTS

Strategy	MAE Train	MAE Test
Random Walk Baseline	0.4406	0.42596
Average Return Baseline	0.5946	0.57264
AR(1)	0.43762	0.42907
EMA Returns	0..43913	0.42585
LR EMA Returns	0.43643	0.42754
LR News, Stock K-means clusters w/ema	0.43198	0.42975
LR News, Stock K-means clusters, Stock Return w/ema	0.42966	0.43307
LR Delta Stock K-means clusters, Last Stock Return	0.43345	0.42495
LR Delta Stock PCA clusters, Last Stock Return	0.43630	0.42988
PR Delta Stock K-means clusters, Last Stock Return	0.43512	0.42464
PR Delta News, Stock K-means clusters, Last Stock Return	0.43577	0.42917
PR Delta Stock K-means clusters, Stock Return w/ema	0.43241	0.42585
PR Delta Stock PCA clusters, Last Stock Return	0.43626	0.43706
LR News, Stock Correlation cutoff	0.43258	0.46518
LR Stock Correlation cutoff	0.43606	0.45345
SM 244 News Features @ 2pm	0.41718	0.43111
SM 55 Return Features for each stock	0.41539	0.43142
SM Last 10 Return Features for each stock	0.43578	0.42881

TABLE III
PREDICTION RESULTS KEY

Abbreviation	What it means
LR	Linear regression
PR	Poisson regression
SM	Soft max
Delta	Regress on the change in return from 2pm to 4pm

V. CONCLUSION

The stock market is a very efficient environment where information is reflected very rapidly in prices. However, there are still ways to process information and produce forecasts that can add value and create more efficient predictions of prices than the current market values.

REFERENCES

[1] https://www.kaggle.com/c/battlefin-s-big-data-combine-forecasting-challenge
[2] http://cs229.stanford.edu/materials.html
[3] http://mdp-toolkit.sourceforge.net/
[4] http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html
[5] http://ufldl.stanford.edu/wiki/index.php/Softmax_Regression