K-Means Clustering of Daily Stock Data

Jessica Kwok, Melody Zhao, Kewei Zhou
Harvey Mudd College

Abstract

The purpose of this paper is to sort general trends of daily stock data into clusters, then use the output to predict future trends and make connections with significant events happening during a specific period of time. K-Means Clustering is used to cluster the general trends, while the Silhouette score is used to optimize the parameters for the distance function. Finally, both the Recurrent Neural Network(RNN) and Markov Chain are used to predict future trends. The result of accuracy versus input sequence length for both methods are compared. The result shows that while Markov Chain might appear to have a higher accuracy with shorter sequence lengths, it decreases linearly as sequence length increase and is sensitive to the data set. On the other hand, the accuracy of RNN has remained relatively constant around 80% for different data sets and sequence length, and therefore is the more appropriate model for the purpose of this paper.

1 Introduction

It is commonly thought that stock indexes are only measures of stock market performance, or the economy in a country. However, they are also indicators of a country's non-economic conditions. For example, a research done by the University of Utara in 2017, by studying the stock prices in Pakistani for a period of time, has concluded that the stock market returns and volatility have noticeable relationship with a country's political instability (Irshad). History often repeats itself. Therefore, it becomes essential to connect unusual yet significant time periods, such as the 911 terrorist attack in 2001 and the outbreak of the contagious illness SARS in 2003, to the change in stock market performances. If similar incidents happen again, the results of this investigation can not only allow government to reflect and update on the effectiveness of their policies and measures during states of emergencies, but also to allow public and private institutions to be more prepared of what to expect.

However, there are so many events that have happened throughout just the past 20 years that it would be impossible to connect each significant event to the stock performance during that period manually. Furthermore, how would one know which events are noteworthy enough such that it had an impact on the stock market? It would also be useful to know which events had similar reactions from the stock market and for the same event, how the countries react differently. For example, is the stock performance during the outbreak of SARS in 2003 and that of COVID19 in 2020 similar? Did the US and China react similarly to either outbreaks? Finally, based on previous stock performances, is it possible to predict future trends? This paper aims to investigate and produce both quantitative and qualitative answers to these questions.

In order to identify unusual events in the stock market, this paper approaches the solution using K-Means clustering, a popular unsupervised machine learning algorithm. It sorts stock performance over short period of times into clusters. In other words, each short period of time is labeled with a number that describes the general shape of its stock performance. Then, by applying either Markov chain or RNN on the time series sequence of labels, the model predicts the nth label based on the previous n-1 number of labels (sequence length). Finally, this paper compares the accuracy between the Markov Model and RNN in different sequence lengths.

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

2 Data Collection

The main source of data collection is Yahoo Finance and Barchart. For the this project, daily historical data from 2000 to current for S&P 500 is used. Historical data for Heng Sung Index (Hong Kong), Nikkei 225, SSE Composite Index (Shanghai) and SZSE Component Index (ShenZhen) are also collected for future usage. These stock indexes are selected because they are some of the most commonly used indexes that are able to represent the general stock market in each country. Historical data is downloaded in the form of csv file and contains the following information: date, open price, high, low, closing price, adjusted close, and volume.

3 Methodology

3.1 Data Pre-Processing

3.1.1 Standardizing Data

As mentioned in the data collection section, the following information for a stock is obtained from financial website from 2000 to current: date, open price, high, low, closing price, adjusted close, and volume. In the attempt to build a model that can identify trends in historical data and apply to similar future occurrences, stock prices and volatility are the most important characteristics. Since it is essential to see how one event may affect the stock prices and volatility, the percent change of price and volume are used to train the model. Since these two components are not included directly in the raw data, percent change for price and volume are calculated. The equations to calculate the percent change in volume and price are defined as the following:

$$\text{Percent Change in Price (T)} = \frac{\text{Price(T)- Price(T-1)}}{\text{Price(T-1)}} * 100$$

$$\text{Percent Change in Volume (T)} = \frac{\text{Volume(T)- Volume(T-1)}}{\text{Volume(T-1)}} * 100$$

where T is the date of interest. This is done directly in the csv file through Microsoft Excel. The calculated values are stored in new columns labeled as "PercentPrice" and "PercentVolume" respectively.

3.1.2 Matching Standard Date

The second step in pre-processing involves standardizing the range of dates that would be used in building the machine learning model. Since one of the ultimate goals in this project is to compare stock performance reactions in different countries, it is important to note that due to a variety of reasons, such as countries having different holidays, the days that each stock exchanges open differ. In order to resolve this issue, a system of standard date is enforced to ensure that data across different stock indexes can be easily compared.

First, a separate csv file containing all standard dates is created. This is done purely on Microsoft Excel, and the column of standard date contains all weekdays from 2000 to current. Then, the csv file containing the standard dates, as well as the csv file containing the calculated percent change in price and volume, are passed into the "preprocessed.py" as input. In the python script, the standard list is looped through, and if the historical data csv file contains information for the date of interest, the percent change in price and volume are assigned to that date. However, if the historical data did not contain the date of interest, the associated percent changes in volume and price of that date are defined to be 0. This is based on the assumption that the economic condition would not alter greatly in a day without significant events, then the change in price and volume should be similar for consecutive days. The output of the python file is the input for the machine learning model, in which the standard dates with their associated percentage changes in price and volume values are written to a new csv file named "newProcessed.csv".

3.2 Principal Component Analysis (PCA)

After pre-processing the stock market data, the percentage change of price and volume every day of a specific market are obtained. The data is divided into smaller chunks of 30 days each so that we can

look at a market at a specific period of time. By the shift window method, we result in a set of 30-day data starting from any day. The shift window method enables us to study the subtle changes in the stock market over time.

We then use principal components analysis to categorize each chunk of 30-day data. Principal components analysis(PCA) is a statistical procedure that converts a set of data pertaining to possibly correlated variables into a set of linearly uncorrelated variables. These uncorrelated variables are called principal components. For a two-dimensional dataset, we result in a main principal axis, which represents the direction with the largest possible variance, and the second principal axis, which represents the direction with the largest variance under the constraint that it is orthogonal to the main principal axis. The lengths of the principal components are the amount of variance explained by each axis.

We calculated the principal axes of our data and their respective variance using the library sklearn.decomposition.PCA. We input a list of paired percentage change in price and percentage change in volume. Each pair corresponds to the stock market data of a specific stock on a specific day. Using the explained_variance_ function, we were able to extract two variances represented by the principal components in decreasing order. We then found the vector representation of the principal components using the components_ function. Since the two principal components are guaranteed to be orthogonal, we could simplify the representation of the principal components with the angle between the main principal component and the horizontal. Thus, we calculated the angle using

$$\tan^{-1} \frac{\text{y component of the main principal axis}}{\text{x component of the main principal axis}} \tag{1}$$

Along with the mean of percentage change in price and the mean of percentage change in volume, we characterized each chunk of data using a five-dimensional vector: mean of change in price, mean of change in volume, main variance, second variance, and angle of the main principal component.

3.3 Visualize Data

In order to visualize the processed PCA data, a plot with percent change in price as its x-axis and percent change in volume as its y-axis is generated. The plot contains:

1. The 30 data points
2. The 2 eigenvectors (Principal and second Principal)
3. The ellipse corresponding to the eigenvectors, with the eigenvalues as its width and height, and the angle between the principal and horizontal axis as its degree of rotation

An example of the plot looks like the following (Figure 1):

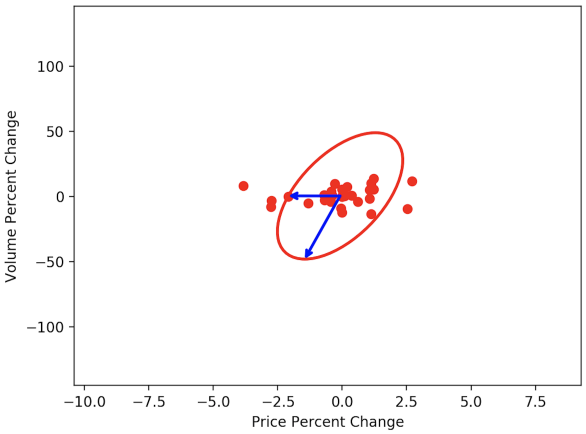


Figure 1

It can be seen that the PCA is performed correctly on the data points since the eclipse is a relatively good representation of the data. In addition, notice that the scale in the y-axis is substantially greater than that of the x-axis. This implies that one of the eigenvalues will also be noticeably bigger than the other.

3.4 K-Means Clustering

K-Means clustering is an unsupervised machine learning technique that uses input vectors in order to label the dataset. K-Means defines k number of centroids and cluster data points according to the distance function. The data point is labeled such that the distance between the point and the centroid of a cluster is minimized. The code for K-Means is adapted from an online source code from Visual Studio Magazine. [1]

3.4.1 Normalize Data

K-Means Clustering is called on an array of array in the following format:

$$\begin{bmatrix} p_1 & v_1 & \lambda_{prin,1} & \lambda_{minor,1} & \theta_1 \\ p_2 & v_2 & \lambda_{prin,2} & \lambda_{minor,2} & \theta_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ p_n & v_n & \lambda_{prin,n} & \lambda_{minor,n} & \theta_n \end{bmatrix}$$

where each row is an output of the PCA and thus a vector that represents the stock performance trend in a period of 30 days. Note that p is the average percent change in price, v is the average percent change in volume, λ_{prin} being the principal eigenvalue, λ_{minor} being the minor eigenvalue, and theta being the angle between the horizontal axis and the principal eigenvector, in degrees. Before running the clustering algorithm, all of the raw data are normalized using the Min-Max normalization method, which utilizes the following technique: $x' = \frac{x - min(x)}{max(x) - min(x)}$. Doing so would scale the range of the data to between 0 and 1. The normalized array of array would then be used as input for clustering.

3.4.2 Distance Function

In order to determine which cluster a data point belongs to, a function is defined to calculate the data point's distance from the mean of each cluster:

$$D = W_{center} \cdot \Delta center + W_{\lambda_1} \cdot \Delta\lambda_1 + W_{\lambda_2} \cdot \Delta\lambda_2 + W_\theta \cdot \Delta\theta$$

where the difference between each variable is

$$\begin{aligned} \Delta center &= \text{norm}(p_i - \bar{p}, v_i - \bar{v}) \\ \Delta\lambda_1 &= \text{abs}(\lambda_{1,i} - \bar{\lambda_1}) \\ \Delta\lambda_2 &= \text{abs}(\lambda_{2,i} - \bar{\lambda_2}) \\ \Delta\theta &= \text{abs}(\theta_i - \bar{\theta}) \end{aligned}$$

and $\bar{p}, \bar{v}, \bar{\lambda_1}, \bar{\lambda_2}, \bar{\theta}$ are mean of each variable in the specific cluster. In addition, the constant weight for the variables are $W_{center}, W_{\lambda_1}, W_{\lambda_2}, W_\theta$ respectively, such that

$$W_{center} + W_{\lambda_1} + W_{\lambda_2} + W_\theta = 1$$

Applying the silhouette function on the clustering done for the midterm project with the previously designed distance function and the parameters described, a silhouette score of 0.27 was achieved. Through changing the distance function above, the silhouette score had a substantial increase, changing from 0.27 to 0.51. The new distance function used in obtaining the result for the final project is the following:

$$D = W_{center} \cdot \Delta center^2 + W_{\lambda_1} \cdot \Delta\lambda_1^2 + W_{\lambda_2}^2 \cdot \Delta\lambda_2 + W_\theta \cdot \Delta\theta^2$$

In comparison to the previously used distance function, all of the Δ values are squared. This would mean that the data points with relatively small distance before get affected less than those with large

[1] Garbade, Michael J. "Understanding K-Means Clustering in Machine Learning." Medium, Towards Data-Science

distances, meaning that the data points that are not similar would seem even further away from each other. Then, it would make sense that the new distance was able to perform better. Just by changing the distance function alone without alternating the parameters, the silhouette score increased greatly, which is why the revised distance function is adopted in obtaining final project results.

3.4.3 Determine K

The parameter K represents the number of clusters the ellipses are divided into. There is no existing theoretical k value, so we determined k value by numerically finding a balance between a small average distance from each ellipse to their corresponding cluster center and a reasonably small k value.

We defined a cluster_distance function that finds the average distance from the ellipses to their corresponding cluster center associated with a specific value. It takes in a list of clusters, where each cluster is a list of 5-dimensional vectors representing the ellipses in the cluster. The function finds the mean of ellipses' centers in each cluster as the cluster center and sums the distance from each ellipse to the cluster center. Cluster_distance then sums the average distance in each cluster and divide it by the number of clusters. The output gives us a measure of how close ellipses in the same cluster are for a specific k value, weighted by the number of clusters. Then, we clustered the ellipses using k values ranging from 1 to 7. We found the cluster distance of each k value and plotted the cluster distance against the k value.

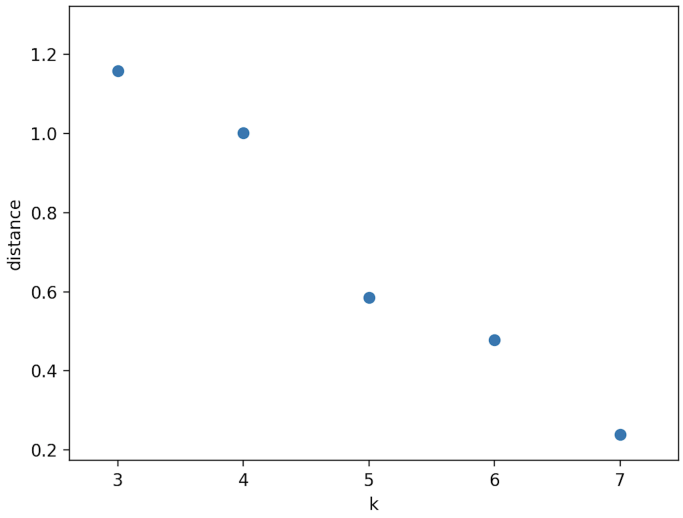


Figure 2

It can be seen that from Figure 2 that the average distance within a cluster is the lowest when k=7. Thus, it can be concluded that k=7 is the optimal k to cluster the data in.

3.5 Optimize Parameters

3.5.1 Preliminary Weights for Midterm Project

For the midterm project, we manually determined the weights for the k-means model such that the result of the clustering makes the most sense. Insights to how parameters were picked can be extracted from looking at the normalized data mentioned in 3.4.1. A snippet of the data is shown below (Figure 4), where the columns are center of price, center of volume, λ of principal axis, λ of minor axis, and the angle respectively. Although each of these values are ranged from 0 to 1, it is obvious that, after normalizing it, the percent change in volume (third column) becomes extremely small comparing to the others. The explanation for this is briefly mentioned in the PCA visualization section (3.3) - it is due to the fact that the range of percent change in volume is considerably bigger than all the other variables. If not dealt with properly, this would result in λ of the minor axis and especially the

angle completely over-powering it. After several trials and errors, it can be concluded that,in order to compensate for the issue, the weight for the principal axis is set to 0.78, while minor_W and angle_W are set to 0.015 and 0.005 respectively. Since the centroid should play a relatively big role in cluster, its weight is set to 0.2.

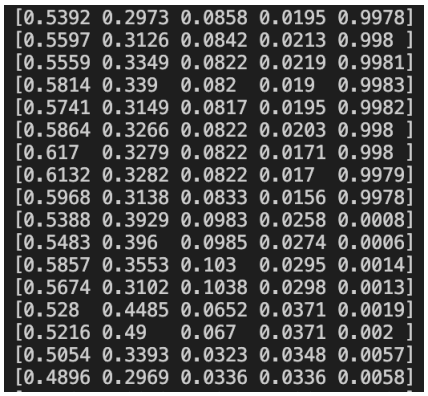


Figure 3

3.5.2 Silhouette Score

In order to verify whether the distance function and hence the clustering, using the preliminary weights, are reasonable, it is necessary to find a measure that can quantify how "good" the clustering is. Thus, we introduced the use of Silhouette Coefficient. The Silhouette Coefficient is a measure of two major characteristics of a cluster - cohesion and separation. Cohesion represents how similar a sample is comparing to other samples in the same cluster, while separation means how different samples from each clusters are. The equation for the Silhouette score s is

$$s = \frac{b - a}{\max(a, b)}$$

where a is the mean intra-cluster distance
b is the mean nearest-cluster distance of all the samples.
Note that the nearest-cluster of a sample is that of the smallest distance with the sample, but cannot be the cluster that the sample is in. The Silhouette score ranges from -1 to 1, such that -1 represents that samples are incorrectly clustered and 1 represents that samples are correctly clustered and the clustering is well-structured. The following table from the text on *Partitioning Around Medoids (PAM)* summarizes a scale of the Silhouette score and what it implies for different ranges:

RANGE OF SC	INTERPRETATION
0.71-1.0	A strong structure has been found
0.51-0.70	A reasonable structure has been found
0.26-0.50	The structure is weak and could be artificial. Try additional methods of data analysis.
Below 0.25	No substantial structure has been found

Figure 4

Having the tool to verify whether the clustering is well-defined, we calculated that the Silhouette score for our previously selected weights ([0.2, 0.78, 0.015, 0.005]) was 0.27, implying that the clustering had a weak structure. In order to resolve this problem, we came up with two methods: changing the overall structure of the distance function (Section 3.4.1) and optimizing the current weights used for the distance function (Section 3.5.3).

3.5.3 Optimizing Weight

In order to optimize the weight, we need to have an objective function and some constraints. We have decided the the objective function will be the negative Silhouette score, since we are minimizing the function. Most systematic methods requires either information related to the gradient of the function or could only find local minimum rather than global ones. Therefore, we decided to first do a manual estimation, then use the results for the systematic estimation.

3.5.3.1 Manual Estimation

Since the Silhouette score is a relatively rough function, it contains a lot of local minimum. The Scipy minimize methods mainly support finding the local minimum. In addition, since the function has a dimension higher than 3, it was impossible to visualize it to spot the global minimum. Therefore, manual estimation is required. We performed this by choosing around 15 different sets of weights, performed clustering using the distance function with the specific weight, then calculated the Silhouette score for the clustering. The 15 weights tested included different types of points:

- evenly distributed weights
  e.g. [0.25, 0.25, 0.25, 0.25]
- one of the four weights dominates
  e.g. The principal eigenvalues dominates [0.15, 0.75, 0.05, 0.05]
- two of the four weights dominates
  e.g. The principal and secondary eigenvalues dominate [0, 0.5, 0.5, 0]
- three of the four weights dominates
  e.g. The principal, secondary eigenvalues and center dominate [0.3, 0.3, 0.3, 0.1]

After calculating the Silhouette score for each of these sets, the results shows that the set of weights [0.15, 0.75, 0.05, 0.05] has the maximum score. This implies that having a dominant weight for the principal eigenvalue results in the best clustering for this specific set of data.

3.5.3.2 Systematic Estimation

Knowing the brief estimation of where the global minimum will be, we can apply the Scipy minimize function on it. Since we want the maximum Silhouette score, we defined the objective function as the negative of the Silhouette score. Note that we used the result from the manual estimation as the initial guess for this function. At first, we approached it using the Nelder-Mead method, a direct search method that did not require any information of the objective function's gradient. However, we realized that there it does not support non-linear constraints, so we moved on the Sequential Least Square Programming (SLSQP) method. The two constraints we had were that the sum of the weights need to be 1 and each weight is bounded from 0 to 1. A difficulty that we have encountered is the inconsistency in the Silhouette score, which possibly caused confusion for the minimize function. This is due to the fact that the Silhouette score is calculated through random sampling. Thus, we attempted to resolve this issue by increasing the sampling size. Although there was some occasional inconsistency, we found that it was sufficient to give us a reasonable estimation. We concluded using this method that the optimal weight for clustering is [0.114, 0.786, 0.0365, 0.0635], which has a Silhouette score of 0.73. This is a 0.22 points increase comparing to the score for the preliminary set of weights. Based on the previous scale (Figure 3), this implies that the clustering has a strong structure.

3.6 Tying Together

Tying all the strategies discussed from 3.4.1 to 3.4.3, K-means clustering is performed. First, the normalization method is applied so that the raw data imported from the csv file is processed and is fit into the range from 0 to 1. Then, using the strategy described in 3.4.3, the optimal K value determined is 7. Next, in sections 3.5, the optimal weight for clustering is determined. As described in section 3.5.3, we manually selected weights and tested the clustering algorithm. Then, we used the Scipy optimize function systematically in determining the optimal weight. Using the optimal weight, clustering is performed once again. The clustering algorithm first randomly labels all of the data into 7 clusters and calculates an initialized mean. Then the cluster will be "updated". The updating cluster technique uses the initialized mean and the distance function to calculate all the data points'

distances relative to the means of each cluster. Then, each data point is re-clustered based on the distance. After re-clustering, a new mean is generated for each cluster, and the process repeats with re-clustering following by updating the means. Clustering is done and exits when there is no change in cluster means, meaning that no data points are being relabeled.

3.7 Markov Model of Time Series

Having grouped the ellipses into 7 clusters, we resulted in a time series of cluster labels. Since clustering a 30-day period stock market data categorizes this period by its volatility, the time series represents the trend of stock market volatility. We believe that the volatility of the market at a given period is dependent on the previous periods, so this time series is likely to repeat itself.

We used the Markov model to predict what clustering the future stock market data would belong to. Our training set is a time series of clusters, and our model consists of a dictionary, of which the keys are chunks of n consecutive cluster labels and the values are an array such that the first element is the most probable cluster that follows this n-cluster prefix and the latter is the probability of the most probable cluster happening. We varied n from 4 to 25 in seeing how the Markov Model may perform differently as sequence length increases. To construct this dictionary, we looked at the time series data in chunks of n+1 clusters using the shift window method. Then, we recorded the type of the n+1 cluster under the label of the n-cluster prefix. We chose to investigate the time series data in chunks of different sequence to see which sequence length would have an optimized balance between a high coverage of possible prefixes and a high probability of the most probable prediction.

We then tested our Markov model on a testing set, which consists of stock market data later in time than the training set. Similarly, we divided the time series of clusters into chunks of n+1 clusters using the shift window. Comparing the n+1 cluster predicted by our model and the actual n+1 cluster, we found the number of correct predictions and wrong predictions, as well as the number of times the n-cluster prefix was not found in the model. With this result, we were able to calculate the accuracy of our model.

3.8 Recurrent Neural Network (RNN)

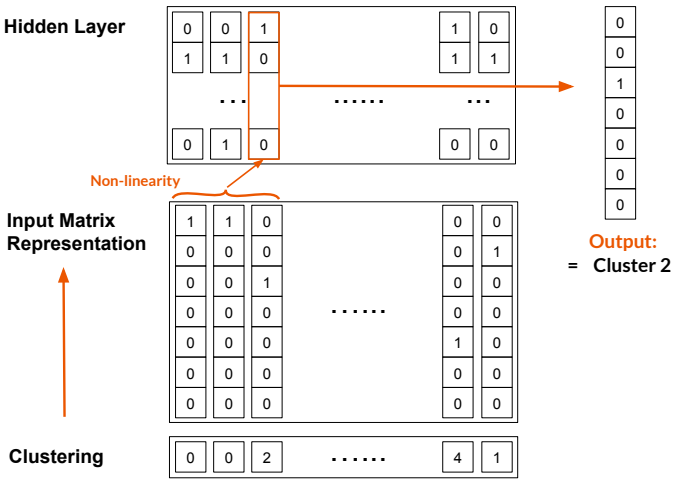


Figure 5: Illustration of the matrix representation of clustering, hidden layers and cluster predictionin Recurrent Neural Network (RNN).

Recurrent Neural Network (RNN) is a class of neural network that allows previous outputs to be used as inputs while having hidden states. It is suitable for modeling sequential data, that is data that

appear in a specific order such as time. Since we have a time series of clusters, RNN is suitable for cluster sequence modeling.

First we represent the time series of clusters in a matrix. Each cluster is represented as a vector of k elements, depending on the number of group they are clustered into using k-means.

Then, several columns of the input matrix is input into a non-linear function, along with previous columns in the hidden layer, to generate the next column in the hidden layer. The number of columns can be manually chosen, which we will refer to as sequence length used in generating the model. Columns in the hidden layer represent hidden characteristics of the corresponding input clusters, which provides insights for next cluster in the time series. A loss function is defined as distance between sum of distances between the hidden layer and the following cluster it predicts. The loss function is minimized to achieve the best accuracy.

For better accuracy of the model, we can stack up multiple hidden layers. To make a cluster prediction, we can find the column in the hidden layer and pass that column into a linear function. The linear function would generate a vector with a size of k, and all elements in the vector are 0 except one of them. This is a vector representation of a cluster. Translating the vector into the cluster, we result in a prediction for the next cluster in sequence.

3.9 Verify Model with New Variable

Previously, we have used volume change and price change as the input characteristics for k-means clustering. These two characteristics, when used as inputs, cause difficulties to arise. The range of volume change is significantly more diverse than the daily price change. This caused the principal eigenvalues to be around three magnitudes bigger than the secondary eigenvalue. This would mean that the theta value, which is an input characteristic for k-means clustering, is always around 90 degrees. This is meaningless for clustering, which is why a new model involving a new variable of volatility and price changed are built to see if clustering would work better. The new variable of volatility, $frac_{high}$, is defined as the following:

$$frac_{high} = \frac{(high - open)}{open}$$

This volatility measure has previously been used as input for published Hidden Markov Models that were built to predict S&P 500 trends and stock prices.

The same routine for the old variable has also been applied to this new measure. This is because we are using this new volatility variable in testing our model. In comparing results obtained from the old variable and this new volatility variable, we would be able to check whether our streamline of procedures are robust and that we are not getting good results by luck. This helps us confirm whether our procedures can be easily adapted and utilized on diverse variables in the future.

4 Results

4.1 Preliminary Result

It is important to note that in this subsection, the results are using the preliminary weights chosen (before optimization of parameters) and the original distance function.

4.1.1 K-Means Clustering

After determining the optimal K value (reference 3.4.3) and performing K-Means clustering, the data is sorted into 7 clusters. In order to verify the correctness of this process, every ellipse from the data is plotted on the same graph, where each color represents a different cluster (see Figure 5).

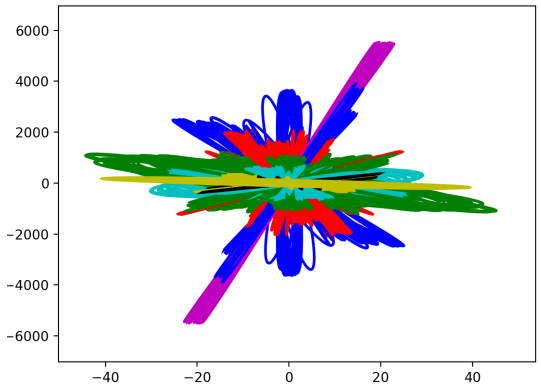


Figure 6

A pattern can be somewhat seen. For example, all the ellipses that are extremely long and in the same direction are colored purple. However, since the scale of the x-axis is considerably smaller than that of the y-axis (around 200 times smaller), this graph is quite misleading and hard to interpret. Thus, an attempt was made to re-scale the plot such that the x and y-axes have the same scale (see Figure 6).

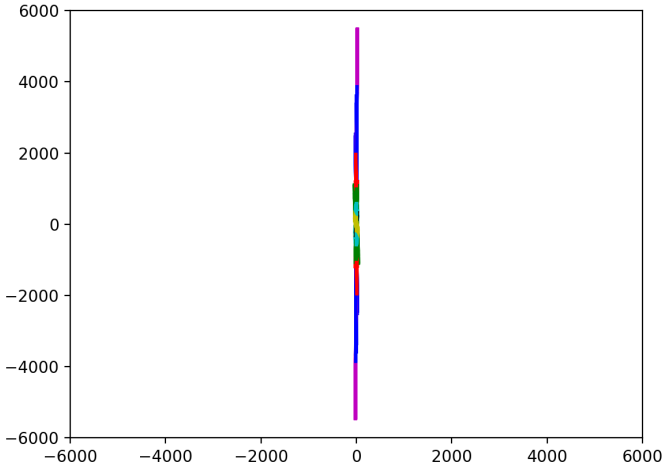


Figure 7

In the scaled graph, there are many things to note of. Similar to what was mentioned in the parameter choosing section, the mean of all the individual eclipses seemed very similar and the relative angles from the horizontal axis to the principal eigenvectors are all close to 90 degrees. The biggest difference among the clusters is the length of the principal eigenvector, or in other words, the length of the major axis for the ellipse. This reflects our choice of having the principal eigenvalue as the most dominate weight among the other parameters.

In addition, Figure 7 shows that from the center of the figure and moving outwards, seven colors can easily be identified in both directions. This may suggest that similar eclipses were successfully classified into their respective clusters based on the distance function.

The K-Means Clustering algorithm seems to be doing its job, but the two figures seem to suggest some problems that must be looked at. First, our data has drastically different variances, which makes choosing the parameters difficult and the distance function unbalance. The variability in the percentage of price change is relatively small in comparison to the variability in the percentage of volume change, which makes the principal eigenvalues to be almost hundreds of times of the the minor eigenvalues. The differences in magnitude of the two eigenvalues cause the theta value to all be close to 90 degrees, which make the theta value less meaningful and less useful in model training. Another strategy must be come up with in order to find the balance between the variability of the two eigenvalues in order to make the distance function more balanced and demonstrative.

4.1.2 Markov Model Result

One of the ways that the K-means model can be used is to build a Markov model. With the trained K-means model, all of the eclipses from the data are sorted into 7 clusters. The Markov model is built in order to test whether using four shifting windows' worth of data, the fifth day's general eclipse shape can be predicted. The dataset is split into training set and test set with a 9:1 ratio. When the model is tested on the test set, the accuracy achieved is 92.01% with 1 case in the test set not being able to use the trained dictionary to make a prediction. This accuracy is very high and may suggest that there is a trend in the dataset. It is worthy to note that four days of shift windows is not significant in tying the sequence to a specific type of event, so more testing needs to be done. With this strategy, the number of shift windows needs be increased in the future in order to identify the longest sequence of trends. Then, the confidence of correlating the trend to a specific event would be higher. On the other hand, it is also likely that many eclipses are being clustered into one cluster, and achieving a high accuracy is due to luck. More extensive testing must be done in order to solidify the conclusion.

4.2 Final Results

Note that in this subsection, the results are produced after applying the optimal parameters for the distance function and refining the distance function.

4.2.1 K-Means Clustering

For the price change versus volume change data set, its Silhouette Score is at 0.73. Figure 7 shows the final K-Mean Clustering result.

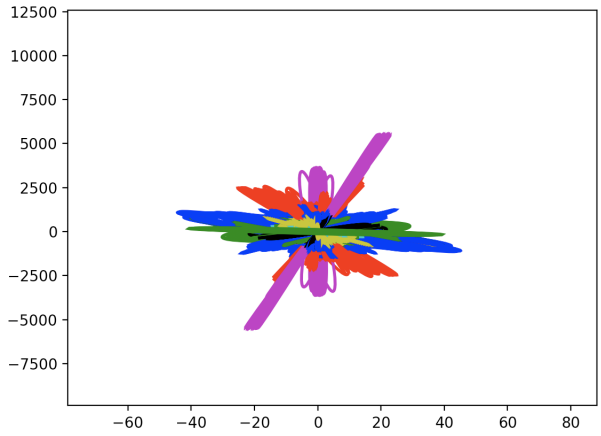


Figure 8

Although the Silhouette score has greatly increased, this figure does not reflect significant differences comparing to Figure 5 (the K-Mean Clustering result without optimal parameters and the refined distance function). This might be due to the fact that our optimal parameters for this data set were somewhat close to the preliminary parameters that we had used initially for Figure 5.

The same process of finding optimal parameters for the distance function had been done on the price change vs volatility data set. The preliminary parameters are [0.1, 0.1, 0.1, 0.7] with a Silhouette Score of 0.50, while the optimal parameters are [0.09, 0.1, 0.1, 0.71] with a Silhouette Score of 0.55. Clearly, both the magnitude and the improvement of Silhouette score for this new data set is lower than that of the previous. This difference in Silhouette Score among the previous and this data set can attribute to a variety of reasons. For instance, it is possible that price change and volatility had a lower correlation comparing to that with volume change. Another reason might be that the optimizing parameters methodology did not produce the global maximum for this data set, due to insufficient coverage during the manual estimation process. Despite the differences, it is important to note that a score of 0.55 still implies that the clustering has a reasonable structure, based on Figure

3 (interpretation of Silhouette score from Section 3.5). The following figure is the final K-Mean clustering result for this data set:

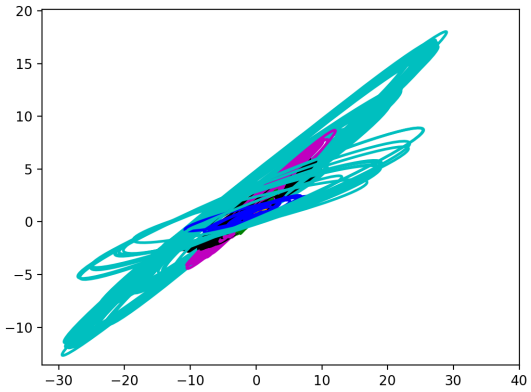
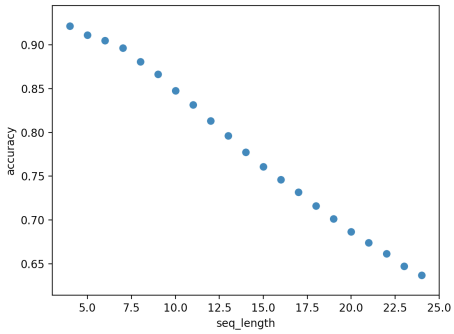


Figure 9

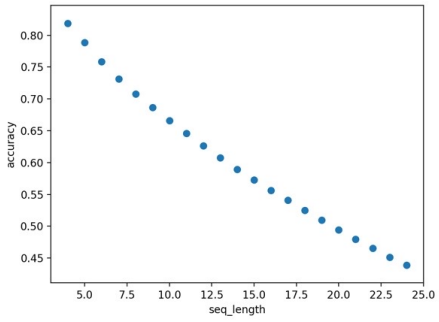
From the figure, we can see that the structure does seem somewhat reasonable - the ellipses with longer lengths and shorter lengths in the major axis were clearly clustered separately. However, we can see that for the cluster represented with the teal color, it seemed to have clustered two groups of ellipses with noticeably different angles together. This corresponds with the Silhouette score that was calculated - the clustering had a reasonable but not strong structure.

4.2.2 Markov Chain

We varied the sequence length from 4 to 25 as input and built a Markov model for each individual sequence length using the K-means clustering result. 90% of the dataset served as the training set while 10% of the dataset served as the test set. This strategy was applied to both of our models. The accuracy of the predictions generated by the Markov models is plotted below:



(a) Volume Change vs. Price Change



(b) Volatility Change vs. Price Change

Figure 10

As we can see from the accuracy vs. sequence length plots for the two models, the accuracy is decreasing linearly as the sequence length increases. This is because the number of possibilities of input is increasing exponentially, thus the chance of having the "not found" case increases drastically, bringing the accuracy down.

We are seeing a similar trend for the results for the two models, indicating that this result did not happen by chance and that we would see similar results if we apply this algorithm to other variables

in the future. In addition, notice that the accuracy for the data sets with input sequence length of 4 are 92% and 83% respectively. This 9% difference in accuracy implies that the accuracy of the Markov Model is greatly sensitive to data sets.

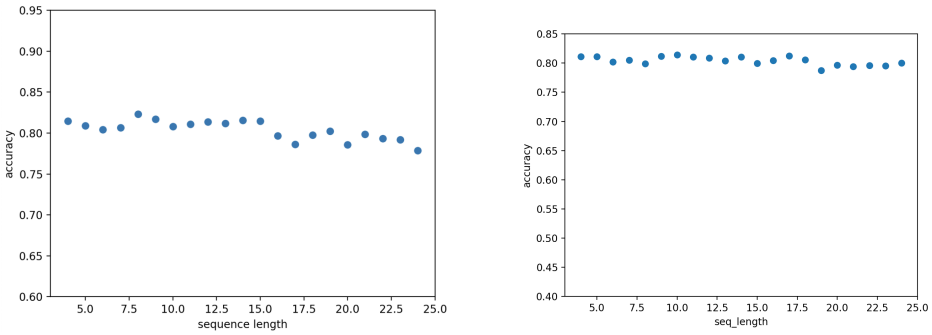
4.2.3 RNN

We used the same range of input sequence length as Markov chain to construct the RNN model, ranging from 4 to 24 labels. We then used each model to predict future clusters based on a list of historical clusters that corresponds to the sequence length used in model. For example, when testing the accuracy of the RNN model with a sequence length of 4, we first constructed the model by looking at pairs of 4-cluster chunk and the following 5^{th} cluster. Then we input clusters of four into the RNN model and predicted the corresponding 5^{th} cluster with the model. We compared the predicted clusters with the actual 5^{th} cluster in the data. Accuracy of the model was calculated by $\frac{\text{Number of correct predictions}}{\text{Total number of tests}}$. Note that since RNN can predict the following cluster even if the 4-cluster chunk is not present in the training set, we would never reach a case where predictions cannot be made like the Markov models.

Cluster predictions based on the original variables are shown below on the left side. Different from the Markov models, the accuracy is relatively constant at 80% as sequence length increases. This shows that we would be able to predict for further in the future with a decent accuracy.

We also constructed RNN model using clustering based on the new variables (price change vs volatility), shown below on the right side. Similar to the predictions generated based on the original variables, the accuracy has little correlation with the sequence length used. Moreover, the accuracy achieved is similar to that of the original variables, averaging about 80%.

The consistency of accuracy in the model of the new variables shows the absence of dependence on sequence length is a property not peculiar to that specific data set. We can potentially test out accuracies of predicting clusters based on other variables and select the best-yielding one.



(a) Volume Change vs. Price Change

(b) Volatility Change vs. Price Change

Figure 11

4.2.4 Comparing Markov Chain and RNN

As shown by the plots in the previous sections, Markov model prediction accuracies decrease linearly as the increase of sequence length, while the accuracies of the RNN model stays relatively constant when sequence length increases from 4 to 24. The high accuracy observed in the RNN model using long sequences allows us to make better predictions about clusters further in the future. It is worthy to note that we can use the predicted cluster along with the original input clusters to generate more predictions further in the future.

In addition, the accuracy of RNN model also shows significantly less dependency from the data sets it is applied on, comparing to the Markov Chain. For the Markov chain, for a sequence length of 4, the original and the new variables had an accuracy of 92% and 83% respectively, while the RNN had an accuracy of around 82% for both variables. This shows that RNN is a more stable model and is more reliant when used on new data sets.

Finally, we noticed that the accuracy predicted by RNN on the original variables is lower than that of the Markov model at short sequence lengths. We suspect that the accuracy of the RNN model can be further improved by running more epochs, retraining or other modifications.

5 Literature Review

We reviewed literature related to analyzing stock market data with similar approaches. We didn't find any existing articles that categorizes stock market data using PCA, but we found several articles that share other aspects with our approach. One of the articles uses multiple regression analysis of stock trend prediction. They conducted dimensional reductions by computing percentage price change, stock volume and dispersion/volatility. These variables are presented in the regression models with optimized regression coefficients from R function Im(). Using the regression model and the KSE 100 index as an input dataset, they obtained a prediction accuracy of 95%. Similarly, when the Lucky Cement Stock dataset was used, the model resulted in an accuracy of 89%. This project uses similar data pre-processing method for dimensional reductions, which serves as indicators for the stock market trend. However, they used a regression model of the market indicators. We could possible use similar indicators in our model, and use similar regression models to train the parameters in our distance function.

The second article uses a hybrid feature extraction method, coupled with Support Vector Machine(SVM), in order to predict stock trends. Algorithms are applied before SVM clustering in order to extract features from the dataset. The features are scored based on F-score, and the best features are used as training data for SVM. When applying to different stock indexes and exchange rate, SVM achieved accuracy ranging from 83.2% to 87.2% in predicting NASDAQ stock trend with an average accuracy of 85.4%. The project the article described had a similar goal to our project and used classifying technique. However, instead of using unsupervised technique of clustering like k-means, SVM, a supervised technique in model training, is used. Then, the feature selection step is vital and labels the data before classifying. Their model and our model achieved similar accuracy in predicting stock trend. We could also apply similar strategies as them such as using feature extractions models and supervised learning classification models.

The last article uses a LSTM approach developed by RNN to forecast stock prices based on attention mechanism. This is done by using a wavelet transform to extract and train features from historical stock data. Then, it is used to create a prediction model for the stock price. The experiment was also ran on the SP500 data set, in addition to the DIJA data set. Finally, the paper compares its results with three other approaches: LSTM, LSTM with wavelet transform, and the gated recurrent unit(GRU) neural network model. It concluded that the model had a coefficient of determination higher than 0.94 and a mean square error lower than 0.05. We could also apply LSTM in the future to better predict stock trends.

6 Future Work

6.1 More Data Collection

The current model is trained on daily data, which consists of only around 5000 data points. Moving forward, we will continue to acquire intra-day data (minutes data, hours data) and make appropriate purchases if necessary. Getting intra-day data would increase our data size to at least half a million. Training on more data would result in a better model and provide more opportunities for trend seeking. For instance, the key to the dictionary in the Markov Model can be increased to 30 instead of 5. This would significantly increase the possible conclusions that can be made based on the results.

6.2 Better Method for Pre-processing

K-Means clustering usually requires variables to have similar standard deviation. However, in this case, the percent change in price is significantly less variable than the percent change in volume. Volume can easily change for around 10% while changing 10% in price is highly unlikely without significant events. We will seek algorithms that will alter the raw date such that each variable would have similar variability. In addition, instead of percent change in price versus volume, we can also attempt to apply our model on other possibly related variables that might tell us useful information.

6.3 Comparing Results from the Two Built Models

The two models, whether it is $frac_{high}$ vs. change in price or volume change vs. change in price, were built based on the same dataset. We performed clustering on both data sets and received results. Theoretically, since $frac_{high}$ and volume change are all measurements of volatility and based on the same dataset, the result for clustering would be similar. We could potentially compare the clustering results and see if is seen that way. We could also work on combining the clustering results in building a stronger model. This could also be a great test to see whether k-means clustering is performing correctly.

6.4 Improvements on RNN

Our current model has 2 hidden layers and employs GRU networks. We can increase the number of hidden layers to increase the accuracy of our model. Since LSTM is found to be more accurate than GRU when dealing with the issue of short term memories, we can replace GRU with LSTM in the model to increase our accuracy. We can also improve our model by warm starting the model, which is to retrain the model with smaller step sizes. Cost functions and parameters used in the RNN model can also be further investigated and optimized.

7 Conclusion

This paper establishes a new approach in using geometry optimization in predicting S&P 500 stock trend. The two models that we built used "volume change vs. price change" and "volatility vs. price change" as input characteristics, respectively. PCA is used to dimensional reduce input data and extract geometrical parameters that served as the input to the K-Means clustering algorithm. A K-means clustering algorithm is built and applied with a newly defined distance function. Scipy library's optimization is then utilized to select the optimal weights for the distance function with the Silhouette score as the objective function. The clustering result was then put into use as the input to train both Markov Model and a Recurrent Neural Network. By varying the sequence length as input for the a Markov Model and RNN, the robustness of the Markov Model and the RNN is determined. Both the Markov Model and RNN achieved great accuracy when the sequence length was short. However, as the length of input sequence increases, the accuracy for Markov Model declined linearly while the accuracy for the RNN was consistently around 80%. This shows that the RNN is more robust than the Markov Model in stock prediction and should be adopted. Furthermore, this streamline process of extracting parameters, clustering and predicting were built to be easily adapted to various variables, as we have shown by applying the algorithm to a completely new variable. Then, our developed a streamline of algorithm can be easily adapted in the future for inputting different characteristics and can achieve a relatively high accuracy in stock trend prediction.

References

[1] Asghar, Muhammad Zubair, et al. "Development of Stock Market Trend Prediction System Using Multiple Regression." Computational and Mathematical Organization Theory, vol. 25, no. 3, 2019, pp. 271–301.

[2] Garbade, Michael J. "Understanding K-Means Clustering in Machine Learning." Medium, Towards Data Science, 12 Sept. 2018, towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1.

[3] Irshad, Hira. "Relationship Among Political Instability, Stock Market Returns and Stock Market Volatility." Studies in Business and Economics, vol. 12, no. 2, 2017, pp. 70–99., doi:10.1515/sbe-2017-0023.

[4] Lee, Ming-Chi. "Using Support Vector Machine with a Hybrid Feature Selection Method to the Stock Trend Prediction." Expert Systems with Applications, vol. 36, no. 8, 2009, pp. 10896–10904., doi:10.1016/j.eswa.2009.02.038.

[5] McCaffrey03/27/2018, James. "Data Clustering with K-Means Using Python." Visual Studio Magazine, visualstudiomagazine.com/articles/2018/03/27/clustering-with-k-means-using-python.aspx.

[6] "Optimization (Scipy.optimize)¶." Optimization (Scipy.optimize) - SciPy v1.4.1 Reference Guide, docs.scipy.org/doc/scipy/reference/tutorial/optimize.html.

[7] Partitioning Around Medoids (Pam), web.archive.org/web/20111002220803/www.unesco.org:80/webworld/idams/advguide/Chapt711.htm.

[8] "Sklearn.decomposition.PCA." Scikit, scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html.

[9] "Sklearn.metrics.silhouettescore¶." Scikit, scikit-learn.org/stable/modules/generated

/sklearn.metrics.silhouettescore.html.
[10] “Text Generation with an RNN: TensorFlow Core.” TensorFlow, www.tensorflow.org/tutorials/text
/textgeneration. [11] Qiu, Jiayu, et al. “Forecasting Stock Prices with Long-Short Term Memory Neural Network
Based on Attention Mechanism.” PLOS ONE, Public Library of Science, 3 Jan. 2020. [12] Shervine Amidi,
"Recurrent Neural Network cheatsheet", https://stanford.edu/ shervine/teaching/cs-230/cheatsheet-recurrent-
neural-networks