CSE 482 Project Proposal

Stock Market Analysis

Hanghang Zhang A47035780

January 29, 2017

Abstract

This project uses historical stock price to predict a probability distribution for upcoming price variance.
The primary purpose of this project is to analysis the behavior of stock market with some machine learning algorithms, and try to compare performances between algorithms.

Contents

Chapter 1

Sources

1.1 Daily Data

Daily data used by this project are public accessible at:
http://www.google.com/finance/historical
With query:
{
q: **Stock Code**,
startdate: **Month Code**+**Date**+%2C+**Year**,
enddate: **Month Code**+**Date**+%2C+**Year**,
output: csv
}

1.2 Hourly Data

Hourly data used by this project are public accessible at:
https://www.google.com/finance/getprices
With query: {
i: 3600,
p: 600d,
f: [d,o,h,l,c,v],
df: cpct,
q: **Stock Code**
}

Chapter 2

Project Plan

2.1 Data Collection

Data collection part will be simply done with python scripts. All data used in this project are quoted from Google Finance(See: https://www.google.com/finance).

2.2 Data Process

2.2.1 Data Normalization

For most of machine learning models, the input data need to be nicely distributed in a certain set or a small range. But the range of stock price is a positive real number, and the average price varies from company to company. So the raw data from the Internet will not be applied to machine learning algorithms directly. The very initial plan to normalize the raw data is, transform the the daily price to daily variance. After doing so, the range of input data theoretically turns to $(-\infty, \infty)$, but for 99% of situations, the average daily variance of certain stock will not excess 15%, base on this assumption, we can fit the daily variance of a stock into a category like $\{(\infty, -15\%], (-15\%, -2\%\}, (-2\%, 2\%), [2\%, 15\%), [15\%, \infty)$. And the category should be customizable from the front end web app.

2.2.2 Similarity

Most supervised machine learning algorithms need a huge amount of training data(historical price in this case) to achieve a decent performance. The idea to improve algorithm performance for some new companies with very few training data, historical data from other similar companies will be used.

So the task here is to determine how similar are two given companies is. One approach in the project plan is to apply cosine similarity. First, looking for some other companies with sufficient amount of data records, and then determine the similarity between it and the new company. A linear combination of those elder companies' result will be used as the new company's distribution, and the weight for each term in the combination depends on the similarity between that company and the new company.
The cosine similarity of two data sets is defined to be:

$$S_{cos}(DataSet_1, DataSet_2) = \frac{< DataSet_1, DataSet_2 >}{\sqrt{< DataSet_1, DataSet_1 >< DataSet_2, DataSet_2 >}}$$

The above formula requires $DataSet_1$ and $DataSet_2$ to have the same dimension, but obviously that the dimension(number of records in this case) of new company is much less than elder companies'
So we need to redefine a new cosine similarity so that it could work for our specific circumstance.
The convolution trick can be used here to deal with data sets of different dimensions. Suppose the smaller data set $DataSet_1$ has dimension of n while the larger set $DataSet_2$ has dimension of $n + k$ for some $k \geq 0$, then we use $DataSet_1$ as convolutional core and define S^*_{cos} as

$$S^*_{cos} := \frac{\sum_{j=1}^{k} S_{cos}(DataSet_1, DataSet_2[j : n + j])}{k}$$

Just like cosine similarity, the idea of co-variance in statistic can be used as a measure of similarity as well, co-variance represent the correlation between data sets, when two sequences increasing or decreasing synchronously, the cov between them will increase. So it seems like cov could be a good measure for similarity, and in the report of this project, an analysis between two different similarities will be given.
Definition of cov similarity that used in this project:

$$S_{cov}(D_1, D_2) = \frac{\sum_{i=1}^{k} cov(D_1, D_2[i : i + k])}{k}$$

2.2.3 Machine Learning Algorithms

Due to the time limit, this project will not use complicated machine learning models, and the prediction of probability distribution will mainly come from naive Bayes classifiers and n-gram algorithm.

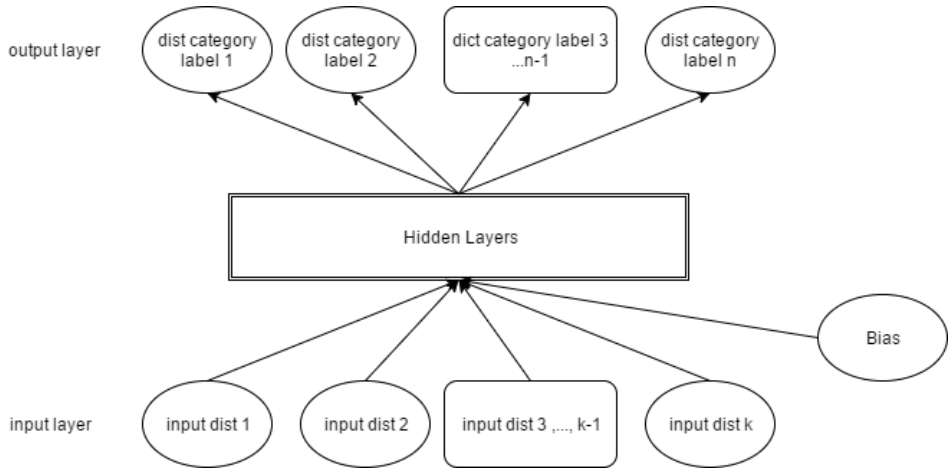
Naive Bayes model for this project assumes that the price variance of the t th day depends on previous n days, and previous n days are independent to each other.
And the n-gram algorithm will calculate

$$\sum_{X \in Category} P(Day_k \in X | Day_{k-1} \in Y_{k-1}, ..., Day_{k-n} \in Y_{k-n}) x^{ord(X)}$$

Where Y_js are some labels, and $ord : Category \to \mathbb{Z}$ can be any injective map.

2.2.4 Refinement of Results

Multiple machine learning models with different parameters will be trained by the same set of training data, thus when input comes, different distributions will be generated. The plan is to select the best one from them or generate a better distribution base on them.
A back propagation neural network with k input nodes and n output nodes can be used to refine k distributions of n categories as shown in the graph below.



Another traditional way to get a result base on the k distributions is using linear combination. Suppose k result distributions are $D_1, ..., D_k$, then the result distribution will be

$$D^* := \sum_{i=1}^{k} D_i \lambda_i$$

where λ_i are weights for D_i. The vector $w = (\lambda_1, ..., \lambda_k)^T$ which fits the previous k days best can be approached by solving the system

$$\begin{bmatrix} XD_{11} & XD_{12} & \dots & XD_{1k} \\ XD_{21} & XD_{22} & \dots & XD_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ XD_{k1} & XD_{k2} & \dots & XD_{kk} \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_k \end{bmatrix} = \begin{bmatrix} YD_1 \\ YD_2 \\ \vdots \\ YD_k \end{bmatrix}$$

YD_i is the solution label for the $(n-k)$-th day, and XD_{ij} is the label with highest probability in the distribution B_j generated at $(n-i)$-th day.
In the above system, all distributions generated in previous k days are used to build the matrix XD, if the matrix XD happened to be invertible, we can use the method of least squares or add a perturbation to the system.

2.3 Front End

Front end web application will be available and accessible on a MSU web server, it will accept users requests in form of post query, and return a json string of the analysis result. If users access the url with a browser, the web app page will translate the result JSON data and present it in graphical chats.