Binghamton University, State University of New York
Department of Electrical and Computer Engineering

EECE 629 – MACHINE PATTERN RECOGNITION (Fall 2017)

Project

Due Date: 12/08/2017 (Friday)

Presentation Date: 11/28/2017 (Tuesday)

Note: This is a group assignment, and each group may consist of up to three students.

The overall goal of this project is to use machine pattern recognition methods to predict stock market changes, up or down. These notes cover the overall organization of this project, and suggestions for the inputs and outputs of each section. Note that many of the details of each part can be customized by each individual performer.

1. Data collection, preprocessing, indicators
2. Calculation of targets
3. Features from Indicators
4. HMM part
5. NN part
6. Post-processing and Testing

1. We will work with predicting price changes of a specific stock randomly chosen by each group. (You may not use AAPL example and please make sure that every group is using a different stock.) This can be downloaded from Google Finance or NASDAQ.

   For example, if we are choosing Apple Inc. (NASDAQ: AAPL)
   You can go to the NASDAQ website to download up to 10 years of daily historical stock prices & volumes.
   http://www.nasdaq.com/symbol/aapl/historical

   At the bottom of the table, you can find "Download this file in Excel Format."

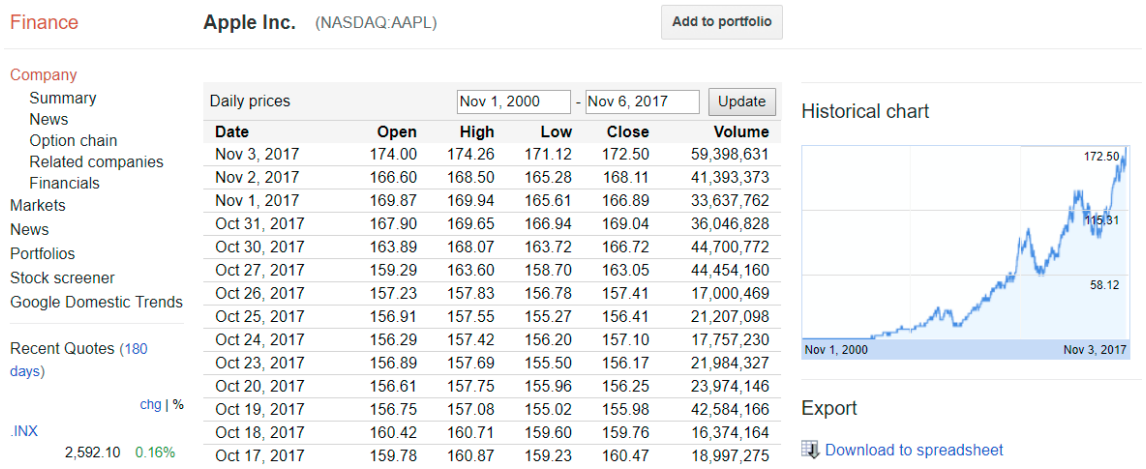
11/08/2007	26.6671	26.7	23.9671	25.0666	471,937,762
11/07/2007	27.23	27.5257	26.59	26.6143	247,734,134
11/06/2007	26.7214	27.4286	26.4671	27.3986	238,372,752
11/05/2007	26.47	26.9943	26.32	26.5971	200,807,100

*This data reflects the latest intra-day delayed pricing.

Download this file in Excel Format

Or you may also use the data provided by Google Finance through the following link: https://finance.google.com/finance/historical?q=NASDAQ:AAPL

In the right column underneath the Historical chart, there is "Export: Download to spreadsheet."



2. Let us use about 10 years of data (say Nov. 1, 2007 to Nov. 1, 2017. You must be careful in later stages to carefully separate training data and test data. I suggest using sliding blocks of 2-3 years of training data, and about 1 month of test data, immediately following the training data (in time). The training and test blocks should then be advanced, so that eventually all the data is used. (Note, previously, we first tried using 17 years of training data, followed by 3 years of test data. This did not work well at all.)

   Please also be careful about the significant changes over the period of 2007-2011, where the economy suffered from the Great Recession. As shown in the figure below, the S&P 500 index has dropped from 1561 (Oct. 12, 2007) to 683 (Mar. 6, 2009), and then bounced back to 2592 (as of Nov. 6, 2017). This may result in a more challenging task if you would simply plot or predict the stock price change using yearly data. Please consider the analysis based on segmented, sliding windows.

3. Please use 10 technical indicators as initial "measurements." It will up to each group to decide which particular indicators to use and to write the MATLAB code to convert raw stock data (highs, lows, opens, close, volume) to these indicators (or use codes already available to do this type of thing—it is out there). You should normalize these indicators so that each one is on a range of approximately -1 to 1. To be useful, the indicator values should also be in the same general range from year to year—this may require you to do some additional long-term averaging.

4. Computing features from indicators. Use whatever techniques you think are promising to convert "blocks" of indicators to features values for each day. The block length can be any value you think is reasonable, but typically on the order of 50 days. The block must consist only of indicator values from past days up to the current day. Typical methods for converting blocks of indicators to features include principal components analysis, discriminant analysis, and/or a cosine basis vector expansion.

5. Let the variables that are to be predicted be the lowest value of the price that occurs over the following *Days_ahead* days, and the highest value that occurs over these *Days_ahead* days. *Days_ahead* should be a variable, but a typical value for this project is 10 days. The high and low values should be in terms of fractional drop and fractional rise. Note these high and low values are not necessarily the values on the 10th day. For example, the low might occur on day 5 and the high on day 8, following the "current" day. Use the convention that rises are positive values and drops are negative values. Typical values for drops are 0 to -0.15, and typical values for rises are 0 to 0.15, if *Days_ahead* = 10.

Note that you can change details here. However, assume that basic strategy will be to either buy on a given day, and sell for a higher price within the next *Days_ahead* days, or to short sell on a given days, and buy for a lower price within the next *Days_ahead* days. Thus the basic idea is to profit from short-term rises in the market (buy low, sell high) or from short-term drops in the market (sell high, buy low). Both of these techniques will be profitable if you can accurately predict short-term fluctuations in the market.

6. These states are NOT the so-called Hidden States in an HMM. Rather these are observable states (at least in training data) that you eventually want to predict automatically. You will be able to accurately label these states in the training data. Hopefully there will be hidden states embedded in the feature values over some number of days prior to the day of the "state" you will try to predict.

   One suggested approach for defining these states is as follows. First quantize the high_rises into 2 categories. Category 1 defines as no large rise over next *Days_ahead* days. Category 2 defines as large high rise over next *Days_ahead* days. A threshold must be used to separate these categories. (Essentially this is a 1 bit quantizer for high rises.). Similarly define two categories for low_drops. Category 1 is for no large low drops over next *Days_ahead* days, while category 2 is for large drops over next ***Days_ahead*** days. Again a threshold is needed to separate the two categories. Then define the states as:

High-Low Condition	High Condition	Low Condition	Description
1	1	1	Placid market – Do nothing
2	1	2	Good day to short sell – market about to drop a lot
3	2	1	Good day to buy – market about to rise a lot
4	2	2	Risky market – both big rises and drops coming

   You could train 4 HMMs to recognize these 4 high_low conditions. Alternatively you could train 2 HMMs, based on high condition only.

7. There should be from 1 to 8 neural networks. Each neural network should have 1 output, trained to predict either a short-term rise or drop in the market. For the simplest case (1 network), this network should be trained to predict the short-term rise only, without regard to the states mentioned for step 6. For the most complex case (8 networks), there would 1 network for highs and 1 for lows, for each of the 4 states mentioned in step 5.

   Important notes for steps 6 and 7 (NN and HMM steps)

   For both NNs and HMMs, you could decide that a certain percentage of data (say first 7 years) is training data and a certain percentage is test data (say last 3 years). However, such an approach has not been found to work well. Rather it seems to work better to use sliding blocks of data for both training and testing. For example, use 1 year for training, and the immediately following month for testing. After that "block" is completed, advance by 1

month and repeat. Continue doing this until all data is used up. With this approach, except for the first year, eventually all data is used as test data. This is the motivation for the `trn_tst` flag mentioned for steps 6 and 7. Note that final buy-sell decisions must be made based only on predictions made when data is considered test data.

Another issue to consider is that the "best" features are likely to be different for the HMM step and NN step. In particular, since the HMM has a temporal aspect embedded in the model, whereas the NN does not, likely the "best" block size and number of features, will be much lower for HMM features vs NN features. As a starting point, I would suggest block length of 50, and 50 total features for NN features, and a block length of 10 and 30 total features for HMM case. If you want to pursue this approach, you should create two feature output files back in step 4.

8. Buy/sell strategy will be based on having access to previously made predictions of the HMM and NN. Since all data is already available, we can store these predictions in files, as described in more detail elsewhere. The testing will consist of using these predictions to make buy, sell or hold decision, and then evaluating how well this actually works. BE SURE THAT FEATURES AND HMMS AND NNS ARE BASED ONLY ON DATA THAT PRECEDES CURRENT DAY.

Detailed Specifications:

Step 1: Data Preparation and Indicator Calculations

Step 1a. Reading a XLSX or CSV file of financial data, downloaded from Google Finance or NASDAQ, and converting to a matrix in MATLAB.

Sample first few lines of financial data of Apple Inc. (NASDAQ: AAPL) from Google Finance:

1	Date	Open	High	Low	Close	Volume
2	3-Nov-17	174	174.26	171.12	172.5	59398631
3	2-Nov-17	166.6	168.5	165.28	168.11	41393373
4	1-Nov-17	169.87	169.94	165.61	166.89	33637762
5	31-Oct-17	167.9	169.65	166.94	169.04	36046828
6	30-Oct-17	163.89	168.07	163.72	166.72	44700772
7	27-Oct-17	159.29	163.6	158.7	163.05	44454160
8	26-Oct-17	157.23	157.83	156.78	157.41	17000469
9	25-Oct-17	156.91	157.55	155.27	156.41	21207098
10	24-Oct-17	156.29	157.42	156.2	157.1	17757230
11	23-Oct-17	156.89	157.69	155.5	156.17	21984327

There are functions to convert these excel files to data matrices (xlsread.m). Also, I think it is more intuitive to arrange data in matrices from older to newer, rather than new to old as the original files are. Please "flip" data to accomplish this with appropriate MATLAB commands.

After data is originally read in and converted to a data matrix, indicators can be computed.

Note that you should think of each row of the data matrix as corresponding to a day. The columns will then give information about that day. For example, the first three columns of each data record (row) will be the month, day, year (date).

The **`rd_feat`** and **`wr_feat`** have many different options. However, we will use only the basic mode, whereby all data are written as ASCII. The only arguments to the routines will be the name of the data matrix and the filename to read or write.

First few lines of sample TYPEA1 data file (MATLAB code wr_feat.m to be given to you to write a matrix in this format)

Step 1b. Computing 10 technical indicators from read in financial data.

Step 1c. Scale the technical indicators—either linearly or nonlinearly, for a range of -1 to +1. That is to compute mean and standard deviation. Then for each indicator for each day, subtract the mean and divide by the 5*standard deviation. This will linearly scale each indicator for an approximate ±1 range. You may also need to do some long-term based normalization.

Step 1d. Write out scaled financial indicators in a file, called 'step1.dat." Include the date, the price and volume information, and the 10 indicators. See other document on file format issues for more details.

First few lines of a step1.dat file:

```
TYPEA1
   12
    1
    1
   18
    1
    1  2596   18                step1.dat
        1        3      2000
  1469.2500  1478.0000  1438.3600  1455.2200  9318.0000
    3.6148       NaN       NaN   -22.7073    77.2927
  9318.0000  1455.2200  9318.0000       NaN -1391.5883

        1        4      2000
  1455.2200  1455.2200  1397.4300  1399.4200 10090.0000
    3.5160       NaN       NaN   -78.3293    21.6707
```

8931.1018 1455.2200 -772.0000 NaN -10786.6895

1 5 2000
1399.4200 1413.2700 1377.6800 1402.1100 10855.0000
3.4672 NaN NaN -75.6479 24.3521
8951.9676 1476.9600 10083.0000 NaN -6739.3209

Step 2. Calculation of Targets for HMM and neural network steps

For this step, you should first compute max of highs and min of lows over the *Days_ahead* that you choose. These should be converted to percentage changes relative to opening price of each day, but on a scale of 0 to 1 for highs and 0 to -1 for lows. Depending on criteria you choose, and some reasonable thresholds, also compute the state of the market for that day:

a) 1 — Good day to buy
b) 2 — Good day to sell short
c) 3 — Best to do nothing

Input for step 2 is the file named step1.dat. Output is the file named step2.dat

Step 3. Compute features from blocks of indicators.

More details on this coming later. However, the idea is to transform blocks of 10 indicators to features more useful for predicting purposes. There may be more or fewer features than indicators.

Input for step 3 is file named step2.dat. Output is file named step3.dat

Step 4. Hidden Markov Modeling

Use the targets from step 2, and features from step 3, to model and predict states of the market, for each day. That is, the actual state will be a determined from step 2. HMM models should be trained based on these known labeled states. The goal of the HMM is to predict states on data it was not trained on.

Notes:
How many Gaussian mixture components to use?
Diagonal or full covariance matrix?
How many hidden states to use in each model?
How many past days to use for each model?
Be sure to only use the training data dates for the HMM training

Inputs: step2.dat, step3.dat; Output: step4.dat

Step 5. Neural network training

Inputs: step2.dat, step3.dat, step4.dat Outputs: step5.dat

Read in the data created in previous steps, as from above.

These notes are based on a paper that suggests using an HMM to first decide general "state" of market, and then use NN to predict details in that state. However, since we will cover NNs before HMMs, you can use just one NN, based on all days (assume state 1 always). In fact, after HMM step is completed, you may only need 1 NN, trained for state 1 data. If the HMM predicts market is in state 2 or 3, your strategy may be to do nothing, unless you try to make a profit by predicting drops in price.

Step 6. Post processing and testing

Step 6a. Decide conceptually on a buy/sell/hold strategy. Assume that you can make money from both stock rises and falls, as long as you predict in the correct direction, and end a transaction "correctly." For example, if you think the stock is going to rise 8%, and it only rises 7%, should you sell or hold on? Also, perhaps you could try to make a profit from rises only.

Step 6b. Read in the files mentioned in previous steps. These files will contain dates, pricing data, feature values, percent rises and falls, predicted percent rises and falls, actual states, and predicted states, for each day.

Make your buy/sell/hold decisions based on predicted values only, being sure that you are not "cheating" by peeking into the future. However, just of checking the logic of your strategy, you could use actual future values rather than the predicted ones to make decisions. You can make your strategy to use NN predictions only, HMM predictions only, or both NN and HMM predictions.

What you need to do:

1. **Project presentation (11/28/2017) — 10 points**
   - 30 minutes for each group.
   - Show the chosen stock to be analyzed.
   - Clearly present the overall methodology and specific approaches.
   - Describe how you segment the training and testing data set and how you make sure all the data have been used in your model.
   - Present the developed NN and HMM models.
   - Demonstrate the effectiveness of your approaches through figure plots and tables, or other ways if necessary.

2. **Final report and code submission (due: 12/08) — 20 points**
   - Problem statement (i.e., scope of project, functional specification).
   - Implementation of all those required models.
   - Testing and evaluation results.