Advanced Big Data - Assignment #2

Maanit Mehra - mm4694@columbia.edu

March 11, 2016

The entire code for this Assignment has been uploaded to: https://github.com/maanitmehra/big-data

Problem 1

In this assignment you will be asked to predict the share price of 10 US Equity stocks at exactly 10:00 a.m. on Friday March 11.

Your predictions / code are due at exactly 8:59 a.m. March 11 (THAT MORNING)

The ticker symbols for these 10 stocks are:

BAC, C, IBM, AAPL, GE, T, MCD, NKE, TWTR, TSLA

This is an open-ended assignment so you can use as many or as few data-sources / methods as you would like in your analysis. The point of this is to do historical regression analysis to build a model an predict the stocks exactly 1 hr in advance (after 30 minutes of activity since markets open at 9:30 am). Although this may sound arbitrary the point is to implement your own creative approach to analyzing different data sources and generating tangible results. Everyone has the same tools and (random luck) for this assignment so consider it a free for all to create the best approach.

`Solution`

We explore three approaches to the problem:

1. Polynomial Regression using only historical Data
2. Supervised Learning and the Least squares algorithm
3. Exploring Twitter based sentiment Analysis on Stock Behaviour.

Part 1

The first part of the approach involves a simplified extrapolation problem. We broke this up in two stages. We first began with a basic linear model adapted over the course of 10 am collected available using the google finance API for a period of 15 days.

```
GOOGLE_QUOTE_URL = "http://www.google.com/finance/getprices?f=d,o,h,l,c,v&df=
    cpct&i="+str(INTERVAL*60)+"&q="

stock_url = GOOGLE_QUOTE_URL + symbol
resp = requests.get(stock_url)
resp_filter = resp.text.split(" ")
```

The shortcomings of the model revolved on the lack of sufficient historical data for calculating generic trends.

To overcome some of these limitations we explored the usage of an adaptive polynomial thresholding. We approach this by doing basic error calculations on a number of data points for a number of stocks.

The thought process behind this entails that each degree of the polynomial best describes the characteristic of the curve.

We explore the approach to the same in our analysis. More volatile stocks (Tesla, for example) are better represented by the higher order polynomial expressions.

```
def extrapol(x, y, x_new, degree):
  f = np.poly1d(np.polyfit(x, y, degree))
  y_new = f(x_new)
  return y_new
```

This approach allows for a optimized version of polynomial regression. And lead to some interesting results.

Part 2

During the course of this project we explored the usage of a Supervised Learning algorithm. The input parameters we consider in this scenario include Volume of trade on the previous day, the day's Highs & Lows, opening & closing prices as features.

This data can be obtained with the help of the pandas data io library.

These dataframes return Open, High, Low, Close, Volume & Adjusted Closing.

```
  df = pio.get_data_yahoo(company,start = dt.datetime(2015, 1,21), end = dt.
    datetime(2016,3,11))

  npMat = df.as_matrix()

  x = npMat[:-1,1:]
```

```
  y = npMat[1:,0]

  # Using the classical Least Squares Regression Approach
    # for quick linear regression.
  xtx = Dot(x.T,x)
  xy  = Dot(x.T,y)
  try:
    w = Dot(np.linalg.inv(xtx), xy)
  except:
    w = Dot(np.linalg.pinv(xtx), xy)

    print w
```

We observe weights such as `[7.82346495e-02 -3.03513473e-02 4.75929400e-01 -1.01093090e-07` with the negative weights indicating an inverse correlation to the highs & volumes of the previous day. Potentially this implies, higher the volume of trade, less likely is the price likely to increase.

Further using cross-validation to train and test the data, we use the approach here to indicate suitable prediction values.

```
## cross validation test Data
  xtest = x[-1,:]
  ytestAct = y[-1]
  ytestPred= Dot(xtest, w)
## Using input parameters taken from the
## 10th of March as input parameters for
## the 11th of March.

  x_pred = npMat[-1,1:]
  Fin_pred = Dot(x_pred,w)
  print "%s\tPred:%.2f\t"%(company, Fin_pred)
```

`Part 3`

Lastly we attempted, unfruitful in the end, the usage of Tweet & news headline based sentiment analysis.

```
from alchemyapi import AlchemyAPI

KEY = open('../../api_key.txt','rb').read()[:-1]


def alcObj():
        return AlchemyAPI(KEY)


def sentAn(obj, text):
  response = obj.sentiment("text", text)
  print response
```

```
  try:
    score     = response[u'docSentiment'][u'score']
  except:
    score     = 0
  type    = response[u'docSentiment'][u'type']
  return score, type
```

This approach was met with two major issues.

We did not appear to find suitable historical data for a detailed tweet based approach. This limited our usage for the following. In the presence of the Twitter data, the planned approach was to involve a weighted mean of the tweet sentiment scores to the regression model. Some of our attempts at this were not very fruitful for lack of carefully calibrated twitter data.

We approached an alternative by attempting to use the information from Google & Yahoo based headlines. These approaches too however were met with lack of quality data feeds. On occasion the data feeds were not synchronised correctly.

```
## Using Google News
GOOGLE_NEWS_URL = 'https://ajax.googleapis.com/ajax/services/search/news?v
    =1.0&userip=INSERT-USER-IP&q='
## Company Names Here
COMPANY = 'GOOG' # for example

def sentAn(obj, text):
        response = obj.sentiment("text", text)
        try:
                score     = response[u'docSentiment'][u'score']
        except:
    ## neutral sentiments return no score.
                score     = 0
        type      = response[u'docSentiment'][u'type']
        return score, type

def collect_data():
        url = (GOOGLE_NEWS_URL + COMPANY)

        request = urllib2.Request(url, None)
        response = urllib2.urlopen(request)

        # Process the JSON string.
        results = simplejson.load(response)

        return results

def parsing(data_collected):
        headline_list=[]
        stories = data_collected['responseData']['results']#['
```

```
    titleNoFormatting']
30          for story in stories:
31                  headline_list.append( story['titleNoFormatting'])
32          return headline_list
```

We also explored the usage of an efficient method involving moving averages & flight towards the golden ratio. Codework for the same exists already, for example at: `https://github.com/lsphate/2`

We chose to not take this approach at this point.

Future explorations would however look at exploring this approach, such as `http://www.scientificpapers.or`

We hope to explore these at a later stage.

`Predictions:`