Stock-Market-Prediction

October 31, 2017

```
In [111]: with open('hw4_data/hw4_nasdaq00.txt') as f:
              content = f.readlines()
          train_data = list(map(float,[line.strip() for line in content]))

          with open('hw4_data/hw4_nasdaq01.txt') as f:
              content = f.readlines()
          test_data = list(map(float,[line.strip() for line in content]))
```

1 Getting Feature Vector & outputs from Training data

Feature Vector : Stock indices for the three preceding days.
Output : Stock index on a particular day.

Conforming to the assumed linear model, we can train our model to learn the weights such that likelihood of a stock index on a day given indices on the three preceding days is maximised. But however, we can only start training from day 3, 2000. Each row in `train_X` is the feature vector for a single data point. `train_y` is the actual stock index values across all business days of the year 2000.

```
In [112]: import numpy as np

          train_y = np.array([d for d in train_data[3:]])
          train_X = np.array([[train_data[i-1],train_data[i-2],train_data[i-3]]\
                              for i in range(3,len(train_data))])
```

2 Test datapoints

Since the graph for 4.4 shows continuity, we can use the last three business days of 2000 to predict the first business day of 2001 and so on. The following is an implementation of this.

```
In [113]: train_data.extend(test_data)

          test_y = np.array([d for d in train_data[249:]])
          test_X = np.array([[train_data[i-1],train_data[i-2],train_data[i-3]]\
                              for i in range(249,len(train_data))])
```

3 Getting Linear Coefficients

Linear coefficients are obtained from `train_X` and `train_y` using this equation.

```
a = inv(train_X.T . train_X).(train_X.T . train_y)
```

```
In [114]: a = np.dot(np.linalg.inv(np.dot(train_X.T, train_X)),np.dot(train_X.T,train_y))
```

```
In [119]: print "a1 :",a[0]
          print "a2 :",a[1]
          print "a3 :",a[2]
```

```
a1 : 0.950673366139
a2 : 0.0156013307754
a3 : 0.031895685159
```

4 Predictions & Mean Squared Error

```
In [116]: train_pred = np.dot(train_X,a)
          test_pred = np.dot(test_X,a)
```

MSE on the training data

```
In [117]: sum((train_pred-train_y)**2)/len(train_pred)
```

```
Out[117]: 13902.401076367871
```

MSE on the test data

```
In [118]: sum((test_pred-test_y)**2)/len(test_pred)
```

```
Out[118]: 3600.9253539330207
```

I would not recommend this linear model of combining three preceding days' indices to predict the stock index on a day. This linear model would not perform well on real world stock index fluctuations. Given data has a largely decreasing trend in the training & testing data, the reason for which it actually has lesser error on test data. This might not be the case with actual stock indices in a long run.