Sayyed Mohsen Vazirizade

23398312

smvazirizade@email.arizona.edu

INFO 521

Introduction to Machine Learning

Assignment 2

Problem 1

We have to complete the script for the calculation of **w**. To this end, the following script is added to the code:

```
   w = numpy.dot(numpy.linalg.inv(numpy.dot(numpy.transpose(X),X)),numpy.dot(numpy.transpose(X),t))  # Calculate w vector (as an
numpy.array)

   print('w.shape', w.shape)

   return w
```

It should be mentioned that the name of the function for calculation of **w** has been changed to mfitpoly. Apart from the fitpoly, the following script is used.

```
# -*- coding: utf-8 -*-
"""
Created on Sat Sep  9 14:05:43 2017
@author: smvaz
"""
import numpy
import fitpoly
A=fitpoly.read_data('data/mens100.csv', d=',')
x=(A[:,0])
t=A[:,1]
print('x=%s' % (x))
print('t=%s' % (t))
fitpoly.plot_data(x, t, title='Data')
print('-----------------------fitpoly')
w=fitpoly.mfitpoly(x, t, 1)
print('w')
print(w)
plot_model(x, w)
```

The following figure, shows the 1st order polynomial for the data, mens100.

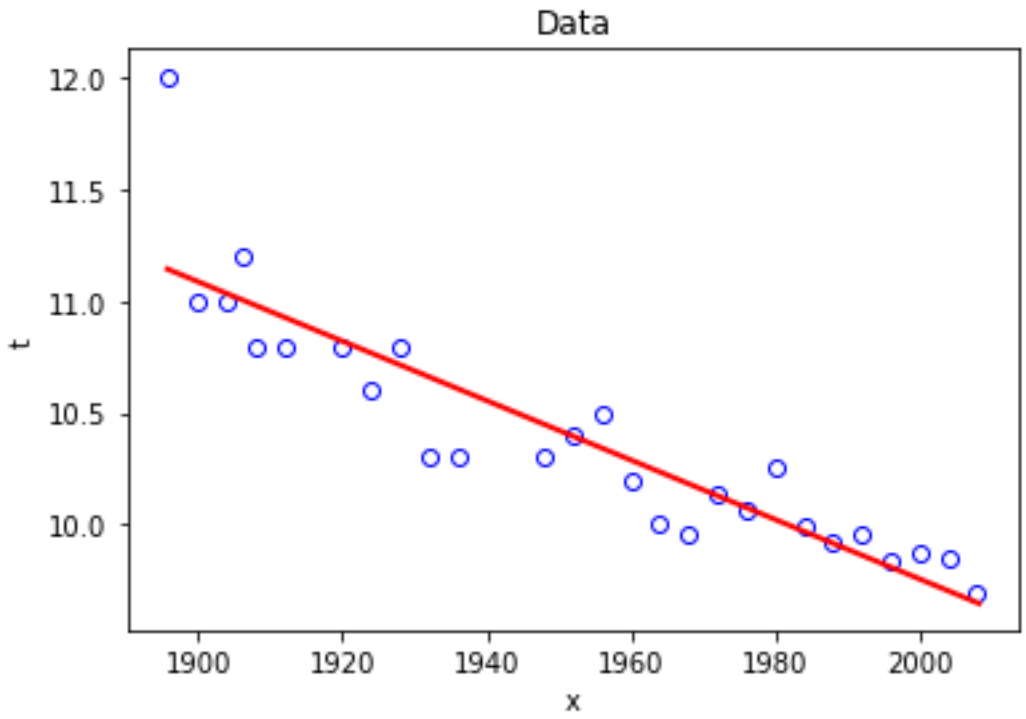

Figure 1 1st order polynomial for mens100.

The figure below is generated by Excel and values of the trendline in middle corroborate our calculation in the previous step.

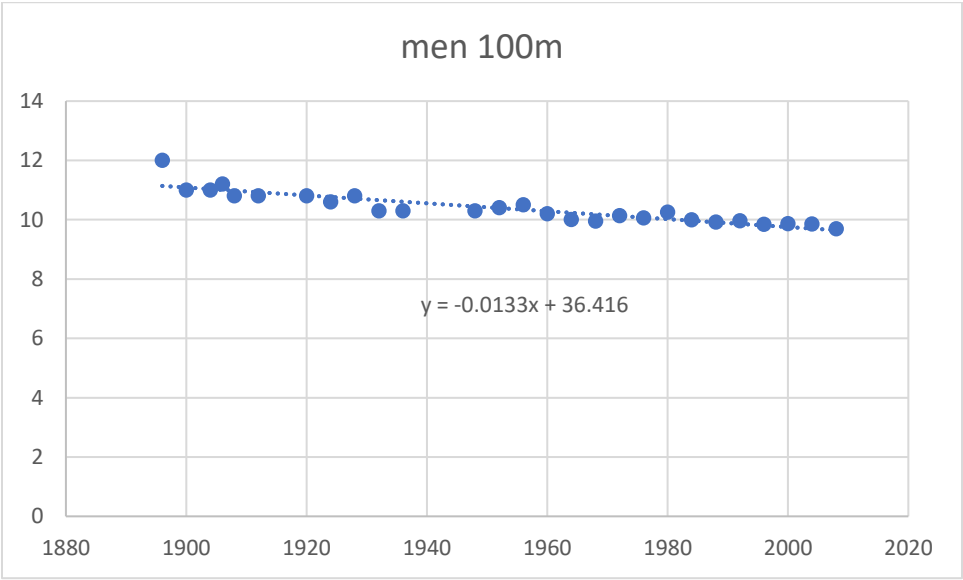

Figure 2 1st order polynomial for mens100 by Excel

The values of **w** are as follow:

[3.64164559e+01 -1.33308857e-02]

Thus, the equation of the fitted line is:

$$t = -0.0133\,x + 36.4$$

Problem 2

The procedure is almost the same as the previous problem. The only required change is changing the address of the input file. In this regard, the values of **w** are:

[4.09241546e+01 -1.50718122e-02]

Thus, the equation of the fitted line is:

$$t = -0.0151\, x + 40.09$$

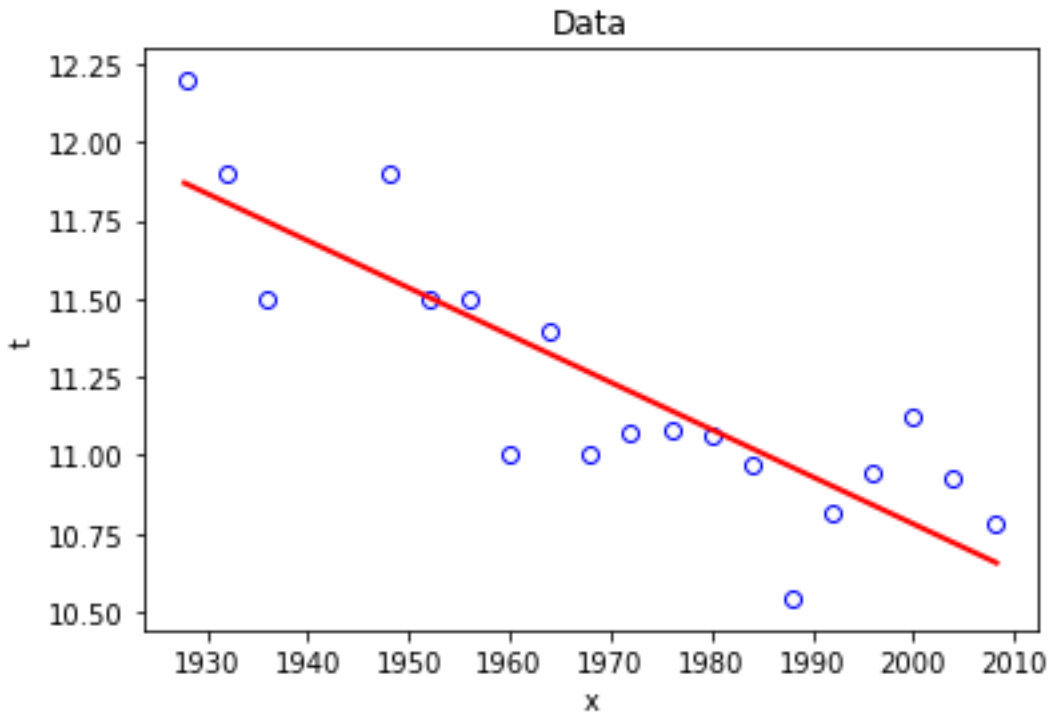

Figure 3 1st order polynomial for womens100

Problem 3

The process is the same as the two previous problems, just we have to change the input address and change the degree of the polynomial fitting curve to 3.

[-1002.32228474 8.75912877 -24.63650296 -4.88946983]

$$t = -1002.32228474\,x^3 \quad 8.75912877\,x^2 \quad -24.63650296\;\; x \;\; -4.88946983$$

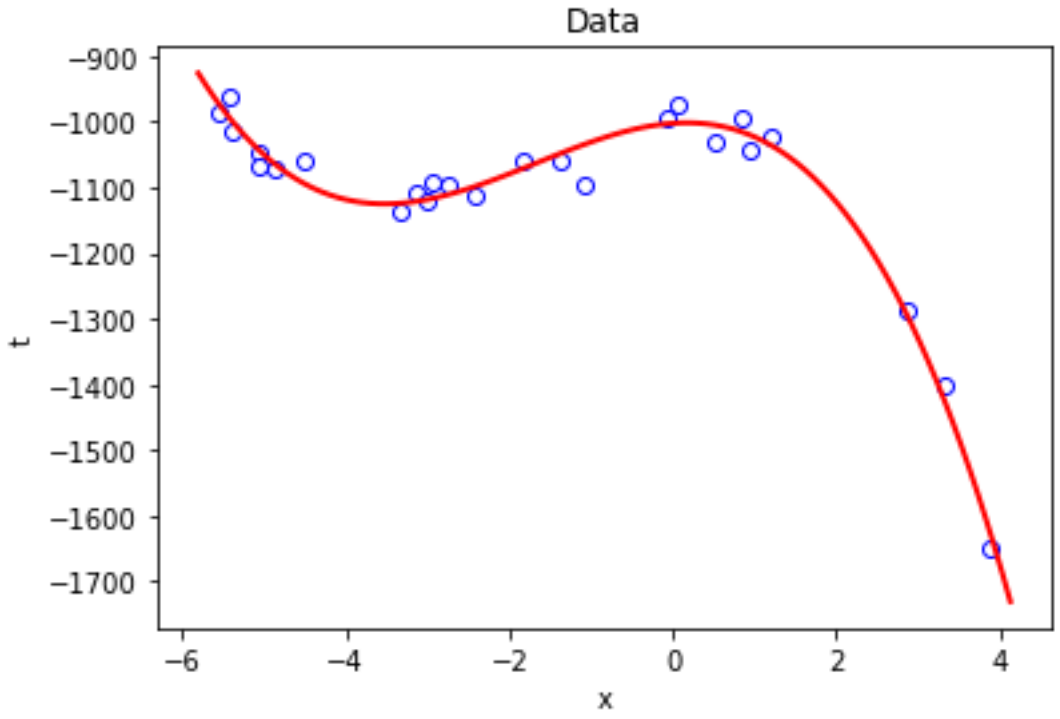

Figure 4 3rd order polynomial for synthdata2017

Problem 4

According to the problem statement we have to distribute the 25 samples in to 5 folds.

Figure 5 Schematic distribution of the data

As mentioned in the problem statement, we used 5 folds and test 0 to 7 order polynomial. As shown in the figures below, although increasing the order of the fitting curve reduce the train loss value; however, it raise the CV loss. In this regard, it seems the 4th order polynomial is the best fit to the data. Furthermore, in the following figures the Independent test loss does not have specific meaning because we used random data just for the sake convenience.

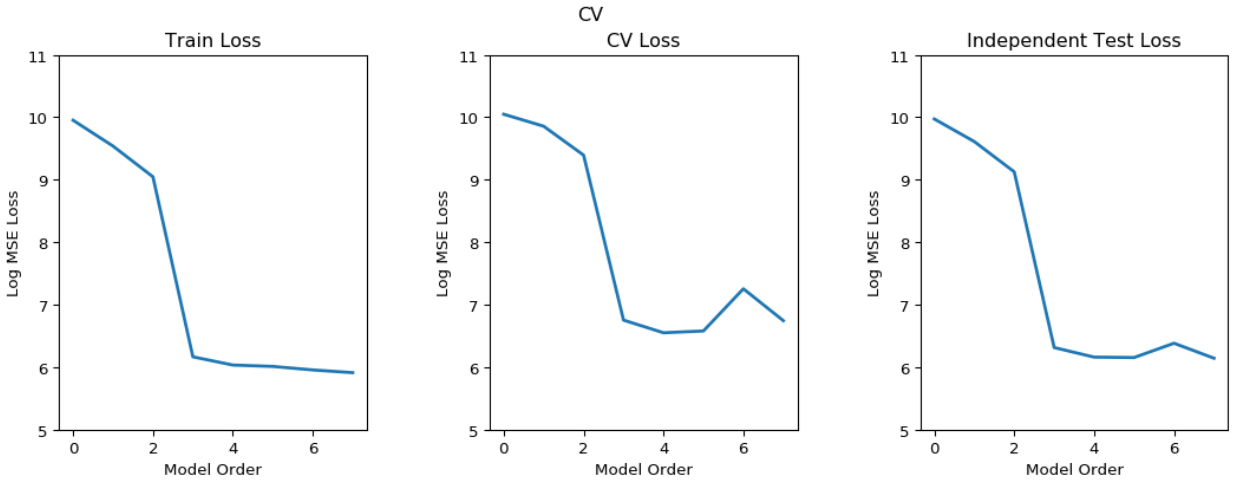

Figure 6 Train loss, CV loss and Independent test loss for 5 folds

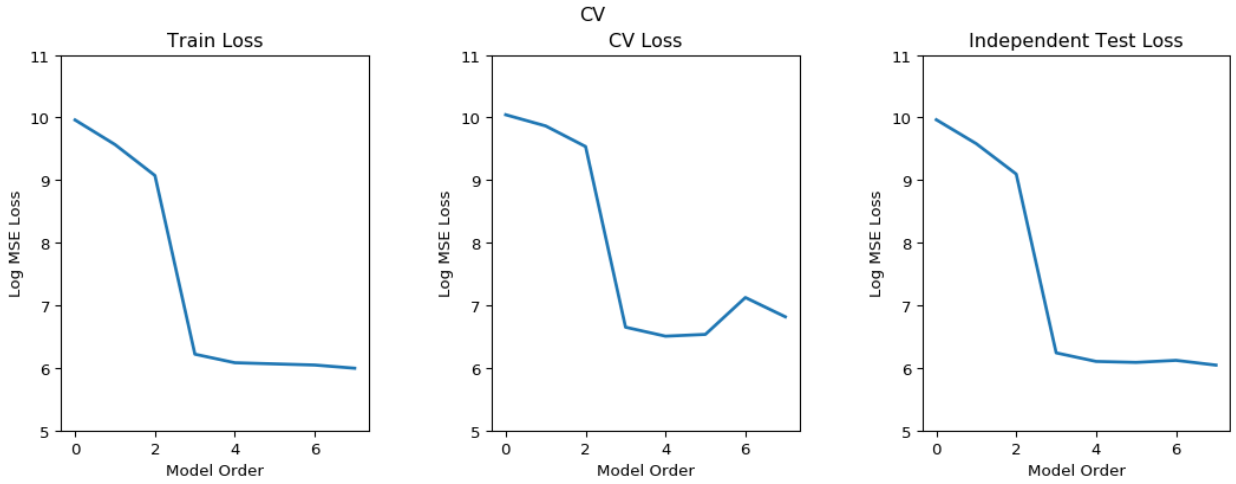

Figure 7 Train loss, CV loss and Independent test loss for 5 LOOCV

In this case, both methods are employed for a couple of times and results are pretty close to each other.

The script which is used for calling cv_demo is:

```
# -*- coding: utf-8 -*-
"""
Created on Sat Sep  9 14:05:43 2017

@author: smvaz
"""
#calling the library
import numpy
import fitpoly
import cv_demo
#calling the data
A=fitpoly.read_data('data/synthdata2017.csv', d=',')
#shuffling the data
RandomA=cv_demo.permute_rows(A, P=None)
TrainA=RandomA[0:20,:]
TestA=RandomA[20:25,:]
#printing the output for checking
print('RandomA=\n %s \n' % (RandomA))
print('TrainA=\n %s \n' % (TrainA))
print('TestA=\n %s \n' % (TestA))
#calling required function from cv demo
maxorder=4
#cv_demo.run_cv( 5, maxorder, TrainA[:,0], TrainA[:,1], TestA[:,0], TestA[:,1], randomize_data=False, title='CV' )
best_poly, min_mean_log_cv_loss, w=cv_demo.run_cv( 5, maxorder, RandomA[:,0], RandomA[:,1], RandomA[:,0], RandomA[:,1],
randomize_data=False, title='CV' )


#calculation and drawing pertinent to the best fitted data
cv_demo.plot_data(RandomA[:,0], RandomA[:,1])
cv_demo.plot_model(RandomA[:,0], w, color='r')
print('w=%s' %(w))
```

and w is defined as the previous example.

As mentioned earlier the best polynomial is the 4^{th} order one and the following matrixes show the value of w for 5-fold and LOOCV respectively.

w=[-1.01058629e+03 1.93350205e+01 -1.95398491e+01 -5.91528282e+00

-3.08537895e-01]

LOOCV

w=[-1.01414590e+03 1.85248461e+01 -2.00744937e+01 -5.79775839e+00

-2.56164582e-01]

w=[w0 w1 w2 w3 w4] where t= w0+w1.x+ w2.x^2+ w3.x^3+ w4.x^4

The following two figures depict the data and the best curve fitted for each method.

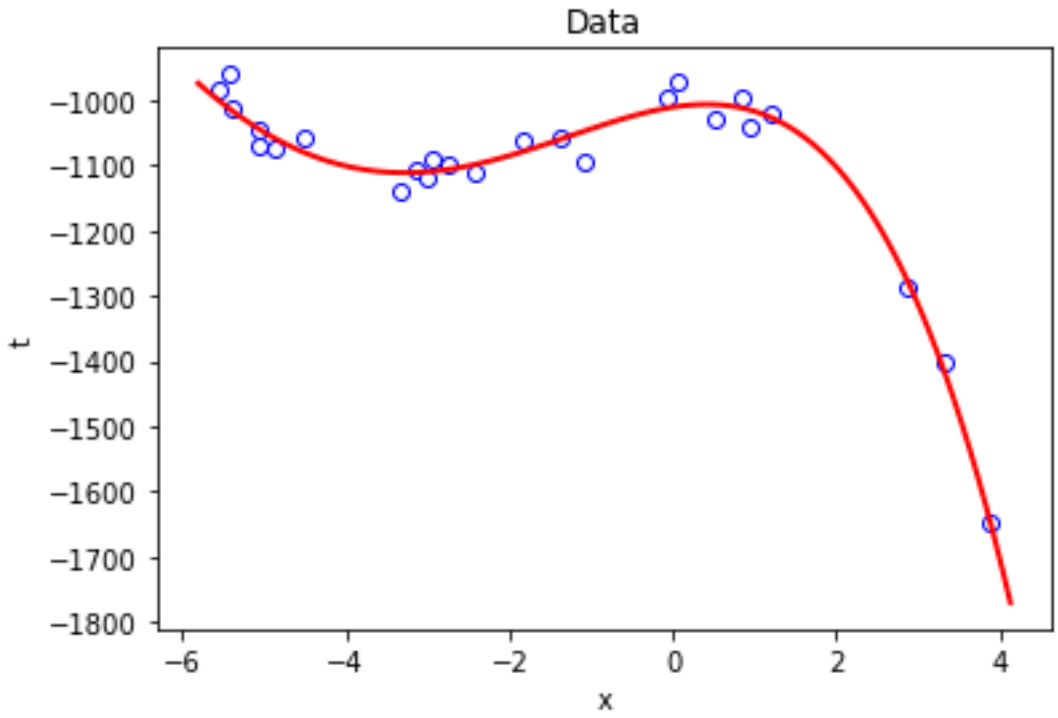

Figure 8 data and 4th order polynomial for synthdata2017 for 5-fold

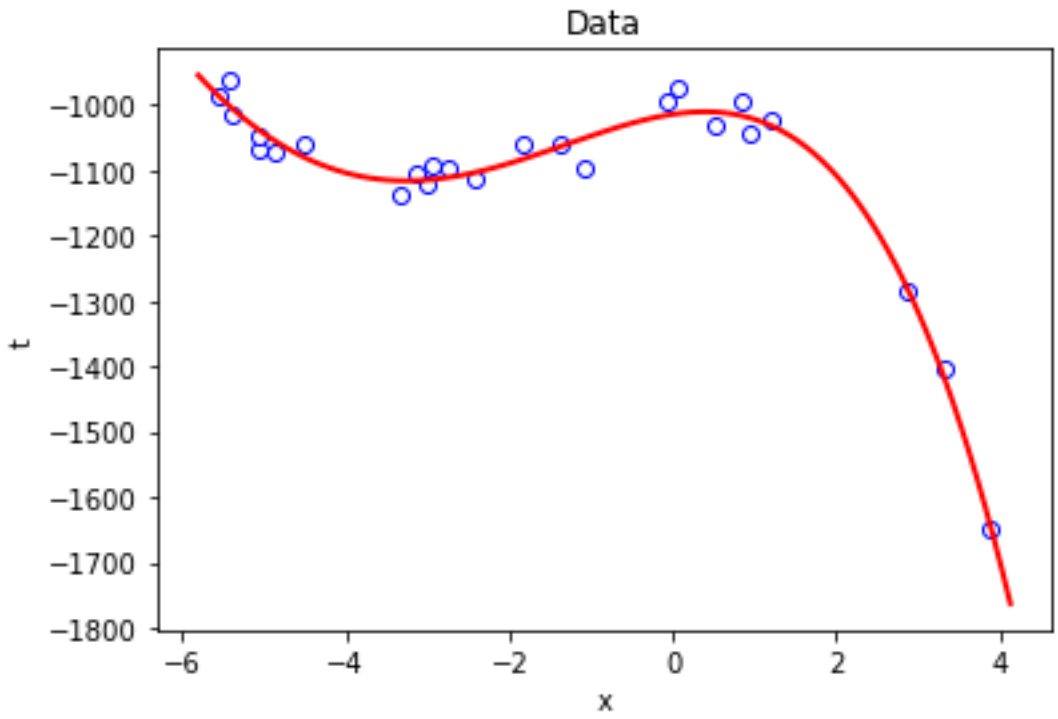

Figure 9data and 4th order polynomial for synthdata2017 for 5-fold

Problem 5

$$\mathcal{L} = \sum_{n=1}^{N} (t_n - w^T x_n)^2 = (xw - t)^T (xw - t)$$

Derivate for w:

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{\partial (xw - t)^T (xw - t)}{\partial w} \Rightarrow \frac{\partial \mathcal{L}}{\partial w} = x^T x\, w - 2\, x^T\, t \Rightarrow Iw = (x^T x)^{-1} x^T\, t$$

As you can see, because N cancels out it does not change the solution. The only matter is that when we use mean loss it makes it independent of the number of input data while when the equation is not divided by N it is obvious that increasing the number of input data will increase the error.

Problem 6

We can write the summation formulation in matrix form as follow:

$$\mathcal{L} = \sum_{n=1}^{N} \alpha_n (t_n - w^T x_n)^2 = (xw - t)^T (\alpha (xw - t))$$

Where α is a diagonal matrix which includes α_n values.

After taking derivation for w:

$$\frac{\partial \mathcal{L}}{\partial w} = 0 \Rightarrow \frac{(xw - t)^T (\alpha (xw - t))}{\partial w} = 0 \Rightarrow \frac{\partial \mathcal{L}}{\partial w} = x^T \alpha x\, w - 2\, x^T \alpha\, t \Rightarrow Iw = (x^T \alpha x)^{-1} x^T \alpha\, t$$