Arsingh3_ECE542_HW01B_Submission

January 29, 2020

```python
[1]: import numpy as np
     import matplotlib.pyplot as plt
     import pandas
     import os
     # Function that creates the X matrix as defined for fitting our model
     def create_X(x,deg):
         X = np.ones((len(x),deg+1))
         for i in range(1,deg+1):
             X[:,i] = x**i
         return X


     # Function for predicting the response
     def predict_y(x,beta):
         return np.dot(create_X(x,len(beta)-1),beta)


     # Function for fitting the model
     def fit_beta(df,deg):
         return np.linalg.lstsq(create_X(df.x,deg),df.y,rcond=None)[0]


     # Function for computing the MSE
     def mse(y,yPred):
         return np.mean((y-yPred)**2)


     # Loading training, validation and test data
     os.chdir(r'C:\1_WorkSpace\Professional\NC States\2nd_Sem_Spring_2020\ECE_542␣
      ↪Neural Networks\HW\HW1\hw01b\hw01_files')
     dfTrain = pandas.read_csv('Data_Train.csv')
     dfVal = pandas.read_csv('Data_Val.csv')
     dfTest = pandas.read_csv('Data_Test.csv')

     ############ TRAINING A MODEL

     # Fitting model
     deg = 1
     X = create_X(dfTrain.x,deg)
     beta = fit_beta(dfTrain,deg)
```

```
# Computing training error
yPredTrain = predict_y(dfTrain.x,beta)
err = mse(dfTrain.y,yPredTrain)
print('Training Error = {:2.3}'.format(err))

# Computing test error
yPredTest = predict_y(dfTest.x,beta)
err = mse(dfTest.y,yPredTest)
print('Test Error = {:2.3}'.format(err))
```
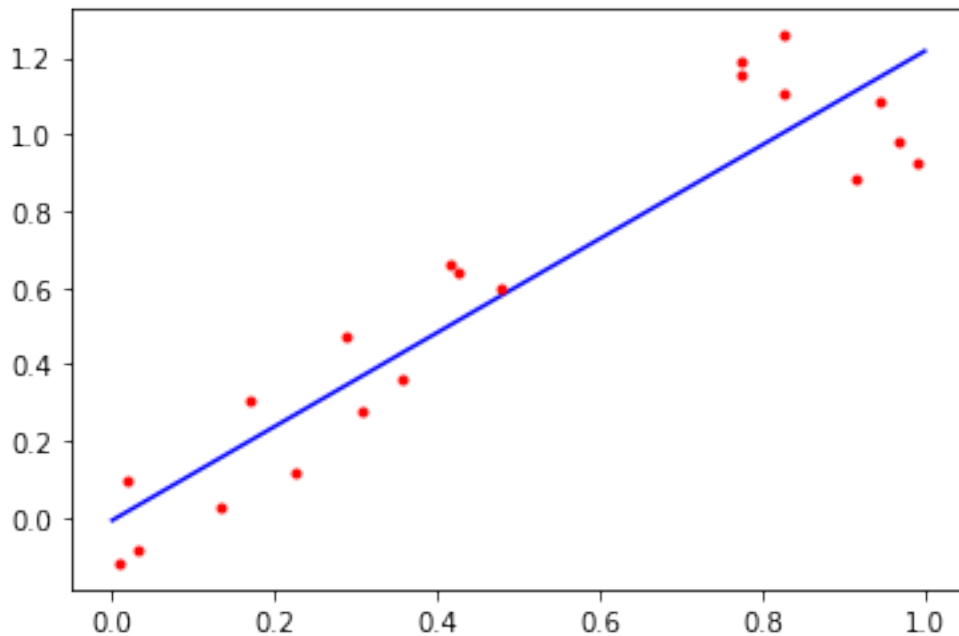
```
Training Error = 0.0258
Test Error = 0.0154
```

[2]:
```
############ PLOTTING FITTED MODEL
x = np.linspace(0,1,100)
y = predict_y(x,beta)

plt.plot(x,y,'b-',dfTrain.x,dfTrain.y,'r.')
plt.show()
```


[3]:
```
############ HYPER-PARAMETER TUNING

# Initializing range of degree values to be tested and errors
degRange = list(range(1,11))
errTrain = np.zeros(len(degRange))
errVal = np.zeros(len(degRange))
```

```python
# Computing error as a function of degree
# ...
for i in range(len(degRange)):
    betaTrain = fit_beta(dfTrain,degRange[i])#finding beta for every degree
    ypredTrain = predict_y(dfTrain.x,betaTrain)#fidning the prediciton for␣
 ↪every degree
    errTrain[i] = mse(dfTrain.y,ypredTrain)#finding error for training data
    ypredVal = predict_y(dfVal.x,betaTrain)#finding y prediction of validation␣
 ↪data for every degree
    errVal[i] = mse(dfVal.y,ypredVal)#calculating validation error
# Plotting training and validation errors
plt.plot(degRange,errTrain,'b-',degRange,errVal,'r-')
plt.legend(('Training Error','Validation Error'))
plt.show()
```
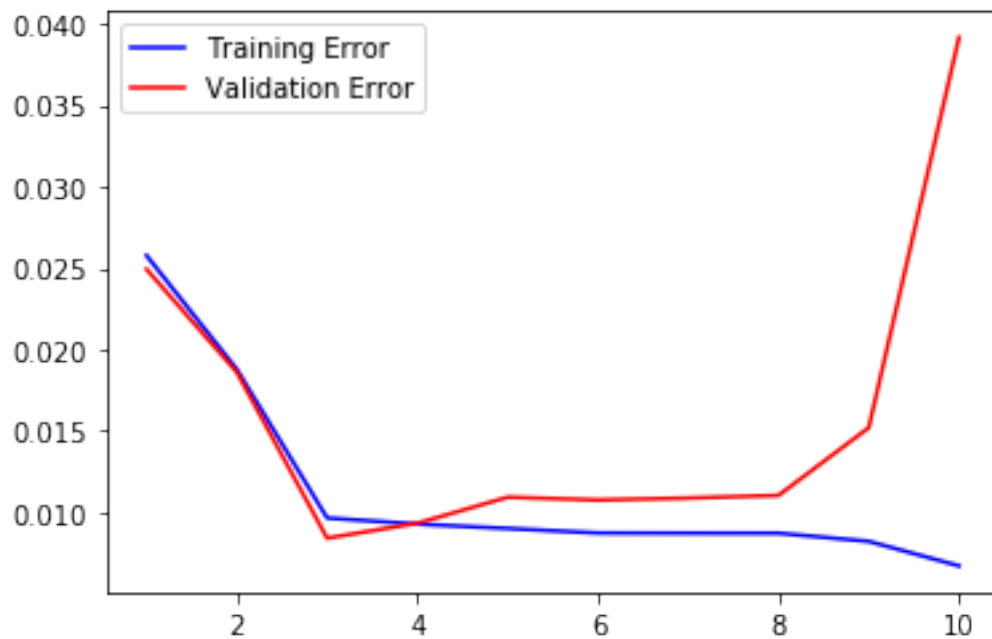


```python
[4]: ########### TRAINING SELECTED MODEL

# Concatenating data training and validation data frames
df = pandas.concat([dfVal,dfTrain])# ...

# Fit model using the optimal degree found in the previous cell
degOpt = 3 # ...Selecting 3 as it avoids over-fitting
X = create_X(df.x,deg)
beta = fit_beta(df,degOpt)
```

```
# ...

# Compute and print training and test errors
# ...
yPredTrain = predict_y(df.x,beta)
err = mse(df.y,yPredTrain)
print('Training Error = {:2.3}'.format(err))

yPredTest = predict_y(dfTest.x,beta)
err = mse(dfTest.y,yPredTest)
print('Test Error = {:2.3}'.format(err))
```
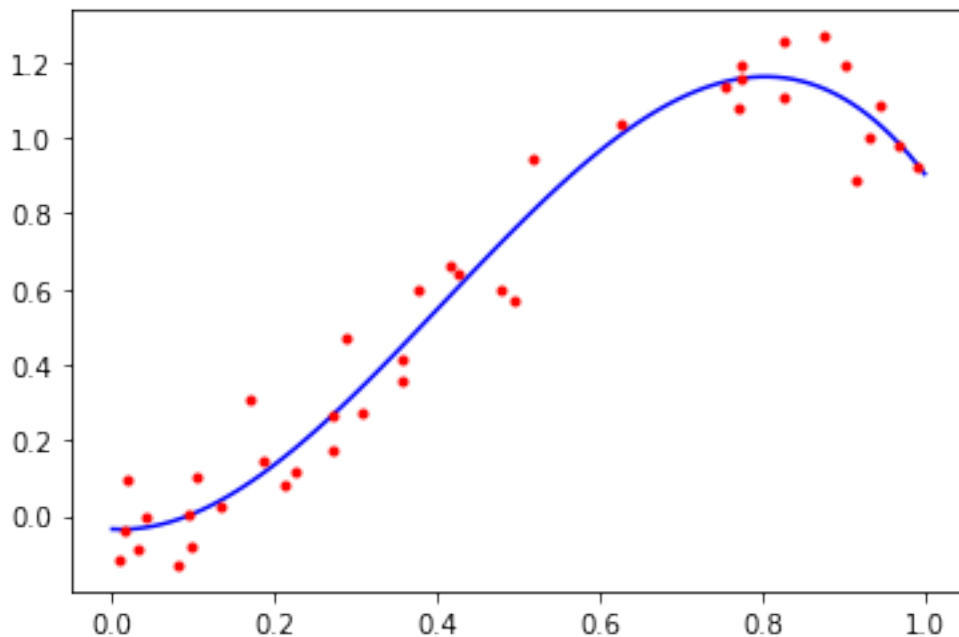
```
Training Error = 0.0087
Test Error = 0.0108
```

[5]:
```
############ PLOTTING FITTED MODEL
# Plot the fitted model as in the second cell
# ...
x = np.linspace(0,1,100)
y = predict_y(x,beta)

plt.plot(x,y,'b-',df.x,df.y,'r.')
plt.show()
```


[ ]: