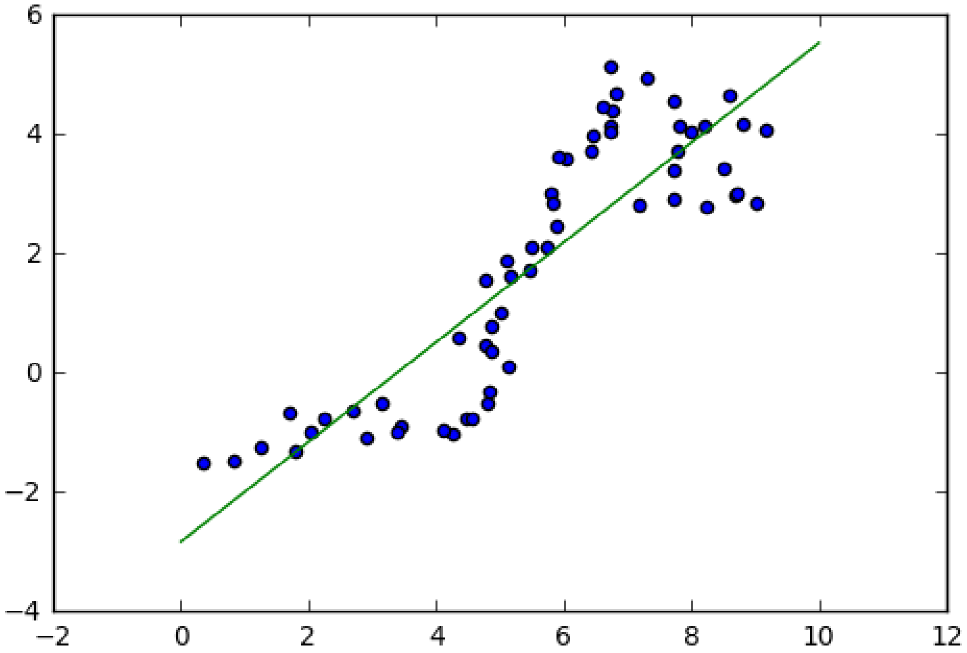
CS 178 HW 2

January 27, 2017

1 Kevin Phung, 47881547

1.1 Problem 1: Linear Regression

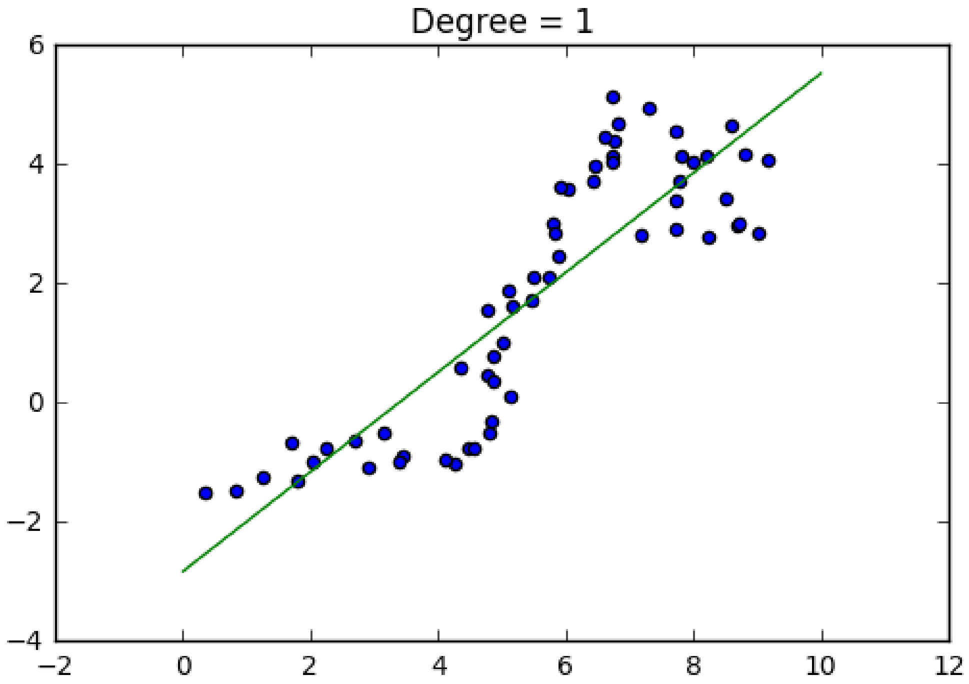
```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        import mltools as ml

        # Problem 1: Linear Regression

        # 1a
        data = np.genfromtxt("data/curve80.txt",delimiter=None)

        X = data[:,0]
        X = X[:,np.newaxis]
        Y = data[:,1]
        Xtr,Xte,Ytr,Yte = ml.splitData(X,Y,0.75)

        # 1b
        lr = ml.linear.linearRegress(Xtr,Ytr)
        xs = np.linspace(0,10,200)
        xs = xs[:,np.newaxis]
        ys = lr.predict(xs)

        plt.scatter(Xtr,Ytr)
        plt.plot(xs,ys,"g")
        ax = plt.axis()
        plt.show()
        print("Linear Regression Coefficients: "+str(lr.theta))
        mseTe = lr.mse(Xte,Yte)
        mseTr = lr.mse(Xtr,Ytr)

        print("Mean Square Error of Test Data: "+str(mseTe))
        print("Mean Square Error of Training Data: "+str(mseTr))
```

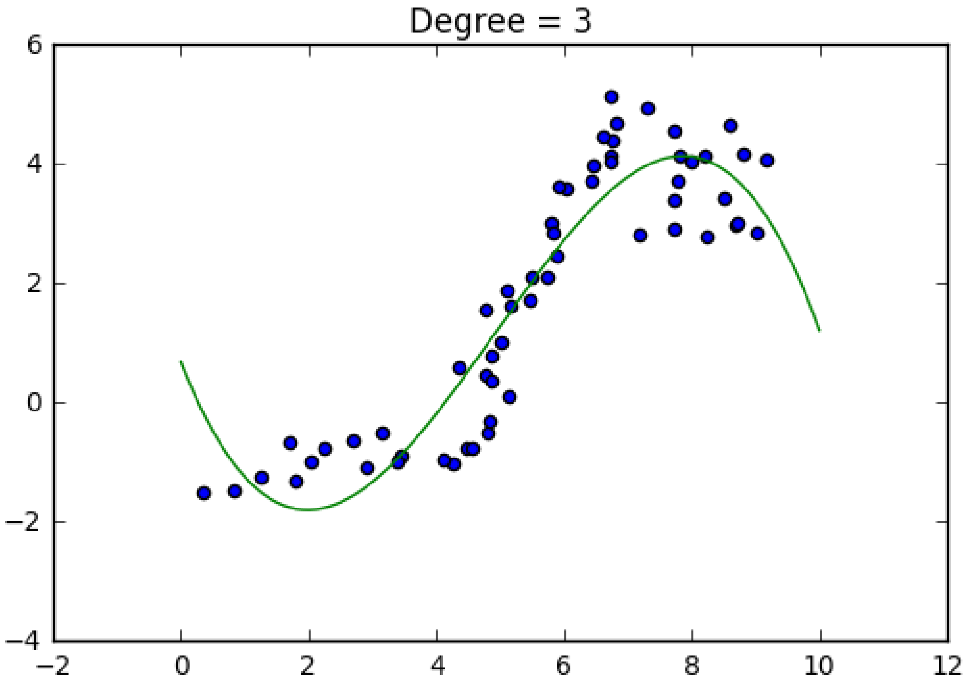


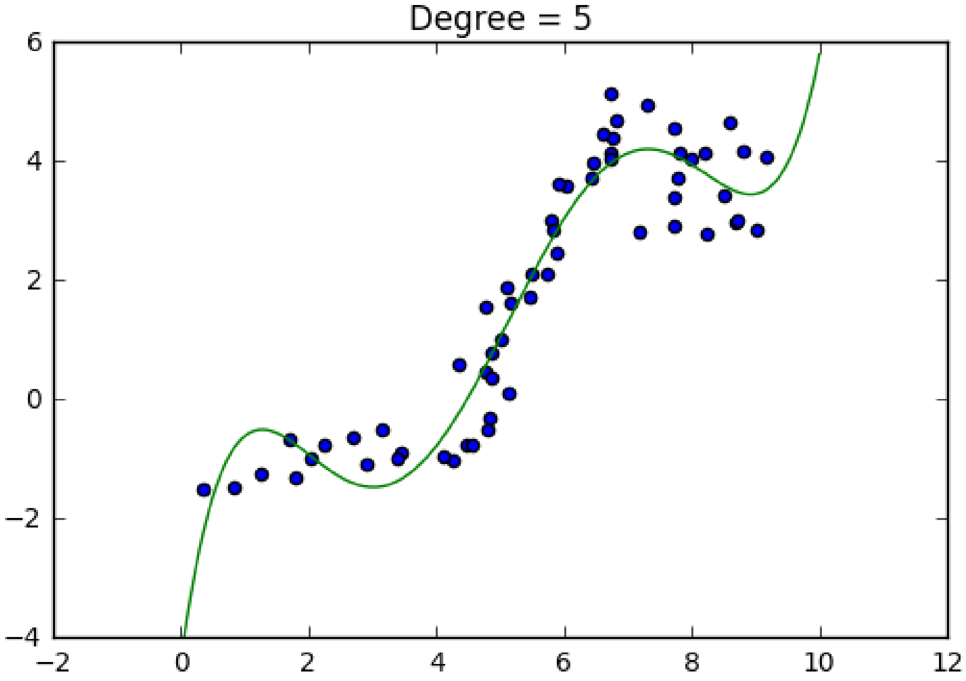
```
Linear Regression Coefficients: [[-2.82765049  0.83606916]]
Mean Square Error of Test Data: 2.24234920301
Mean Square Error of Training Data: 1.12771195561
```

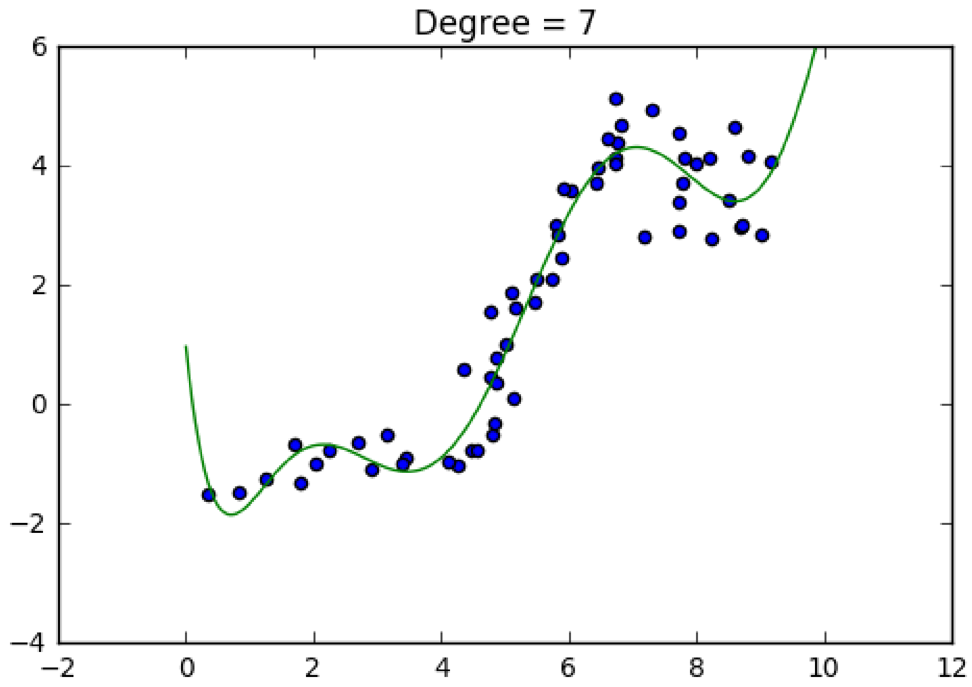
```
In [2]: # 1c
        degrees = [1,3,5,7,10,18]
        mseTeList = []
        mseTrList = []

        for degree in degrees:

            XtrP = ml.transforms.fpoly(Xtr,degree,bias=False)

            XtrP,params = ml.transforms.rescale(XtrP)

            lr = ml.linear.linearRegress(XtrP,Ytr)

            XteP,params = ml.transforms.rescale(ml.transforms.fpoly(Xte,degree,
                                                                     False),params)

            Phi = lambda X: ml.transforms.rescale(ml.transforms.fpoly(X,degree,
                                                                       False),params
            YhatTrain = lr.predict(Phi(xs))
            YhatTest = lr.predict(Phi(Xte))
```

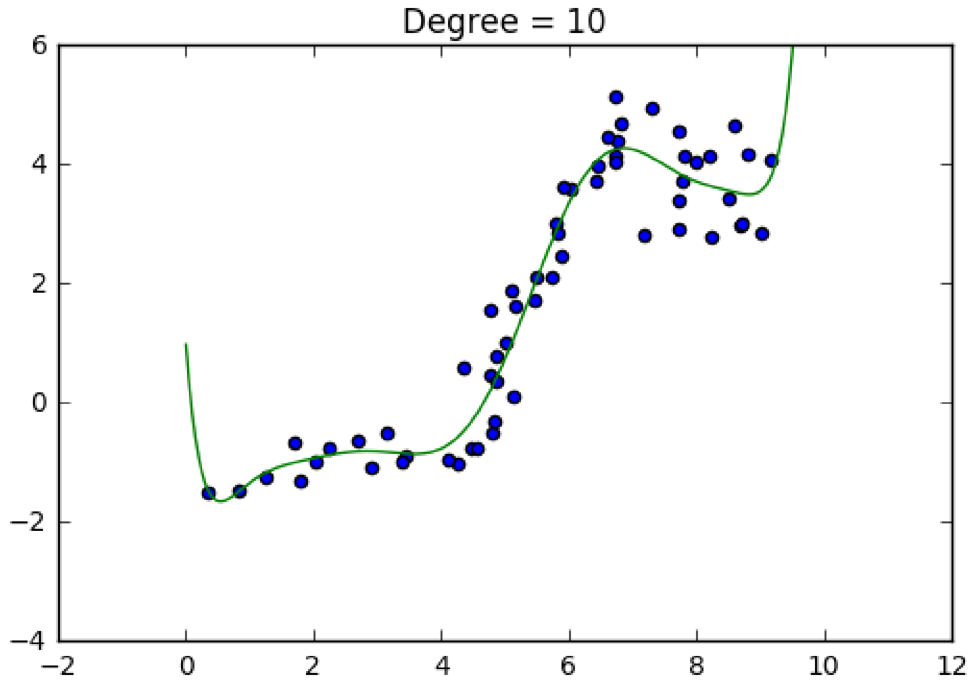
```
    plt.plot(xs,YhatTrain,"g")
    plt.scatter(Xtr,Ytr)
    plt.axis(ax)
    plt.title("Degree = "+str(degree))
    plt.show()

    mseTe = lr.mse(Phi(Xte),Yte)
    mseTr = lr.mse(Phi(Xtr),Ytr)

    mseTeList.append(mseTe)
    mseTrList.append(mseTr)

plt.title("Test MSE vs Training MSE")
plt.ylabel("MSE")
plt.xlabel("Degree")
plt.semilogy(degrees,mseTeList,'g-',degrees,mseTrList,'r-')
plt.show()
print("Green = Test Data, Red = Training Data")
```

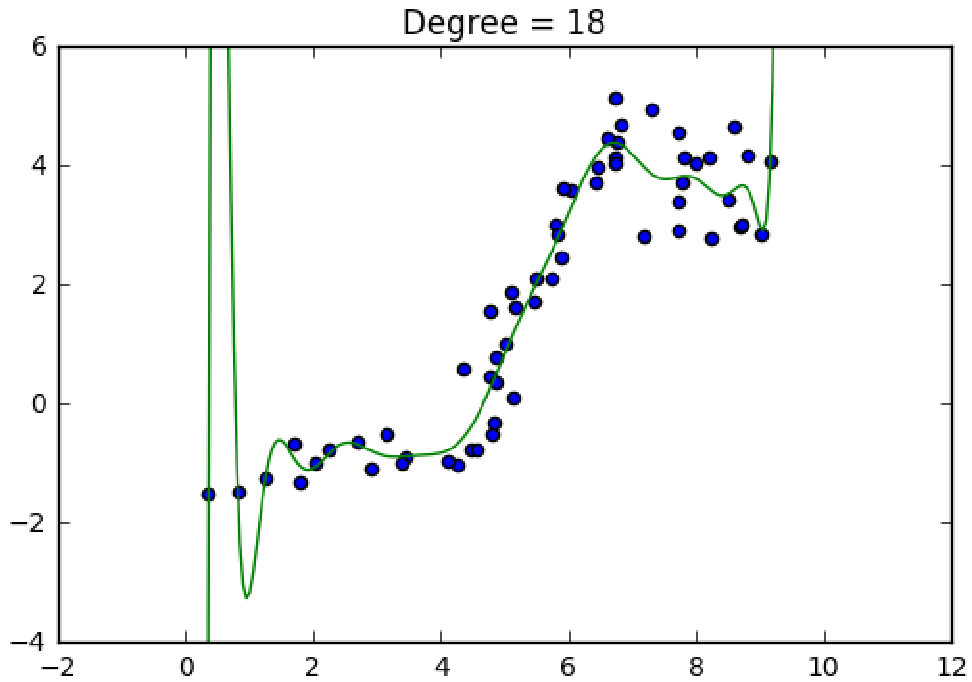


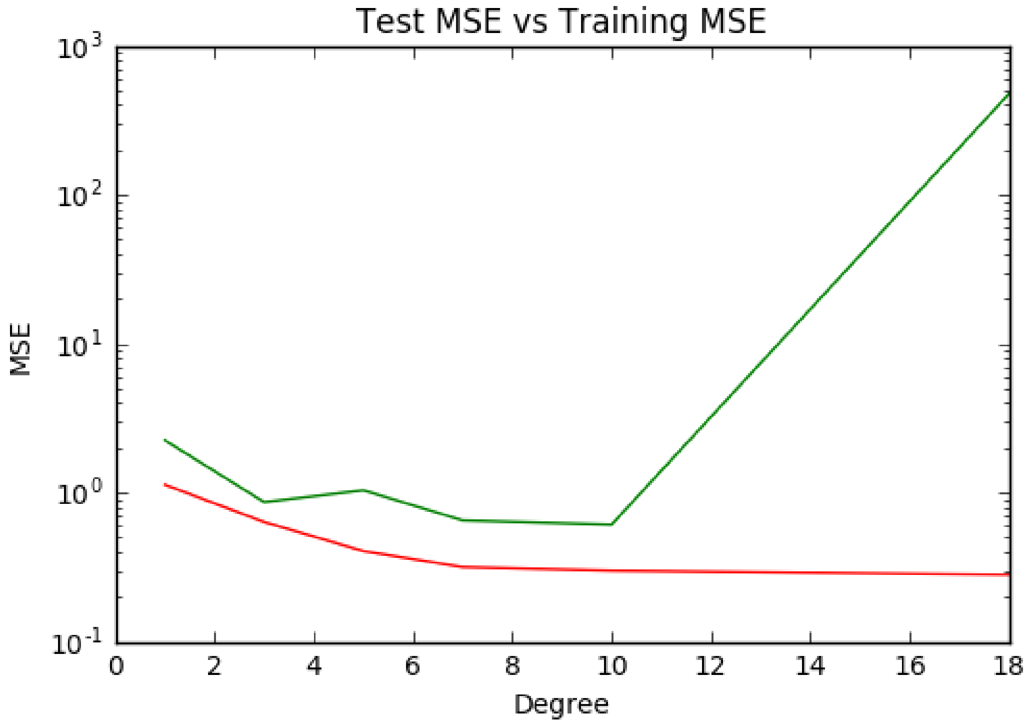

Degree = 3
6
4
2
0
−2
−4
−2
0
2
4
6
8
10
12


Degree = 5
6
4
2
0
−2
−4
−2
0
2
4
6
8
10
12


Degree = 7
6
4
2
0
−2
−4
−2
0
2
4
6
8
10
12


Degree = 10
6
4
2
0
−2
−4
−2
0
2
4
6
8
10
12

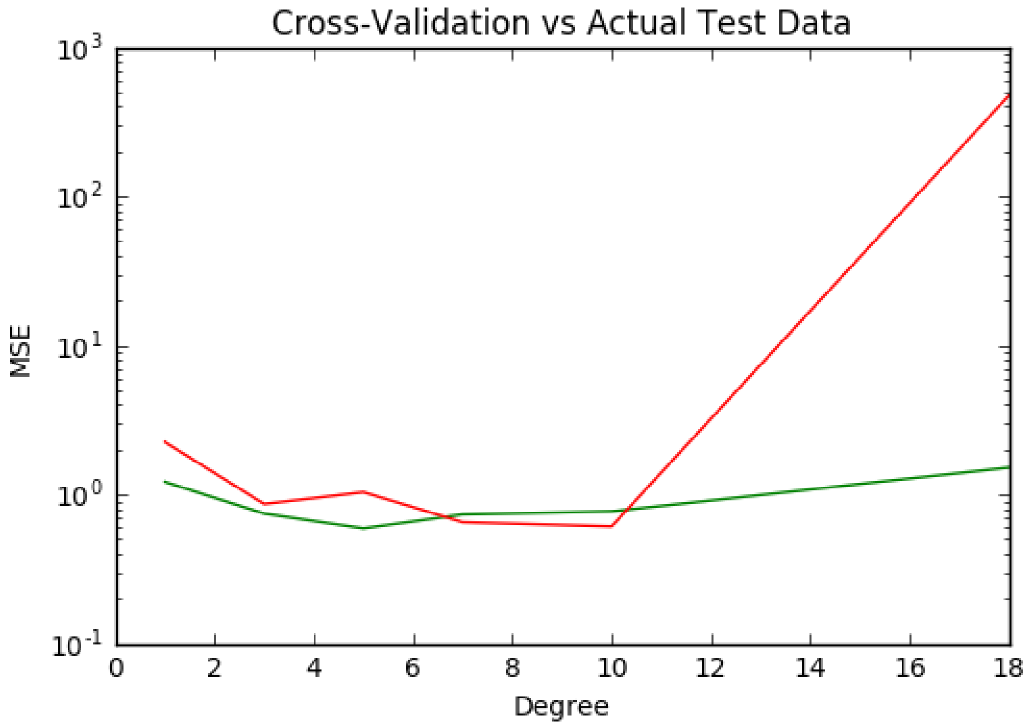

Degree = 18
6
4
2
0
−2
−4
−2
0
2
4
6
8
10
12


Test MSE vs Training MSE
10^3
10^2
10^1
10^0
10^{-1}
MSE
0
2
4
6
8
10
12
14
16
18
Degree

```
Green = Test Data, Red = Training Data
```

1.2 Problem 2: Cross-Validation

```
In [3]: #Problem 2: Cross-validation

        nFolds = 5;
        J = []

        for degree in degrees:
            XtrP = ml.transforms.fpoly(Xtr,degree,bias=False)
            temp = [0,0,0,0,0]
            for iFold in range(nFolds):
                Xti,Xvi,Yti,Yvi = ml.crossValidate(XtrP,Ytr,nFolds,iFold)
                learner = ml.linear.linearRegress(Xti,Yti)
                temp[iFold] = (learner.mse(Xvi,Yvi))
            J.append(np.mean(temp))

        plt.title("Cross-Validation vs Actual Test Data")
        plt.ylabel("MSE")
        plt.xlabel("Degree")
        plt.semilogy(degrees,J,'g-',degrees,mseTeList,'r-')
        plt.show()
        print("Green = Cross-Validation, Red = Actual Test Data")

        '''
        According to the graph, degree 5 has the minimum cross-validation error.
        The MSE estimated from cross-validation is lower compared to the actual
        test data on higher degrees. Starting from around 10 degrees the MSE of
        the actual test data is substantially higher than the cross-validation.
        '''
```



Green = Cross-Validation, Red = Actual Test Data