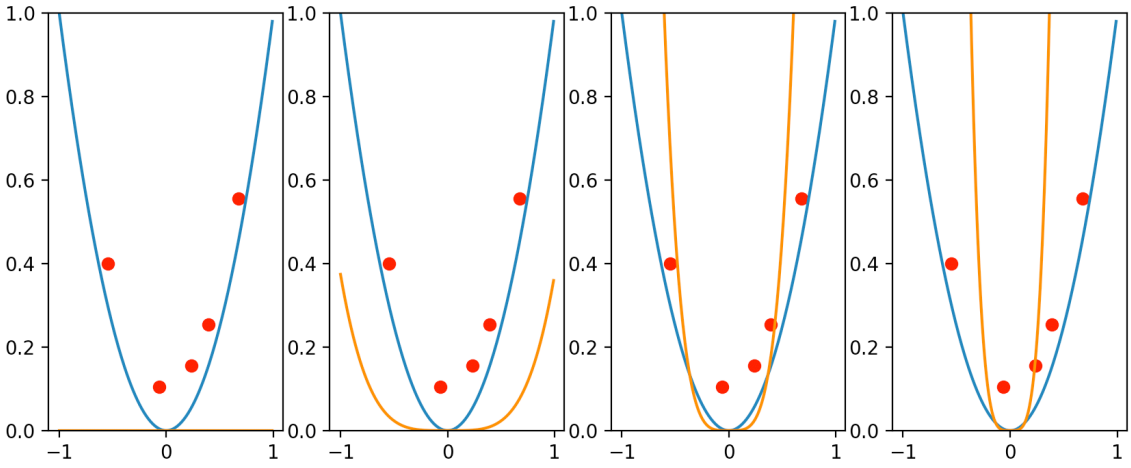
Samantha Rothman

Homework 4

Problem 1

a) The graphs below show the target function (blue) and the hypothesis (orange) using regularization. The red points are the random data points with noise. The graphs go in order of lambdas [0, 1E-5, 1E-2, 1E0].



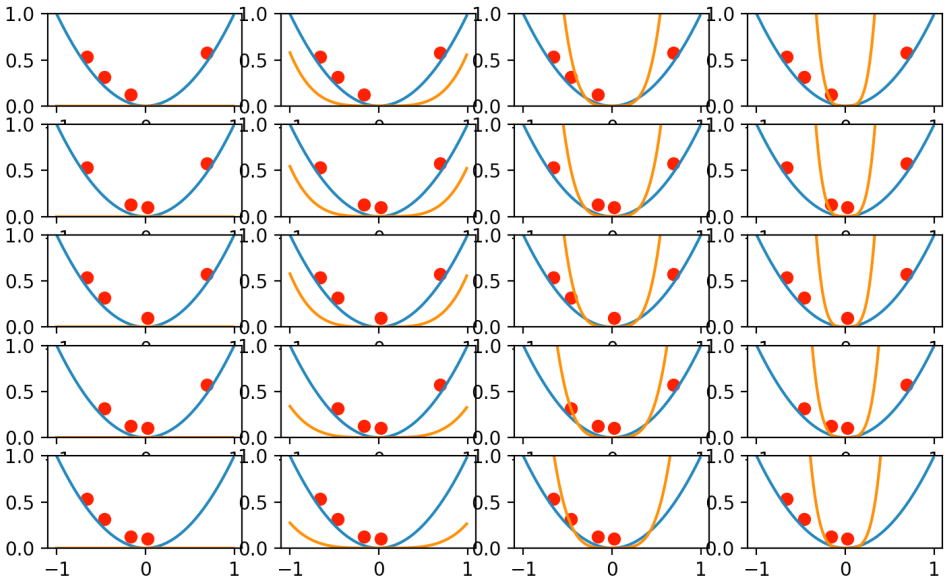
b) Using the program, I calculated the out of sample error for each lambda. The results are below. Lambda=0 gave the best results.

```
total error =  [[0.01]] for lambda =  0
total error =  [[1.45253309e+13]] for lambda =  0.006737946999085467
total error =  [[5.85993653e+15]] for lambda =  0.1353352832366127
total error =  [[3.19941694e+17]] for lambda =  1.0
```

Problem 2

a) The graphs below show leave-one-out cross validation. The target is in blue, the hypothesis is in orange, and the random points with noise are in red. They go from left to right in order [0, 1E-5, 1E-2, 1E0].

Samantha Rothman



b) For each lambda, I calculated the validation error. The validation error is typically higher than part b. However, lambda=0 had the lowest error again.

```
Validation error:  6.091812478666517 for lambda  0
Validation error:  17115984838519.469 for lambda  0.006737946999085467
Validation error:  6905085002845160.0 for lambda  0.1353352832366127
Validation error:  3.770048757170435e+17 for lambda  1.0
```

c) When I used lambda = 0 to retrain the data, I calculated the following weights using the full D.

final w: [[2.11111111e-01] [-4.26325641e-14][2.22222222e-01][8.88178420e-16] [4.33680869e-19]]

Samantha Rothman

Code to set up data points (both questions)

```
import numpy
import matplotlib.pyplot as plt
D = numpy.random.uniform(low=-1, size=(5, 2))
for point in D:
    xi = point[0]
    point[1] = (xi**2) + 0.1
y = D[:, 1:2]
print(y)
X = D[:,0:1]
Z = numpy.empty((5,5))
for index in range(5):
    x = D[index][0]
    z1 = x
    z2 = (1/2)*(((3*x)**2)-1)
    z3 = (1/2)*(((5*x)**3)-(3*x))
    z4 = (1/8)*(((35*x)**4)-((30*x)**2)+3)
    z = [1, z1, z2, z3, z4]
    Z[index] = z
n, m = Z.shape
```

Samantha Rothman

Code for regression

```
plt.figure(0)
plots = []
val = numpy.arange(-1,1, 0.01)
for i in range(4):
    lam = L[i]
    ax = plt.subplot2grid((1,4), (0,i))
    plt.plot(val, val**2)
    plt.scatter(X, y, color = 'r')

    w = numpy.matmul((numpy.linalg.inv(numpy.matmul(Z.transpose(), Z))+ lam * I),
        numpy.matmul(Z.transpose(), y))
    #plt.scatter(val, w[0] + w[1]*(val) + w[2]*(val**2) + w[3]*(val**3) +
        w[4]*(val**4))
    c = []
    for a in val:
        #a = Z[i][1]
        b = w[0] + w[1]*a + w[2]*(1/2)*(((3*a)**2)-1) +
            w[3]*(1/2)*(((5*a)**3)-(3*a)) +w[4]*(1/8)*(((35*a)**4)-((30*a)**2)+3)
        b = b/100000000
        c.append(b)
    plt.plot(val, c)
    error = []
    for j in range(5):
        yi = numpy.square(D[j][0])
        zi = Z[j]
        err = numpy.square(yi - numpy.matmul(w.transpose(), zi))  # how to
            calculate error
        error.append(err)
    final = numpy.mean(error) + lam*(numpy.matmul(w.transpose(), w))
    print("total error = ", final, "for lambda = ", lam)
    plt.ylim([0,1])

plt.show()
```

Samantha Rothman

Code for cross validation

```
L = [0, numpy.exp(-5), numpy.exp(-2), numpy.exp(0)]
val = numpy.arange(-1,1, 0.01)
plt.figure(0)
plots = []
for i in range(4):
    lam = L[i]
    error = []
    for j in range(5):
        tempd = D
        tempZ = Z
        yj = D[j][1]
        tempd = numpy.delete(tempd,j , axis = 0)
        tempX = tempd[:, 0:1]
        tempy = tempd[:, 1:2]
        zj = Z[j]
        tempZ = numpy.delete(tempZ, j, axis=0)
        w = numpy.matmul((numpy.linalg.inv(numpy.matmul(tempZ.transpose(), tempZ))+
            lam * I), numpy.matmul(tempZ.transpose(), tempy))
        err = numpy.square(yj - numpy.matmul(w.transpose(), zj))
        error.append(err)
        ax = plt.subplot2grid((5,4), (j,i))
        plt.plot(val, val**2)
        #plt.scatter(val, w[0] + w[1]*(val) + w[2]*(val**2) + w[3]*(val**3) +
            w[4]*(val**4))
        c = []
        for a in val:
            #a = Z[i][1]
            b = w[0] + w[1]*a + w[2]*(1/2)*(((3*a)**2)-1) +
                w[3]*(1/2)*(((5*a)**3)-(3*a))
                +w[4]*(1/8)*(((35*a)**4)-((30*a)**2)+3)
            b = b/100000000
            c.append(b)
        plt.plot(val, c)
        plt.scatter(tempX, tempy, color = 'r')
        plt.ylim([0,1])
    print("Validation error: ", numpy.mean(error), "for lambda ", lam)
        #print(err)

finalw = numpy.matmul((numpy.linalg.inv(numpy.matmul(Z.transpose(), Z))+ 0 * I),
    numpy.matmul(Z.transpose(), y))
print("final w: " , finalw)
plt.show()
```