EECS 349 (Machine Learning) Homework 3

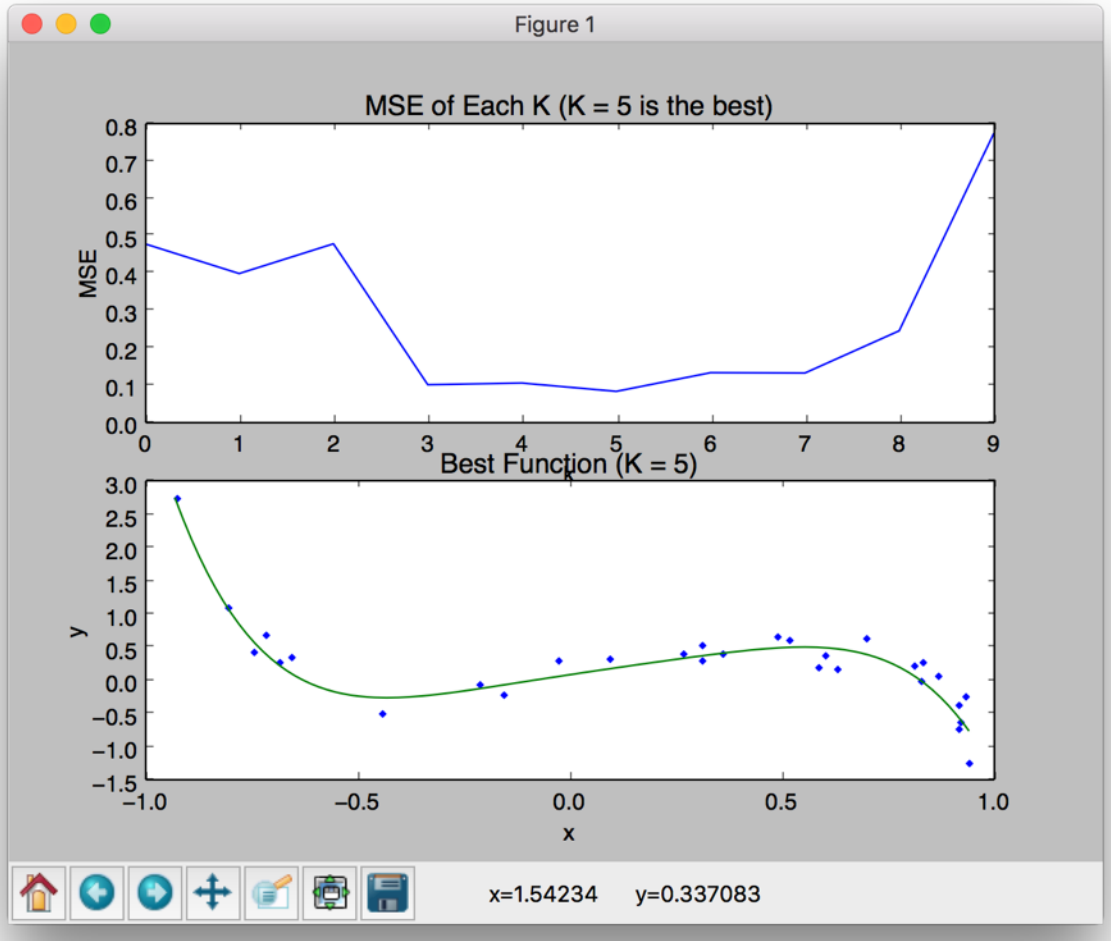
Xinyi Chen

Problem 1

A)

I choose n = 10 experientially. Though higher n can give more samples to estimate a better regression(less bias), but higher variance and higher running time. So 90% of data set to be training set is reasonable and the bias can still be small.

Please run python *nfoldpolyfit.py* <csvfile> <maxdegree> <numberoffolds> <verbose>. E.g. python *nfoldpolyfit.py linearreg.csv* 9 10 1

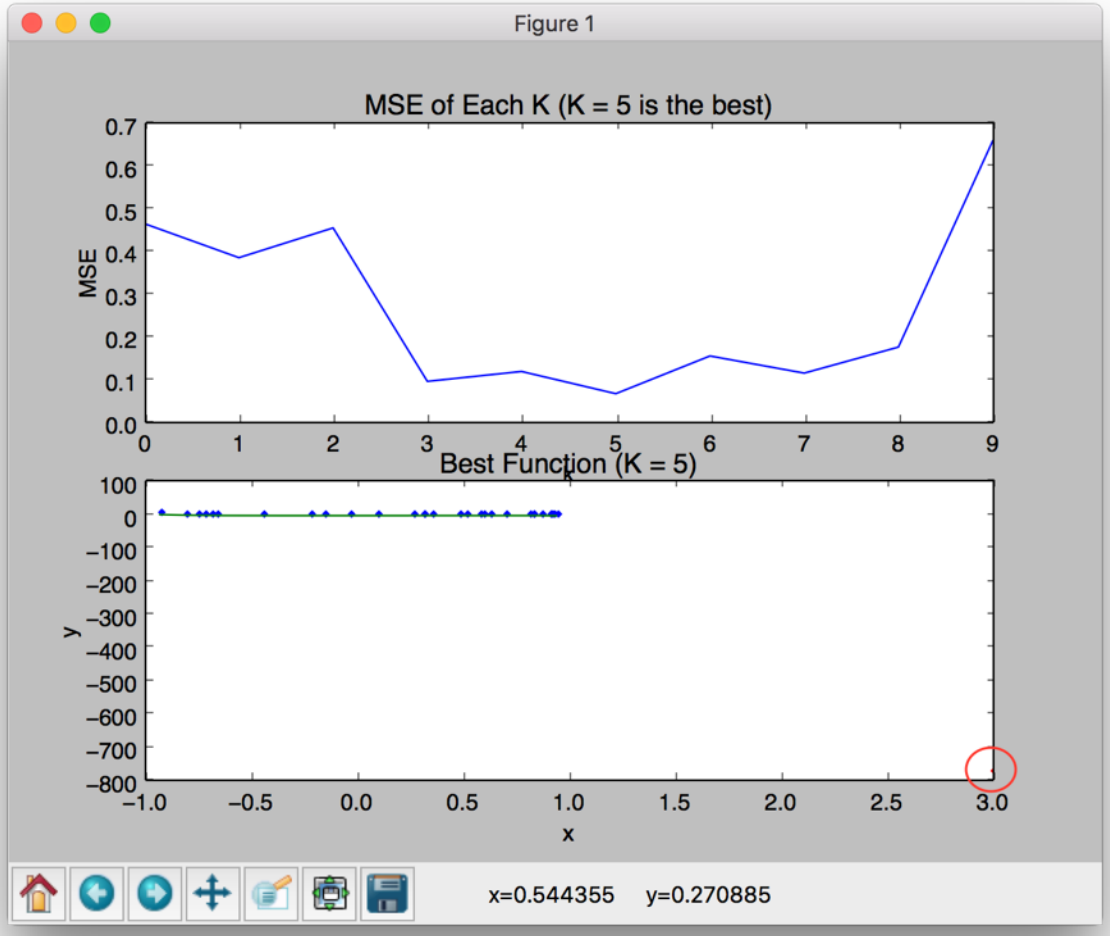


B)

k = 5 is the best. Because all the tests shows that best k that has the lowest mean square error is mostly between 5 to7, and in most cases it is 5. Also to avoid overfit, we should choose 5 not 7.

C)

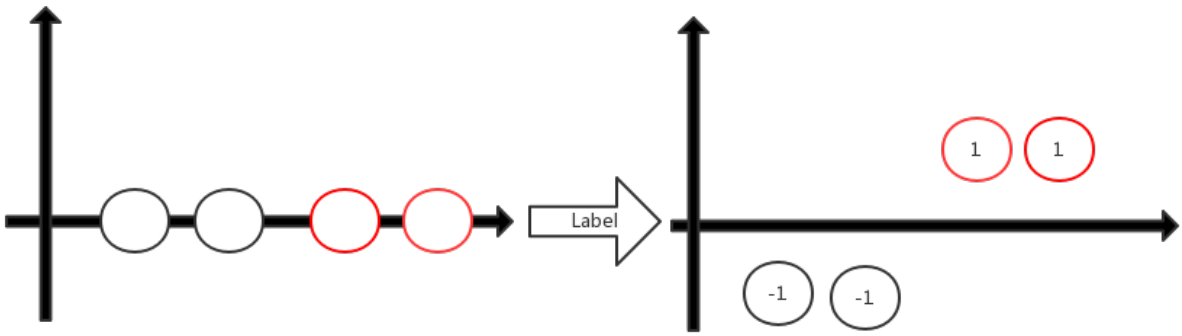
I don't think it is an accurate prediction. As we can see the data point x= 3 is in the red circle. This data point is too far away from others, so I don't think it is reasonable. It is because all of the training data X are between 1 to -1, so the program cannot learn the situation around x = 3, so it cannot give the reasonable prediction for x = 3.



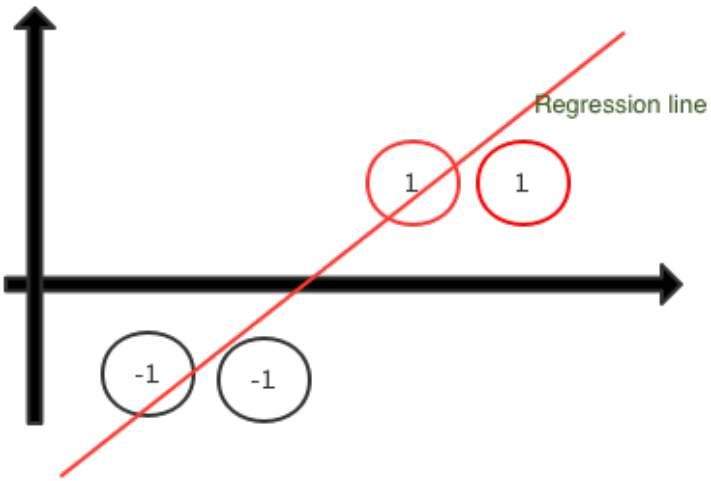
Problem 2

A)

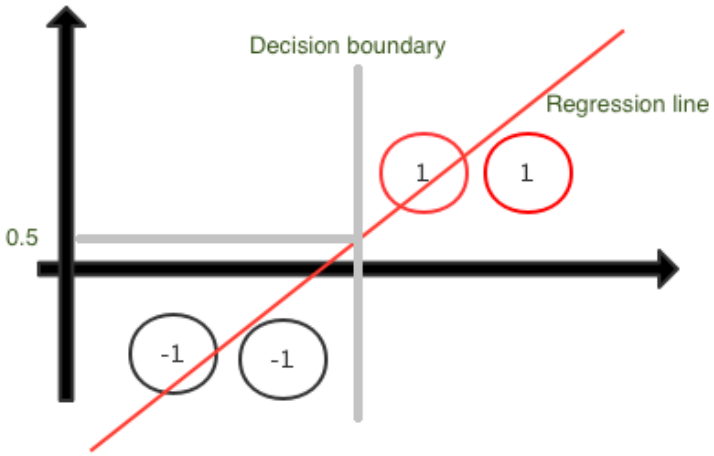
1. Label the data by number. E.g. There's some patients in the left, black circles are patients with malignant tumor, red circles are the patients with malignant tumor. Then label them with {-1,1}, as you can see in the right, they are labeled.



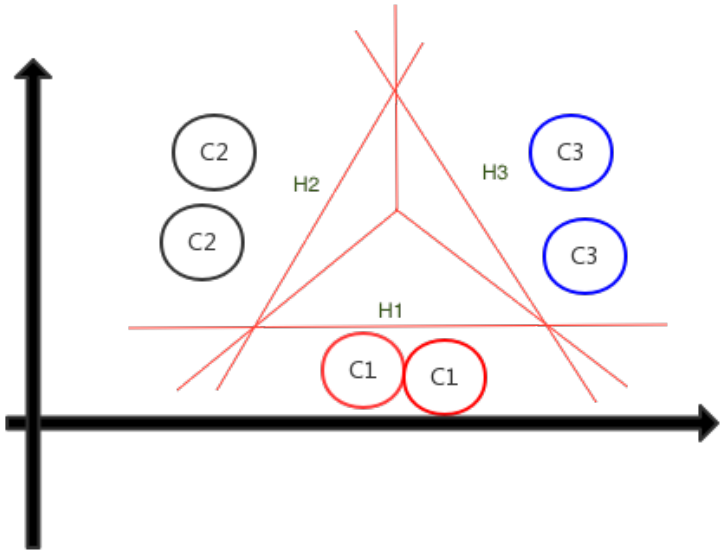
2. Derive a regression line(the red line) by linear regression, minimizing sum of squared residuals.



3. Choose a threshold (e.g. threshold = midpoint = 0.5) to decide a decision boundary. Since our data is one dimensional, so the decision boundary is vertical to X axis(grey line vertical to X axis).

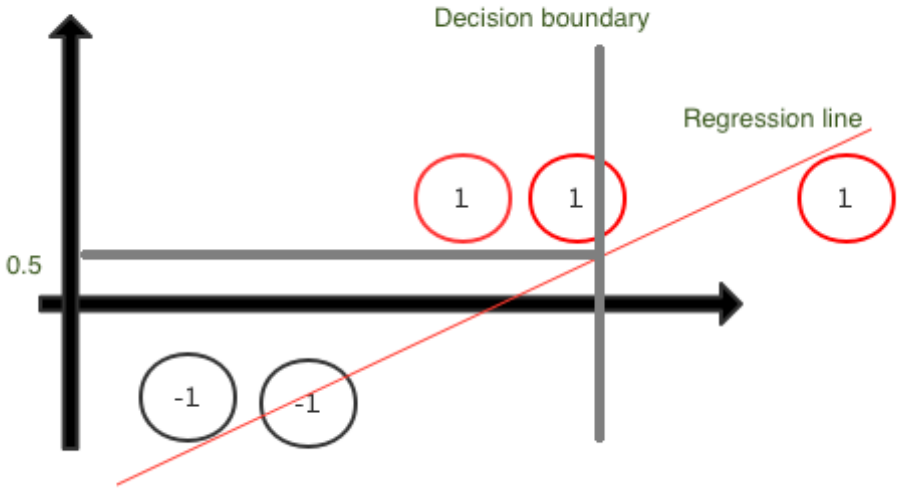


4. If N(number of classes) > 2, we can use N discriminant functions. We should choose the class with the maximum output, geometrically divides feature space into N convex decision regions. E.g. There are three classes, so we use three discriminant functions. For data point x, if g2(x) is the maximum output comparing with g1(x) and g3(x), we classified x as class 2.



B)

If there's some data points that is far away from the most, the regression line will be affected, due to the fixed threshold we cannot pick a decision boundary to classify data well. For example:



Problem 3

A)

One or more classes can be masked by others because of the rigid nature of the regression model, in this situation LDA will do better job than classification via regression.

B)

LDA assumes that we model each class density as multivariate Gaussian and the classes have a common covariance matrix.

C)

Compared to LDA, in QDA classes have different covariance matrixes. The downsides of relaxing this assumption is that separate covariance matrices must be estimated for each class in QDA, this means a dramatic increase in parameters when p in large, so QDA will take more runtime. The upside is that QDA can deal with polynomial situation, but LDA can only deal with linear separable situation if didn't do transformation.

D)

Because LDA can only deal with linear separable situation, we should do a transform Φ (like a kernel trick), for example maybe we can transform space from w0+w1*x1 to w0+w1*x1+w2*w3 just like what we did in regression, because data will be linear separable in higher dimensional space. In this way we can find a decision surface that could be modeled with a polynomial using LDA instead of QDA.

Problem 4

A)

Please run python *perceptrona.py* <csvfile>. E.g. python *perceptrona.py linearclass.csv*

My output (w,e), w is an array contains polynomial coefficients in descending powers.

B)

The program cannot return any result, because it runs into an infinite loop(can never find a line that separates two classes), since X2 is linearly non-separable and perceptron algorithm can only deal with linear separable datasets.

C)

I've transformed the perceptrona function from 1 degree space into 2 degrees space, w_init changed from np.array([0.0, 0.0]) to np.array([0.0, 0.0, 0.0]), and every time when x is not classified, w = w + w0 × x^2 + w1 × x + w2. (w contains polynomial coefficients in descending powers.). Because data will be linear separable in higher dimensional space.

Please run python perceptronc.py <csvfile>. E.g. python *perceptronc.py linearclass.csv*

My output (w,e), w is an array contains polynomial coefficients in descending powers.