Homework 3, Marc Gyongyosi, Net ID: mtg035

Problem 1:

a) The function is implemented in the included python file. From testing, I determine that setting n to 10 is ideal for minimizing the MSE (mean squared error). The mean square error is minimized for a first order approximation, it is then ~0.10 on the training set. By setting n to 10, the training set is of size 3 and the testing set of size 27. This seems to be a reasonable choice for splitting the data. The required plot is shown upon completion of the program if verbose = 1.

b) In all experiments, the linear regression program returned k = 1 as the approximation with the least mean squared error. Looking at the data set and different fits over the data for all the different orders in k = {0,1…,8,9} it becomes clear that for the provided data, the variance increases a lot around the higher order polynomials. This is because the linear regression fit might be roughly for the data, but it's not consistently "wrong" on it. In other words, looking at the plots for higher order polynomials, one can see that the fit looks like the data and resembles the overall shape of it (low bias), but that the individual training points are neither consistently over, nor under the estimated values, but rather mixed (high variance).

c) If we use x = 3, then we have to be very careful with our prediction. X = 3 is far outside any of the provided training data. While our 10-fold cross validation fit might be very good on the data provided (which is roughly in the range -1 to 1), we can't guarantee that it will be very accurate with data from outside that range. If we were to blindly apply our function to x = 3 then the value y for this x would be ~4.36.

Problem 2:

a) Classification via regression: Let's assume we have a classification problem which groups people into the groups {rich, not rich} based on the number of years they studied. Initially, a linear regression fit (yellow) for the data could look like Figure 1. We could determine that if $h(x) > 0.5 \rightarrow x$ *is rich, else x is not rich*.

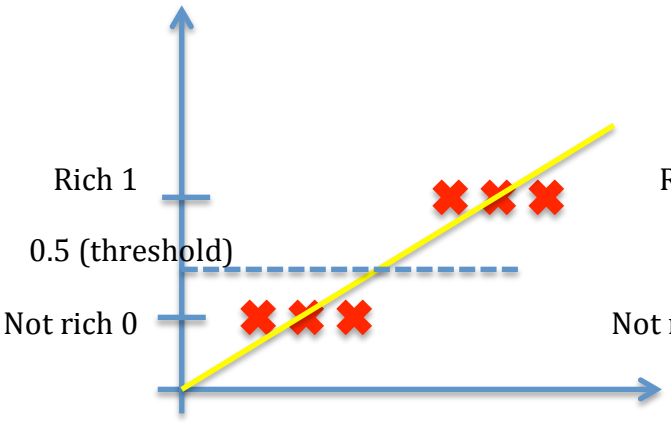


Figure 1

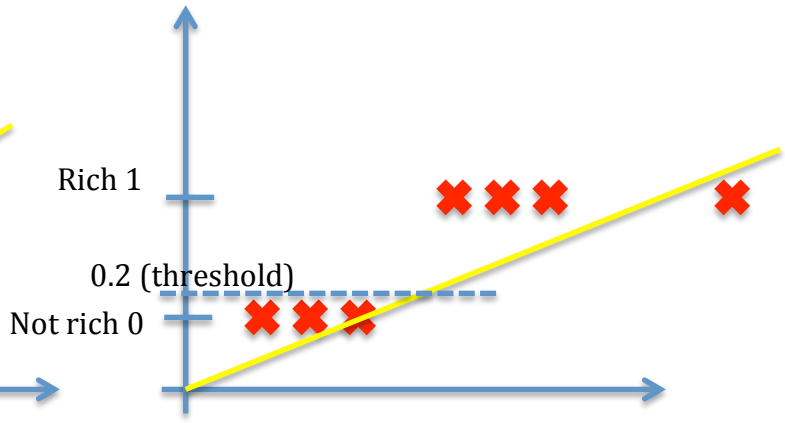


Figure 2

b) For the first case, the linear regression classification works properly. However, let's assume somebody is from Germany or Austria where people usually spend more time at University (Figure 2). Suddenly, the

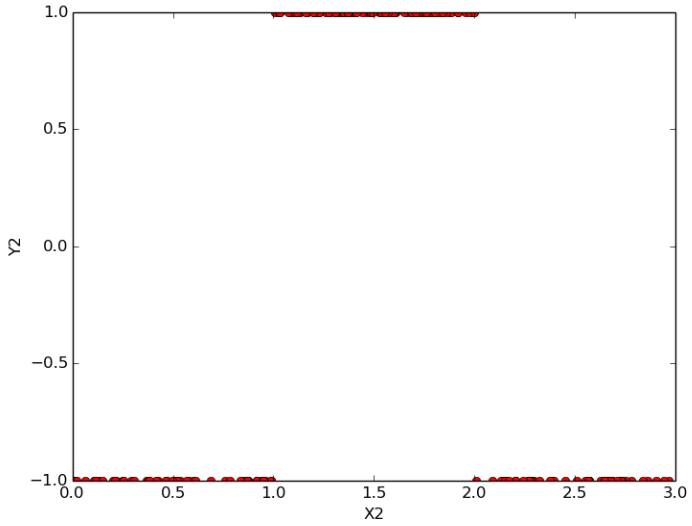
linear regression doesn't seem to classify everything properly anymore if we use *h(x) > 0.5 → x is rich, else x is not rich.* We would have to shift that limit down to something around maybe 0.3. As we introduce more data on either side of the spectrum, the linear regression classification boundary will have to shift around dynamically. This is one of the major weaknesses of classification via linear regression. For this kind of question logistic regression makes more sense.

Problem 3:

a) The main problem with using linear regression on multi-class problems is that classes that are in between others and have a relatively low variance will get "underclassified" with linear regression. For example, imagine a dataset where there are 3 classes, two with high variance and a third one squeezed in the middle. The middle class, since it has low variance will only be classified if the linear regression returns a value from within that relatively small range of the low variance middle class. Thus, it will be very inaccurate. LDA assumes the same variance across all classes and is thus better for multi-class problems.
b) LDA assumes that the independent variables are normally distributed. LDA also assumes that the class covariances are identical and that they have full rank. Only then can it guarantee a solution.
c) QDA does not assume that the covariance of the classes is equal. Compare to LDA, QDA has many parameters to estimate and as such is less accurate. However, QDA is more flexible in that it can have quadratic decision boundaries and as such better
d) In order to use LDA to find a decision surface modeled as a polynomial, you would do what I also did in Problem 4c: a kernel transformation, which raises everything to a higher dimension until linear separation is possible. One could (for example) augment the example vector x by x^2, x^3, … x^n. Then, using LDA, one can find the weights w on this polynomial to model the decision surface.

Problem 4:

a) The perceptron algorithm is implemented and commented in the provided source file.
b) The first perceptron algorithm cannot take the X2 data and produce a proper classification line (it runs forever). This is because the data does not allow for a linear regression classification. When plotting the data points, it becomes apparent why:



The data distribution for the the first order linear regression classification won't allow to classify this data since there can be no linear line of form a*x+b fitted in between all of the data points. See c on how to fix this.

c) It is possible to find a proper solution when augmenting w by another dimension, as well as augmenting the example vector which before was {1, x} to {1, x, x^2}. The perceptron is able to find a solution once modified this way.