**Problem 1**

Question 1

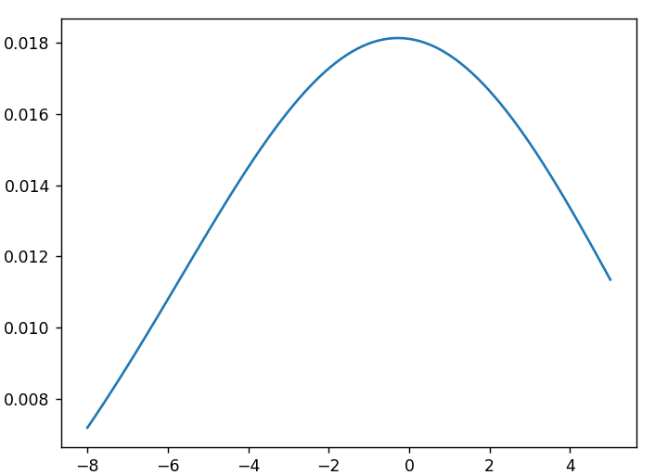
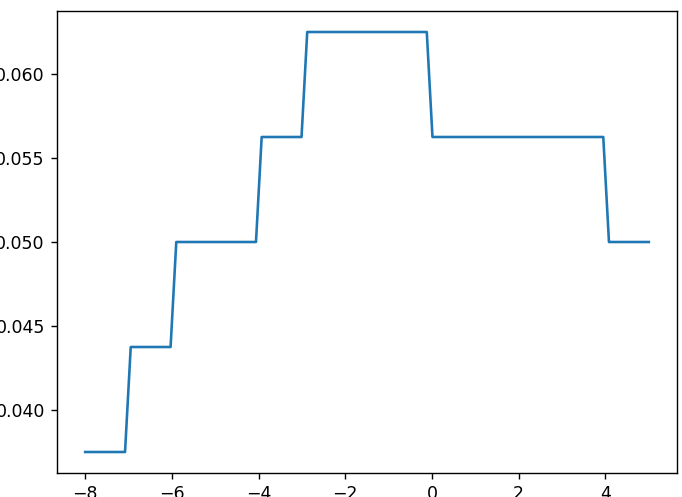
Notice that the risk is:

Substituting 0-1 loss function we get:

We want to minimize the risk, which means to maximize the probability . Notice that:

Notice that when , , and therefore we're simply counting the number of votes of each class.

Question 2

Here's a drawing of the probability estimation:

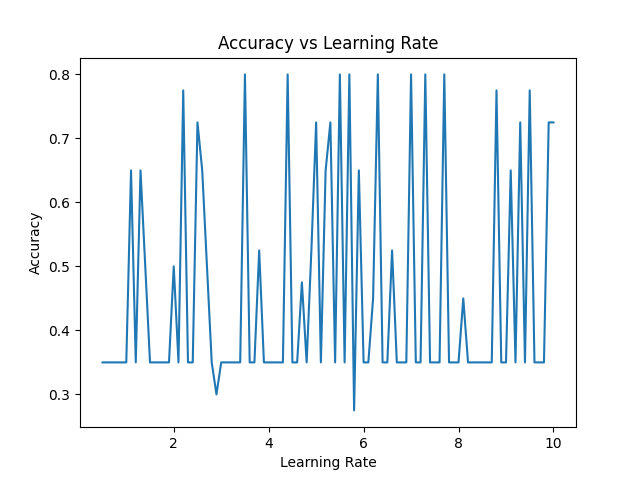
We assume the distribution is gaussian. Here are the MLE estimators:

**Problem 3**

We will be doing logistic regression. For that purpose, we define a class which has 3 attributes: learning rate (etta), convergence constant, and LDF vector (w).

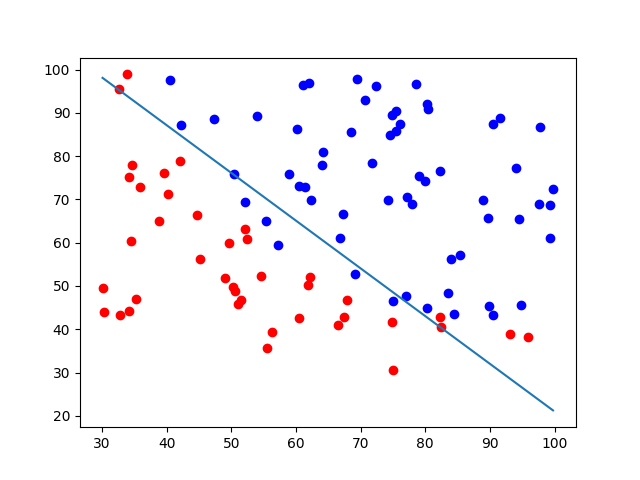
Then, we load the data from the CSV file, and split it into 90-10 train-test sets in the function train. Then, we iterate 9,000 times with gradient descent. This gives us a value for w, which we use to test our model with the function test, which uses a vectorized function of the function classify, then calculates the mean.

This whole thing is done within the scope of the function 'evaluate', which tests the model epoch\_num times, and returns the average accuracy rate.

I split the train set into validation and training set (0.9-0.1), and ran over different values. This is the average validation accuracy rate as a function of the learning rate:

As noisy as it is, the optimal accuracy is 80%, which can be accomplished for different values. The program returned 3.5, and so it is the learning rate in the code.

Note I noticed that gradient descent gets crazy when using a constant learning rate (overshoots), so I divided by 2 each time we progress, so that we progress more cautiously.

Average accuracy rate is 80%, using epochs. Here’s the best separating line: