**Problem 1**

Question 1

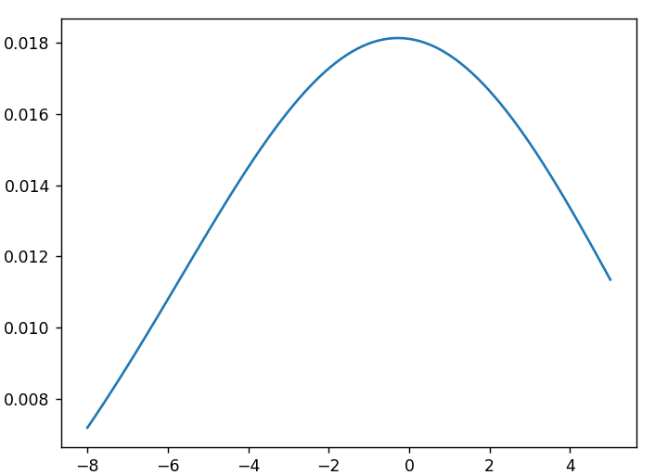
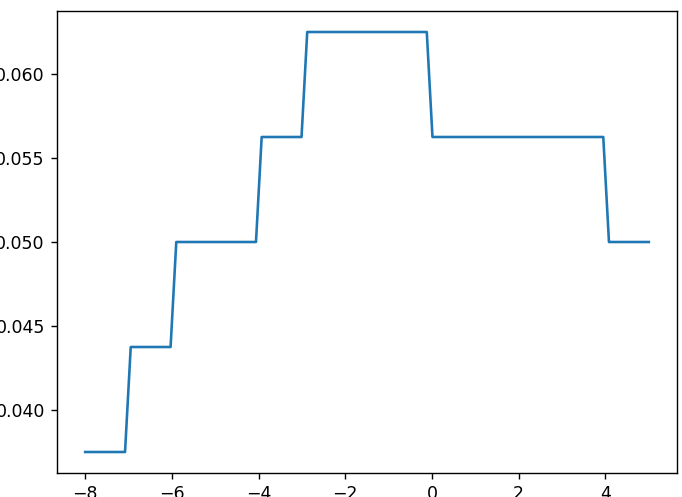
Notice that the risk is:

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, and choosing the one with the maximal votes.

Question 2

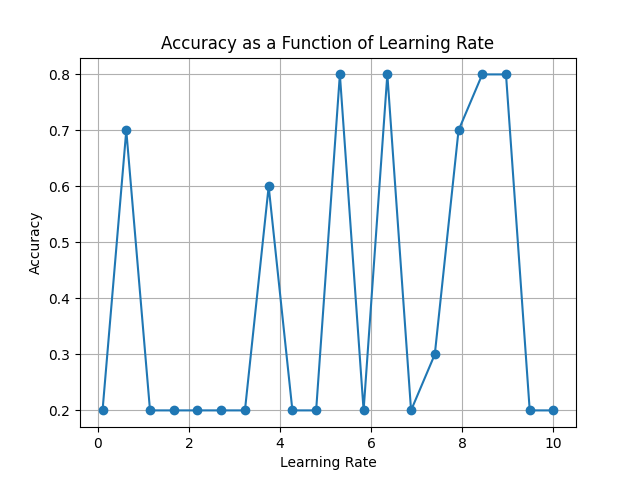
Here's a drawing of the probability estimation:

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

**Problem 3**

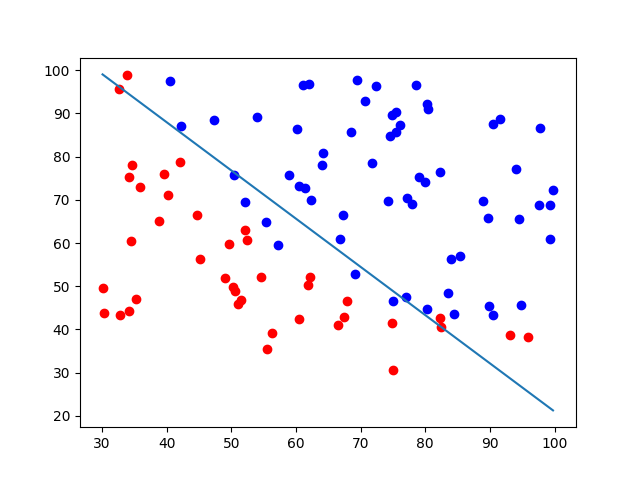
We will be doing logistic regression. For that purpose, we define a class which has 3 attributes: learning rate (etta), convergence constant, and LDF vector (w).

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 28,000 iterations (or until convergance) 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 90%, which can be accomplished for different values. The program returned 6.35, and so it is the learning rate in the code.

Average running time is 40 seconds.

Average accuracy rate is 90%. Data is not linearly separable. Best classifier for this dataset is some special kernel with SVM. Here’s the best separating line:

**Problem 5**

We run for each dataset.

The first one is obviously not RBF(not radical), but could be separated by linear or polynomial kernels with 95% accuracy. The arguments are: .

The second one is almost linearly separable, and so the best kernel is linear with 90% accuracy. The arguments are: .

The third one looks radical, so using RBF is reasonable, and gives 92.5% accuracy, which is good. The arguments are:

Last but not least, the forth one, obviously not linear, or radical, so by trying polynomial kernel we can get to 96.55% with the parameters: .

Here are all 4 separations aligned in one picture:

