Name: Chandramohan Natarajan

Student number: 190617866

Assignment number: 2

Module: ECS766

Exercise 0: What are the new correctly classified instances, and the new confusion matrix? Briefly explain the reason behind your observation. [1 mark]

When KNN gets assigned with k=999 we get 700 correctly classified instances, that is 70% accuracy. The corresponding confusion matrix is as shown below. It shows that all of the instances were classified good. KNN counts the neighbouring samples and their classes and assigns the sample top the majority class. In the data provided, the original dataset has 1000 instances with 700 good and 300 bad samples. When we select k=999, every sample will be labelled as good, due to its majority. None of the instances are classified as bad here, because of a simple lack of majority. So, it is imperative to pick an optimum value for k.

When K= 999

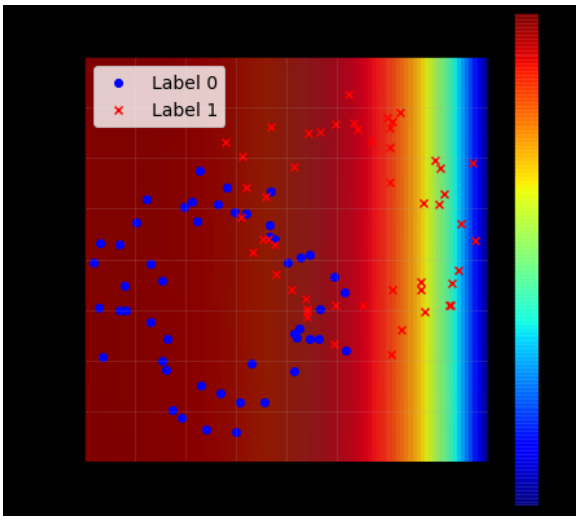
a	b	<-- classified as
700	0	a = good
300	0	b = bad

When K=1

a	b	<-- classified as
567	133	a = good
147	153	b = bad

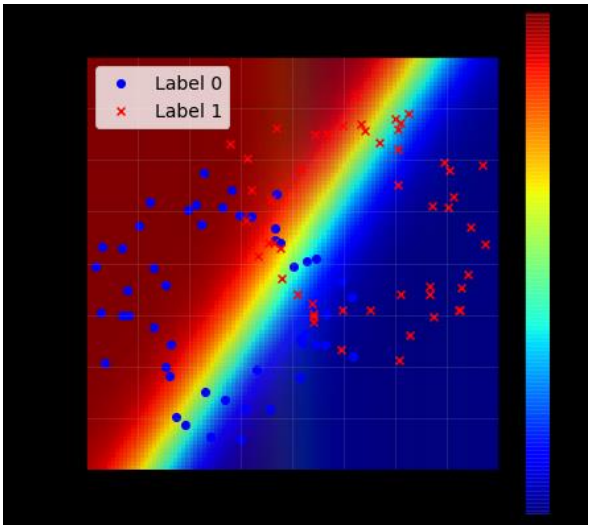
Exercise 1: Try (by varying the values of weights in the code), and describe the effect of the following actions on the classifier (with a brief explanation): [0.5 mark]

- changing the first weight dimension, i.e., w0



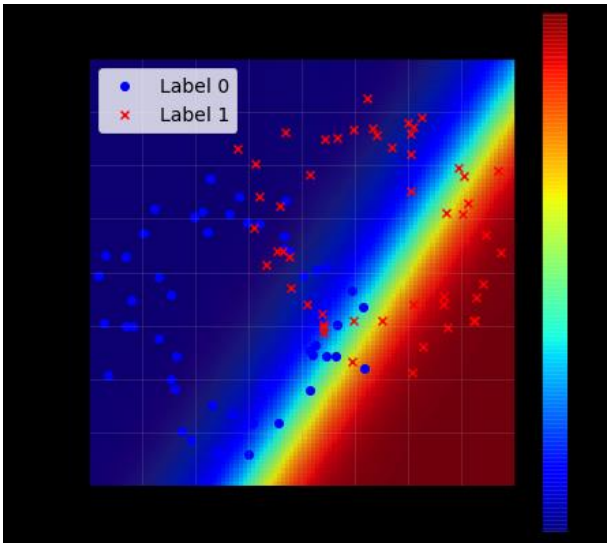
When we change w0 , a proportional difference can be seen in the probability region of a sample belonging to either label 0 or label 1. If we increase w0, the probability region of the sample being labelled as 0 increases and vice versa. Ie if we decrease w0, the probability region of the sample being labelled as 0 decreases and increases for label 1.

- changing the second two parameters, w1, w2?



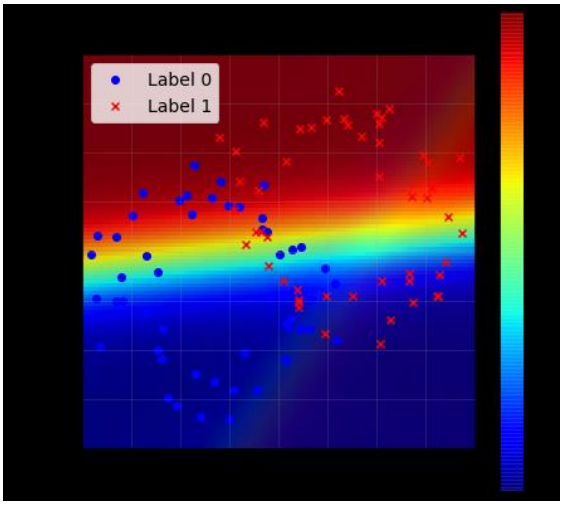
When we change w1 and w2 the angle of intercept for the probability region changes with respect to (0,0)

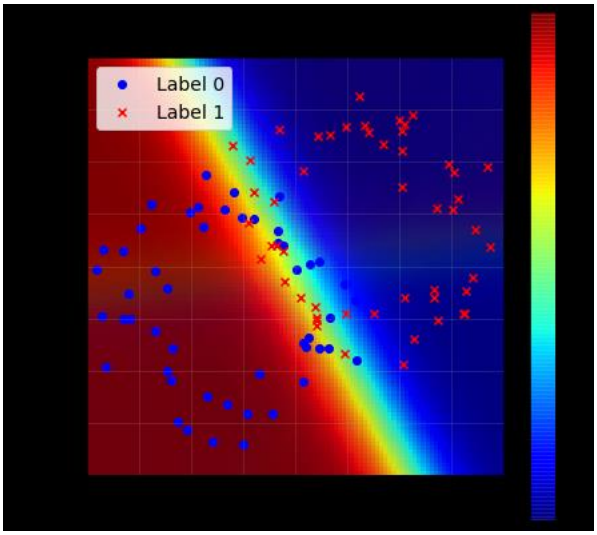
- negating all the weights?



If we negate all the weights, the probability regions are inverted, the blue dots are in blue region and red cross in red region.

- scaling the the second two weights up or down (e.g. dividing or multiplying both w_1 and w_2 by 10)?





When we scale up w1 and w2 and w2 by a factor of 10, the two probability regions red and blue become linearly separable with a plain line. On the contrary , if we scale down the weights, the intermediate probability regions start expanding,I.e. with values0.2,0.4,0.6 etc. Thus it becomes difficult to predict to which class a sample belongs to.

Exercise 1: (a) How many times did your loop execute? (b) Report your classifier weights that first get you at least 75% train accuracy. [0.25 + 0.25 = 0.5 mark]

a) The loop in the code executes 8000 times

b) The weights that first get atleast 75% accuracy are [-0.579,1.000,0.789]

Exercise 2: (a) Notice that we used the negative of the gradient to update the weights. Explain why. (b) Given the gradient at the new point, determine (in terms of "increase" or "decrease", no need to give values), what we should do to each of w_0, w_1 and w_2 to further improve the loss function. [0.5 + 0.5 = 1 mark]

a) We are using the negative of gradient to update the weights because we are calculating the negative log likelihood i.e. -np.matmul(X.T, Xprob) + 2.* LAMBDA * w, to determine the direction for the gradient descent to obtain the global minima.

```
[gradient_neg_log_likelihood = -
np.matmul(X.T, Xprob) + 2. * LAMBDA * w]
```

When we try with positive function, (np.matmul)we do witness drastic downtrend in Train accuracy.

b) Considering that we have to move in the opposite direction of the gradient, w0 should be decreased, while w1 and w2 should be increased in order to improve the current loss function.

Exercise 3: Suppose we wanted to turn this step into an algorithm that sequentially takes such steps to get to the optimal solution. What is the effect of "step-size"? Your answer should be in terms of the trade-offs, i.e., pros and cos, in choosing a big value for step-size versus a small value for step-size. [0.25 + 0.25 = 0.5 mark]

Step size plays the prime role as controlling factor in Gradient Descent, and determines the rate of movement for the gradient. The real goal behind the gradient is to reach the global minima.It is the point with minimum error which is the primary objective. Step size determines how fast or slow we move towards that minima.

- ➔ For large step size , the process is quick and requires less resources but there is a high cost of overshooting of the desired minima.
- ➔ If the step size is too small, then we ought to get accurate answer. But this becomes time consuming, as calculation part becomes more.

Exercise 4: Answer this question either by doing a bit of research, or looking at some source codes, or thinking about them yourself: (a) Argue why choosing a constant value for `step_size`, whether big or small, is not a good idea. A better idea should be to change the value of step-size along the way. Explain whether the general rule should be to decrease the value of step-size or increase it. (b) Come up with the stopping rule of the gradient descent algorithm which was described from a high level at the start of this section. That is, come up with (at least two) rules that if either one of them is

reached, the algorithm should stop and report the results. [0.5 + 0.5 = 1 mark]

A) It is not wise idea to have a constant value for the step size as gradient descent tries to minimise the loss function and at each step it optimizes the function towards best output. In the case of constant step size, we may not arrive at best desired output. So we have to decrease the step size over the computation to prevent the values not to skip optimum minima.

B) It is a good option to keep the iterations as found in the loop to be finite to keep tab on results. So the algorithm pause at right step to output desired optimized result.

   This can be followed as a rule to have controlled algorithm and well manipulated error values.

Exercise 5: Provide one advantage and one disadvantage of using exhaustive search for optimisation, especially in the context of training a model in machine learning. [0.25+0.25 = 0.5 mark]

Advantage → Looks bit simple and widely applicable for search-oriented problems with pretty good accuracy,

Disadvantage → When we think about handling capacity of exhaustive search model, it is not very convincing at this stage of understanding. Because it looks time consuming and resource consuming process, and possibility of more cost involving.

Exercise 6: (a) What is the best value for K? Explain how you got to this solution. (b) Interpret the main trends in the plot. In particular, specify which parts correspond to under-fitting and which part with over-fitting. [0.5+0.5 = 1 mark]

a) The best value for k is 6 as seen from the graph.
   The error looks minimum at k=6 after which validation error rises.

b) When K is minimum we can witness overfitting because closest point to any data is itself. Once it reaches minima, it steadily increases . This shows the elbow trend method. And underfitting is evident from rising values of K and curve looks overfitting well before the validation error reaches its minima,

Exercise 7: Given what you learned from this entire lab(!), provide one advantage and one disadvantage of using MaxEnt vs KNN. [0.25+0.25 = 0.5 mark]

MaxEnt advantage over KNN → Looks good when training data is large.

MaxEnt disadvantage over KNN → Not a good option to arrive at optimum solution for Non linear problems.