AUC Course: Machine Learning

Written Assignment 4

Simon Fang
simon.fang@student.auc.nl
10898492

Exercise 1:

Part (a)

1-Nearest Neighbour

In figure 1, you will see the voronoi diagram using the 1-nearest neighbour algorithm, where the shaded light gray area is classified as the green class, i.e., the class of 1, and the unshaded area is the red class, i.e, class of 0. We calculated the areas by computing the boundaries of the different classes.

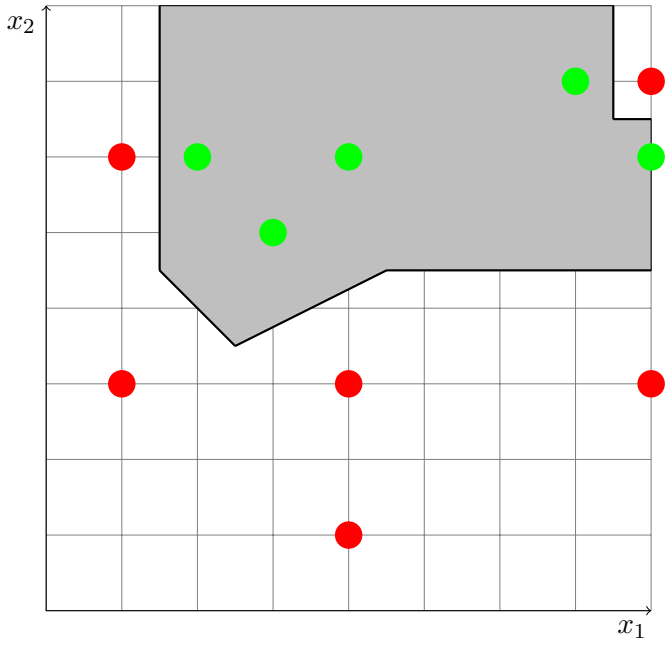
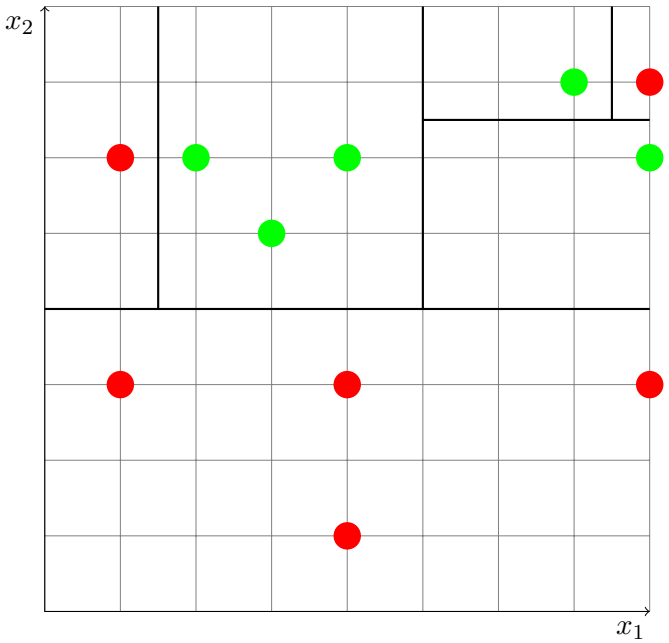


Figure 1: Nearest neighbour voronoi diagram

Decision Tree Learning

Below, you will find the diagram that is found after using the decision tree learning algorithm. We found this diagram by splitting the sets iteratively in subsets, such that we end up with pure subsets. First, we splitted the diagram based on x_2 and then we splitted the subset based on x_1. We end up with 6 subsets.

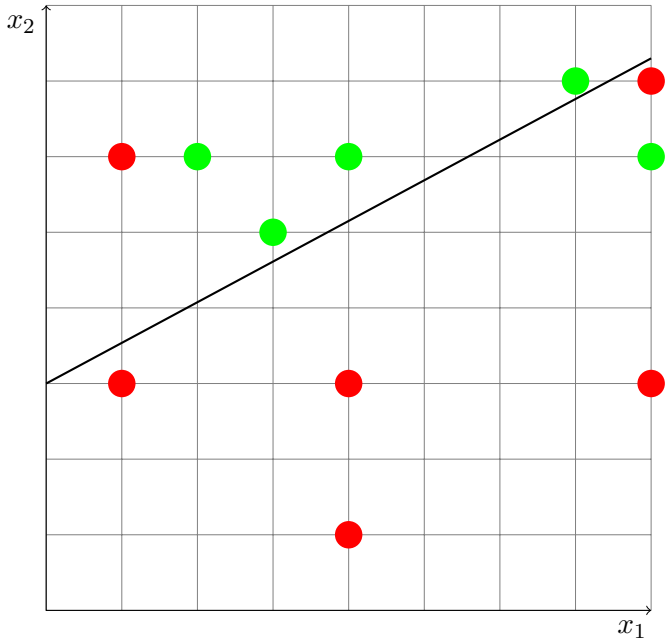
Figure 2: Decision tree learning diagram



Plain logistic regression

For the plain logistic regression, we tried to fit a linear decision boundary through the datapoints such that everything above the decision boundary would be classified as class 1 and everything under the decision boundary as class 0. In our case, it was not possible to fit a straight line such that every data point would be correctly classified. Instead, there is one data point of each class that is misclassified.

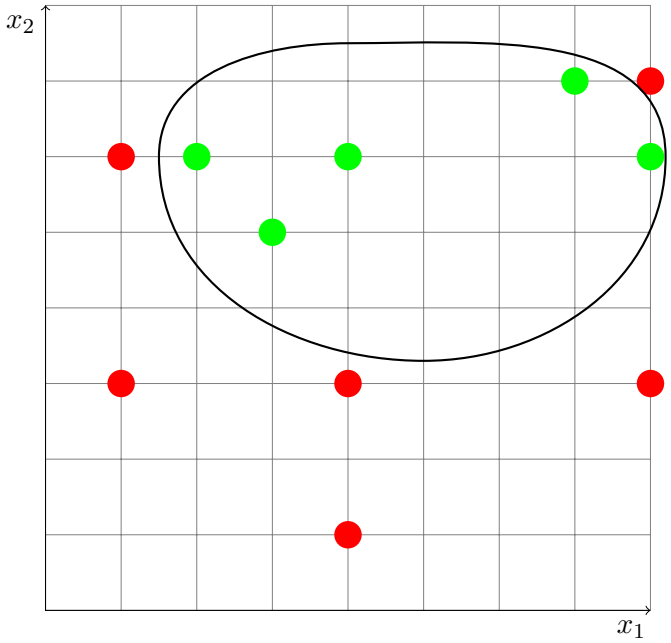
Figure 3: Logistic regression diagram



Quadratic logistic regression

In figure 4, we have attempted to fit a decision boundary with quadratic terms. Our improvement compared the plain logistic regression is that we have only misclassification, i.e., we wrongly classify one class 1 that should have been class 0.

Figure 4: Quadratic Logistic Regression



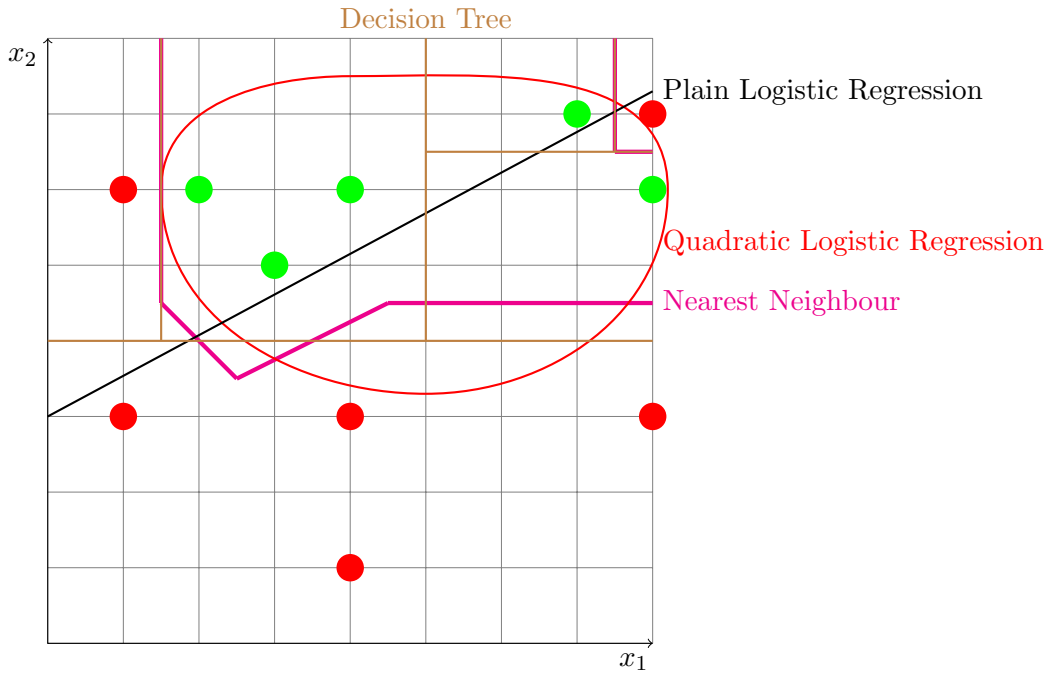
Combined

In figure 5, we have combined all the figures in one plot. This way, we can compare the different classification algorithms and argue which one is better than the other.

Part (b)

When I look at the decision tree learning and nearest neighbour learning algorithms, then I can observe that the algorithms are over-fitting. It is unlikely that datapoint $\{8, 7\}$ has its own subset or voronoi cell, while it is surrounded by class 1 datapoints. Therefore, I believe that either logistic regression algorithms have better boundaries, even though the bias of these boundaries are higher. From my observations, I believe that the logistic regression with quadratic terms has the best boundary, considering it has no misclassifications. Although, also here it is questionable how feasible the boundary is, i.e., perhaps the boundary is over-fitting as well and a boundary with one quadratic term would be better. One possible solution would be to invent a new method that would combine two different learning algorithms or make a weighted boundary such that it combines the best of

Figure 5: All the classification algorithms in one plot



both algorithms, e.g., we could combine for example nearest neighbour and a logistic regression algorithm.

Exercise 2:
We are given the following dataset:

$$\{x^{(1)}, \ldots, x^{(16)}\} = \{1, 2, 3, 3, 4, 5, 5, 7, 10, 11, 13, 14, 15, 17, 20, 21\} \tag{1}$$

and we may assume that there are 3 clusters with initial means 1, 3 and 8. We plot the datapoints from our dataset and the cluster centroid in one graph, where we denote the datapoints as red dots and the cluster centroids as black squares.

Figure 6: Plot of dataset and centroid clusters with mean 1, 3 and 8

We use the following set of formulae in order to compute 1 iteration of the k-means clustering algorithm:

$$\begin{aligned}
&\text{For every } i, \text{ set} \\
&c^{(i)} \equiv \arg\min_j ||x^{(i)} - \mu_j||^2 \\
&\text{For each } j, \text{ set} \\
&\mu_j \equiv \frac{\sum_{i=1}^m 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^m 1\{c^{(i)} = j\}}
\end{aligned}$$

First, we assign every datapoint to a cluster centroid, which is based on the distance between the datapoint and the cluster centroid. In Table 1, you can find the indices of the cluster centroid that each number is assigned to.

Table 1: Each column contains the indices of the cluster centroids that each number is assigned to, which is the minimum argument of the distance between each centroid and point.

Index 1	Index 2	Index 3
$c^{(1)} = 1$	$c^{(3)} = 2$	$c^{(8)} = 3$
$c^{(2)} = 1$	$c^{(4)} = 2$	$c^{(9)} = 3$
	$c^{(5)} = 2$	$c^{(10)} = 3$
	$c^{(6)} = 2$	$c^{(11)} = 3$
	$c^{(7)} = 2$	$c^{(12)} = 3$
		$c^{(13)} = 3$
		$c^{(14)} = 3$
		$c^{(15)} = 3$
		$c^{(16)} = 3$

Having assigned all the datapoints to a cluster centroid, we proceed with the algorithm and we compute the new mean of the cluster centroid. This is essentially equivalent to moving the cluster centroid. In practice, we would continue this algorithm until it either reaches a maximum number of iterations or the centroid 'stops moving', i.e. if the mean of the cluster centroids do not change significantly anymore up to a small $\epsilon > 0$.

$$\mu^{(1)} = \frac{1+2}{2} = 1.5$$
$$\mu^{(2)} = \frac{3+3+4+5+5}{5} = 4$$
$$\mu^{(3)} = \frac{7+10+11+13+14+15+17+20+21}{9} = 14.2$$