Student ID: 201680340

1 Question 1.

The perceptron is used within a supervised learning context to learn the weight and bias parameters that correctly classifies via a hyperplane that forms the linear decision boundary between a positive and negative class.

1.1 Training Procedure

The weights and biases which are initialised with random values. Next, the activation score for the first object is calculated by multiplying each input feature value by its corresponding weight value in the weight vector. These values are summed together along with a bias term to produce an activation score.

In application, the bias is included as the first term of the weight vector and a value of 1 is added as the first input feature in order to allow the activation score to be calculated through dot product multiplication.

$$a = \sum (x_i \times wi_) + b \tag{1}$$

If the activation score is larger than the threshold, the output of the perceptron will be 1, otherwise the output will be -1. This output is multiplied by the training set Y value (equation 2)to which if the signs are different, the product will be negative and a misclassification is detected.

$$y_i \times a < 0 \tag{2}$$

In the case of misclassification, the weight is adjusted according to equation 3 and the bias is adjusted according to equations 4. Geometrically, the vector addition of the input feature vector to the old weight vector rotates the hyperplane, that is perpendicular to the weight vector, in the direction that would classify that object correctly. The bias can be interpreted as moving the hyperplane in axis perpendicular to the weight vector. More adjustments may be necessary before this correct classification occurs.

$$w_i = w_i + (y_i \times x_i) \tag{3}$$

$$b = b + y_i \tag{4}$$

1.2 Testing Procedure

The input features from the objects within the unseen testing set are passed to the perceptron. This input feature vector is multiplied by the weights and bias parameters within the weight vector that was learnt in the training stage. This dot product multiplication gives the activation score.

The activation score is multiplied by the test label as in equation 2. If the product has a negative sign, misclassification has occurred.

1.3 Pesudo-code

Algorithm 1: Perceptron Training Algorithm

```
initialization;
$w_i = R$
 b = R
 for iteration in Maximum Iteration do
    for $(X, y) \in Data_{train}$ do
        $a = W^T X + b$
        if $y \cdot a \leq 0$ then
            $w_i = w_i + y_i \cdot x_i$ for all $i \in d$
            b = b + $y_i$
        end
    end
end
Return $(b, W_1, W_2, W_3, ...W_d)$
```

Algorithm 2: Perceptron Test Algorithm

```
for $(X) \in Data_{test}$ do
    $a = W^T X + b$
end
Return sign(a)
```

2 Question 3)

Table 1: Binary Classifier Accuracies(left) and C2C3 Test Confusion Matrix(right)

	Train	Test
C1C2	99.4%	100%
C2C3	50%	48.7%
C1C3	85%	81.6%

Predicted/ Actual	Positive(C2)	Negative(C3)
Positive(C2)	19	20
Negative(C3)	0	0

The test and train accuracy's are shown in Table 1 to which The class-1, class-2 classifier showed the best performance. The marginally better accuracy for the training set to test set suggest a small amount of overfitting by the class-1, class-3 classifier.

The class-2 and class-3 is the most difficult to separate as revealed by the test and train accuracy's for 'C2C3' being no better than random chance. The lack of perceptron convergence was removed as a possible cause of this after changing the maximum iteration from 20 to 200 had no impact on the accuracy's.

The confusion matrix shows that the trained perceptron was only making positive class label (C2) predictions for the test set. Whilst this behaviour could arise from the order of the objects in the training data meaning all the class-2 objects were seen at the end, the fact the dataset was shuffled removes this is an explanation.

Instead, it is possible that class-2 and class-3 are not linearly separable and as such, an optimal decision boundary determined by the hyperplane for the perceptron could not be learnt. A possible solution for this would to use a step-rate variable or use a non-linear models such as decision trees or Support Vector Machines using the kernel trick.

3 Question 4.

The one-vs-rest approach consists of training a binary for perceptron for each class on a dataset where its class is the positive class label and all other classes have a negative class label. For prediction, the activation score for a given test object of each classifier is compared amongst the classifiers and the highest becomes the prediction label.

Table 2: Multiclass Classifier Accuracy's(left) and OVR Test Confusion Matrix(right)

	Train	Test
OVR	66.3%	67.2%

Predicted/ Actual	C1	C2	C3
C1	19	0	7
C2	0	20	12
C3	0	0	0

The accuracy's are fairly high for both train and test data sets, whilst the lack of significant difference between the two accuracy's signals an absence of overfitting. The confusion matrix in Table 2 reveals the composition of the rest accuracy, showing that the OVR classifier was able to distinguish between class-1 and class-2 with approximately the same amount of accuracy (shown by True Positive rate). It should be noted however that the classifier missclassified objects with class-2 labels more than class-1.

The OVR classifier never predicted Class 3 as a label. This is due to the trained weight vector for the class-3 OVR classifier has its weights outsized by the magnitude of the weights of the class-2 and class-3 OVR classifiers. The resultant activation score of the class-3 OVR classifier is therefore the lowest and does not become the prediction. l2 regularisation could improve this by reducing the difference in weight magnitudes between the classifiers.

4 Question 5.

Table 3: Multiclass Classifier Accuracy's

	0.01	0.1	1	10	100
Train	38.3%	33.3%	33.3%	33.3%	33.3%
Test	37.9%	34.5%	34.5%	32.8%	32.8%

Train and test accuracy's fall to, or close enough to, that of random guessing (33.3%) when a lamda coefficient larger than 0.01 is used. This is because for the larger the lamda coefficient, the more restricted the magnitude of the weight parameters become. Here, the restriction has reduced overfitting to the extent that underfitting has occured. As such, it is possible to conclude that l2 regularisation, even for the lowest coefficient value, has severely effected both train and test accuracy's.