COMP337 - CA Assignment 1
Data Classification
Implementing Perceptron algorithm

Name: Sami Idreesi
Student ID: 201348278

1. **(20 marks)** Explain the Perceptron algorithm for the binary classification case, providing its pseudo code.

   Perceptron algorithm pseudo code:

   PerceptronTrain(D, numFeatures, maxIter)
   1: $\bar{W}$ = zeros(numFeatures)
   2: $b = 0$
   3: for i in range (MaxIter) do
   4: for all $(\bar{X}, y)$ in D do
   5: a = $\bar{W}^T X + b$
   6: if $y.a \leq 0$ then
   7: $\bar{W} = \bar{W} + y \cdot \bar{X}$
   8: $b = b + y$
   9: return $(b, \bar{W})$

   Perceptron algorithm explanation:

   The perceptron algorithm is inspired by a biological model of the human nervous system where the nervous system is a collection of many neurons (nerve cells). These neurons are connected to each other via synapses and learning is done by changing synaptic connections strength between neurons. The strength of the synaptic connections change is response to the external stimuli. Perceptron is a model of a single neuron.

   Firstly, the algorithm takes an input, D: the training dataset, numFeatures: the number of features of each object in D and maxIter: the maximum number of iterations we want to run the algorithm on the D.

   The weight vector $\bar{W}$ {w1,w2, …, wd} is used for the strengths of the synaptic connections for the external stimulus. In line 1 of the algorithm this is initially set as all 0's. It's then updated accordingly when the algorithm makes an incorrect prediction.

   b in line 2 is the bias term (initially set as 0) which is also updated when the algorithm makes an incorrect prediction.

In line 3, the algorithm makes maxIter number of iterations to run the algorithm. Then, in line 4 tries classify each object ($\bar{X}$) which is the incoming external stimulus provided by the training data that holds the synaptic connections/features: {x1,x2, …} to the Perceptron Algorithm (neuron). In line 5, a (the activation score) which is the overall signal the neuron receives from its incoming synaptic connections in object $\bar{X}$ and the current weights in $\bar{W}$ is calculated: a = $\bar{W}^T X + b$. Then in line 6 it checks if the object is classified incorrectly by using the objects actual class label y and then updates the weights in line 7 and bias term in line 8. If the object was classified correctly, the algorithm just moves onto the next training object. Finally in line 9 after MaxIter iterations the algorithm returns the final bias term and weights/strengths of the synaptic connections.

Further remarks on misclassification (line 6): $y.a \leq 0$ in line 6 is able to detect misclassification as the predicted class, sign of the activation score a, is different from the actual class of the current object, y, if and only if $y.a \leq 0$. This works as, if we predicted the class to be +1 and the actual class is -1 $y.a$ will be -1 (negative) and if we predicted the class to -1 and the actual class was +1 $y.a$ will again be -1 (negative). So in the case of misclassification $y.a \leq 0$ always holds. The opposite is also true when we classify instances correctly, e.g. if we predict the class to be +1 and the actual class is +1 and if we predict the class to be -1 and the actual class is -1 $y.a$ will always be +1 (positive) so condition $y.a \leq 0$ doesn't hold and the algorithm decides the current object was predicted correctly and moves onto the next object.

Further remarks on the update rule (lines 7 and 8): There are 2 cases of misclassification for the binary Perceptron: we incorrectly classify a positive object as negative or we misclassify a negative object as positive. In the first case we should have a higher activation score to avoid this, so in line 7 we increase the weight vector by adding $y \cdot \bar{X}$ to it and also adding y to the bias term b in line 8. For the second case we should have a lower activation score as we misclassified a negative instance as positive. So we need to decrease the weight vector and the bias term. We can do the same calculations as the first case because the actual class label y is -1 so this decreases both the weight vector $\bar{W}$ and the bias term y.

Key features of the Perceptron algorithm:

- It is an online algorithm - it processes objects from the training data one at a time as opposed to other methods such as batch learning that requires access to the entire data set.
- It's error driven - the parameters: b and weight vector $\bar{W}$ are only updated when the algorithm makes an incorrect prediction

2. (See code in .py file)

3.

Dataset type	Classes to separate	Accuracy on unrandomized datasets	Accuracy when datasets randomized	Comments
Training data	class 1 and class 2	100%	100%	Perceptron gave 100% accuracy on multiple runs, when the training data was randomized , when discriminating between class 1 and class 2 and also 100% when the data wasn't randomized
Training data	class 2 and class 3	50%	Ranging from 50%-97.5%	Perceptron gave inconsistent accuracies ranging from 50%-97.5% on multiple runs, where the training data was randomized , when discriminating between class 2 and class 3 and 50% when the data wasn't randomized.
Training data	class 1 and class 3	100%	100%	Perceptron gave 100% accuracy on multiple runs, where the training data was randomized , when discriminating between class 1 and class 3 and also 100% when it wasn't randomized.
Test data	class 1 and class 2	100%	100% and sometimes 95%	Perceptron gave 100% accuracy and sometimes 95% on multiple runs, where the training and test data was randomized, when discriminating between class 1 and class 2 and 100% when datasets weren't randomized.
Test data	class 2 and class 3	50%	Mostly in the range 75%-100% but sometimes between 50% and 70%	Perceptron gave inconsistent accuracies ranging from 50%-97.5% on multiple runs, where the training and test data were randomized, when discriminating between class 2 and class 3 and 50% when datasets weren't randomized.
Test data	class 1 and class 3	100%	100%	Perceptron gave 100% accuracy on multiple runs, where the training and test data was randomized , when discriminating between class 1 and class 3

				and 100% when datasets weren't randomized.

Screen shot from output:

```
Q3. Binary Perceptron accuracies: Unrandomised

Training data accuracy: class 1 and class 2:  100.0 %
Training data accuracy: class 2 and class 3:  50.0 %
Training data accuracy: class 1 and class 3:  100.0 %

Test data accuracy: class 1 and class 2:  100.0 %
Test data accuracy: class 2 and class 3:  50.0 %
Test data accuracy: class 1 and class 3:  100.0 %
```

```
Q3. Binary Perceptron accuracies: Randomised

Training data accuracy: class 1 and class 2:  100.0 %
Training data accuracy: class 2 and class 3:  95.0 %
Training data accuracy: class 1 and class 3:  100.0 %

Test data accuracy: class 1 and class 2:  100.0 %
Test data accuracy: class 2 and class 3:  90.0 %
Test data accuracy: class 1 and class 3:  100.0 %
```

The pair of classes that were the most difficult to separate were class 2 and class 3 when calculating accuracies for both test and train classifications. Class 1 and class 2, and class 1 and class 3 both gave 100% accuracy consistently on both randomized and unrandomized datasets. Although class 2 and class 3 gave 100% on some runs with certain randomized datasets (when testing both test and training data) it wasn't to the same consistency as between class 1 and class 2 and between class 1 and class 3.

4.

Dataset type	Accuracy on unrandomized datasets	Accuracy on randomized datasets	Comments
Training data	66.66666666666666 %	Mostly 66.67% but sometimes ranging between 44% and 97%	
Test data	66.66666666666666 %	Mostly 66.67% but sometimes ranging between 33% and 100%	

Screen shot from output:

```
Q4. Multi-class Perceptron accuracies: Unrandomised

Test data accuracy: 66.66666666666666 %

Training data accuracy: 66.66666666666666 %
```

```
Q4. Multi-class Perceptron accuracies: Randomised

Test data accuracy: 66.66666666666666 %

Training data accuracy: 66.66666666666666 %
```

5.

```
Q5. Multi-class Perceptron accuracies using l2 values: Unrandomised

Training data accuracies:
l2 value: 0.01 Accuracy: 66.66666666666666 %
l2 value: 0.1 Accuracy: 66.66666666666666 %
l2 value: 1.0 Accuracy: 33.33333333333333 %
l2 value: 10.0 Accuracy: 33.33333333333333 %
l2 value: 100.0 Accuracy: 33.33333333333333 %
best l2 value: 0.1 Accuracy: 66.66666666666666 %

Test data accuracies:
l2 value: 0.01 Accuracy: 66.66666666666666 %
l2 value: 0.1 Accuracy: 66.66666666666666 %
l2 value: 1.0 Accuracy: 33.33333333333333 %
l2 value: 10.0 Accuracy: 33.33333333333333 %
l2 value: 100.0 Accuracy: 33.33333333333333 %
best l2 value: 0.1 Accuracy: 66.66666666666666 %
```

Train and test classification accuracies gave same accuracies for each l2 term with 0.1 and 0.01 as the best l2 values on unrandomized datasets.

```
Q5. Multi-class Perceptron accuracies using l2 values: Randomised

Training data accuracies:
l2 value:  0.01 Accuracy: 66.66666666666666 %
l2 value:  0.1 Accuracy: 33.33333333333333 %
l2 value:  1.0 Accuracy: 33.33333333333333 %
/Users/Sami/Desktop/Data Mining/CA1data/Perceptron.py:208: RuntimeWarning: overflow encountered in multiply
  weightVector = (1-2*l2term)*weightVector + classLabel*features
l2 value:  10.0 Accuracy: 33.33333333333333 %
l2 value:  100.0 Accuracy: 33.33333333333333 %
best l2 value:  0.01 Accuracy: 66.66666666666666 %

Test data accuracies:
l2 value:  0.01 Accuracy: 66.66666666666666 %
l2 value:  0.1 Accuracy: 33.33333333333333 %
l2 value:  1.0 Accuracy: 33.33333333333333 %
l2 value:  10.0 Accuracy: 33.33333333333333 %
l2 value:  100.0 Accuracy: 33.33333333333333 %
best l2 value:  0.01 Accuracy: 66.66666666666666 %
```

Train and test classification accuracies gave the same accuracies for each l2 term with 0.01 as the best l2 value on randomised datasets.