CA Assignment 1, Data Classification (Implementing Perceptron algorithm)

Saeth Wannasuphoprasit
Student ID: 201585689

Question 1: Explain the Perceptron algorithm for the binary classification case, providing It's pseudocode.

Pseudocode:

```
PerceptronTrain(Training data, MaxIter)
# Initialize weights and bias
weights = vector(0), dimension = numbers of features
bias = 0

# Compute activation score (a)
for iteration = 1,2,...,MaxIter:
        for record (X, y) in Training data:
                a = (weights dot X) + bias

                # update weights and bias (misclassification)
                if y×a <= 0:
                        weights = weights + y X
                        bias = bias + y
return weights, bias

PerceptronTest(weights, bias, X)
# Compute activation score (a)
a = (weights dot X) + bias
return sign(a)
```

Explanation:

The PerceptronTrain algorithm receives
1. training data
    1.1 feature X (x1, x2,..., xn)
    1.2 classes y (possitive: 1, negative: -1)
2. MaxIter (numbers of iterations).

Firstly, weights and bias are set to 0. Then, the algorithm iterates for MaxIter times, each time the features in the record of the training dataset (X) are used to calculate the activation score (a) by dot product of X with weights vector and then plus the bias value. If the value of y×a is greater than 0, the model classifies the target record correctly. However, if the value of y×a is less than or equal to 0, the model adjusts their weights and bias with the above equations in the # update weights and bias (misclassification) section. Finally, the latest weights and bias are the output of the algorithm.

The PerceptronTest algorithm receives weights vector, bias, and the test data X (features: x1, x2, …, xn). Then, it calculates the activation score (a) with the same equation as PerceptronTrain. Finally, the output is determined by sign(a) which is 1 if a > 0 and -1 if a < = 0.

Question 2: Implement a binary perceptron

The binary perceptron algorithm is implemented in the python file (PerceptronTrain function).

Question 3: Use the binary perceptron to train classifiers to discriminate between class 1 and class 2, class 2 and class 3, and class 1 and class 3.
Report the train and test classification accuracies for each of the three classifiers after training for 20 iterations. Which pair of classes is most difficult to separate?

The code is given in the python file (# Question 3 section) and the results are shown below.

```
1 vs 2
train accuracy:  1.0
test accuracy:  1.0
```

```
2 vs 3
train accuracy:  0.775
test accuracy:  0.75
```

```
1 vs 3
train accuracy:  1.0
test accuracy:  1.0
```

So, according to the train and test accuracy, class 2 vs class 3 is the most difficult one to separate because it has the lowest accuracy in both train and test (0.775, 0.75) compared to the other two (1.0, 1.0).

Question 4: Extend the binary perceptron that you implemented in part 3 above to perform multi-class classification using the 1-vs-rest approach. Report the train and test classification accuracies after training for 20 iterations.

The code is given in the python file (# Question4: 1 vs the rest approach section) and the results are shown below.

```
Q4: 1 vs the rest method
train accuracy:  0.6833333333333333
test accuracy:  0.6333333333333333
```

Question 5: Add an L2 regularisation term to your multi-class classifier implemented in part 4. Set the regularisation coefficient to 0.01, 0.1, 1.0, 10.0, 100.0 and compare the train and test classification accuracies.

The code is given in the python file (# Question 5: L2 regularization section) and the results are shown below.

```
lambda =  0.01
train accuracy:  0.9416666666666667
test accuracy:  1.0
```

```
lambda =  0.1
train accuracy:  0.6666666666666666
test accuracy:  0.6666666666666666
```

```
lambda =  1
train accuracy:  0.3333333333333333
test accuracy:  0.3333333333333333
```

```
lambda =  10
train accuracy:  0.3333333333333333
test accuracy:  0.3333333333333333
```

```
lambda =  100
train accuracy:  0.3333333333333333
test accuracy:  0.3333333333333333
```