Data Mining and Visualization – Assignment 1 – Report (201682772)

1.

The Perceptron algorithm is an easy method for predicting which of two classes an input belongs to in binary classification problems. It tries to find the linear decision boundary to separate the two classes.

Training Data: Set the bias term and the weights vector w to zero during initialization.

The training examples (x, y), where x is the input vector and y are the desired output (either +1 or -1) should be iterated over. Calculate the output A by adding the bias term b to the dot product of the weights vector w and the input vector x. Predict A as +1 if it is larger than or equal to zero, else consider A as -1. Update the bias term b and weights vector w in accordance with the following principle: Do nothing if y and A have the same sign. Update the weights vector w and bias term b as follows if y and A have different signs (or) find the dot product between y and A if the computed value is less than or equal to zero, then update the weights vector w and bias term b as follows if y and A have different signs (or) find the dot product between y and A if the computed value is less than or equal to zero. Weights = weights + y * x, Bias = bias + y

Repeat until all the training data are correctly classified (or) continue till it reaches the given number of iterations.

Testing data: Given a new input vector x, compute the output A by taking the dot product between the weights vector w and the input vector x and adding the bias term b. Predict A as +1 if it is larger than or equal to zero, else anticipate A as -1. Predict A as +1 if it is larger than or equal to zero, else anticipate A as -1. For binary classification problems, the Perceptron method is a straightforward and effective approach, but the algorithm does not give an optimal solution when two classes are not linearly separable.

PSEUDO CODE FOR TRAINING:

X: training (input vectors) and Y: Target output labels (+1 or -1) and Iterations: Maximum number of iterations to run the algorithm.

Initialization: weights = [0, 0, ..., 0] (vector of zeros with the same dimension as input vectors) and bias = 0 and Iterations = 20.

```
for i in range(iterations):
    for count, values in enumerate(x):
        A = (weights. x) + bias
         if A * y <= 0:
         weights = weights + y * x
        bias = bias + y
```

Returns (weights, bias)

PSEUDO CODE FOR TESTING:

X (input), weights and bias (from the values we get from the training data)

A = (weights. x) + bias

Returns sign(A)

3.

	ACCURACY FOR TRAIN	ACCURACY FOR TEST
CLASS 1 VS CLASS 2	99.37	100.0
CLASS 2 VS CLASS 3	90.68	92.5
CLASS 1 VS CLASS 3	94.37	89.74

I believe that class 1 versus class 3 is most difficult to differentiate since the accuracy for the train data is high and the accuracy for the test data is low. This could lead to an issue called Overfitting. Data included in these classes may not be linearly separable and these two classes could be challenging for the model to separate compared to the other classes.

4.

	ACCURACY FOR TRAIN	ACCURACY FOR TEST
CLASS 1 VS REST	90.04	93.22
CLASS 2 VS REST	91.70	93.22
CLASS 3 VS REST	66.80	66.10

One vs Rest:

One vs Rest is also termed as 1 vs all. It's a technique which employs binary classification algorithm to conduct multiclass classification. Fundamentally, a multiclass dataset is separated into distinct binary classification tasks. Each binary classification task is trained using binary classification algorithm and then the estimates are obtained.

For example, let's imagine there are three classes such as A, B and C. This might be separated into,

1. Classifier 1: A versus [B and C]
2. Classifier 2: B versus [A and C]
3. Classifier 3: C versus [A and B]

5.

- The best classification accuracy is attained from the Class 1 versus Rest technique. Both the train and test accuracies are 76.76 and 86.44 by utilising Regularization coefficient as 0.01 which is high compared to the other classes.
- There is a progressive drop in the accuracy for all the classes as the Regularization coefficient grows from 0.01 to 100.0. Hence, raising the Regularization coefficient leads to under fitting of data.

- Consequently, I can conclude that the model did well both on training and test data in Class 1 versus Rest method with Regularization coefficient at 0.01.