CSC 635 Data Mining

Assignment 4 Report

Submitted to:

Dr. Jamil Saquer

Author(s):

Sharmin Sultana

**Perceptron Implementation**

**Introduction**

For this assignment, single layer perceptron algorithm has been implemented on the given datasets. There are two datasets provided for this task. One dataset will be used for training and the other one for testing the algorithm. In the datasets, there are three columns: first two columns represent the features, and the third columns is the class label based on the features. The class labels are categorized as either -1 or +1. In the train dataset, there are in total 80 data whereas the test dataset contains 20 data. Both datasets are given in .txt format.

In the Part 1, by using these two datasets, we performed perceptron algorithm until stopping criteria and after that stepwise threshold activation function was used to categorize the testing features. Based on the predicted classes of the test features, accuracy report for the algorithm was also generated.

In part 2, we were asked to plot the train and test dataset by using the scatter plot. For this part, we needed to find out the hyperplane that linearly separates the data based on their class label.

**Background**

Perceptron is a neural network consisting of one neuron that takes inputs, aggregates them (weighted sum), and returns 1 only if the aggregated sum is more than some threshold else returns 0. Here, a short description is given how perceptron algorithm works.

1. At the very first step, generate k random weights equal to (n+1). Here, n is the number of feature columns in the dataset. For example, if one is given a dataset of three (3) features then s/he needs to generate four (4) random weights. The extra weight i.e., weight [0] is used for bias. Bias weight is an adjustable, numerical term used for increasing the classification model accuracy.

For each training sample

1. We need to calculate the weighted sum of the perceptron’s weights and input value. The default input value for bias weight is 1.

[**Equation 1**]

1. In this step, error is calculated by subtracting the values of from the expected class label which is y.

**[Equation 2]**

1. If error is zero (0), do nothing else we need to update next weights based on the following equation:

**[Equation 3]**

Here, defines the learning rate which should be in between [0,1]. Smaller learning rate tends to make the perceptron more stable and noise resistant.

1. Follow steps (II – IV), until stopping condition is fulfilled. Stopping condition means total number of iterations which is also known as epoch. It indicates how long this training process will continue.
2. After certain number of iterations, we need to set a threshold value to predict the class label for the test dataset. Usually step/threshold activation function is used to classify the objects.

**[Equation 4]**

Here,

After training the dataset by using perceptron algorithm, we need to plot the data. To illustrate the linear separability of the datasets, we need to draw hyperplane that separates the positive and negative data in a best possible way. Hyperplane is measured as follows:

1. First, deduce the value of X intercept and Y intercept from the learned weights and bias.

**[Equation 5]**

1. For each training samples, we need to calculate the position of hyperplane by using the formula of [5] and [6].

**[Equation 7]**

**Implementation**

At the beginning, I loaded all the libraries that may require to carry on the task. I used *numpy* for numerical operations, *pandas* to import the given datasets and *matplotlib* to plot the diagram. With the libraries imported, now my environment is ready for implementation.

Since both datasets are in .txt format, I read the features and the class label separately by opening the files, splitting each lines and store these lines in a separate list. This list is then converted to an array for convenient manipulation. Here map function is used to convert the string value of feature into float type.

Logo, company name

Description automatically generated with medium confidence

*figure 1: Storing the data in a separate array*

Same procedure is done for all the features and class label of training\_data and testing\_data. Now we have x1, x2, y that represent feature1, feature2 and class\_label for the dataset training\_data and x\_test1, x\_test2 and y\_test for feature1, feature2 and class\_label of testing\_data.

After having all the feature data and class data, I defined a function named products that calculates the weighted sum by using formula [1] and returns the result for further calculation. As arguments, I pass the features and corresponding weights to do the aggregation.

Then I define another function which calculates the error by using the equation [2]. This function takes the class\_label of the dataset and the outcome of products function as parameters.

In this phase, I define function named **‘perceptron\_train’** that will be used for training the data. The features, learning rate and the stopping condition are passed as parameters in this function. Here, at the beginning I took three random weights because we have two features in both datasets. Then I pass this in a loop that will continue for a certain number of iterations. In the body of the loop, I called the function products and calculate error after that. If error is not zero, the I updated the weights again by using formula [3]. At the end of the iterations, I stored the last learned weights in a variable and the learned bias in another variable.

After training, I called a threshold activation function to assign labels to the testing objects. This labels are stored in an array named y\_pred[]. The label of this is then compared with the actual label and based on the comparison I calculated the accuracy of my algorithm. In my implementation accuracy fluctuates in between 60 – 95% most of the time.

In part 2, I plotted the training\_data and testing\_data after implementation of perceptron algorithm. Here feature1 is placed in x-axis and feature2 is placed in y-axis. After certain number of iterations, I got three weights: first weight represents the last bias weight, and the next two weights represent the updated weights after complete iteration. Now I calculated X intercept and Y intercept. Then I calculate slope and c from minimum(feature1) to maximum (feature2) and plot it. This is the way I calculate the hyperplane. This process is followed to plot both training\_data and testing\_data. For both dataset, if target value is +1 then it is classified as positive and if it is -1, then it is negative class and I separated this by writhing a simple if-else logic.

Text

Description automatically generated with medium confidence

*figure 2: separating positive and negative values*

**Experimental Results**

After training the dataset, accuracy for the perceptron algorithm ranges from 60-90% if I set the learning rate 0.01 and number of iterations 100. Sample output is:

Text

Description automatically generated

*figure 3: Sample output for learning rate 0.01 and number of iterations = 100*

With this learning rate and epoch, the plotted diagram for training and testing dataset look like the below:

Chart, scatter chart

Description automatically generated

*figure 4: Perceptron for training data*

Chart, scatter chart

Description automatically generated

*figure 4: Perceptron for testing data*

**Conclusion**

In this assignment, we got familiar with the implementation of the perceptron algorithm and learned how to check linear separability by drawing hyperplane. For the given task, I set the stopping criteria i.e., the number of iterations 100 and learning rate 0.01. It is also checked that if the value of epoch is reduced and the learning rate is set 0.03 then the accuracy rate is 100%.