HW4 – Perceptron, Linear Regression and Logistic Regression

ISHAN PRADEEP JOSHI

USC ID: 3451334123

Index

Perceptron Learning Algorithm

Outline:

The perceptron learning algorithm can be split into two steps of execution:

1. Computation of the summation of the product of the data instance with the weights
   Mathematically:
   $$S = \sum_{i=1}^{N} xi * w'$$
   Where,
   N – total number of data instances
   xi- data instance i
   w'- transpose of weights matrix
2. Calculation of predicted output labels using the sign function.

The data matrix is initially of (N x d) dimensions. However, each data instance is padded with a "1.0" at the beginning making the new dimensions of each data instance 1x(d+1). Similarly, the initial values of the weights are initialised in (d+1) dimensions. The first step computes the value of S as shown by the equation and the second step computes $\theta(S)$, which is the result from the sign function.

Member Functions:

1. **perceptron_train():** This is the main control function of the program. It initialises the weights and runs the above two steps continually until we reach the optimal hyperplane for the given linearly separable data. It also has an outer loop of 10000 iterations as a fail safe in case the initialised weights take too long to converge.
2. **predict():** This function computes step one and returns the predicted output labels for the specific set of weights in a current iteration.

Output:

For a specific run of the code, the following were the initial weights and the final weights and accuracy:

Initial weights[[-86.50443697 15.27696807 -80.84038782 89.37611539]]

Final weights[[-0.09443697 59.08147119 -47.30780643 -35.3286515]]

Accuracy is 100.0

Pocket Perceptron

Outline:

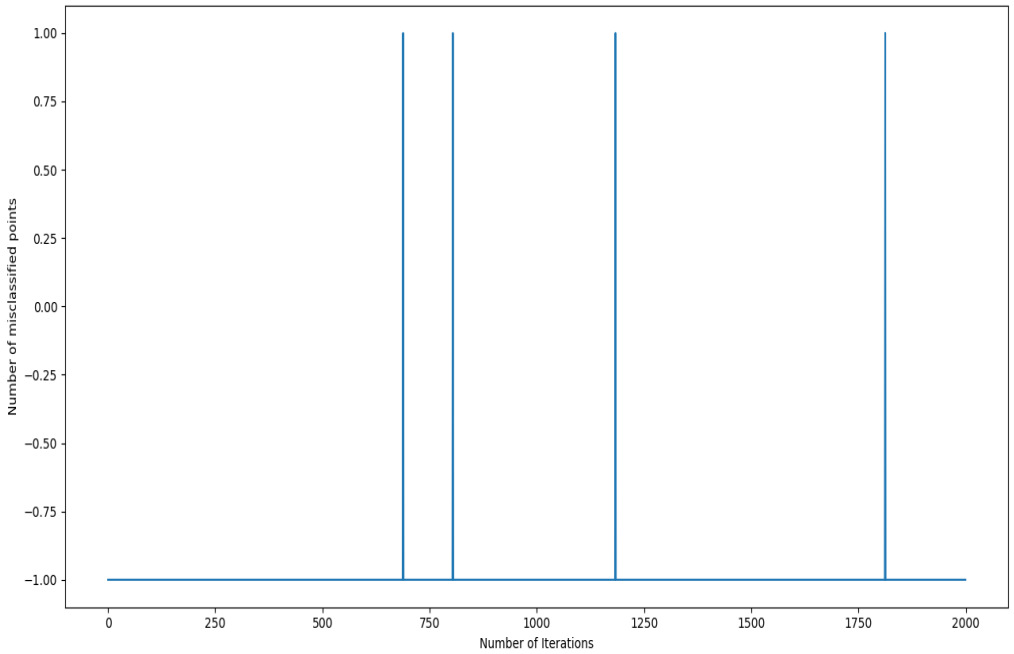
The pocket perceptron is a modification on the perceptron learning algorithm. In a situation where the number of iterations for which the perceptron algorithm can run is limited, the pocket perceptron algorithm stores the weights which best fit the given dataset i.e. the weights that give the lowest misclassified instances. The implementation of the pocket perceptron algorithm is almost identical to that of the perceptron learning algorithm barring this modification. For this assignment, the iterations have been limited to 7000 and the output provides us the accuracy, final weights and a graph of the misclassified instances per iteration.

Output:

Initial weights[[-67.31858346 -14.07919383 0.40662669 82.01578524]]

Final weights[[10.39141654 -12.9561178   4.44806207 3.69233338]]

Accuracy is 50.7



Logistic Regression

Outline:

The logistic regression algorithm computes the output label of each data instance as a probability of it being +1 or -1, based on the dataset of the assignment. It's functioning is similar to the perceptron algorithm wherein it functions in two steps:

1. Computation of the summation of the product of the data instance with the weights
   Mathematically:
   $$S = \sum_{i=1}^{N} xi * w'$$
   Where,
   N – total number of data instances
   xi- data instance i
   w'- transpose of weights matrix

2. Calculation of the output values using the sigmoid function.

Sigmoid Function:

Mathematically, the sigmoid function can be defined as:

$$\theta(S) = e^S/(1 + e^S)$$

Where S- result from step 1 per data instance

This gives us a probability of a data instance having a label +1. 1 - $\theta(S)$ would give us the probability of it giving us a label of -1.

Weights:

The weights are updated using gradient descent. Mathematically,

Weights = weights – (learning rate * gradient error)

Member Functions:

1. **logistic_train():** This is the main control function of the code. It runs a loop for 7000 iterations as mentioned in the assignment and displays the initial and final value of weights.
2. **predict():** This function computes and returns the values of the sigmoid function as well as the gradient for a data instance.

Output:

Initial weights[[-5.05761824 -20.00880494 40.55456476 -90.55956326]]

Final weights[[5.73131128 -6.06885647   8.34542065 -13.56923892]]

Accuracy is 51.800000000000004

Linear Regression

Outline:

Linear Regression provides a continuous value as an output for each data instance within a dataset. Theoretically, it involves the minimisation of the sum of least squared errors of the data instances from an optimal hyperplane.

Mathematically,

$$Ein = \sum_{i=1}^{N} (xi * w' - yi)^2$$

Where, xi – data instance i

w'- transpose of weights matrix

yi-output value for data instance i

The minimization of the above Ein error can be seen as a simple linear system of equations in (d+1) dimensions. Thus, the solution to find the optimum weights is:

$$X * w = Y$$

Where, X- data input matrix

w-weights

Y-Output Labels

The result of the above equation will provide us the optimal weights, without the need of running the algorithm for multiple iterations.

Output:

[[0.01523535]

[1.08546357]

[3.99068855]]

Challenges

1. A major challenge with implementing perceptron and logistic regression was tuning the hyperparameters to find the maximum accuracy possible. For perceptron, an initial accuracy of 99.65 was consistent over multiple attempts and initializations for the weights.
2. The logistic regression algorithm had similar but compounded issues. Running the algorithm for 7000 iterations was troublesome as the 7000 * 2000 calculations required a lot of time to execute and present results. The algorithm did not perform particularly well on the given dataset, as can be seen by the average accuracy of about 50 % over multiple runs. Using neural networks might give better results.
3. The pocket perceptron algorithm took a long time to execute as well, although not as much as logistic regression. This algorithm also performed poorly on the dataset, with an approximate accuracy of only 50%.