Neural Networks Project 1

Final Report

Ashwin Vaidya S3911888

Hari Vidharth S4031180

1 Introduction

Frank Rosenblatt simulated an artificial neuron (perceptron) in the Cornell Aeronautical Laboratory in 1957 [1]. The perceptron algorithm was one of the earliest artificial neural network models. It aims to create a linear classifier inspired by a biological neuron. In our project, we implement a single perceptron to perform a linearly separable operation and train it on numerous randomly generated datasets with the Gaussian distribution. For the training/learning an iterative Hebbian update is performed to update the weights. The capacity of the hyperplane is calculated as per the convergence of the perceptron for the various parameters and datasets. We experimented by varying the size of the dataset for the number of features(N) by varying the Alpha values(α) given by the formula discussed below. The perceptron training algorithm was then trained for multiple epochs for each training dataset. This was done to plot the graphs which represent the accuracy of the perceptron algorithm and also the ability of the perceptron to converge for a linearly separable dataset. Python was used to build and run the experiments and the capacity of the hyperplane and the convergence of the perceptron for different parameters and datasets is analyzed and the results are presented.

2 Method and Implementation

2.1 Generating the Data set

An artificial dataset is generated with the size of PxN, where P is the number of rows (examples) and N is the number of columns (Features), N-dimensional feature vector. The data points ξ^{μ} are randomly generated as per the Gaussian distribution with mean 0 and standard deviation 1. The labels for the dataset ξ^{μ} belongs to the Class +1, -1 with equal probability which is randomly assigned to the data points.

2.2 Training, Accuracy, and Plotting.

The generation of the dataset using the above procedure is repeated for N_d number of different datasets. N_d is the parameter to be set which accepts an integer and creates N_d many randomly generated datasets for training. During each sweep of the perceptron training algorithm, an individual feature vector is introduced from the dataset one at a time. A single sweep through the entire dataset is called an epoch. Iterative Hebbian training is used to perform updates to the weights. The training is performed for N_d number of datasets and on each dataset, the training is performed for N_{max} number of epochs. During each epoch, the algorithm sweeps through 1 to P data points in the dataset, calculates the Local Potential (E) according to the formula given below and updates the weight vector (W) for the P examples for the time step 't'.

The Local Potential is calculated using the formula: $E^{\mu(i)} = w(i).\xi^{\mu(i)} S_T^{\mu(i)}$

If the calculated Local Potential for the P example is lesser than or equal to zero the weight vector is updated according to the formula given below. If the calculated Local Potential for the example P is greater than zero, there is no update performed on the weight vector and the weight vector remains the same as in the previous time step.

The weight vector is updated using the formula: $w(t+1) = w(t) + \frac{1}{N} \xi^{\mu(i)} S_T^{\mu(i)}$

For all the P examples if the calculated local potential is greater than zero, there is no update in the weight vector and this means that the solution exists for which the algorithm can linearly separate the data points, and the training process can be stopped and need not run until the maximum number of epochs. In case the Local Potential E values are still lesser than zero the training runs up till the maximum number of epochs and then comes to a stop. The training is performed for different values of alpha with a constant increment in step size. After the training is completed, using the final weights the predictions are made. Next, the true Y values are compared with the Predicted Y values and the training accuracy is determined. The above-mentioned process is repeated for N_d number of random datasets and the list of accuracies is obtained. This process is continued for different alpha values within a given range with constant increments for the alpha value during each run. With the final list of accuracies for different random datasets and different alpha values, the linear separability of the algorithm is checked and the graph is plotted. This whole process is again repeated for different values of N (N is the number of features and P is the number of data points) where P = α * N and the final graphs are plotted for the different values of N between α and $Q_{l.s}$.

The training algorithm is implemented using Algorithm 1.

Algorithm 1: Rosenblatt Perception Algorithm

for $i \leftarrow 0$ **to** *length(dataset)* **do**
  $\xi^{\mu(i)} = dataset(i)$;
  Compute the local potential $E^{\mu(i)} = w(i).\xi^{\mu(i)} S_T^{\mu(i)}$;
  if $E^{\mu(i)} <= 0$ **then**
    $w(i+1) = w(i) + \frac{1}{N}\xi^{\mu(i)} S_T^{\mu(i)}$;
  else
    w(t+1) = w(t);
  end
end

Algorithm 2: Structure of our Algorithm

for $N \leftarrow 10, 20, 50, 100$ **do**
  for $\alpha \leftarrow$ *0.25* **to** *4.00 with 0.25 increments* **do**
    for *datasets* $\leftarrow 0$ **to** *100* **do**
      $D = \{\xi^\mu, S^\mu\}_{\mu=1}^P$, $\xi_j^\mu \sim N(0,1)$;
      $S^\mu = \pm 1$;
      for *epochs* $\leftarrow 0$ **to** n_{max} **do**
        Update weights using the Rosenblatt's Perceptron algorithm
      end
    end
  end
end

3 Results

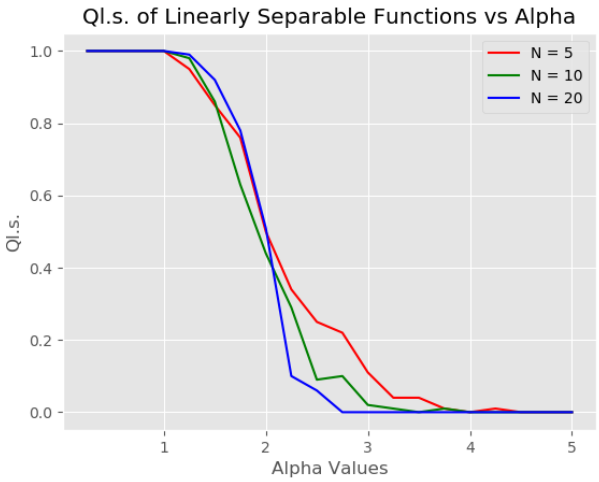

Figure 1: Fraction of $Q_{l.s}$ of Linearly Separable Functions vs Alpha for different values on N. Perceptron Training Algorithm C = 0

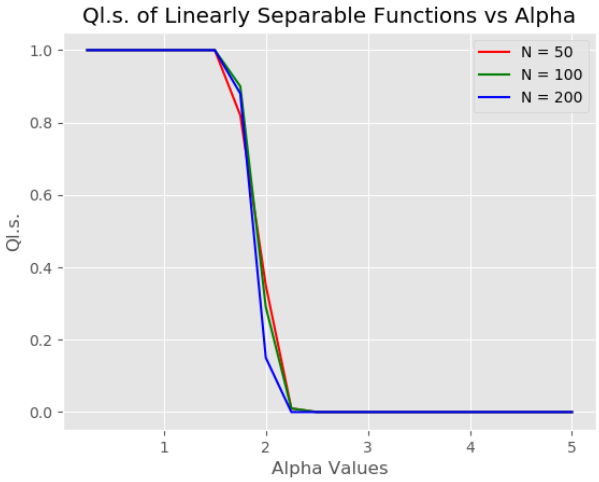

Figure 2: Fraction of $Q_{l.s}$ of Linearly Separable Functions vs Alpha for different values on N. Perceptron Training Algorithm C = 0

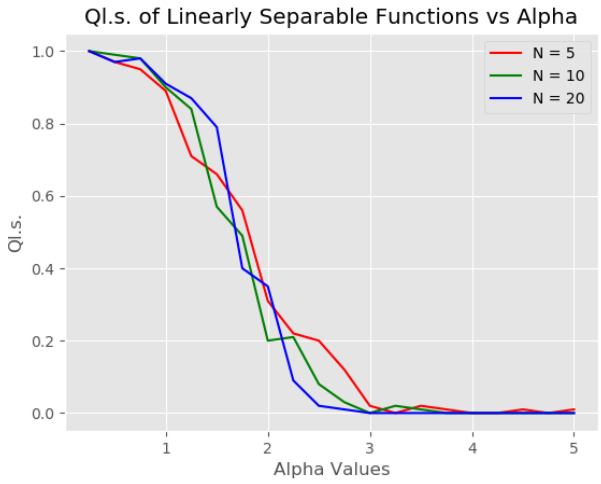

Figure 3: Fraction of $Q_{l.s}$ of Linearly Separable Functions vs Alpha for different values on N. C = +1

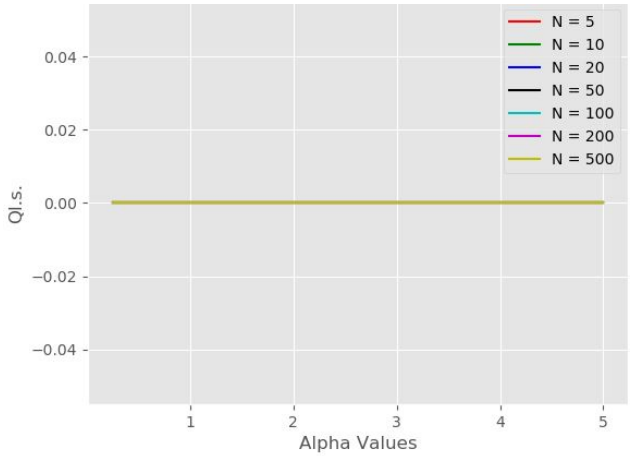

Figure 4: Fraction of $Q_{l.s}$ of Linearly Separable Functions vs Alpha for different values on N. C = -1

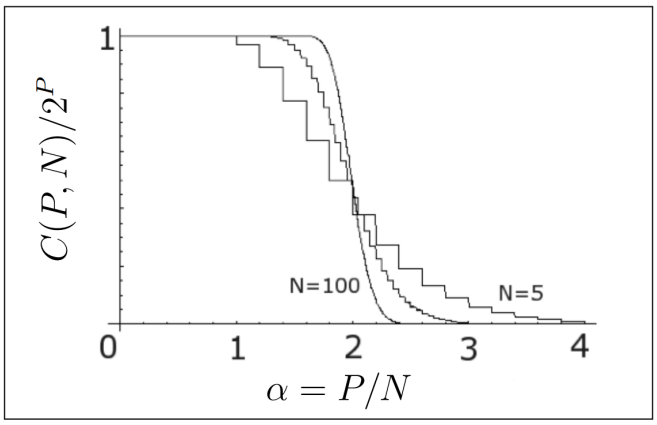

Figure 5: The fraction of $P_{l.s} = C(P,N)/2^P$ of Linearly Separable Functions vs α for different values on N (5, 20, 100). Source: Lecture Notes

Figure 1 and 2 represents the experimental results of the fraction of $Q_{l.s}$ of Linearly Separable Functions vs α for different values of N. The values of α ranges from 0 to 5.0 with a step increment size of 0.25. The different values of N used for the experiment are N = 5, 10, 20, 50, 100, 200 represented by the two graphs with max epochs of 1000. Increasing the α values by 0.25 the perceptron algorithm is able to classify the data sets correctly for α values in the range 0.25 $< \alpha <$ 1.5 for N = 5. For the remaining values of N the range of α becomes $0.25 < \alpha < 1.75$ for convergence and can classify the data sets correctly with 100 percent accuracy. From α greater than 1.5 and 1.75 respectively there is a decrease in value of accuracy of the perceptron algorithm till α greater than 3.0 for N values 5, 10, 20 and α greater than 2.0 for higher values of N that is 50, 100, 200. During these periods there is a steep drop in performance as shown by the graphs. For α values greater than 3.0 and 2.0 respectively as shown in the graphs for the different values of N the accuracy becomes almost 0 or close to 0 percent, that is the perceptron algorithm fails to classify the dataset correctly.

3.1 Bonus Questions:

For the Bonus questions, The experiment was carried out for different values of N represented by the graphs of Figure 1 and 2. Looking at the graphs we can conclude than by increasing the values of N the plot starts to resemble a step function, similar to the results predicted by the theory as shown by Figure 5. Also, an experiment was conducted for different values of C, the values of C for this are -1, 0, +1. The C value of 0 represents the perceptron algorithm as represented by the graphs of Figure 1 and 2, the C value of -1 as represented by the graph 4 shows that there is no convergence achieved and the accuracy is always 0 for all values and parameters. For convergence, the value of C should

be at least 0 or greater than 0. Figure 3 shows the results for the value of C equal to +1. The algorithm converges and can reach 100 percent accuracy for the different values of N for α values up till 1.0 and soon after starts decreasing for α greater than 1.0 and reaches 0 percentage accuracy for α greater 3.0.

4 Discussions

From Figure 1 it can be seen that for all cases where $P \leq N$, we get 100 percent accuracy, which shows that the algorithm can linearly separate the data. In the above case, the α values are less than 1. For α values greater than 1 the accuracy starts to decrease over the different datasets finally reaching zero for higher α values telling us that the datasets are no longer linearly separable by a single perceptron. The Figure 1 and 2 are similar to the Theoretical Figure 5, which shows the function of $P_{l.s}$ vs α. The Theoretical represents a step function where if α value is set to a small enough value, we get exact points where the performance increase or decreases. Thus, we can argue that if we increase the number of epochs and keep the value of α small enough then we may get a figure of step function just like The Theoretical Figure 5.

5 Conclusion

The experimental results obtained in Figure 1 and 2 is similar to the theoretical results obtained shown in Figure 5. To conclude, when we increase the α value by a small number for a very large number of N, The Rosenblatt perceptron Algorithm will linearly separate the data sets till alpha equal to the value of around 2.0, which can be seen from the 2, there is a steep decrease in performance for α values greater than 2.0 till around α equal to 2.25 for higher values of N. Hence from our Experimental results, the storage capacity of The Rosenblatt perceptron Algorithm is equal to 2.0.

6 Distribution of Workload

Both members were responsible for implementing different modules equally. Even the report was divided into equal sections and each member filled in his part.

References

[1] Gary Marcus.(2019 October 17). Hyping Artificial Intelligence, Yet Again, https://www.newyorker.com/tech/annals-of-technology/hyping-artificial-intelligence-yet-again.

[2] Michael Biehl. (2019, January 1). Learning from Examples. Lecture Notes.