Neural Networks and Computational Intelligence
Assignment 1: Perceptron Training

Ayça Avcı (S4505972)
Ethelbert Uzodinma (S3886026)

January 24, 2020

1 Introduction

In this report, we applied the Rosenblatt perceptron algorithm on a dataset of randomly generated data with P number of examples, each of N dimensional features, to observe the capacity of a hyperplane [1]. A set of binary labels S^{μ} of -1 or 1 was randomly assigned to each example. Both labels with an equal probability of occurrence of 0.5.

The Rosenblatt perceptron algorithm is used for binary classification of weight vectors. It is designed to handle only classification tasks that are linearly separable. This is achieved by separating the two dichotomies with a hyperplane.

We first run our experiment on the dataset and observe the capacity of the hyperplane. We then sweep through the entire dataset n times, which corresponds to the number of epochs while observing the changes in the hyperplane until it becomes linearly separable, or, if the algorithm converges before the maximum epoch n_{max} is reached. We then plot a graph of linearly separable success fractions against α.

2 Methods and Implementation

2.1 Dataset

The dataset consists of feature vectors that are randomly generated from a normal distribution with a size of P x N were P is the number of examples and N the number of dimension. The randomly generated data set is Gaussian distributed. it was chosen to ensure that the vector $\xi^{\mu} \in \mathbb{R}^N$ are vectors of independent random components ξ_j^{μ} with a mean of zero and a standard deviation of 1. In this experiment, initial value of $N = 20$ and P depends on the function $P = \alpha * N$.

2.2 Rosenblatt Perceptron Algorithm

The Rosenblatt perceptron algorithm was applied to the dataset of $\mathbb{D} = \{\xi^{\mu}, S^{\mu}\}_{\mu=1}^{P}$ which are Gaussian components $\xi_j^{\mu} \sim N(0, 1)$ by sequentially presenting the feature vectors in a loop (epoch). In each epoch, the network is trained by presenting feature vectors and their corresponding labels $S^{\mu(t)}$, and updating the weight vector of the network. The loop runs n times, and is limited to $n \leq n_{max}$, the total update step on each training example is $n_{max} * P$. The training is repeated until the maximum number of epochs n_{max} is reached, or until all the labels are correct then convergence is reached. The training is repeated several times on different datasets, and what fractions of these datasets were linearly separable is determined.

The steps taken to implement the algorithm on the dataset include:

- We generate artificial datasests containing P randomly generated N-dimensional feature vectors and binary labels: $\mathbb{D} = \{\xi^{\mu}, S^{\mu}\}_{\mu=1}^{P}$. The $\xi^{\mu} \in \mathbb{R}^{N}$ are vectors of independent random components ξ_j^{μ} with a mean of zero and variance of 1.
- We implemented sequential perceptron training by cyclic representation of the P examples. At time step $t = 1, 2, ...$ present example $\mu(t) = 1, 2, ...P, 1, 2, ...$ We implemented the above using nested loops where the inner one runs from 1 to P and the outer loop counts the number n of epochs, i.e sweeps through the data set $\mathbb{D}$. We limited the number of sweeps to to $n \leq n_{max}$ so that the total number of update steps will be at most $n_{max}P$.
- Initialization of the weight as $w(0) = 0$, assuming a tabula rasa while given the time steps $t = 1, 2, 3...$
- On each training step, we perform Hebbian update on the weight vector based on:

$$w(t+1) = w(t) + \frac{1}{N}\Theta[c - E^{\mu}]\xi^{\mu(t)}S^{\mu(t)} \tag{1}$$

  where $\Theta[c - E^{\mu}]$ is a threshold function, c is a constant.
- Update the weight vector $w(t+1)$ based on the condition below:

$$f(E^{\mu}) = \Theta[c - E^{\mu}] = \begin{cases} 1, & \text{if } E^{\mu(t)} < c \\ 0, & \text{if } E^{\mu(t)} \geq c \end{cases} \tag{2}$$

- Previous steps are repeated on several randomly generated datasets, using different values of P and N, until the P number of examples is linearly seperable and then determine the success fraction $Q_{l.s}$ against $\alpha = P/N$ which is the capacity of the perceptron.

  The $P_{l.s.}$ is the probability of a set of randomly assigned labels $\{S^{\mu} = \pm 1\}_{\mu=1}^{P}$ to be linearly separable. We use it to compare and represent the result for different N dimensions as shown in the Figure 2 below. For values of $P \leq N$ and high dimensions the fraction of linearly separable dichotomies should increase up till a value of approximately 1, for $P > N$ the value of $P_{l.s.}$ decreases and should approach its limiting value $P_{l.s.}(\alpha) \to 0$ for $\alpha \to \infty$ (see equation 4).

3 Results

First, the algorithm is run for $N = 20$, $P = \alpha N$ with $\alpha = 0.75, 1.0, 1.25, ...3.0$, $n\_D = 50$, $n\_max = 100$. It means that we used a dataset containing 20 feature vectors, for the maximum epoch number 100. P is determined by the following equation:

$$P = \alpha * N \tag{3}$$

The below $P_{l.s.}$ formula [2] determines the probability that a randomly generated feature set with binary labels is linearly separable:

$$P_{l.s.}(P, N) = \frac{C(P, N)}{2^P} = \begin{cases} 1 & \text{for } P \leq N \\ 2^{1-P} \sum_{i=0}^{N-1} \binom{P-1}{i} & \text{for } P > N \end{cases} \tag{4}$$

For every values of α between 0.75, 1.0, 1.25, ...3.0, training step is repeated for 50 times, and each time, newly generated data set is used. Following graph is obtained for the success fractions that are obtained for different values of α. Comparison between the experimental $Q_{l.s.}$ and the $P_{l.s.}$ formula is plotted below:

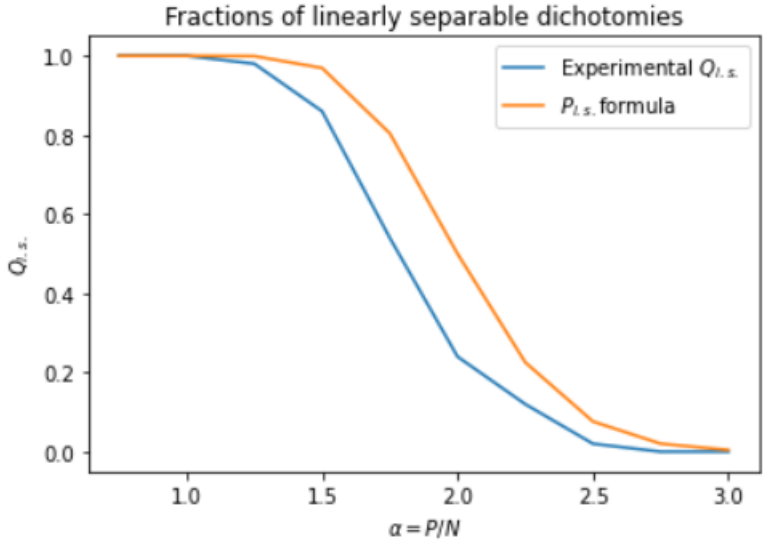

Figure 1: The fractions of datasets which are linearly separable for different α values for $N = 20$.

3.1 Bonus Extension 1

We observed the behavior of $Q_{l.s.}(\alpha)$ for different system sizes N. We set α values between $1.5 \leq \alpha \leq 2.5$ with 0.1 increment (smaller increment compared the initial experimentation as advised) in each iteration, and ran the algorithm for values of $N = 50$, 100 and 300 respectively. We obtained the following results:

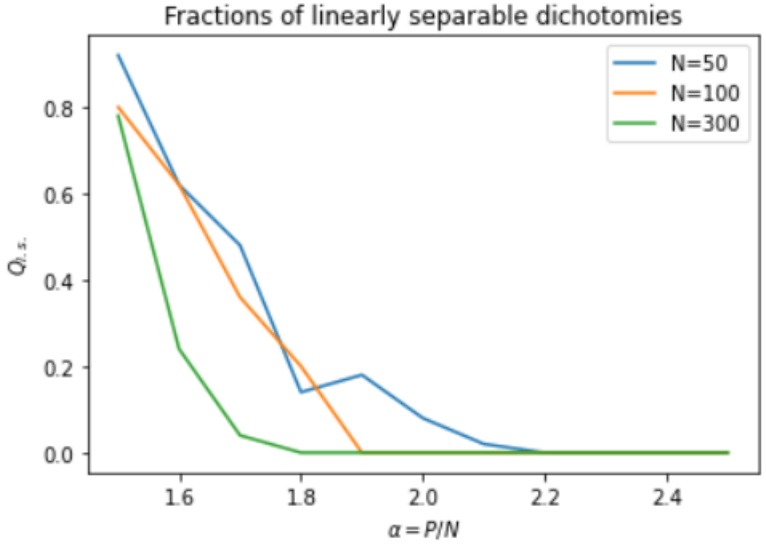

Figure 2: The fractions of datasets which are linearly separable for different α values for $N = 50$, 100, 300.

3.2 Bonus Extension 2

We considered a non-zero value of c, which is greater than 0, for the updates when $E_\mu < c$ to see whether the value of c influences the results in terms of $Q_{l.s.}$. We used values of $c = 0.5$, 0.8, 1, 2, 5, 10, 20 respectively.

We obtained the following results:

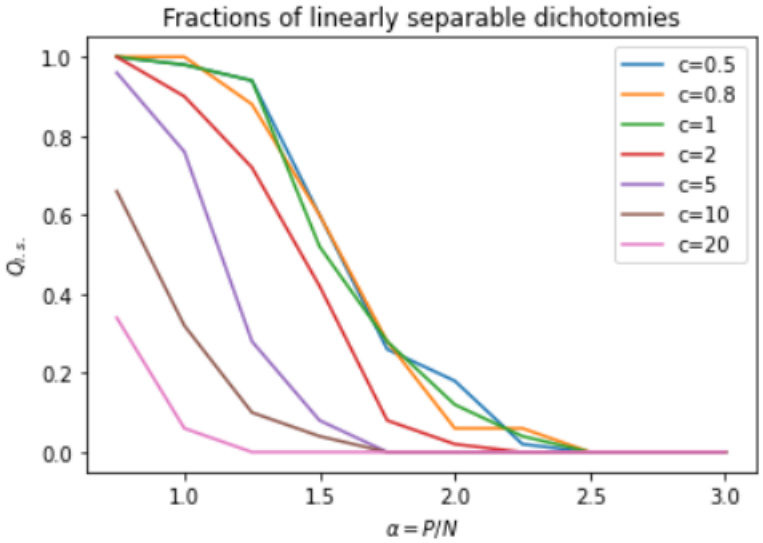

Figure 3: The fractions of datasets which are linearly separable for different α values for $N = 20$, $c = 0.5$, 0.8, 1, 2, 5, 10, 20.

4 Discussion

After running the algorithm for $N = 20$, $P = \alpha N$ with $\alpha = 0.75, 1.0, 1.25, ...3.0$, $n\_D = 50$, $n\_max = 100$, it is found that obtained success fraction is 1 when $P \leq N$ as described in equation 4. For P > N, it is obtained that results approach a logarithmic function that corresponds to the equation 4. Capacity of the hyperplane decreases with the increasing α values in each iteration.

However, the obtained success fractions for the experimental $Q_{l.s.}$ differ slightly from the $P_{l.s.}$ formula in Figure 2. The difference can be explained by relatively small dataset, small number of runs and randomized data. This difference can be alleviated by increasing the maximum number of epochs, the number of training runs and the number of features that are used.

4.1 Bonus Extension 1

As a first possible extension, we observed the behavior of $Q_{l.s.}(\alpha)$ for different system sizes N. For $N = 50$, 100, 300 and α values between $1.5 \leq \alpha \leq 2.5$ with 0.1 increment in each iteration, computed success fraction became 0 after the values of $\alpha = 2.2$, 1.9, 1.8 in Figure 2. As predicted by the theory, we concluded that success fraction approaches a step function (even if we could not obtain the exact discontinuous graph) with increasing N.

4.2 Bonus Extension 2

As a second possible extension, we used a non-zero value of c for the updates when $E_\mu < c$ to see whether the value of c, which is greater than 0, influences the results in terms of $Q_{l.s.}$. c values are determined by looking the range of local potentials E^μ in each iteration. For values of α between $0.75 \leq \alpha \leq 3.0$ with 0.25 increment for $N = 20$, $c = 0.5$, 0.8, 1, 2, 5, 10, 20, computed success fraction became 0 after the values of $\alpha = 2.5$, 2.5, 2.5, 2.25, 1.75, 1.75, 1.25 respectively in Figure 3. Increasing choice of c shows that for the values of $c \leq 1$, results in terms of $Q_{l.s.}$ do not change significantly. The reason is most of the E^μ values lie between (0, 1]. When higher values are chosen for c, the $Q_{l.s.}$ function starts behaving as a step function with decreasing α values where $Q_{l.s.}$ become 0. The experiment implies that a value of 0 for the $Q_{l.s.}$ for all

values of α, which means that the given dataset is not linearly separable, and the exact step function can be obtained by applying higher c values. Depending on our results from the experiment, we concluded that the choice of c influences the results in terms of $Q_{l.s.}$. With higher values of c, $Q_{l.s.}$ becomes 0 for decreased α values.

5 Conclusion

We implemented the Rosenblatt perceptron algorithm on a dataset of randomly generated data with P number of examples, each of N dimensional features, to observe the capacity of a hyperplane [1]. First, we run the algorithm for $N = 20$ and α values between $0.75 \leq \alpha \leq 3.0$ with 0.25 increment in each iteration. The experiment illustrates that success fraction $Q_{l.s.}$ is 1 when $P \leq N$ as described in equation 4, and the capacity of the hyperplane decreases with the increasing α values in each iteration. For P > N, the obtained success fractions for the experimental $Q_{l.s.}$ differ slightly from $P_{l.s.}$ formula in Figure 2, but shows a similar behaviour. Hence, linear separability in our implementation resembles the theoretical probabilities in the given equation 4. The difference between two experimental $Q_{l.s.}$ and $P_{l.s.}$ formula can be explained by relatively small dataset, small number of runs and randomized data. This difference can be alleviated by increasing the maximum number of epochs, the number of training runs and the number of features that are used.

As predicted by the theory, we concluded that for the different values of N and a limited range of α values with a smaller increment, success fraction approaches a step function with increasing N.

For different values of $c > 0$, depending on our results from the experiment, we concluded that the choice of increasing c values influences the results in terms of $Q_{l.s.}$ approaching the step function. For higher values of c, the dataset will become not linearly separable, which means the success fraction $Q_{l.s.}$ will be zero for all values of α.

Better results can be obtained by increasing the the maximum number of epochs, number of features and number of training runs.

References

[1] Michael Biehl, "D1-Perceptron: history, storage problem, Rosenblatt Perceptron Algorithm, convergence proof ", *"An Introduction to Neural Networks and Computational Intelligence, Materials from MSc level Course"*, 2020.

[2] Michael Biehl, "D2-Perceptron: capacity of a hyperplane, learning a rule in version space, optimal stability ", *"An Introduction to Neural Networks and Computational Intelligence, Materials from MSc level Course"*, 2020.

Appendix A Distribution of the workload

The main part of the project is completed with the equal contribution by each group member.

Ayça Avcı implemented the bonus extensions and added them to the report.