Neural Networks and Computational Intelligence
Assignment I: Perceptron Training
Group 09

David Fernandez Chaves (P284762), Shafeef Omar (S4171853)

December 13, 2019

1 Introduction

Today, neural networks are used in a multitude of applications due to the good results achieved in data classification. There are algorithms capable of learning to make correct classifications from a training with multiple input data. The most basic case of this was found in [1]. The algorithm proposed by Rosenblatt aims to imitate the behavior of a brain neuron, using pre-classified data for learning (called supervised learning). Rosenblatt demonstrated that if the input dataset is linearly separable, its algorithm will always converge. This is known as the Perceptron Convergence Theorem and is one of the fundamental results in Machine Learning.

The accomplishment of this assignment consists of a study on the linear separability of data sets using the perceptron theory. The study would also give an insight about the capacity of the hyperplane. The assignment requires us to find the weights of the hyperplane that would binary classify the data. i.e. Given a data set containing P examples of N-dimensional feature vectors and binary labels: $\mathbb{D} = \{\boldsymbol{\xi}^\mu, S^\mu\}_{\mu=1}^{P}$, where $\boldsymbol{\xi}^\mu \in \mathbb{R}^N$ and $S^\mu = \pm 1$, we must find the weight vectors of the hyperplane $\boldsymbol{w} \in \mathbb{R}^N$ such that

$$S(\boldsymbol{\xi}^\mu) = sign(\boldsymbol{w} \cdot \boldsymbol{\xi}^\mu), \text{ for all } \mu = 1, 2, ..., P \tag{1}$$

This is realised by using the Rosenblatt algorithm which is an iterative algorithm involving sequential perceptron training by cyclic representation of the P examples till it converges. The data set that we're using is an artificially generated data set containing P examples of N-dimensional feature vectors and binary labels: $\mathbb{D} = \{\boldsymbol{\xi}^\mu, S^\mu\}_{\mu=1}^{P}$, where $\boldsymbol{\xi}^\mu \in \mathbb{R}^N$ are vectors of independent random components ξ_j^μ with mean *zero* and variance *one* of Gaussian components ($\xi_j^\mu \sim \mathcal{N}(0, 1)$) and the binary output labels, $S^\mu = \pm 1$, are taken to be independent random numbers with equal probability 1/2.

Next, we define the term, $\alpha = P/N$, and vary it for a fixed value of N(=20 in our case) in order to study the capacity of hyperplane. For this section, we specify that it is essential to assume that the data set is in general position. The number $C(P, N)$ of linearly separable binary class assignments or dichotomies of P feature vectors in an N-dimensional input space is of interest to us to study the capacity of the hyperplane. For vectors in general position, the value of $C(P, N)$ is defined as:

$$C(P,N) = \begin{cases} 2^P & \text{for } \alpha \leq 1, \\ 2\sum_{i=0}^{N-1} \binom{P-1}{i} & \text{or } \alpha > 1 \end{cases} \tag{2}$$

The fraction of the linearly separable dichotomies out of all possible dichotomies is given by:

$$P_{ls}(P,N) = \frac{C(P,N)}{2^P} = \begin{cases} 1 & \text{for } \alpha \leq 1, \\ 2^{1-P} \sum_{i=0}^{N-1} \binom{P-1}{i} & \text{or } \alpha > 1 \end{cases} \tag{3}$$

This fraction P_{ls} can also be interpreted as the probability of the randomly assigned labels $S^\mu = \pm 1$ to be linearly seperable.x

2 Our Solution

This problem is addressed by the Rosenblatt Perceptron algorithm as mentioned earlier. As mentioned about our objective in Eq. (1), we reformulate the problem slightly. We note that $sign(\boldsymbol{w} \cdot \boldsymbol{\xi}) = S \Leftrightarrow \boldsymbol{w} \cdot \boldsymbol{\xi} S > 0$. Hence we define the term E^μ called the local potential, defined as:

$$E^\mu = \boldsymbol{w} \cdot \boldsymbol{\xi}^\mu S^\mu, \text{ for all } \mu = 1, 2, ..., P \tag{4}$$

The problem is now reformulated such that we must find $\boldsymbol{w} \in \mathbb{R}^N$ such that $E^\mu \geq c > 0$ where constant $c > 0$ is introduced as a margin in terms of the conditions $E^\mu > 0$. Keeping this in mind, let's proceed to the Rosenblatt Perceptron Algorithm.

The Rosenblatt Perceptron Algorithm is a Hebbian Iterative Training algorithm which is provided with sequential perceptron training by cyclic representation of the P examples till it converges. This is realised using nested loops where the inner loop runs from 1 to P and the outer loop counts the number n of epochs, i.e. number of sweeps through the dataset $\mathbb{D}$ until $n \leq n_{max}$ so that the total number of individual update steps will be at most $n_{max}P$. The algorithm is presented as below:

ROSENBLATT PERCEPTRON ALGORITHM

Begin with *tabula rasa* initialization, i.e. $\boldsymbol{w}(0) = 0$
At each discrete time step t:
— Determine the index $v(t)$ of the current example $(\xi^{v(t)})$
— Compute the local potential $E^{v(t)} = \boldsymbol{w} \cdot \boldsymbol{\xi}^{v(t)} S^{v(t)}$
— Update the weight vector according to

$$\boldsymbol{w}(t+1) = \begin{cases} \boldsymbol{w}(t) + \frac{1}{N} \boldsymbol{\xi}^{v(t)} S^{v(t)} & \text{if } E^{v(t)} \leq 0, \\ \boldsymbol{w}(t) & \text{else} \end{cases} \tag{5}$$

— Repeat till convergence or until the maximum number of sweeps n_{max} is reached.

We perform n_D repetitions in which we generate random data sets from the previously mentioned configuration parameters to train algorithm. In addition, we have performed each batch of experiment with multiple alpha values so that $\alpha = P/N$ and therefore different values for P.

During each repetition, we count how many times the algorithm converges and divide it by the number of experiments to get Q_{ls}, which is the convergence probability. For each batch of experiments we have also calculated P_{ls} which is obtained using the Eq. (3). A graph of P_{ls} and Q_{ls} vs. alpha values is plotted to study the comparison.

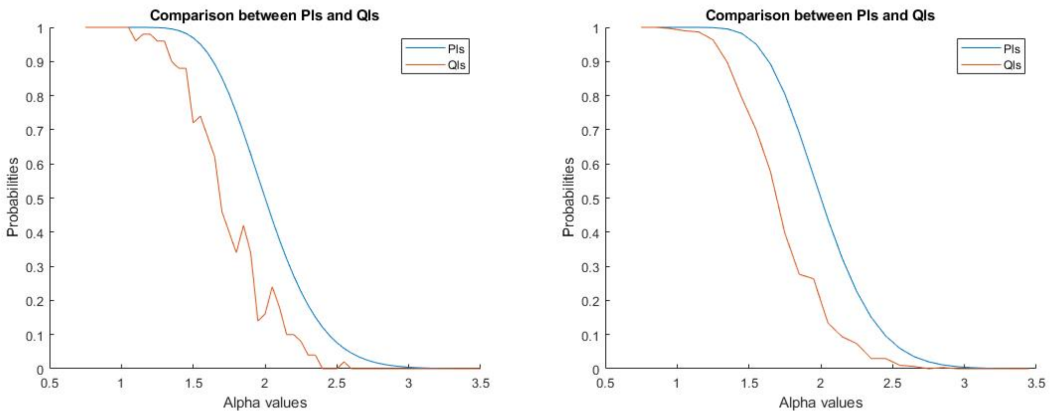

Figure 1: Comparison between the theoretical probability that the data are linearly separable (P_{ls}) and and the experimental value (Q_{ls}) obtained from the data that we have been able to separate linearly. On the left: $n_D = 50$ and $\alpha Step = 0.25$, on the right: $n_D = 300$ and $\alpha Step = 0.1$.

3 Results and Conclusions

Fig. 1 shows the comparison between the theoretical probability (P_{ls}) and the experimental value (Q_{ls}) obtained from the data that can linearly separate our randomly generated data. As evident from the figure, the experimental value of probability is always less than or equal to the theoretical probability ($Q_{ls} \leq P_{ls}$) for every value of alpha.This is because for infinite number of tests, Q_{ls} and P_{ls} are supposed to converge. The convergence of the algorithm to find a solution within our specified number of iterations (n_{max}) is guaranteed up to $\alpha = 1$ as evident from the figure in which Q_{ls} and P_{ls} are 1 upto this point. Beyond this point, the convergence probability for the fixed number of iterations decrease with increase in value of α. For the given value of n_{max}, our algorithm won't converge after $\alpha = 2.6$ practically and won't converge after $\alpha = 3$ theoretically.

4 Workload

Tab. 1 shows the working percentage of each of the integrates in the group. In general we have had a good communication and the work has been equitable.

Table 1: This table shows the percentage of the distribution of tasks in the performance of the assignment.

	David Fernandez Chaves	Shafeef Omar
Code programming	50	50
Plots creation	50	50
Writing	50	50

References

[1] F Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain., 1958.