Linear separability of randomized Gaussian data

Neural Networks and Computation Intelligence

Ivo de Jong (s3174034)
Travis Hammond (s2880024)

December 2019

1 Introduction

Perceptrons are the core and origin of modern neural networks. The way a perceptron works as a classifier is that it draws a hyper-plane through the input space. Items on one side of the plane are linearly separable from the items on the other side of the plane.

Normally the items are given a classification in advance, somehow related to the data that formed them. In that case the perceptron has to learn what plane exists to separate the two classes. Fortunately they are very good at this. It has been mathematically proven that if the two classes are linearly separable, then a perceptron trained with the Rosenblatt algorithm will eventually find an appropriate hyper-plane (though this may take an indeterminable amount of time). Because of this proof, one can consider the perceptron's ability to find the appropriate hyper-plane as a proxy for whether the classes are linearly separable to begin with.

This is precisely what the current experiment does. It generates a number of N dimensional data-points which are randomly assigned a binary class. Then the perceptron will try to find a linearly separating hyper-plane, which will be used as an argument that the data was linearly separable.

For this we may vary the number of data-points per dimension, which will be referred to as α. Mathematical proof exists that when $N \to \infty$ linear separability is well defined by α. If $\alpha \leq 2$ the data will be linearly separable, whereas if $\alpha > 2$ it will not be linearly separable. When N is less than infinite we will find probabilities of being linearly separable. At $\alpha = 1$ we may expect a probability of 1 that the data is linearly separable, and as α increases we may expect the probability of linear separability (P_{LS}) to approach 0. When N is small we expect the approach to be fairly slow, whereas a more step-like function as described with $N \to \infty$ will form as the number of dimensions increases. This experiment will vary both the number of dimensions, as well as the number of data-points per dimension to see how they affect the probability of linear separability.

Unfortunately it is not possible to determine true probabilities with such an experimental set up. Instead the empirical probablility Q_{LS} is defined as the emperical probability. The purpose of this is the approximate P_{LS} as well as possible, but keeping aware that the experiment measures the empirical probability Q_{LS}, rather than the true probability P_{LS}.

To further expand upon this we will also add the consideration of inhomogeneous perceptrons, and how that affects the linear separability. The previously described homogeneous perceptron is only able to draw hyper-planes that cut through the origin. With inhomogeneous perceptrons a bias is added, so that it is possible to move away from the origin which grants additional degrees of freedom and will likely increase the probability of linear separability.

2 Methods

The method section will consist of three components. Firstly it will discuss the way that the data is generated. Secondly it will discuss the way the Rosenblatt perceptron algorithm works, and lastly it will discuss which parameters were used for the experiment.

2.1 Generating data

As alluded in the introduction, the data is randomly generated. P values are drawn with a Gaussian distribution, with a mean of 0 and a variance of 1, each with N dimensions. With a sufficient number of datapoints this resembles a Gaussian distributed N-dimensional cloud around the origin with a variance of 1. Each of these datapoints is will be randomly assigned a label 0 or 1.

This way of generating data allows the perceptron to show the number of linearly separable dichotomies for P datapoints in N dimensions. Since the preceptron forms a hyperplane through the origin, we also want the data centered around the origin. The variance is essentially irrelevant, as the separation plane is infinite.

Naturally, new data is generated for each instance it is required, regardless of whether that is a repetition with the same parameters, or perhaps the equivalent repetition with an inhomogeneous perceptron.

2.2 Rosenblatt algorithm

The Rosenblatt algorithm is iterative, so the best way to present it's functioning is with the pseudo-code shown in algorithm 1. The algorithm defines some variables, of which most notable is the N dimensional 0 weight vector w. It then describes a number of repetitions over the data-set. In each repetition it determines for every data-point the local potential $E^{v(t)} = w \cdot \xi^{v(t)} S^{v(t)}$. Where $\xi^{v(t)}$ is the N-dimensional feature vector for the given data point, and $S^{v(t)}$ is the label for that data point.

If it is wrong ($E^{v(t)} >= 0$) the weight vector is updated according to

$$w(t+1) = w(t) + (\xi^{v(t)} * S^{v(t)})/N.$$

If no mistakes were made in a round over the data a solution exists, and the loop ends. If after a maximum number of iterations no separation is found, the algorithm determines that the data-points were likely not linearly separable.

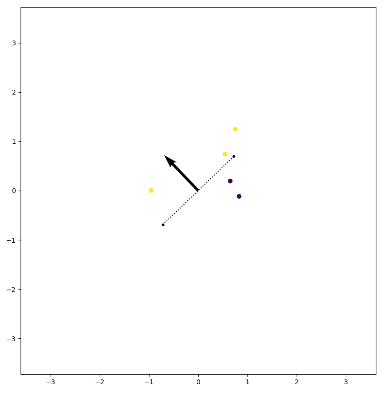
Algorithm 1: Rosenblatt Perceptron Algorithm

```
Result: A vector defining the weights of the perceptron, a Boolean
        whether linear separation was successful
$epoch_{max} \leftarrow$ maximum repetitions;
$N \leftarrow$ number of Dimensions;
$\xi$, $S \leftarrow$ $N$-dimensional data with labels as generated;
$w \leftarrow$ $N$-dimensional 0 vector;
while no solution found and epoch < $max_{epoch}$ do
    for $\xi^{v(t)}$, $S^{v(t)}$ do
        $E^{v(t)} \leftarrow w(t) \cdot \xi^{v(t)} S^{v(t)}$;
        if $E^{v(t)} >= 0$ then
            $w(t+1) \leftarrow w(t) + (\xi^{v(t)} * S^{v(t)})/N$ $m \leftarrow$ True
        end
    end
    if $m == False$ then
        return w, success;
    end
    $epoch \leftarrow epoch + 1$
end
return w, failure;
```

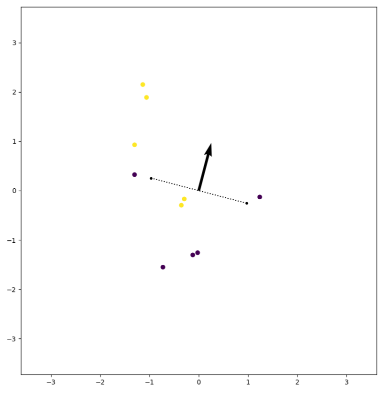
To expand this algorithm to support inhomogeneous perceptrons a value of 1 is added as an $N+1$th dimension to each data-point, and the weights vector will then also be $N+1$ dimensional.

2.2.1 Perceptron visualized

In 2 dimensional space it is very easy to visualize what the perceptron and the data look like. To illustrate this effect two plots are given in Figures 1a and 1b. They each show some datapoints with their classes labeled as yellow or purple. The weight vector of the perceptron is visualized by the arrow. Consequently, the decision boundary, which would be the hyper-plane in higher dimensions, is shown orthogonal to the weight vector. Figure 1a shows a linearly separated scenario, where all the yellow points are on one side of the decision boundary, and all the purple ones are on the other side. In Figure 1b we see the opposite, where linear separation was not achieved. In this case we see some purple and yellow dots on either side of the decision boundary. In fact, for the data shown it is impossible to draw any linear separation which homogeneously separates all the data points.



(a) Linearly separable



(b) Not linearly separable

2.3 Parameters

In order to make some reasonable analyses of how Q_{LS} is determined by α and the number of dimensions N we need to identify some appropriate parameters. In this experiment you can identify the independent variables as the number of dimensions N, the number of datapoints per dimension α, and whether the data perceptron has an additional bias node that allows for inhomogeneous solutions. The dependent variable that follows from these would be the probability that the data is linearly separable (Q_{LS}).

Through some exploratory tests we determined that an appropriate set of dimensions would be $N = 5, 20, 150$. This allows us to show a clear difference between low, medium and large number of dimensions. We found that the step function really becomes apparent at very high dimensionalities, which is why we picked $N = 150$ as an upper bound. Testing with more possible values of N would be computationally costly.

For the range and interval of α we found that a range $[0.75, 5]$ is appropriate. Below $\alpha = 1$ we find that $Q_{LS} = 1$, which will be briefly visualized in the range $[0.75, 1]$. Most dimensionalities will have approached a $Q_{LS} = 0$ by the time $\alpha \to 5$, which is why 5 can be reasonably picked as an upper limit. The interval chosen for α was 0.1. Particularly for the large number of dimensions this allows for a smoother presentation to show that there is not a perfect step function yet. Any smaller interval would be needlessly computationally expensive.

Of course these parameters were applied for both the biased and unbiased perceptron.

Two additional relevant parameters are the maximum number of epochs before the training is determined to be a failure, and the amount of times each specific setting is tested. A sufficiently large number of repetitions is needed to allow Q_{LS} to converge towards P_{LS}. The maximum number of epochs before training is considered a failure can be referred to as $epoch_{max}$ and is set to be 100. While this still leaves some space for failures when it is possible that the perceptron would have found a solution after more than 100 epochs the risk is fairly minimal. Increasing the maximum number of epochs could very well be a waste of processing power as most tasks were solved in far fewer. The maximum number of tests for each training is perhaps more influential in this. It's easy to understand that doing one trial will always result in a Q_{LS} of 100% or 0%, as the number of trials increases the Q_{LS} can approach P_{LS} better. Of course even then it will remain susceptible to swaying with unevenly divided measurements. To gain some reasonable approximation each setting was tested 100 times. Increasing this would allow for a more accurate evaluation of the actual probability of any setting, but 100 is sufficient to show some general trends.

3 Results and Discussion

All of the results are visualized together in a single graphic as Figure 2. This figure shows that regardless of whether a clamped input is added (inhomogeneous perceptrons) a higher dimensionality approaches a step function near $\alpha = 2$, as predicted in the introduction. We see that as the number of dimensions decreases the slope becomes much gentler.

For a comparison between the empirical probability of linear separability Q_{LS} and the true probability P_{LS} figure 3 shows an analytically determined graph of P_{LS} courtesy of Michael Biehl.

Both graphs show that as the number of dimensions increases, the probabilities reach a step function.

On first sight the Q_{LS} graph appears rather noisy. This is actually not just noise though, but a staircase-shape with a bit of noise. This is largely because despite fairly small steps in alpha, the number of data-points has to be rounded to an integer. More descriptively, when we measure $\alpha = 2.1$ with $N = 5$ we have $2.1 * 5 = 10.5$ data-points, which is rounded to 11. However, when we set alpha to the next step we get $\alpha = 2.2$, so $2.2 * 5 = 11$ data-points. This means that we are getting 2 separate measurements of the same scenario. In this case the α increases, but the actual recorded data should stay the same. As α increases further at some point P will reach 12, so it "steps down" as a staircase. This is also more perfectly presented in the P_{LS} graph.

When comparing the inhomogeneous solutions to the homogeneous solutions we find that the extra flexibity given by inhomogeneity translates noteably in performance. Particularly for lower dimensions we see that inhomogeneous solutions allow for a slower decay of Q_{LS}. However, when considering a larger number of dimensions we find that there is hardly any difference. This is likely because, while normally the decay is slowed by allowing inhomogeneous solutions, this does not translate to a difference when decay approaches an infinite speed. Unfortunately this cannot be compared to the P_{LS}, as we don't have an analytically determined graph for this.

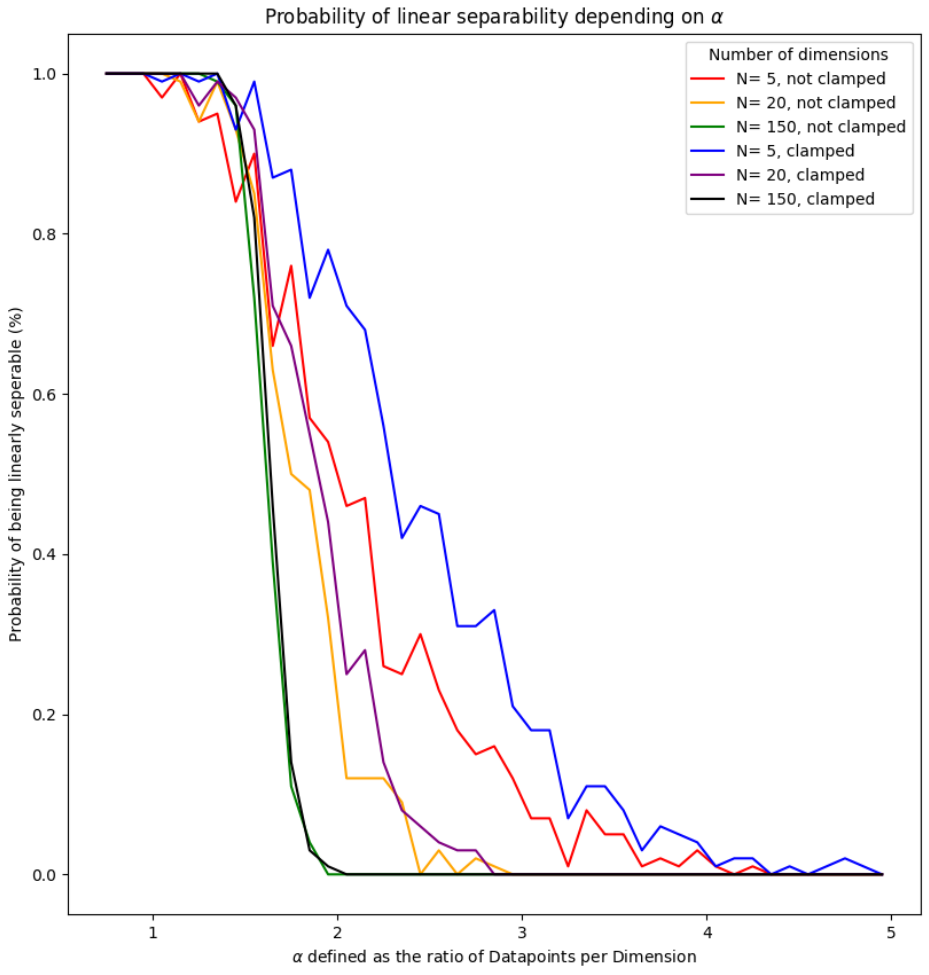


Figure 2: The empirical probabilities Q_{LS} of (in)homogeneous linear separation across different dimensionalities and data densities

A close inspection of Q_{LS} shows that the step function when N increases actually occurs slightly before $\alpha = 2$, whereas in P_{LS} it occurs exactly at $\alpha = 2$. This is likely due to the limited number of epochs allowed to find a solution. This means that there may be a number of linearly separable sets that get marked as inseparable because the number of epoch required to find the correct

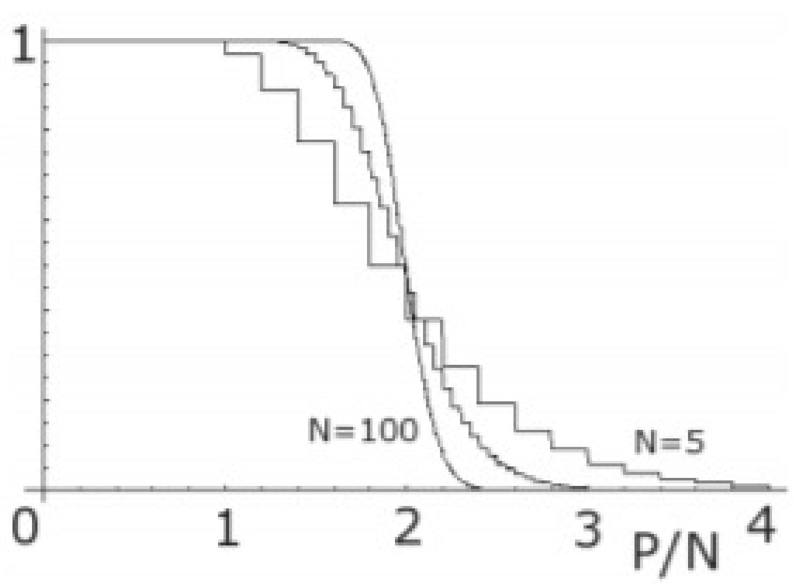


Figure 3: The true probabilities of linear separation P_{LS} across different dimensionalities and data densities, courtesy of Michael Biehl

hyper-plane was larger than 100. Increasing this maximum number of epochs may bring the step function in Q_{LS} closer to $\alpha = 2$ in Q_{LS}, but to make Q_{LS} approach P_{LS} arbitrarily well it is required that the maximum number of epochs is also arbitrarily high.

In conclusion, the empirical function $Q_{LS}(\alpha)$ approximates $P_{LS}(\alpha)$ pretty well. All the most interesting behaviour, is sufficiently well represented, but the exact values will remain an approximation. This is generally acceptable though, as the most insightful conclusions to be drawn from $P_{LS}(\alpha)$ is not the specific numbers and values, but rather the general behaviour.

4 Teamwork

There was a proximate between divide writing the report and developing the experiment. Specifically, Travis was mainly responsible for developing the experiment, while Ivo was mainly responsible for writing the report. This was not fully divided (linearly separable) though.

While Travis wrote the Rosenblatt code and the data generation, Ivo did contribute in parallelization and running the experiments, as well as visualizing and describing the results. Travis also spent additional time on the 2D graphic of the perceptron learning over time, and on developing support for inhomogeneous perceptrons.

Similarly, Ivo wrote the core of the report, but largely based on the code and information given by Travis. Travis also checked the report and took out some final problems.