
Our main goal is to design a perceptron for binary number clasification. This perceptron takes as an input a vector $\vec{S}$ of 20 elements in which the first 10 digits correspond to the first number n_T and the 10 last digits correspond to the second number n_B (the elements of the vector will be $S_i = \{\pm 1\}$, being +1 the 1 on binary notation and -1 the 0 on binary notation, so this way the sequence -1-1-1-1-1-1-1-1+1+1 would be the 0000000011 on binary, hence the 3 on decimal notation). With this information the perceptron has to give as an output $\sigma = +1$ if $n_T > n_B$ and $\sigma = -1$ if $n_T \leq n_B$ using this algoritm for the output $\sigma(\{S_i^\mu\}, \{J_i\})$ = $\text{sign}(\sum_{i=1}^{N} S_i J_i)$.

Problem 2. *Define and evaluate a test error (counting the frequency of errors) of the perfect perceptron over a large number of trials where two random numbers between 0 and 2^{10-1} are ranked. The test error should be obtained comparing the result of the perfect perceptron with an independet benchmark method.*

Comment. Using if-else statements as our benchmark method to tell which number is larger, the perfect perceptron obtained a relative error of 0.07% over 10^4 repetitions of the experiment.

Problem 4. *Evaluate a test error (counting the frequency of errors) of the random perceptron over a large number of trials where two random numbers between 0 and 2^{10-1} are ranked. The test error should be obtained comparing the result of the random perceptron with those of the perfect perceptron.*

Comment. Generating the weights vector $\vec{J}$ using random values extracted from a Gaussian distribution $\mathcal{N}(0, 1)$ the random perceptron obtained a relative error of 49.49% over 10^4 repetitions of the experiment. An error 70700% higher in percentage than the perfect perceptron in comparisation with the benchmark method. This lead us to the conclusion that the untrained random perceptron is useless compared with the perfect perceptron. Even without comparing them, the untrained random perceptron has a failure rate of almost 50%, so if we would pass it 10 couples of numbers, only 5 couples would be well ordered, pointless.

Problem 7. *To study the evolution of the J_i, plot sign($J_i^* J_i$)log$_2$($|J_i|$) versus i, the index of the synaptic weight, for the weights obtained with P = 500 and P = 2000 instances in the training set and comment on the results.*

Comment. Let $\vec{J}^*$ be the weights vector of the perfect perceptron and $\vec{J}$ the trained weights vector. This way on Figure 1 it is represented $\text{sign}(J_i^* J_i)\log_2(|J_i|)$ vs each index of the synaptic weight in order to study the evolution of the J_i from the ones in the perfect perceptron. On this type of representations, a good indicator would be that the function is overall decreasing. This is due to the fact that the weight vector components of the perfect perceptron $\vec{J}^*$ are always decreasing compared with their previous component on absolute value (always but in one transition, the one that goes from 1 to -2^9, clearly reflected on the plots on the 10th transitions), so for the orders of magnitudes in which we are working, $\log_2(|J_i|)$ will aproximate to negatives values. As trained $\vec{J}$ would tend to mimic

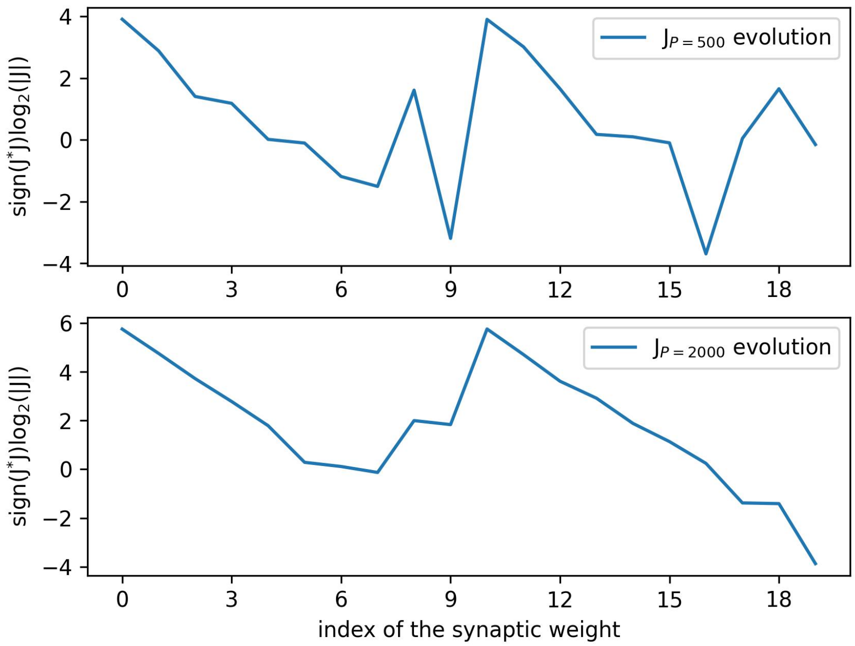

Figure 1: $\text{sign}(J_i^* J_i)\log_2(|J_i|)$ vs index of the synaptic weight for P = 500 (up) and P = 2000 (down) training examples.

this behavior, negative values tendency would translate to $\text{sign}(J_i^* J_i) = +1$, meaning that J_i^* and J_i have the same sign.

By contrary, a bad sign of the training would be oscillations, which might suggest instability on the convengence of the training due to the fact that J_i^* and J_i signs do not match.

Given all of this, we could observe that top plot oscillates more compared with the bottom plot, which converges most notably. This is what we would expect if our training model is effective, given that the top plot is trained with 500 couples of binary numbers while the bottom plot is trained with 2000 couples.

Problem 8. *Evaluate a test error (counting the frequency of errors) of the two trained perceptrons over a large number of trials where two random numbers between 0 and* 2^{10-1} *are ranked. The test error should be obtained comparing the result of the trained perceptrons with those of the perfect perceptron*

Comment. Lastly, we are going to compare the relative error of this two new trained (remember that this perceptrons originates from random values extractedfrom a Gaussian distribution $\mathcal{N}(0,1)$) perceptrons to the relative error obtained for the untrained perceptron (remember that was a 49.49% relative error). For the trained perceptron over 500 couples of binary numbers, the relative error descent to 2.96% (An error 4229% higher in percentage than the perfect perceptron in comparisation with the benchmark method), while for the trained perceptron over 2000 couples of binary numbers, the relative error descent to 0.69%

(An error 986% higher in percentage than the perfect perceptron in comparisation with the benchmark method, notably high but an enormous difference with the untrained model for only 2000 couples used on the training).