# Assignment 8

1. Learning to classify random patterns using a perceptron

   Simulate a perceptron to estimate the maximal number of random patterns that can be correctly classified. The patterns to be classified are vectors of binary activities. More specifically, $\xi_i^\mu = \pm 1$ is the activity of neuron $i$ for pattern $\mu$. The $\xi_i^\mu$ variables are random and uncorrelated (i.e. set them randomly to either +1 or -1 with equal probability). When one of these patterns is imposed to the $N$ input neurons, the output is:

   $$y = \text{sign}\left(\sum_{j=1}^{N} w_j \xi_j^\mu - \theta\right)$$

   where $w_j$ are the weights and $\theta$ is a threshold. For each pattern the desired output $y^\mu$ which indicates the class to which the pattern belongs, is also chosen randomly. Determine the parameters $w_j$ and $\theta$ using the perceptron learning rule discussed in the class, so that the perceptron produces the desired output $y^\mu$ when pattern $\xi_j^\mu$ is imposed to the inputs.

   The procedure is to start from one pattern, randomly chosen, say pattern $\nu$. Compute the corresponding $y$ and compare it the desired $y^\nu$. If it is the same, then do not modify the parameters. Otherwise, use the perceptron learning rule to update the parameters:

   $$w_j \to w_j + \alpha y^\nu \xi_j^\nu$$
   $$\theta \to \theta - \alpha y^\nu$$

   Repeat the same procedure at least 100 times the number of total patterns $p$. Start with $N = 100$ and $p = 10$ to test the program: plot the fraction of correctly classified patterns as a function of the number of learning iterations. It should quickly converge to 1 (i.e. the fraction of errors should to zero).

   Then choose ten values of $N$ between 100 and 1000 and for each value of $N$ determine $p_{max}$, the maximum number of correctly classified patterns. To determine $p_{max}$, start from a small $p$ and progressively increase it until the error does not go to zero (e.g when the minimal fraction of errors never goes below 10% during all the $100p$ learning iterations).

   Plot $p_{max}$ vs $N$ and show that it is approximately linear.

2. Non-linearly separable patterns

   Build now a new perceptron that has $2N$ input neurons. Construct correlated inputs as follows: take the $p$ random patterns of the previous exercise, and concatenate pairs of input vectors. For example one input could be $\xi_i^\mu, \xi_i^\nu$, which is a vector with $2N$ components whose the first $N$ components are $\xi_i^\mu$ and the components from $N+1$ to $2N$ are the $\xi_i^\nu$s. When you consider all possible pairs of random vectors, you get highly correlated input patterns (e.g. $\xi_i^1, \xi_i^2$ will be similar/correlated to $\xi_i^1, \xi_i^3$). These patterns could describe a representation in which half of the neurons encode one feature of a sensory stimulus (e.g. the color of an object) and the other half another feature (e.g. the shape). The two subpopulations of neurons would be "specialized" in the sense that they encode one feature without being affected by the other.

   For $p$ patterns, there will be $p^2$ possible pairs. Consider all possible pairs for small $p$ (e.g. $p = 5$). Try now to train a perceptron to classify the patterns using random desired $y$s, as in the previous exercise (use a different random $y$ for each pair of inputs, and the learning procedure of the previous exercise). Show that even when $N$ is very large (it can be arbitrarily large), the fraction of errors typically never goes to zero (repeat the experiment many times for different choices of random patterns). Show that the situation gets even worse when $p$ increases. That indicates that non-linear separability is not a pathological situations, but it is what typically happens in "realistic" situations, like the one studied in this exercise.

   The classification of these correlated patterns is similar to the XOR problem discussed in the class. The patterns are low dimensional and for large $p$ the majority of random classification problems are non-linearly separable.

3. Advanced: in the case of non-linearly separable patterns described in the previous exercise, determine the dimensionality of the input patterns (when all pairs of random patterns are considered) and show that it is much smaller than the maximal dimensionality, especially when $p$ becomes large. The dimensionality can be estimated using the matlab function that performs Principal Component Analysis.

4. Advanced: solving the problem of non linear separability with mixed selectivity neurons

   Introduce $M$ additional neurons make them part of the input vector (the

total number of input neurons would be $M + 2N$). Denote the activity of these additional neurons with $s_i$, with $i = 1, ..., M$. $s_i = 1$ if $\xi_i = 1$ AND $\xi_j = 1$, otherwise $s_i = -1$. $i$ and $j$ are random indexes denoting two input neurons, with $i = 1, ..., N$ and $j = N + 1, ..., 2N$.

The $s_i$ neurons are mixing non linearly the inputs that come from the two components of the input. In other words, they exhibit mixed selectivity to the "features" represented by the two components of the input. With a number of $s_i$ units that is comparable to $p^2$, it should be now possible to solve the random classification problem of the previous exercise.

Show that the error actually goes to zero as $M$ grows and that it is much smaller than in the case in which the $M$ mixed selectivity neurons are replaced with non-mixing neurons (e.g. neurons $s_i = \xi_i^\mu$).

Replace the $s_i$ neurons with randomly connected neurons (i.e. neurons that are connected with random gaussian weights to 10 randomly selected $\xi$ neurons). Shows that also in this case the problem can be solved. The number of randomly connected neurons will also have to scale as $p^2$.