Simple Artificial Neural Network

Sergio Peignier

1 Introduction

In this practical course we will analyze the analogical neuron, one of the most simple models of neurons. Let $x_i(t)$ denote the average activity of neuron i at time t (the higher the value of $x_i(t)$, the stronger the activity of the neuron). A set of N neurons are organized in a Neural Network. The neurons are interconnected by means of synaptic connections, and each neuron influences directly the activity of the neurons with which it is connected (directed connections). In this simplistic case, we consider that each neuron in the network is connected to every other neuron (even to itself). Therefore, the activity $x_i(t)$ of neuron i at time t is influenced by each other neuron $j \in [1, \ldots, N]$ from the network. Let W be the matrix encoding the strength of the connections. Hence $W_{i,j}$ denotes the *synaptic weight* of the connection that goes from neuron j to neuron i. We consider here that the final impact of the neural network on a single neuron can be computed as the sum of the neuronal activities weighted by the corresponding synaptic weights. Formally, the equation hereafter governs the activity of neuron i :

$$\tau \times \frac{dx_i(t)}{dt} = -x_i(t) + F\left(g \times \sum_{j=1}^{N} W_{i,j} x_j(t)\right) \tag{1}$$

In the previous equation, function F is called the *transfert function*, in practice an hyperbolic tangent function is used :

$$F(x) = tanh(x) \tag{2}$$

Notice that this function behaves somehow as an amplification function or a filtering function, annihilating low and high values of x and mapping linearly values in the middle. In fact, parameter g corresponds to the slope of amplification of the value of the weighted sum (action of the neural network on the neuron). Finally the variable τ corresponds to a relaxation constant of the system (time measure).

2 Euler method

In this section we recall and program a simple Euler method for numerical approximation of first-order integration problems. Let x be the function we want to approximate, let $\frac{dx}{dt} = x'$ be the derivative of x and let $x(0)$ be the initial condition (known). The Euler method is based on the tangent line approximation :

$$x(t) \simeq x'(t^*) \times (t - t^*) + x(t^*) \tag{3}$$

This holds for t close to t^*, i.e., $t - t^* = \Delta_t$, with Δ_t "small".

— Write a function that applies the Euler method iteratively to compute $x(t)$ for $t \in [0, t_{max}]$. This function requires as input the initial conditions $x(0)$, the step of the integrator Δ_t value, t_{max} the last temporal value and an instance of the derivative function to integrate. The derivative function takes as input the current value t^* and $x(t^*)$ to compute $x'(t^*)$. This function should work in a vectorial way (use numpy !).

— Test your method with a simple case (e.g., $x'(t) = -x(t)$ with $x(0) = [0, 3, 5]$).

— If one day you need to work with numerical integration of differential equations, use more accurate numerical methods and use dedicated libraries.

3 Constant weight matrix

In this section we consider that $W_{i,j} = k$, $\forall i, j$. In addition, we take $N = 10$, $g = 1$, $\tau = 10$ and the the initial activities are taken uniformly in $[-0.5, 0.5]$.

— Write a function that computes the derivative of x as defined in Equation 1, this function takes as parameters $x(t)$, t, g and W.

— Write a function that takes as parameters $x(t)$, t, g and W, calls the function defined in the previous step and returns the derivative function that only takes as an input $x(t)$ and t (the other parameters being set within the function). To do this use the function called "partial" that is in the "functools" library. This new function will be used to send a suitable derivative function to the Euler method.

— Make different tests for $k \in -1, -0.5, 0.5, 1$ and plot the results.

— For one of the values of k, take a different value for τ.

— Plot also in each figure the average activity of the neural network : $m(t) = \frac{1}{N}\sum_{i=1}^{N} x_i(t)$.

— Plot in a diagram $m(t + \Delta t)$ as a function of $m(t)$

4 Random weight matrix

In this case we consider that the values of W are i.i.d. and follow a gaussian distribution $\mathcal{N}(0, 1)$. Do the same as in the previous example, for different values of g from 0 to 10.

Acknowledgement

This practical course is subtantially based on a former practical course for matlab made by Hedi Soula.