# Kohonen neural networks

Brunello Tirozzi[1,2]

[1] Enea Research Center, Frascati, Italy
[2] Department of Physics, University La Sapienza of Rome, Italy

October 13, 2019

## 1 The Kohonen network

### An intuitive description

The Kohonen network is formed by a single layered neural network.

1. Low dimension of the network and its simple structure.
2. Simple representation of clusters by means of vectors associated with each neuron.
3. Topology of the input data set is somehow mapped in the topology of the weights of the network.
4. Learning is unsupervised.
5. Self-organized property.

1. $N$ units (neuron) $i$ in a geometrical configuration.
2. According to the geometry each has a definite number of neighbors.
3. To each neuron $i$, is associated a number $\omega_i$ called weight
4. There is a sequence of data $x_k$ distributed in an interval $A$ randomly.

### Learning procedure

1. Fix the neurons number $N$
2. choose randomly initially,$n = 0$ ,the $\omega_i$ , $(1 \leq i \leq N)$
3. extract randomly $x(1)$ from the data set

4. compute the distances from the weights of all the neurons

$$|\omega_i(0) - x(i)|, i = 1, \ldots N$$

5. choose the neuron $v$ such that

$$|\omega_v - x(1)| \leq |\omega_i - x(1)|$$

for $i \neq v$. $v$ is the index of the winner neuron, i.e. the neuron with the weight which has the minimum distance from the data |x(1)|.

6. update only the weight of the winner neuron

$$\omega_v = \omega_v + \eta(1)(x(0) - \omega_v)$$

7. iterate $n = n + 1$

**The learning process is the sequence** $\omega(n)$

**The S.O. (self-organizing property) is the almost everywhere convergence of** $\omega(n)$

**General case**

1. The Kohonen network single layer of output units $\mathcal{O}_i$, $i = 1, ..., N$ each being fully connected to a set of inputs $\xi_j(n)$, $j = 1, ..., M$.
2. An $M$-dimensional weight vector $\omega_i(n)$, $\omega_i(n) = (\omega_{ij}(n), j = 1, ..., M)$ is associated with each neuron.
3. $n$ indicates the $n$-th step of the algorithm.
4. the inputs $\xi_j(n)$, $j = 1, ..., M$ are independently chosen according to a probability distribution $f(x)$.
5. For each input $\xi_j(n)$, $j = 1, ..., M$ we choose one of the output units, called the *winner*, the output unit with the smallest distance between its weight vector $\omega_v(n)$ and the input

$$||\omega_v(n-1) - \xi(n)||$$

6.

$$\omega_{ij}(i+1) = \omega_{ij}(n) + \eta(n)\Gamma(i,v)\overline{I}(\omega_v(n), \xi(n+1))\ (\xi_j(n+1) - \omega_{ij}(n)) \qquad (1.1)$$

5.1

$i = 1, ..., N$ and $j = 1, ..., M$

7. where $\eta(n)$ is the positive learning parameter $\eta(0) < 1$
8. $\Gamma(i, v)$ is a non-increasing function of $|i - v|$, the distance among the neuron $i$ and $v$

where $\eta(n)$ is the positive learning parameter $\eta(0) < 1$, $\eta(n) \geq \eta(n+1)$ and on the lattice where the neurons of the network are located.

$\Gamma = \Lambda$ where:

$$\Lambda(i, v) = \begin{cases} 1 \text{ if } |i - v| \leq s \\ 0 \text{ otherwise} \end{cases} \qquad (1.2)$$

5.3

where $|.|$ represents the distance between the neuron $i$ and the *winner* neuron $v$.

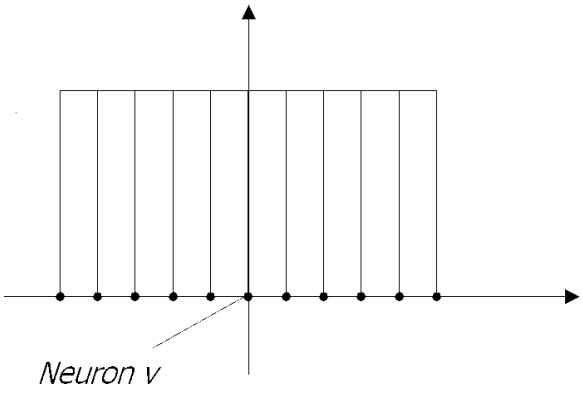


fig6.2

Figure 1: Neighborhood function $\Lambda(i, v)$.

$$h(i, v, n) = \exp\left(-\frac{|i - v|^2}{\sigma(n)^2}\right) \tag{1.3}$$ 5.4

where $\sigma(n)$ is a decreasing function.

$$\sigma(n) = \sigma_i \left(\frac{\sigma_f}{\sigma_i}\right)^{\frac{n}{n_{max}}}$$

where $n_{max}$ is the maximum number of iterations of the algorithm and $\sigma_f$, $\sigma_i$ are respectively the final and initial value of the parameter $\sigma$

We see that the algorithm does not even converge in mean (and so also not a.e.) if:

1. $\eta(n)$, the learning parameter, decreases too fast.
2. The neighborhood functions ($\Lambda(i, v)$ , or $h(i, v, n)$ have a range of action too small or too large.

I We analyzed the following $\eta(n)$:

1. $\eta(n) = \frac{1}{\sqrt{log(n)}}$
2. $\eta(n) = \frac{1}{log(n)}$
3. $\eta(n) = \eta_i(\frac{\eta_f}{\eta_i})^{\frac{n}{n_max}}$
4. $\eta(n) = \frac{1}{\sqrt[3]{n}}$

(where $\eta_i$ and $\eta_f$ are respectively the initial and final values of the function $\eta$ and $n_{max}$ the maximum number of iterations). For all these cases we have convergence in mean, but for each case there is a different accuracy.

Choosing

1. $\eta(n) = \eta_i(1 - \frac{n}{n_max})$
2. $\eta(n) = \frac{\sqrt{6*log(n)}}{\sqrt{(n)}+1}$

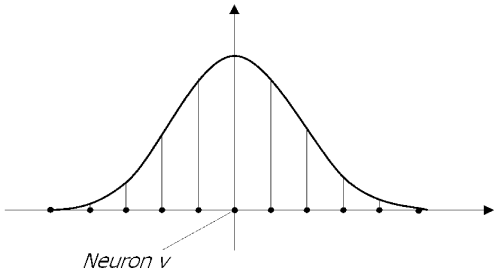


Figure 2: Neighborhood function $h(i, v, n)$. fig6.3

In the case of the $h$ neighborhood function the best choices of $\sigma_i$ and $\sigma_f$ are the following:

$$\sigma_i = \frac{\sqrt{N}}{2} \tag{1.4}$$ sigma1

$$\sigma_f = 0.01 \tag{1.5}$$ sigma2

where $N$ is the number of weights.

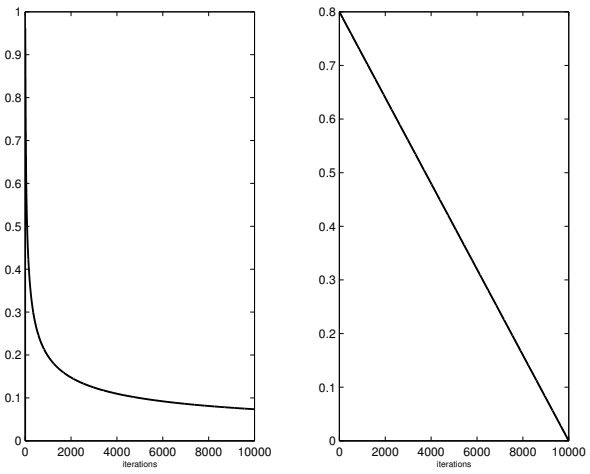


Figure 3: On the left we have $\eta(n) = \frac{\sqrt{6*log(n)}}{\sqrt{(n)}+1}$, on the right $\eta(n) = \eta_i(1 - \frac{n}{n_{max}})$. fig6.4

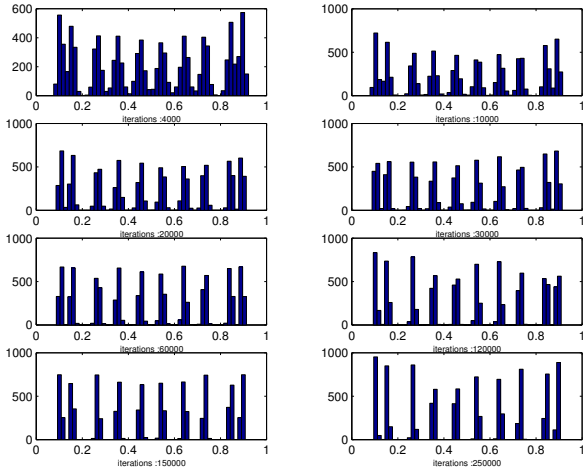


Figure 4: Histograms in the case of $\eta(n) = \eta_i(1 - \frac{n}{n_{max}})$, uniformly distributed data and for different numbers $M$ of iterations. fig6.5