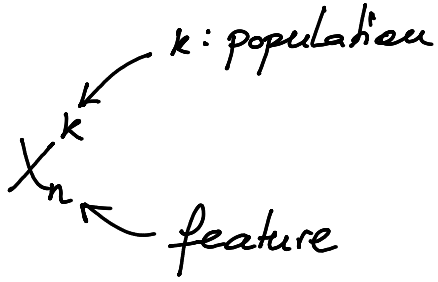
⑧ Data



Population : A set of x_n that have similar features

Feature : A property that identifies a particular population and differentiates it with other populations.

↳ Discrimination

Example: Star catalogue

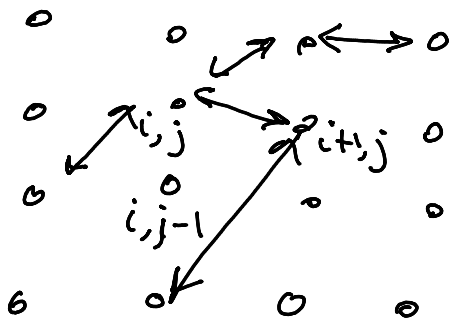
X_n : Color – brightness – age – distance

X^k : WDs, RG, NS, BH, HB, ...

⊗ Self Organized Feature Maps (SOFM)

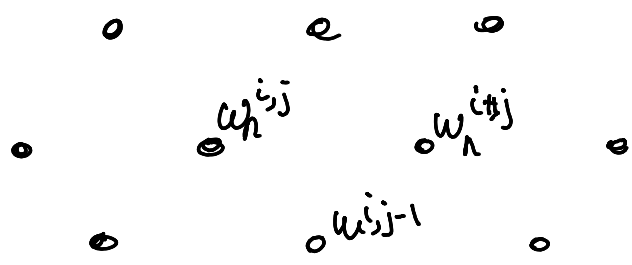
SOFM is a classification technique based on the field of neural networks.

In this method / technique we define a topological network with nodes having a relation of proximity. Therefore, we will define a distance or metric that quantifies that proximity using a coordinate space in N dimensions.



The nodes are kept fixed in the topological network. Their distances do not change.

Each node, or neuron, is identified with a weights vector "w_n".



Initially the weights vector is completely uncorrelated to the characteristics of our populations. If a feature vector component is a representative measure of one of the characteristics of the sample then that component average can be used to quantify if a given sample x_n^k is closer to one population or another.

Therefore, the most convenient is to initialize the weights network using the populations averages

aug sigmas.

$$w_n^{i,j} = \langle x_n^k \rangle + Rndm\left(\sigma_n^k\right)^{i,j}$$

↑ Average of all data I want to use (for initialization)

↑ Sigma of all data I want to use

↑ It should contain all populations.

⊛ Winning neuron (n_g)

For any particular element "n" from a population "k", there is a neuron that its proximity to the x_n^k is the highest.

If we define a distance:

$$\min \left\| x_n^k - w_n^{i,j} \right\| \longrightarrow w^{I,J}$$

I, J is the winning neuron, n_g.

⊛ Updating the network

The SOFM consists on updating in an iterative way the weights of the network, $w_k^{(i)}$, so that a group of nodes, or neurons, which have a relation of proximity, specialize to identify a particular population "k", in such a way that the winning neuron will fall with a high probability within that group of neurons.

The key of SOFM is then to iteratively update the weights of the winning neuron so that it gets even closer to the x_n vector that was producing the neuron to win.

x_n vector (or pattern)

In order to create domains or regions with a proximity relation that identify with a particular population "k", it is necessary that we update the weights of the winning neuron and those neurons that are around.

We may introduce then the following expression:

$$w_n^{ij}(t+1) = w_n^{ij}(t) + \alpha(t) * h(d_{ig}, RG)) \cdot |X_n^k - w_n^{ij}|$$

- t: It is the iteration number. At each iteration all data trainning samples are shown to the network.

- $\alpha(t) < 1$: Learning rate. It is decreasing with increasing iteration number.

- d_{ig}: It is the topological distance between the neuron we are updating and the winning neuron

- $h(d_{ig}, R(t))$: It is the neighborhood function. It defines a scaling factor that decreases as the distance d_{ig} gets lesser.

We may define any arbitrary function that decreases with distance

For example:

$$h(d_{ig}, R(t)) = \exp\left(-d_{ig}/R(t)\right)$$

- $R(t)$: It is the neighborhood radius. And it defines the influence of "h".

Tipically we stert with a radius that is large enough to influence a large number of neurons and we then stert decreasing its value for each iteration.

$$R(t) = R_0 + \frac{I_t}{I_f}(R_f - R_0) \qquad R_f < R_0$$

- I_t: Iteration number
- I_f: Total number of iterations
- R_f: Final radius
- R_0: Initial radius

We do it in a similar way with the learning rate $\alpha(t)$

$$\alpha(t) = \alpha_0 + \frac{I_t}{I_f}(\alpha_f - \alpha_0)$$

That helps to smooth the streight of the update as the iterative process evolves.

⊛ Trainning set and populations.

The trainning set MUST be built with representative datasets from ALL populations "k". If possible, same number of datasets "N" for each "k".

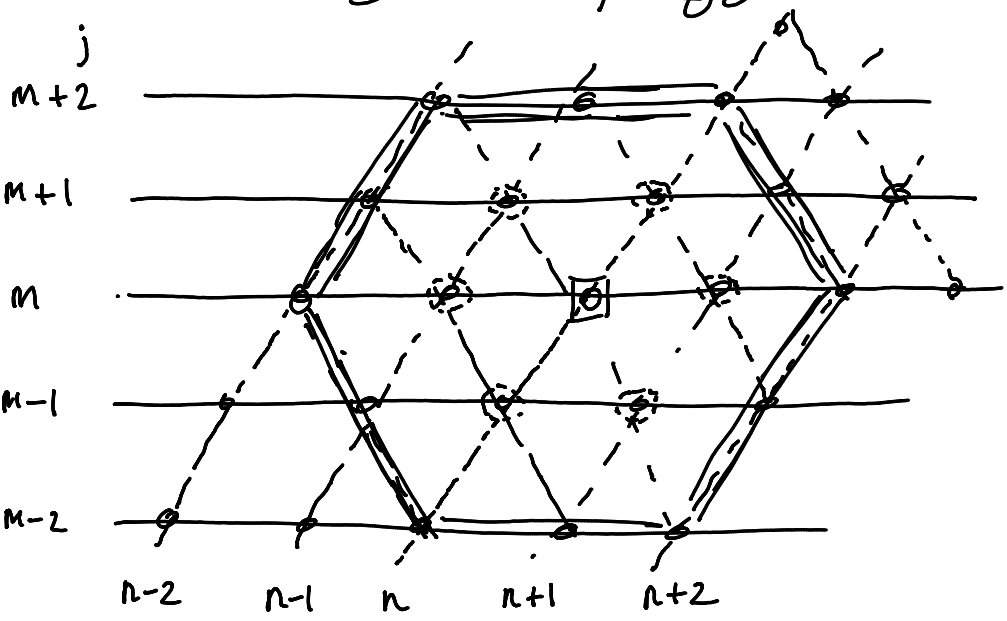
This might not be always guaranteed

Example → Background detector data.

Those datasets are given for trainning. Then, we need a calibration data set to identify the group of neurons that got specialized on identifying a particular group.

Github codes

Hexagonal topology



$D \equiv$ Hexagon dimension. Or number of neighbours.
$j \equiv m \pm j$. Row of hexagone nodes.

Points on each row: $N_p = 2D + 1 - j$

$X \equiv$ node coordinate

Top hexagon:
- Starts always at $X = n - D$

Bottom hexagon:
- Starts at $X = n - D + j$

For the training set we calculate the average and deviation for each variable.

We get the $\langle x_j \rangle$ and σ_j

The network is initialized using the mean and sigma values.

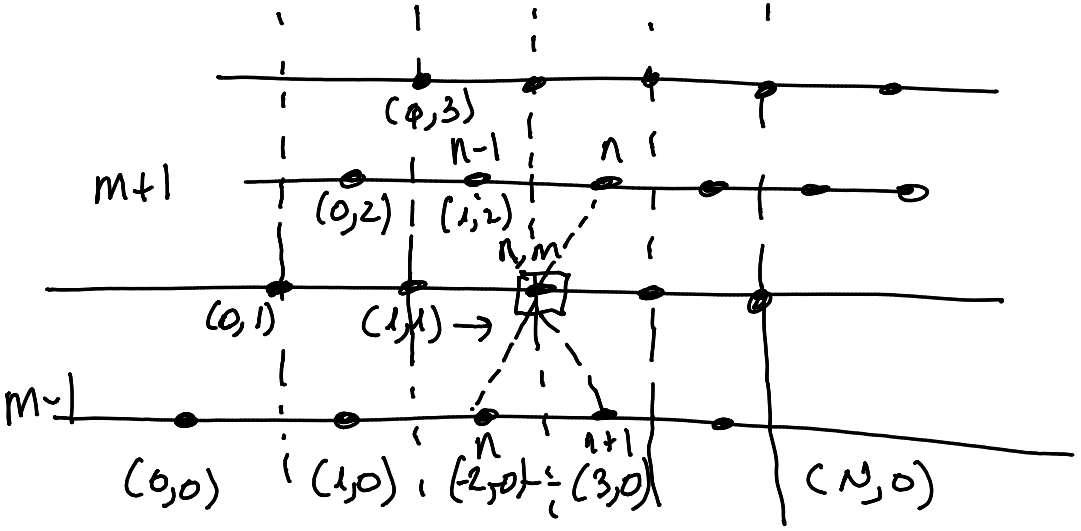
In each iteration the weights vector is updated as follows.

$$w_{ik}(t+1) = w_{ik}(t) + \alpha(t) * h(d_{is}, R(t)]$$

$$w_{ik}(t+1) = w_{ik}(t) + \alpha(t) \cdot h(d_{ig}, R(t)] \cdot (x_k^n - w_{ik})$$

$$h(d_{ig}, R(t)) = \exp(-d_{ig}/R(t))$$

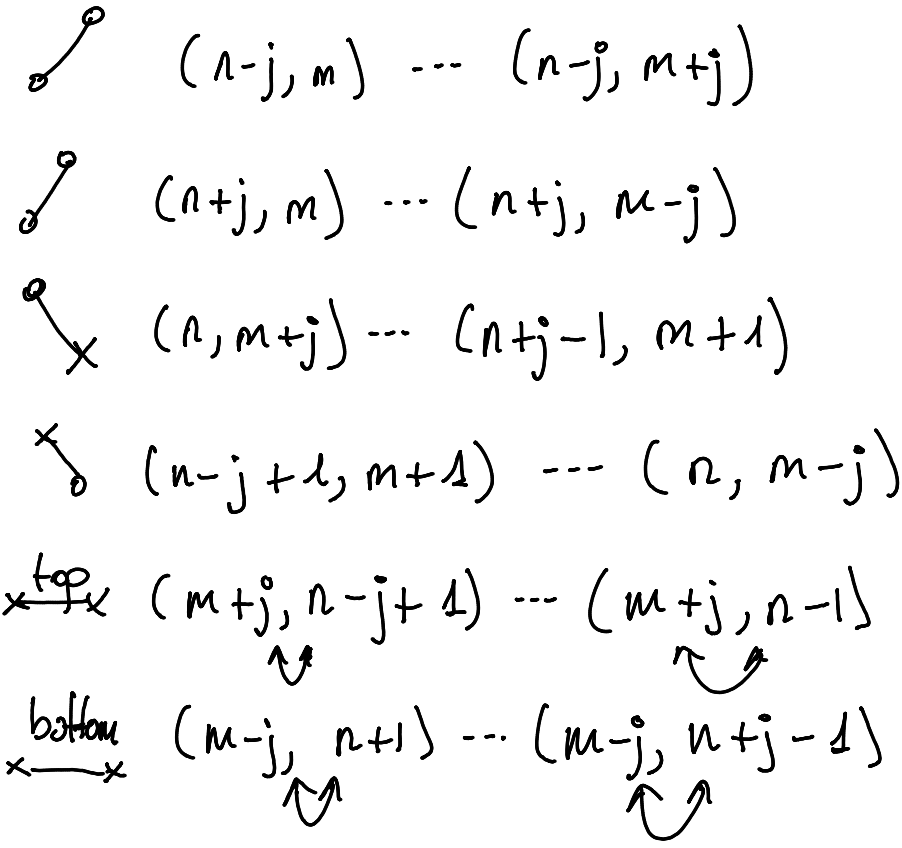
$$R(t) = R_0 + \frac{I_t}{I_f}(R_f - R_0)$$



$(n-1, m)$ $(n+1, m)$

$(n-1, m+1)$ $(n, m+1)$

$(n, m-1)$ $(n+1, m-1)$



Left (n-j, m) --- (n-j, m+j)
Right (n+j, m) -- (n+j, m-j)
Right (n, m+j) -- (n+j-1, m+1)
Left (n-j+1, m+1) -- (n, m-j)
Top (n-j+1, m+j) -- (n-1, m+j)
Bottom (n+1, m-j) --- (n+j-1, m-j)