Lec 6 : Self-Organizing Maps

Previous

Learning NN

① Supervised → desired output

MLP RBFN

⇕

error signal → adjust weight

② Without a teacher → reinforcement Learning → $\boxed{\text{Critic: } \left\{ \begin{array}{l} \text{reward} \\ \text{penalty} \end{array} \right. \text{ signal}}$

↓

adjust weight

③ unsupervised Learning → { no teacher, no critic } (no feed back)

↓

Biologically more adequate (so is RL)

↓

Self-Organized Maps (SOMs)

↓

Motivation (Brain) topographic map

change by how you train your brains

map → (SOM)

output: motor cortex ← input: somatosensory cortex

↓

learn by 后天

different regions, different functions → well-organized !

Topology-conserving mapping achieved by SOMs

① 2 layers { input
output

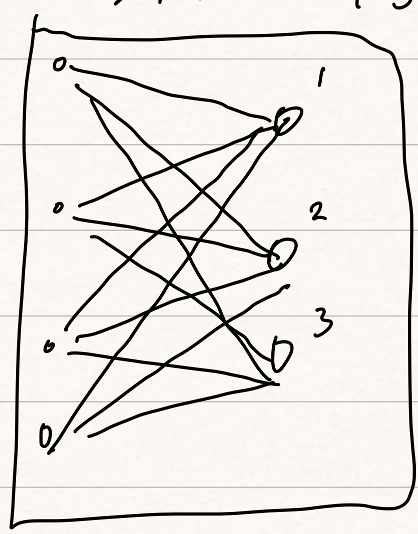
② fully connected

③ topology (neighbourhood relation) → unique property in SOMs

↓

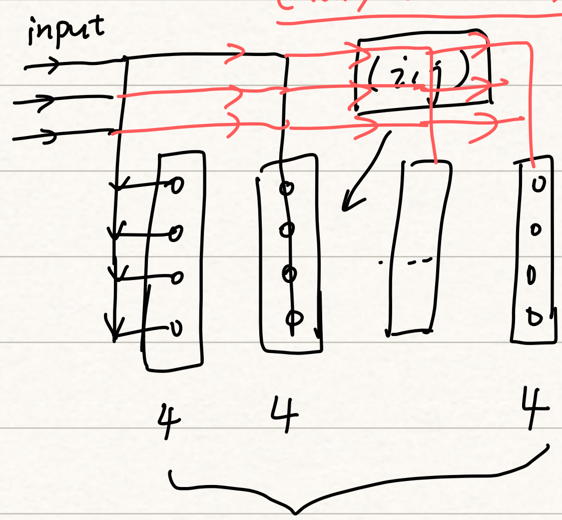
defined in output layer

SOM 4×3



3×16

all input will be transport to all neurons (fully connected)

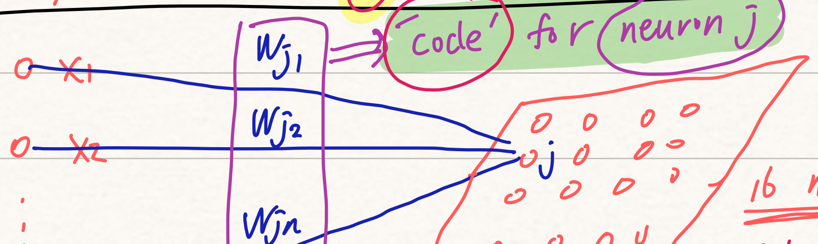


Principle of SOMs

input signal ⟶ output (1 or 2 dim)

(arbitrary dimension)

① 'code' for neuron j → same dim as input signal ⇒ n-dim 'codes'



2d array of neuron

$o \; X_n$

n-dimension input

Purpose of weight

underlying codes of neurons

$\boxed{\text{Structure of SOM}}$

$\boxed{\text{not like Mc \& Pitts Model !}}$

$\Updownarrow$

$$\boxed{Y_k = \phi\left(\sum_{i=1}^{n} W_{ki} X_i + b_k\right)}$$

has topological location

2 output neurons (array)

1. no synaptic weight!
2. related to each other by 2D-map
   - neighbourhood play similar role!

Algorithm: (Sequential Learning)

j-th neuron code!

1. randomly initialize $\{ \underline{W}_j = (W_{j1}, \dots W_{jn}) \text{ for } j = 1, 2, \dots, 16 \}$
2. select $\underline{X}(i)$ from traing set (sample) randomly or one by one
3. find winner neuron $j \leftrightarrow \underline{X}(i)$ (Competition)
4. update { winner, neighbourhood } (Cooperation) & Adjust $\underline{W}$
5. adjust paras { Learng rate, neighbourhood f^n }

① Find winner

↓

Best match neuron $\text{index}(\underline{x}) = \arg\min_{j} \| \underline{x} - \underline{w_j} \|_2$ $\boxed{\text{Eudidean}}$

$\text{index}(\underline{x}) \rightarrow$ the winner's index

↕

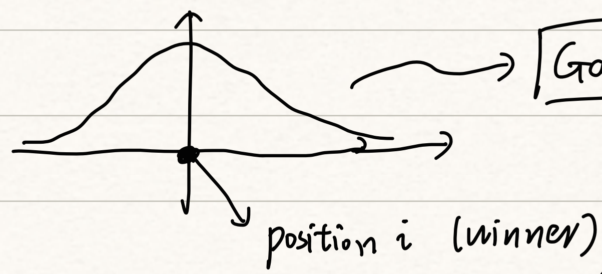
closet to input

② Cooperation

↓

The closer to winner, the more impact it receives!

1. symmetric
2.



→ $\boxed{\text{Gaussian}}$

$$h_{j,i(x)} = \exp\left(-\frac{d_{j,i(\underline{x})}^2}{2\sigma^2}\right)$$

distance between $(j, i(\underline{x}))$

σ → effective width (量纲)

measure the $\boxed{\text{effectiveness}}$

$$d_{j,i(\underline{x})}^2 = \| j - i(\underline{x}) \|_2^2$$

$$= (j_1 - i_1(\underline{x}))^2 + (j_2 - i_2(\underline{x}))^2$$

if array, then

$$\Rightarrow \begin{cases} j = (j_1, j_2) \\ i(\underline{x}) = (i_1(\underline{x}), i_2(\underline{x})) \end{cases}$$

BIG

$$\text{if} \sim \begin{cases} d_{j,i}(\underline{x}) = 6 \Rightarrow \boxed{h_{j,i} = 0.6} \\ d_{j,i}(\underline{x}) = 26 \Rightarrow h_{j,i} = 0.14 \end{cases}$$

if set $\sigma = 1$

SMALL

0.61

0.61 winner 0.61 0.14

0.61

0.14

Note

BIG sigma σ

↓

more neurons will be affected

↓

neighbour

σ: effective width

$$\sigma(n) = \sigma_0 \exp\left(-\frac{n}{\tau_1}\right) \rightarrow \text{time-vary width}$$

τ_1: decay rate of effective width

decrease with time

⇓

☆ with time goes on, less & less neurons are influenced!

Note

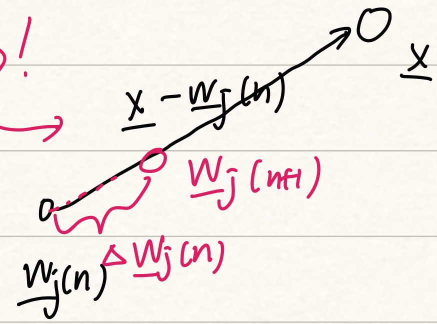
③ Adaption ($\underline{x}$ is train sample, $i(\underline{x}) \rightarrow$ winner index)

$$\boxed{w_j(n+1) = w_j(n) + \eta(n)\, h_{j,i(\underline{x})}\, (x - w_j(n))}$$

Geometrically

$$\Delta w_j(n) = \eta(n)\, h_{j,i(\underline{x})}(n)\, (x - w_j(n))$$

Move towards $\underline{x}$!



All neurons in the neighbourhood of winner will all move towards $\underline{x}$

$\Downarrow$

Synaptic weight vector tend to follow the dist. of input

$$\boxed{\eta(n) = \eta_0 \exp\left(-\frac{n}{\tau_2}\right)}$$

Decay of learning rate

τ_2: decay rate of η

Alg.

Step 1 phase 1 (don't want convergence)

$$\begin{cases} \eta(n): & \eta_0 = 0.1 \quad \tau_2 = T \\ \sigma(n): & \sigma_0 = \dfrac{\sqrt{M^2+N^2}}{2} \end{cases}$$

matrix $M \times N$

phase 2: convergence phase

① $\eta(n) \to$ maintain small

② neighbourhood f^n $\Rightarrow$ only the nearest will be affected

---

1. Randomly initialize
2. sample x
3. winner neuron k
4. update

$$W_j(n+1) = W_j(n) + \eta(n)\, h_{j,k}(n)\, (x - W_j(n))$$

Terminate: Convergence

How we use it?

$\downarrow$

① Visualize

② $\| \underline{x}(k) - \underline{w}_j \| = \min_i \| \underline{x}(i) - \underline{w}_j \|$

$\downarrow$

j-th neuron's winner signal (k)

$\downarrow$

then mark j-th neuron with k

$\Downarrow$

this map: contexual map

9. min