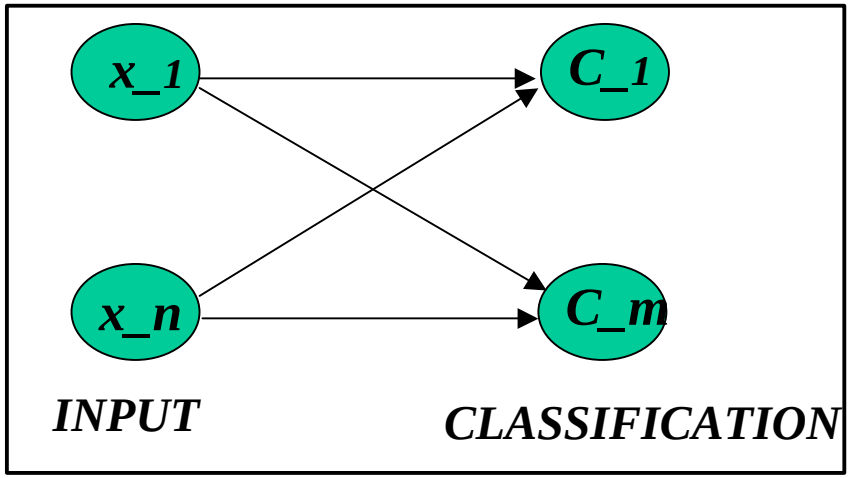
Chapter 4. Neural Networks Based on Competition

- Competition is important for NN
  - Competition between neurons has been observed in biological nerve systems
  - Competition is important in solving many problems

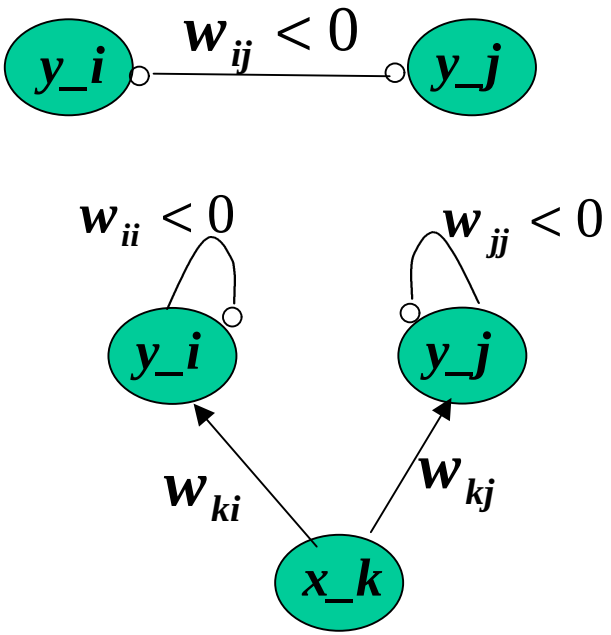
To classify an input pattern into one of the m classes
- idea case: one class node has output 1, all other 0 ;
- often more than one class nodes have non-zero output



  - If these class nodes compete with each other, maybe only one will win eventually (**winner-takes-all**). The winner represents the computed classification of the input

- Winner-takes-all (WTA):
  - Among all competing nodes, only one will win and all others will lose
  - We mainly deal with single winner WTA, but multiple winners WTA are possible (and useful in some applications)
  - Easiest way to realize WTA: have an external, central arbitrator (a program) to decide the winner by comparing the current outputs of the competitors (break the tie arbitrarily)
  - This is biologically unsound (no such external arbitrator exists in biological nerve system).

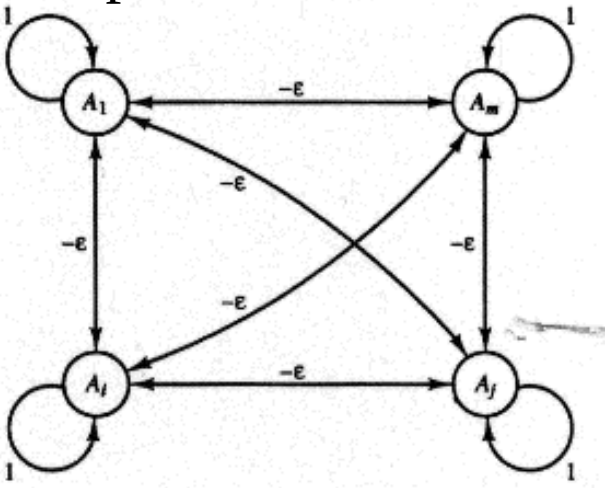
- Ways to realize competition in NN
  - **Lateral inhibition** (Maxnet, Mexican hat)
    output of each node feeds to others through inhibitory connections (with negative weights)
  - **Resource competition**
    - output of x_k is distributed to y_i and y_j proportional to w_ki and w_kj, as well as y_i and y_j
    - self decay
    - biologically sound



- Learning methods in competitive networks
  - Competitive learning
  - Kohonen learning (self-organizing map, SOM)
  - Counter-propagation net
  - Adaptive resonance theory (ART) in Ch. 5

Fixed-weight Competitive Nets

- **Maxnet**
  - Lateral inhibition between competitors



$$\text{weights} : w_{ij} = \begin{cases} 1 & \text{if } i = j \\ -e & \text{otherwise} \end{cases}$$

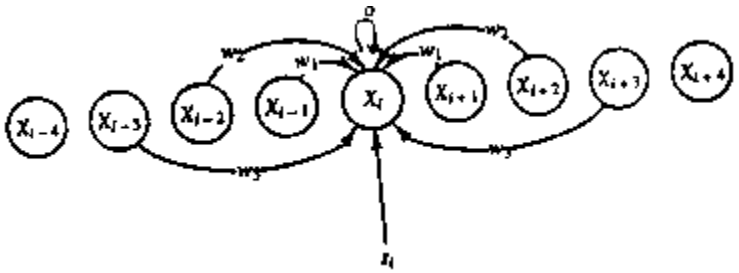
activation function :

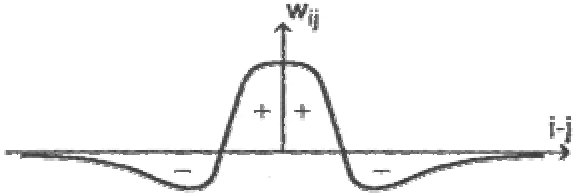
$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

- Notes:
  - Competition:
    - iterative process until the net stabilizes (at most one node with positive activation)
  - $0 < e < 1/m$, where m is the # of competitors
    - e too small: takes too long to converge
    - e too big: may suppress the entire network (no winner)

Mexical Hat

- **Architecture**: For a given node,
  - close neighbors: cooperative (mutually excitatory , w > 0)
  - farther away neighbors: competitive (mutually inhibitory,w < 0)
  - too far away neighbors: irrelevant (w = 0)





- Need a definition of distance (neighborhood):
  - one dimensional: ordering by index (1,2,…n)
  - two dimensional: lattice

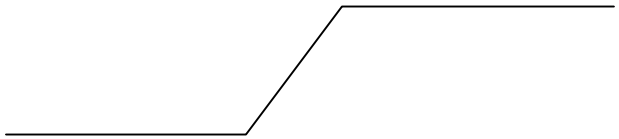
weights

$$w_{ij} = \begin{cases} c_1 & \text{if} \quad \text{distance}(i, j) < k \\ c_2 & \text{if} \quad \text{distance}(i, j) = k \\ 0 & \text{if} \quad \text{distance}(i, j) > k \end{cases}$$

activation function

$$f(x) = \begin{cases} 0 & \textit{if} \quad x < 0 \\ x & \textit{if} \quad 0 \le x \le \max \\ \max & \textit{if} \quad x > \max \end{cases}$$

ramp function :



example:

$$\boldsymbol{x}(0) = (0.0, 0.5, 0.8, 1.0, 0.8, 0.5, 0.0)$$
$$\boldsymbol{x}(1) = (0.0, 0.38, 1.06, 1.16, 1.06, 0.38, 0.9)$$
$$\boldsymbol{x}(2) = (0.0, 0.39, 1.14, 1.66, 1.14, 0.39, 0.0)$$

- Equilibrium:
  - negative input = positive input for all nodes
  - winner has the highest activation;
  - its cooperative neighbors also have positive activation;
  - its competitive neighbors have negative activations.

Hamming Network

- **Hamming distance** of two vectors, $\boldsymbol{x}$ and $\boldsymbol{y}$ of dimension n,
  - Number of bits in disagreement.
  - In bipolar:

$$\boldsymbol{x} \cdot \boldsymbol{y} = \boldsymbol{a} - \boldsymbol{d}$$

where : $\boldsymbol{a}$ is number of bits in agreement in $\boldsymbol{x}$ and $\boldsymbol{y}$

$\boldsymbol{d}$ is number of bits different in $\boldsymbol{x}$ and $\boldsymbol{y}$

$\boldsymbol{d} = \boldsymbol{n} - \boldsymbol{a}$ hamming distance

$$\boldsymbol{x} \cdot \boldsymbol{y} = 2\boldsymbol{a} - \boldsymbol{n}$$

$$\boldsymbol{a} = 0.5(\boldsymbol{x} \cdot \boldsymbol{y} + \boldsymbol{n})$$

larger $\boldsymbol{x} \cdot \boldsymbol{y} \Rightarrow$ larger $\boldsymbol{a} \Rightarrow$ shorter Hamming distance

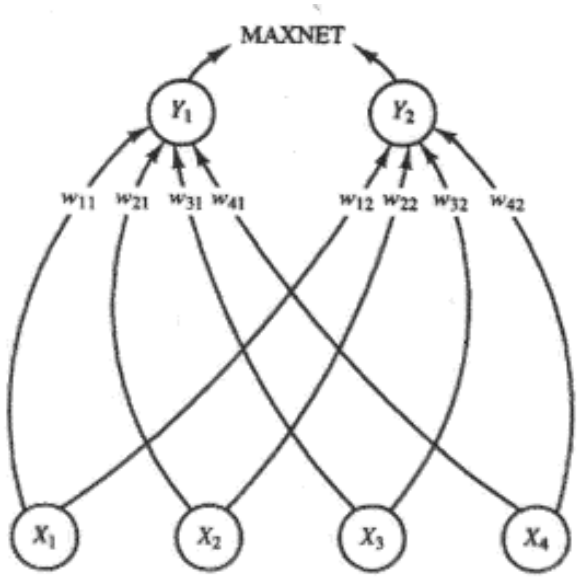
- Suppose a space of patterns is divided into k classes, each class has an exampler (representative) vector $\boldsymbol{e}_j$.
- An input $\boldsymbol{x}$ belongs to class i, if and only if $\boldsymbol{x}$ is closer to $\boldsymbol{e}_i$ than to any other $\boldsymbol{e}_j$, i.e., $\boldsymbol{x} \cdot \boldsymbol{e}_i \geq \boldsymbol{x} \cdot \boldsymbol{e}_j \quad \forall \boldsymbol{j} \neq \boldsymbol{i}$
- Hamming net is such a classifier:
  - Weights: let $\boldsymbol{Y}_j$ represent class j

    $$\boldsymbol{w}_{\bullet j} = 0.5\boldsymbol{e}_j, \quad \boldsymbol{b}_j = 0.5\boldsymbol{n}$$

  - The total input to $\boldsymbol{Y}_j$

    $$\boldsymbol{y}\_\boldsymbol{in}_j = \boldsymbol{b}_j + \sum_{\boldsymbol{i}} \boldsymbol{w}_{ij}\boldsymbol{x}_i$$

    $$= \frac{1}{2}(\boldsymbol{x} \cdot \boldsymbol{e}_j + \boldsymbol{n}) = \boldsymbol{a}_j$$



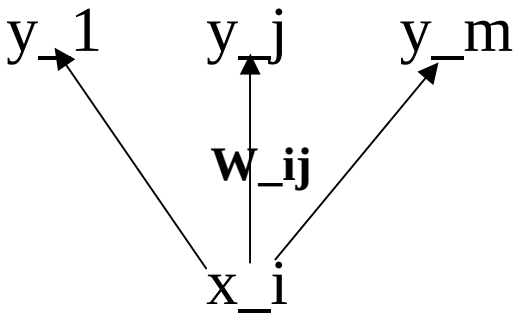
- Upper layer: MAX net
  - it takes the y_in as its initial value, then iterates toward stable state
  - one output node with highest y_in will be the winner because its weight vector is closest to the input vector
- As associative memory:
  - each $\boldsymbol{Y}_j$ corresponds to a stored pattern;
  - pattern connection/completion;
  - storage capacity

    total # of nodes: k

    total # of patterns stored: k

    capacity: k (or k/k = 1)

- Implicit lateral inhibition by competing limited resources: the activation of the input nodes



$$y\_in_j = \sum_i y\_in_{ij}$$

$$y\_in_{ij} = b\,(w_{ij}x_i \cdot y_j) - a\,y_j$$

decay

larger y_j takes larger share of $w_{ij}x_i$

Competitive Learning

- Unsupervised learning
- Goal:
  - Learn to form classes/clusters of examplers/sample patterns according to similarities of these exampers.
  - Patterns in a cluster would have similar features
  - No prior knowledge as what features are important for classification, and how many classes are there.
- Architecture:
  - Output nodes:

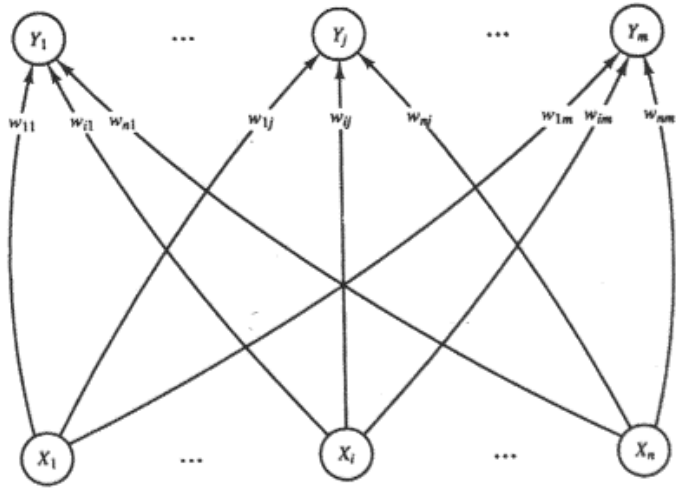
    Y_1,…….Y_m,

    representing the m classes
  - They are competitors

    (WTA realized either by

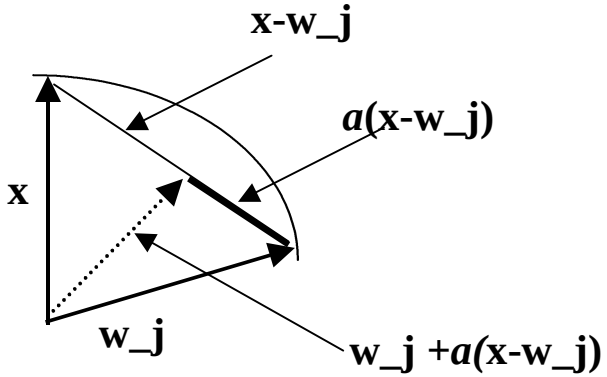
    an external procedure or

    by lateral inhibition as in Maxnet)

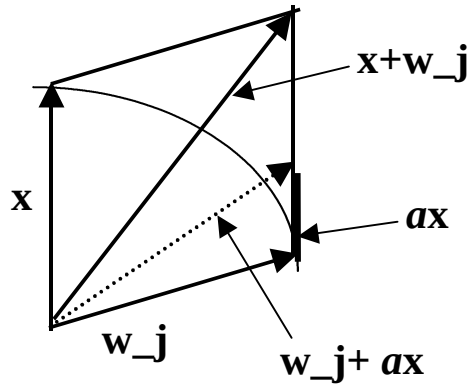


- Training:
  - Train the network such that the weight vector w.j associated with Y_j becomes the representative vector of the class of input patterns Y_j is to represent.
  - Two phase unsupervised learning
- ***competing phase***:
  - apply an input vector $\boldsymbol{x}$ randomly chosen from sample set.
  - compute output for all y: $\boldsymbol{y}_j = \boldsymbol{x} \cdot \boldsymbol{w}_{\bullet j}$
  - determine the winner (winner is not given in training samples so this is unsupervised)
- ***rewarding phase*:**
  - the winner is reworded by updating its weights (weights associated with all other output nodes are not updated)
- **repeat** the two phases many times (and gradually reduce the learning rate) until all weights are stabilized.

- Weight update: $\boldsymbol{w}_{\bullet j} = \boldsymbol{w}_{\bullet j} + \Delta \boldsymbol{w}_{\bullet j}$
  - Method 1: $\Delta \boldsymbol{w}_{\bullet j} = a\ (\boldsymbol{x} - \boldsymbol{w}_{\bullet j})$



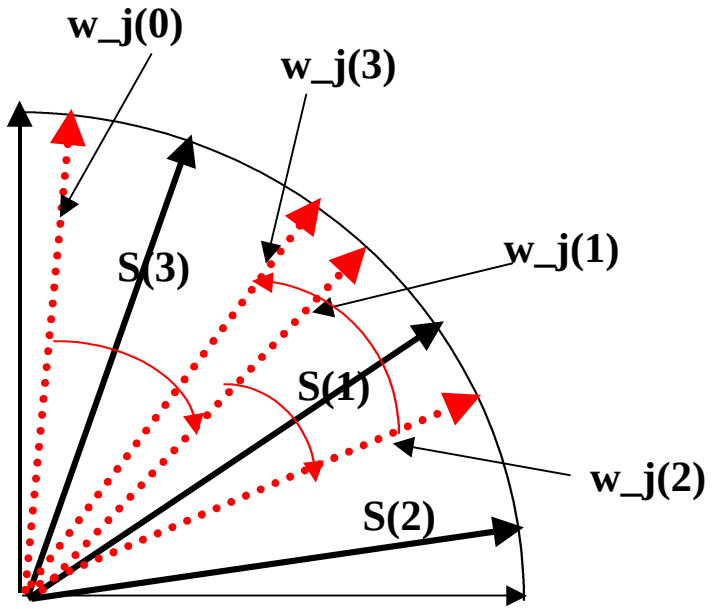
  Method 2 $\Delta \boldsymbol{w}_{\bullet j} = a\ \boldsymbol{x}$



  In each method, $\boldsymbol{w}_{\bullet j}$ is moved closer to $\boldsymbol{x}$
  - Normalizing the weight vector to unit length after it is updated

$$\boldsymbol{w}_{\bullet j} = \frac{\boldsymbol{w}_{\bullet j}}{\left\| \boldsymbol{w}_{\bullet j} \right\|}$$

- $\boldsymbol{w}_{\bullet j}$ is moving to the center of a cluster of sample vectors after repeated weight updates
  - Three examplers: S(1), S(2) and S(3)
  - Initial weight vector w_j(0)
  - After successively trained by S(1), S(2), and S(3), the weight vector changes to w_j(1), w_j(2), and w_j(3)



Examples

- A simple example of competitive learning (pp. 172-175)
  - 4 vectors of dimension 4 in 2 classes (4 input nodes, 2 output nodes)

    S(1) = (1, 1, 0, 0) S(2) = (0, 0, 0, 1)

    S(3) = (1, 0, 0, 0) S(4) = (0, 0, 1, 1)
  - Initialization: $a = 0.6$, weight matrix: $\boldsymbol{W} = \begin{bmatrix} .2 & .8 \\ .6 & .4 \\ .5 & .7 \\ .9 & .3 \end{bmatrix}$
  - Training with S(1)

$$\boldsymbol{D}(1) = \left\| \boldsymbol{S}(1) - \boldsymbol{w}_{\bullet 1} \right\|$$

$$= (.2-1)^2 + (.6-1)^2 + (.5-0)^2 + (.9-0)^2 = 1.86$$

$$\boldsymbol{D}(1) = \left\| \boldsymbol{S}(1) - \boldsymbol{w}_{\bullet 2} \right\| = 0.98, \text{ class 2 wins}$$

$$\boldsymbol{w}_{\bullet 2} = \begin{bmatrix} .8 \\ .4 \\ .7 \\ .3 \end{bmatrix} + 0.6\left(\begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} .8 \\ .4 \\ .7 \\ .3 \end{bmatrix} \right) = \begin{bmatrix} .92 \\ .76 \\ .28 \\ .12 \end{bmatrix} \quad \text{then } \boldsymbol{W} = \begin{bmatrix} .2 & .92 \\ .6 & .76 \\ .5 & .28 \\ .9 & .12 \end{bmatrix}$$

– Similarly, after training with S(2) = (0, 0, 0, 1) , in which class 1 wins, weight matrix becomes

$$W = \begin{bmatrix} .08 & .92 \\ .24 & .76 \\ .20 & .28 \\ .96 & .12 \end{bmatrix}$$

– At the end of the first iteration (each of the 4 vectors are used), weight matrix becomes

$$W = \begin{bmatrix} .032 & .970 \\ .096 & .300 \\ .680 & .110 \\ .980 & .048 \end{bmatrix}$$

– Reduce a

$$a = 0.5 \cdot a = 0.5 \cdot 0.6 = 0.3$$

– Repeat training. After 10 iterations, weight matrix becomes

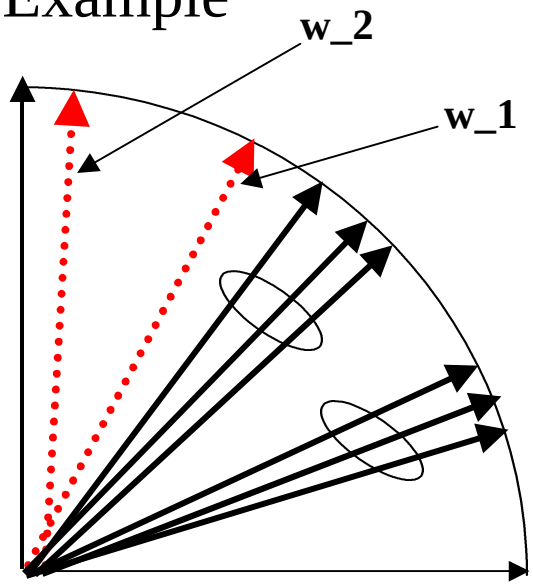
$$W = \begin{bmatrix} 6.7e-17 & 1.000000 \\ 2.0e-16 & .4900000 \\ .5100000 & 2.3e-16 \\ 1.000000 & 1.0e-16 \end{bmatrix} \rightarrow \begin{bmatrix} 0.0 & 1.0 \\ 0.0 & 0.5 \\ 0.5 & 0.0 \\ 1.0 & 0.0 \end{bmatrix}$$

- S(1) and S(3) belong to class 2
- S(2) and S(4) belong to class 1
- w_1 and w_2 are the centroids of the two classes

Comments

1. Ideally, when learning stops, each $\boldsymbol{w}_{\bullet j}$ is close to the centroid of a group/cluster of sample input vectors.
2. To stabilize $\boldsymbol{w}_{\bullet j}$, the learning rate a may be reduced slowly toward zero during learning.
3. # of output nodes:
   - too few: several clusters may be combined into one class
   - too many: over classification
   - ART model (later) allows dynamic add/remove output nodes
4. Initial $\boldsymbol{w}_{\bullet j}$:
   - training samples known to be in distinct classes, provided such info is available
   - random (bad choices may cause anomaly)

Example



$\boldsymbol{w}_{\bullet 1}$ will always win no matter the sample is from which class

$\boldsymbol{w}_{\bullet 2}$ is stuck and will not participate in learning

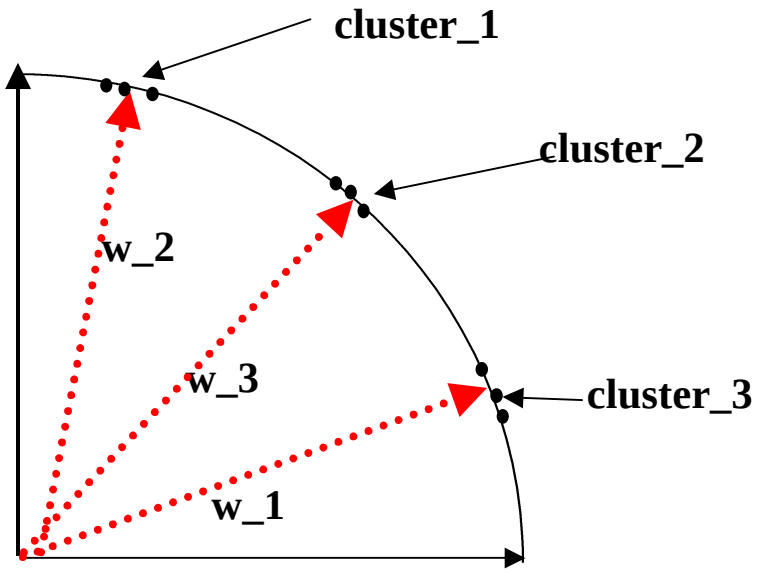
unstuck:

let output nodes have some conscience

temporarily shot off nodes which have had very high winning rate (hard to determine what rate should be considered as “very high”)

Kohonen Self-Organizing Maps (SOM)

- Competitive learning (Kohonen 1982) is a special case of SOM (Kohonen 1989)
- In competitive learning,
  - the network is trained to organize input vector space into subspaces/classes/clusters
  - each output node corresponds to one class
  - the output nodes are not ordered: ***random map***



- The topological order of the three clusters is 1, 2, 3
- The order of their maps at output nodes are 2, 3, 1
- The map does not preserve the topological order of the training vectors

- ***Topographic map***
  - a mapping that preserves neighborhood relations between input vectors, (topology preserving or feature preserving).
  - if $\boldsymbol{x}1$ and $\boldsymbol{x}2$ are two neighboring input vectors (by some distance metrics),
  - their corresponding winning output nodes (classes), i and j must also be close to each other in some fashion
  - one dimensional: line or ring, node i has neighbors $\boldsymbol{i} \pm 1$ or $\boldsymbol{i} \pm 1 \bmod \boldsymbol{n}$
  - two dimensional:grid.

    rectangular: node(i, j) has neighbors:

    $(\boldsymbol{i}, \boldsymbol{j} \pm 1), (\boldsymbol{i} \pm 1, \boldsymbol{j})$, (or additional $(\boldsymbol{i} \pm 1, \boldsymbol{j} \pm 1)$)

    hexagonal: 6 neighbors

- Biological motivation
  - Mapping two dimensional continuous inputs from sensory organ (eyes, ears, skin, etc) to two dimensional discrete outputs in the nerve system.
    - Retinotopic map: from eye (retina) to the visual cortex.
    - Tonotopic map: from the ear to the auditory cortex
  - These maps preserve topographic orders of input.
  - Biological evidence shows that the connections in these maps are not entirely “pre-programmed” or “pre-wired” at birth. **Learning must occur** after the birth to create the necessary connections for appropriate topographic mapping.

SOM Architecture

- Two layer network:
  - Output layer:
    - Each node represents a class (of inputs)
    - Node activation : $y_j = \boldsymbol{x} \cdot \boldsymbol{w}_{\bullet j} = \sum_i w_{ij} x_i$
    - Neighborhood relation is defined over these nodes
      Each node cooperates with all its neighbors within distance R and competes with all other output nodes.
    - Cooperation and competition of these nodes can be realized by Mexican Hat model

    R = 0: all nodes are competitors (no cooperative)
    → random map

    R > 0: → topology preserving map

SOM Learning

1. Initialize $W_{\bullet j}$ for all output nodes, and α to a small value
2. For a randomly selected input sample/exampler x

   determine the winning output node J

   either $x \cdot W_{\bullet J}$ is maximum or

   $$\|x - w_{\bullet J}\| = \sum_i (w_{ij} - x_i)^2 \text{ is minimum}$$
3. For all output node j with $|J - j| \le R$, update the weight

   $$w_{ij} = w_{ij} + \alpha\ (x - w_{ij})$$
4. Periodically reduce α and R slowly.
5. Repeat 2 - 4 until the network stabilized.

Notes

1. Initial weights: small random value from (-e, e)
2. Reduction of a :

   Linear: $a\ (t+\Delta t) = a\ (t) - \Delta a$

   Geometric: $a\ (t+\Delta t) = a\ (t) b$

   Δt may be 1 or greater than 1
3. Reduction of R: $R(t+\Delta t) = R(t) - 1$ while $R(t) > 0$

   should be much slower than a reduction.

   R can be a constant through out the learning.
4. Effect of learning

   For each input x, not only the weight vector of winner J

   is pulled closer to x, but also the weights of J's close neighbors (within the radius of R).
5. Eventually, $w_{\bullet j}$ becomes close (similar) to $w_{\bullet j\pm 1}$. The classes they represent are also similar.
6. May need large initial R

Examples

- A simple example of competitive learning (pp. 172-175)
  - 4 vectors of dimension 4 in 2 classes (4 input nodes, 2 output nodes)

    S(1) = (1, 1, 0, 0) S(2) = (0, 0, 0, 1)

    S(3) = (1, 0, 0, 0) S(4) = (0, 0, 1, 1)
  - Initialization: $a = 0.6$, weight matrix: $\boldsymbol{W} = \begin{bmatrix} .2 & .8 \\ .6 & .4 \\ .5 & .7 \\ .9 & .3 \end{bmatrix}$
  - Training with S(1)

$$\boldsymbol{D}(1) = \left\| \boldsymbol{S}(1) - \boldsymbol{w}_{\bullet 1} \right\|$$

$$= (.2-1)^2 + (.6-1)^2 + (.5-0)^2 + (.9-0)^2 = 1.86$$

$$\boldsymbol{D}(1) = \left\| \boldsymbol{S}(1) - \boldsymbol{w}_{\bullet 2} \right\| = 0.98, \text{ class 2 wins}$$

$$\boldsymbol{w}_{\bullet 2} = \begin{bmatrix} .8 \\ .4 \\ .7 \\ .3 \end{bmatrix} + 0.6\left(\begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} .8 \\ .4 \\ .7 \\ .3 \end{bmatrix} \right) = \begin{bmatrix} .92 \\ .76 \\ .28 \\ .12 \end{bmatrix} \quad \text{then } \boldsymbol{W} = \begin{bmatrix} .2 & .92 \\ .6 & .76 \\ .5 & .28 \\ .9 & .12 \end{bmatrix}$$

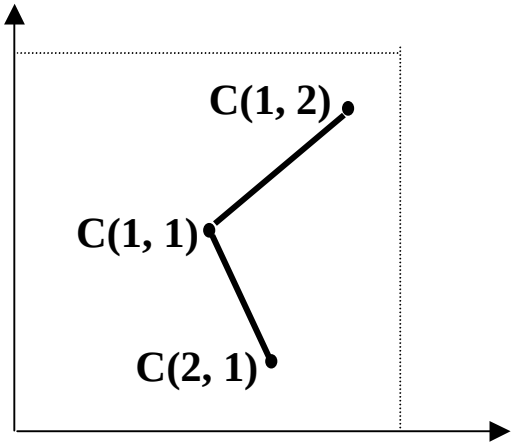
- **How to illustrate Kohonen map**
  - Input vector: 2 dimensional

    Output vector: 1 dimensional line/ring or 2 dimensional grid.

    Weight vector is also 2 dimension
  - Represent the topology of output nodes by points on a 2 dimensional plane. Plotting each output node on the plane with its weight vector as its coordinates.
  - Connecting neighboring output nodes by a line

    output nodes: (1, 1) (2, 1) (1, 2)

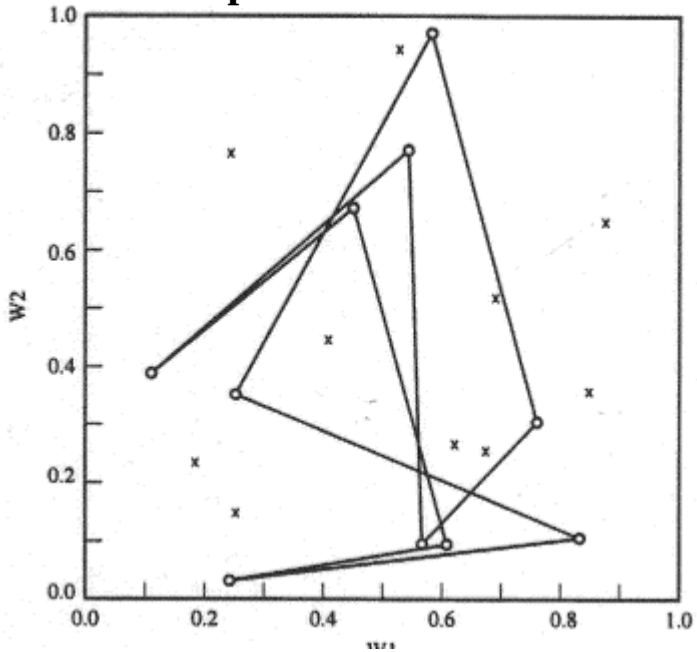
    weight vectors: (0.5, 0.5) (0.7, 0.2) (0.9, 0.9)

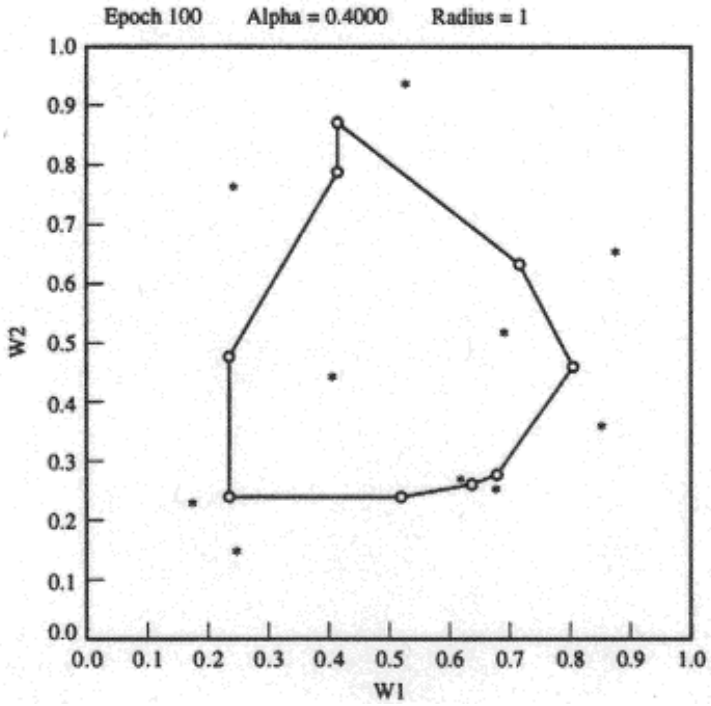


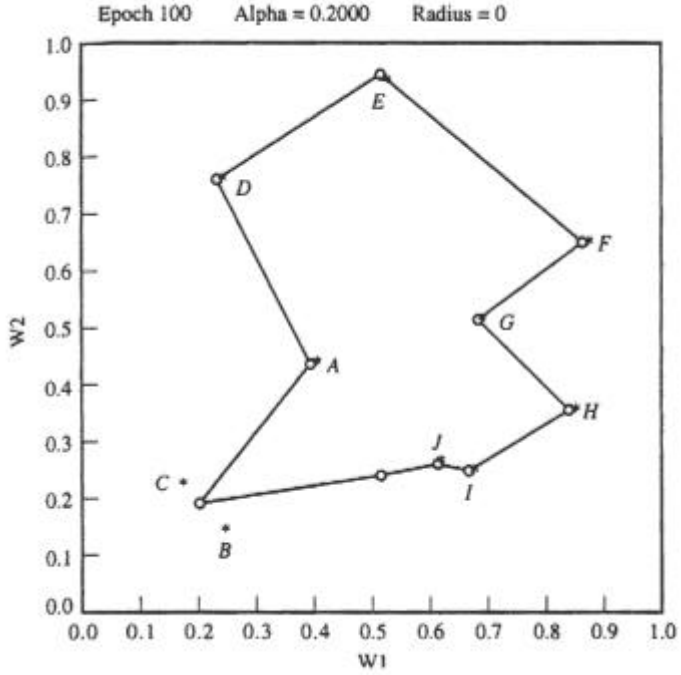
Traveling Salesman Problem (TSP) by SOM

- Each city is represented as a 2 dimensional input vector (its coordinates (x, y)),
- Output nodes C_j form a one dimensional SOM, each node corresponds to a city.
- Initially, C_1, ... , C_n have random weight vectors
- During learning, a winner C_j on an input (x, y) of city I, not only moves its w_j toward (x, y), but also that of of its neighbors (w_(j+1), w_(j-1)).
- As the result, C_(j-1) and C_(j+1) will later be more likely to win with input vectors similar to (x, y), i.e, those cities closer to I
- At the end, if a node j represents city I, it would end up to have its neighbors j+1 or j-1 to represent cities similar to city I (i,e., cities close to city I).
- This can be viewed as a concurrent greedy algorithm

Initial position





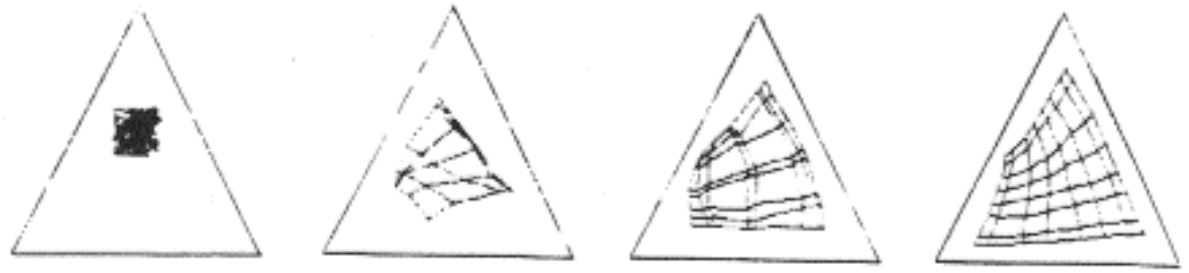


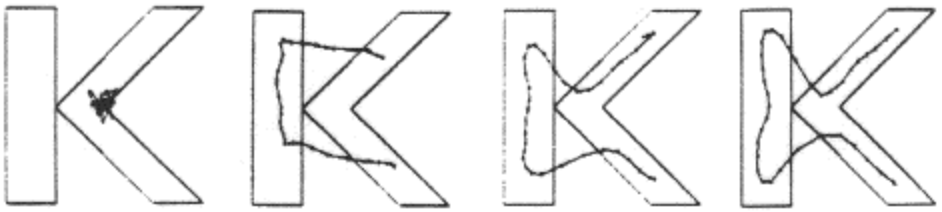
Two candidate solutions:

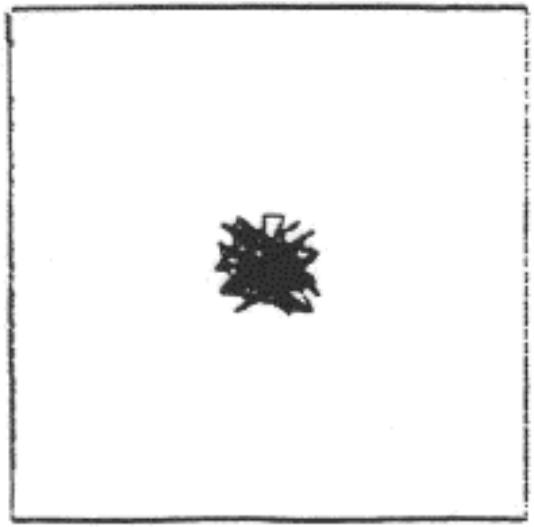
ADFGHIJBC

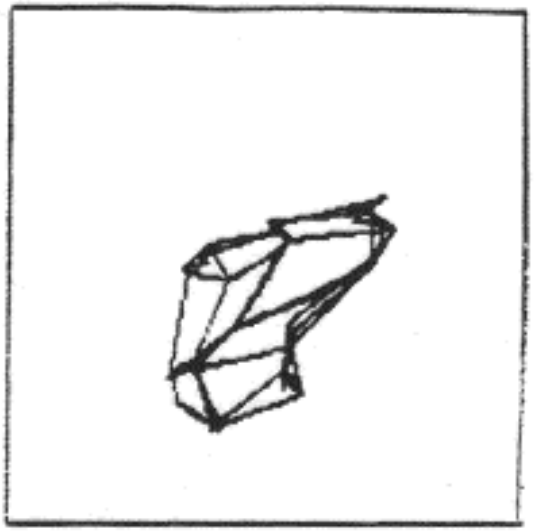
ADFGHIJCB

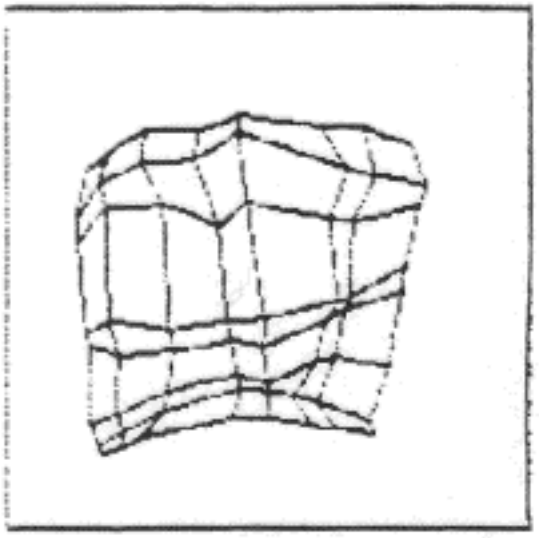
Additional examples

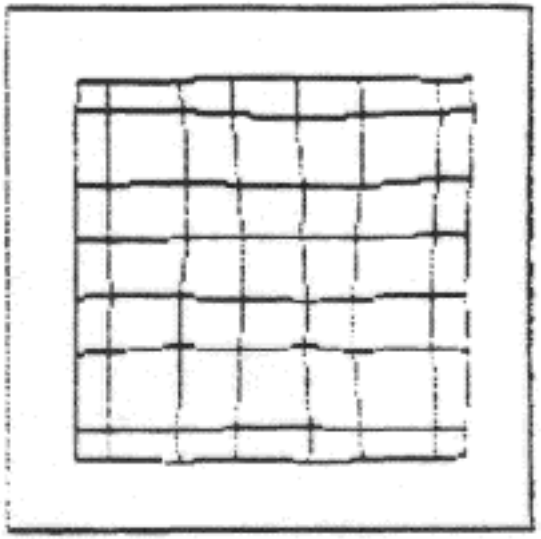






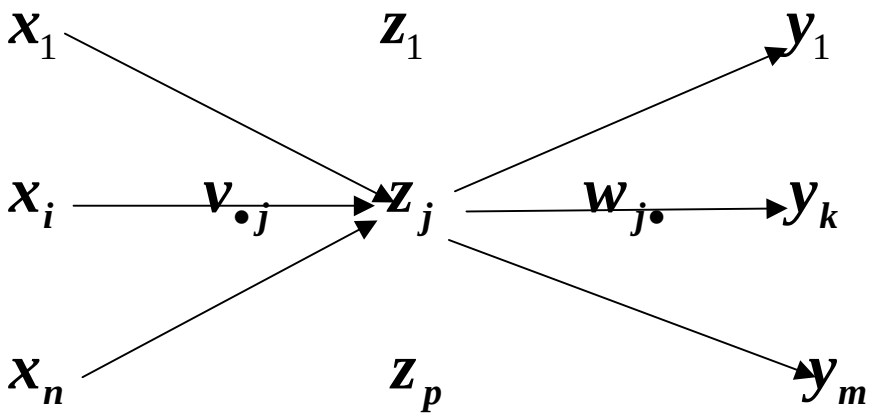






Counter propagation network (CPN)

- Basic idea of CPN
  - Purpose: fast and coarse approximation of vector mapping $y = f(x)$
    - not to map any given x to its $f(x)$ with given precision,
    - input vectors x are divided into clusters/classes.
    - each cluster of x has one output y, which is (hopefully) the average of $f(x)$ for all x in that class.
  - Architecture: Simple case: **FORWARD ONLY CPN**,



- Learning in two phases:
  - training sample $x:y$ where $y = f(x)$ is the precise mapping
  - Phase1: $v_{\bullet j}$ is trained by competitive learning to become the representative vector of a cluster of input vectors x (use sample x only)

    1. For a chosen x, feedforward to determined the winning z_j
    2. $v_{ij}(new) = v_{ij}(old) + a\ (x_i - v_{ij}(old))$
    3. Reduce a , then repeat steps 1 and 2 until stop condition is met
  - Phase 2: $w_{j\bullet}$ is trained by delta rule to be an average output of $f(x)$ where x is an input vector that causes z_j to win (use both x and y).

    1. For a chosen x, feedforward to determined the winning z_j
    2. $v_{ij}(new) = v_{ij}(old) + a\ (x_i - v_{ij}(old))$ (optional)
    3. $w_{jk}(new) = w_{jk}(old) + a\ (y_k - w_{jk}(old))$
    4. Repeat steps 1 – 3 until stop condition is met

Notes

- A combination of both unsupervised learning (for $\boldsymbol{v}_{\bullet j}$ in phase 1) and supervised learning (for $\boldsymbol{w}_{j\bullet}$ in phase 2).
- After phase 1, clusters are formed among sample input $\boldsymbol{x}$, each $v_{\bullet j}$ is a representative of a cluster (average).
- After phase 2, each cluster j maps to an output vector y, which is the average of $\{ f(\boldsymbol{x}) : \boldsymbol{x} \in \boldsymbol{cluster\ \ j} \}$
- View phase 2 learning as following delta rule

  $\boldsymbol{w}_{jk} = \boldsymbol{w}_{jk} + \boldsymbol{a}\ (\boldsymbol{y}_k - \boldsymbol{w}_{jk})$ where $\boldsymbol{y}_k - \boldsymbol{w}_{jk} \propto -\dfrac{\partial \boldsymbol{E}}{\partial \boldsymbol{w}_{jk}}$, because

  $$\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{w}_{jk}} = \frac{\partial}{\partial \boldsymbol{w}_{jk}} \sum (\boldsymbol{y}_k - \boldsymbol{w}_{jk}\boldsymbol{z}_j)^2 = -2(\boldsymbol{y}_k - \boldsymbol{w}_{jk}\boldsymbol{z}_j) \text{ when } \boldsymbol{z}_j \text{ wins}$$
- It can be shown that, when $\boldsymbol{t} \to \infty$,

  $\boldsymbol{v}_{\bullet j}(\boldsymbol{t}) \to \langle \boldsymbol{x} \rangle$ and $\boldsymbol{w}_{j\bullet}(\boldsymbol{t}) \to \langle f(\boldsymbol{x}) \rangle$

  where $\langle \boldsymbol{x} \rangle$ is the mean of all training samples that make $\boldsymbol{x}$ win

Show only on v_{ij} (proof of w_{jk} is similar.) Weight update rule can be rewriteenas $v_{ij}(t+1) = (1-a)v_{ij}(t) + a\ x_i(t+1)$

$$
\begin{aligned}
v_{ij}(t+1) &= (1-a)v_{ij}(t) + a\ x_i(t+1) \\
&= (1-a)((1-a)v_{ij}(t-1) + a\ x_i(t)) + a\ x_i(t+1) \\
&= (1-a)^2 v_{ij}(t-1) + (1-a)a\ x_i(t) + a\ x_i(t+1) \\
&\approx a\left[x_i(t+1) + x_i(t)(1-a) + x_i(t-1)(1-a)^2 + \ldots x_i(1)(1-a)^t\right]
\end{aligned}
$$

If x are drawn randomly from the training set, then

$$
\begin{aligned}
E[v_{ij}(t+1)] &= E[a\ (x_i(t+1) + x_i(t)(1-a) + \ldots x_1(1-a)^t] \\
&= a\ [E(x(t+1)) + (1-a)E(x(t)) + \ldots (1-a)^t E(x_i(1))] \\
&= a\ \langle x \rangle [1 + (1-a) + \ldots (1-a)^t] \\
&\Rightarrow a\ \langle x \rangle \frac{1}{1-(1-a)} = \langle x \rangle
\end{aligned}
$$

- After training, the network works like a look-up of math table.
  - For any input ***x,*** find a region where ***x*** falls (represented by the wining ***z*** node);
  - use the region as the index to look-up the table for the function value.
  - CPN works in multi-dimensional input space
  - More cluster nodes (***z***), more accurate mapping.

Full CPN

- If both $\boldsymbol{y} = f(\boldsymbol{x})$ and its inverse function $\boldsymbol{x} = f^{-1}(\boldsymbol{y})$ exist
  we can establish bi-directional approximation
- Two pairs of weights matrices:
  $\boldsymbol{V}$ ($\boldsymbol{x}$ to $\boldsymbol{z}$) and $\boldsymbol{U}$ ($\boldsymbol{z}$ to $\boldsymbol{y}$) for approx. map $\boldsymbol{x}$ to $\boldsymbol{y} = f(\boldsymbol{x})$
  $\boldsymbol{W}$ ($\boldsymbol{y}$ to $\boldsymbol{z}$) and $\boldsymbol{T}$ ($\boldsymbol{z}$ to $\boldsymbol{x}$) for approx. map $\boldsymbol{y}$ to $\boldsymbol{x}^* = f(\boldsymbol{x})$
- When $\boldsymbol{x}$:$\boldsymbol{y}$ is applied ($\boldsymbol{x}$ on $\boldsymbol{X}$ and $\boldsymbol{y}$ on $\boldsymbol{Y}$), they can jointly determine the winner J or separately for $\boldsymbol{z}_{Jx}$, $\boldsymbol{z}_{Jy}$
- pp. 196 –206 for more details