Competitive Learning with Explicit Output Classification

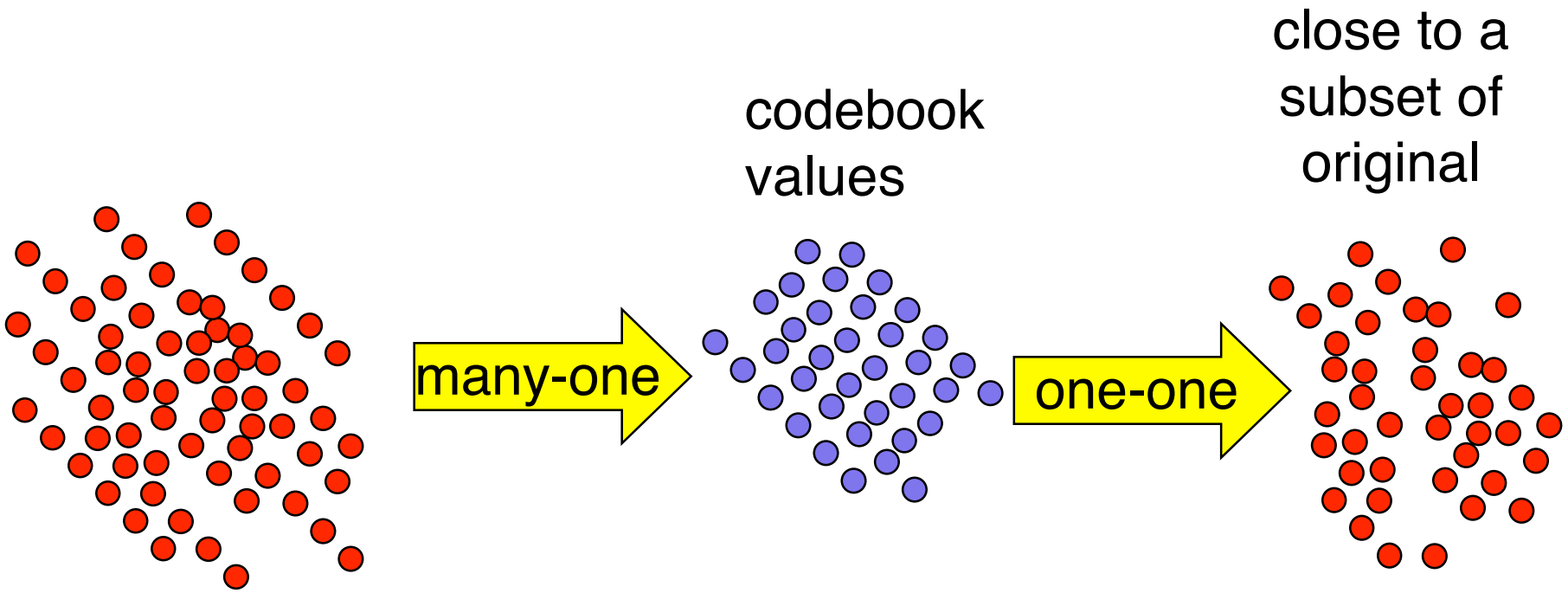
Learning Vector-Quantization

Counterpropagation

Vector Quantization

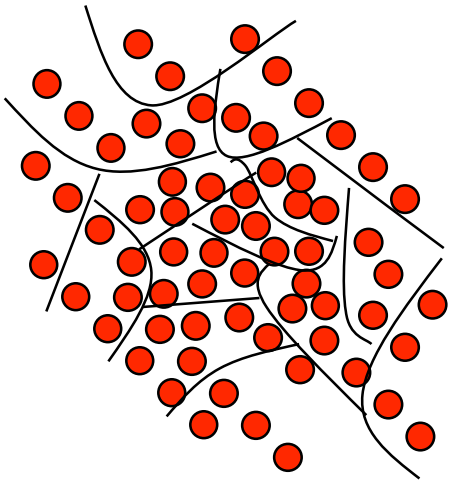
- VQ is a technique used to **reduce the dimensionality of data**.
- The original data is a set of vectors, of dimension n, say.
- The vectors are mapped into a set of smaller dimension, m, of **codebook values**.
- The codebook values are used for storage or transmission (some information content is lost).
- The codebook values, upon receipt or retrieval, are re-translated into values close to the original data values (will not be exact).

Codebook


codebook
values
close to a
subset of
original
many-one
one-one

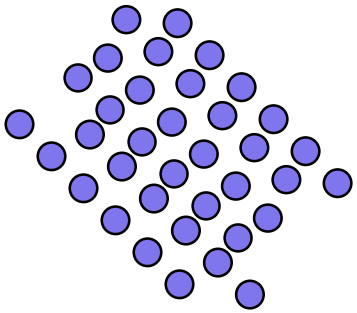
Codebook

inverse-image = some kind of clustering



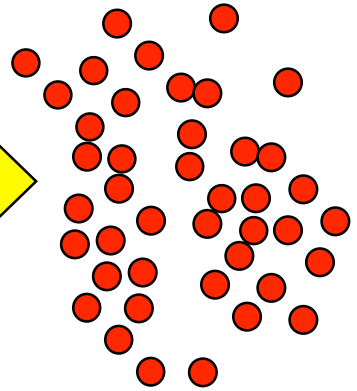
many-one

codebook values



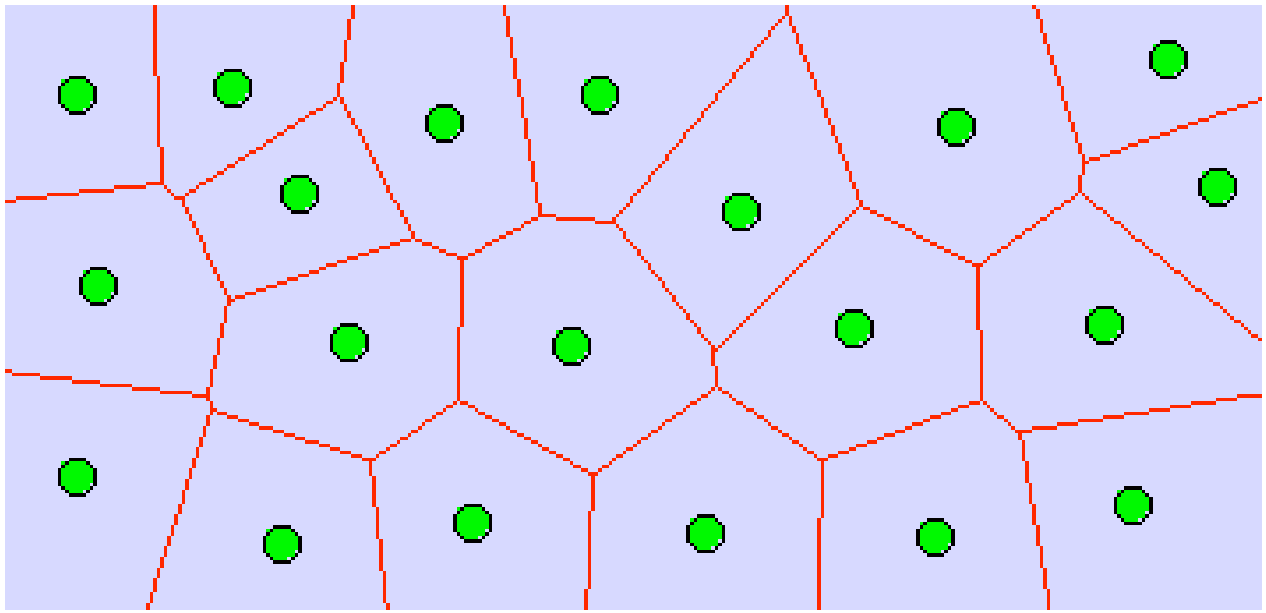
one-one

close to a subset of original



Voronoi Regions

Typically codebook code representatives are centers of ***Voronoi Regions*** in the data space. Voronoi regions are sets of points **closest** to one representative vs. another. Below each polygon is a Voronoi region, with representatives as dots.



closest ⇒ error is minimized

Chicken vs. Egg

- Which comes first, the Voronoi regions or the representatives?
- Initially we just have a set of points with neither.
- We specify the maximum size of the codebook (assume the number of points is larger).
- Then use a clustering technique, such as **k-means clustering**.

k-means clustering

- Choose an **initial** set of representatives by some method.
- Iterate the following:
  - For each point, find its closest representative; this determines a set of Voronoi regions.
  - Compute the *centroids* of the regions and use them as the new representatives.
- Until the desired error bound is reached.
- Note: The representatives need not be actual data points.

Termination Condition

- Compute MSE = $\frac{\sum\{\text{distance-from-rep}(p)^2 \mid p \text{ in region}\}}{\#\text{regions}}$

  where the sum is over all regions.

- We want MSE to be below a certain bound.

Computational Cost

- $O(n^2)$ per iteration, where n is the number of points.

Learning Vector-Quantization (LVQ)

Learning Vector Quantization (LVQ)

- Combine **competitive** learning with **supervision**.
- Competitive learning achieves clusters.
- Assign a **class** (or output value) to each cluster
- Reinforce cluster representative (a neuron) when it **classifies** input in the **desired** class:
  - **Positive reinforcement**: pull the neuron weights toward the input.
  - **Negative reinforcement**: push the weights away.

Learning Vector Quantization

Training examples:

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \square \ , \{\mathbf{p}_Q, \mathbf{t}_Q\}$$

If the input pattern is classified ***correctly***, then move the *winning* weight toward the input vector according to the Kohonen rule.

$$_{i\square}\mathbf{w}^1(q) = {}_{i\square}\mathbf{w}^1(q-1) + \square(\mathbf{p}(q) - {}_{i\square}\mathbf{w}^1(q-1)) \qquad a^2_{k\square} = t_{k\square} = 1$$

If the input pattern is classified ***incorrectly***, then move the *winning* weight *away* from the input vector.

$$_{i\square}\mathbf{w}^1(q) = {}_{i\square}\mathbf{w}^1(q-1) - \square(\mathbf{p}(q) - {}_{i\square}\mathbf{w}^1(q-1)) \qquad a^2_{k\square} = 1 \neq t_{k\square} = 0$$

Learning Vector Quantization

- The classification mapping from selection of cluster representative to output can be accomplished by a second “layer”.

- This second layer is essentially a matrix multiply, so can be represented by neural weights as usual. These weights may be pre-assigned and fixed.

LVQ xor Example

Example patterns with targets:

$$\left\{\mathbf{p}_1 = \begin{bmatrix}0\\1\end{bmatrix}, \mathbf{t}_1 = \begin{bmatrix}1\\0\end{bmatrix}\right\} \quad \left\{\mathbf{p}_2 = \begin{bmatrix}1\\0\end{bmatrix}, \mathbf{t}_2 = \begin{bmatrix}1\\0\end{bmatrix}\right\} \quad \left\{\mathbf{p}_3 = \begin{bmatrix}1\\1\end{bmatrix}, \mathbf{t}_3 = \begin{bmatrix}0\\1\end{bmatrix}\right\} \quad \left\{\mathbf{p}_4 = \begin{bmatrix}0\\0\end{bmatrix}, \mathbf{t}_4 = \begin{bmatrix}0\\1\end{bmatrix}\right\}$$

Competitive layer weights:

$$\mathbf{W}^1(0) = \begin{bmatrix}({}_1\mathbf{w}^1)^T\\({}_2\mathbf{w}^1)^T\\({}_3\mathbf{w}^1)^T\\({}_4\mathbf{w}^1)^T\end{bmatrix} = \begin{bmatrix}0.25 & 0.75\\0.75 & 0.75\\1 & 0.25\\0.5 & 0.25\end{bmatrix}$$

Classification layer weight (fixed):

$$\mathbf{W}^2 = \begin{bmatrix}1 & 1 & 0 & 0\\0 & 0 & 1 & 1\end{bmatrix}$$

Iteration 1, first layer

$$\mathbf{a}^1 = \mathbf{compet}(\mathbf{n}^1) = \mathbf{compet}\begin{bmatrix} -\|_1\mathbf{w}^1 - \mathbf{p}_1\| \\ -\|_2\mathbf{w}^1 - \mathbf{p}_1\| \\ -\|_3\mathbf{w}^1 - \mathbf{p}_1\| \\ -\|_4\mathbf{w}^1 - \mathbf{p}_1\| \end{bmatrix}$$

$$\mathbf{a}^1 = \mathbf{compet}\begin{bmatrix} -\left\|\begin{bmatrix}0.25 & 0.75\end{bmatrix}^T - \begin{bmatrix}0 & 1\end{bmatrix}^T\right\| \\ -\left\|\begin{bmatrix}0.75 & 0.75\end{bmatrix}^T - \begin{bmatrix}0 & 1\end{bmatrix}^T\right\| \\ -\left\|\begin{bmatrix}1.00 & 0.25\end{bmatrix}^T - \begin{bmatrix}0 & 1\end{bmatrix}^T\right\| \\ -\left\|\begin{bmatrix}0.50 & 0.25\end{bmatrix}^T - \begin{bmatrix}0 & 1\end{bmatrix}^T\right\| \end{bmatrix} = \mathbf{compet}\begin{bmatrix} -0.354 \\ -0.791 \\ -1.25 \\ -0.901 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Iteration 1, second layer

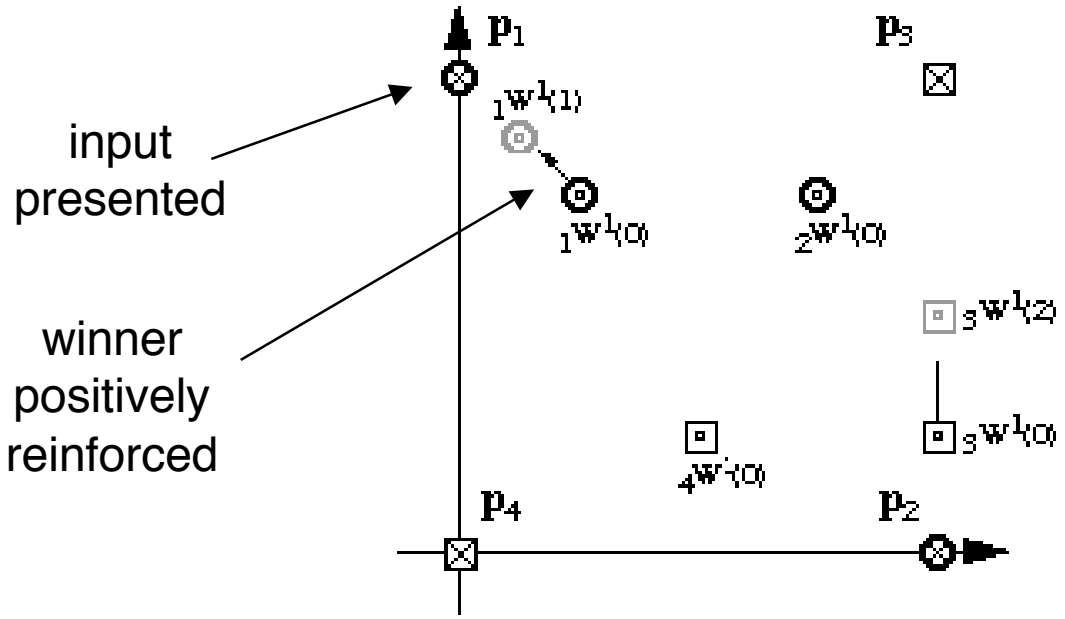
$$\mathbf{a}^2 = \mathbf{W}^2\mathbf{a}^1 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

This is the correct class, therefore the weight vector is moved toward the input vector.

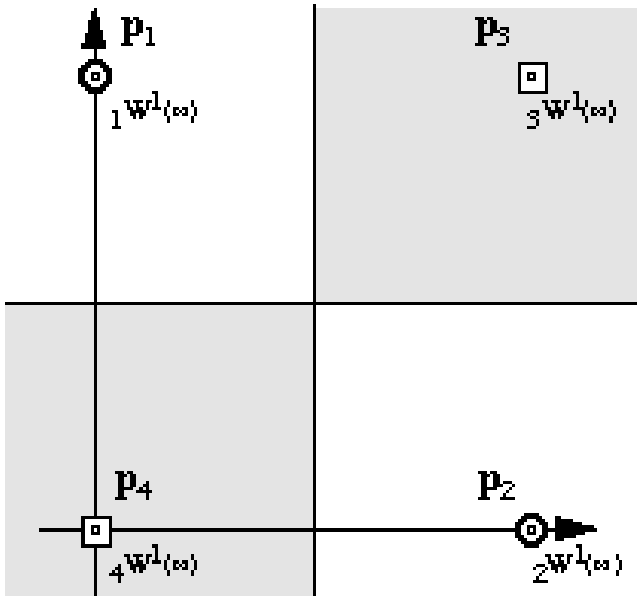
$$_1\mathbf{w}^1(1) = {}_1\mathbf{w}^1(0) + \alpha(\mathbf{p}_1 - {}_1\mathbf{w}^1(0))$$

$$_1\mathbf{w}^1(1) = \begin{bmatrix} 0.25 \\ 0.75 \end{bmatrix} + 0.5\left(\begin{bmatrix} 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 0.25 \\ 0.75 \end{bmatrix}\right) = \begin{bmatrix} 0.125 \\ 0.875 \end{bmatrix}$$

Iteration 1 illustrated



Final Outcome after many iterations



Variant: LVQ2

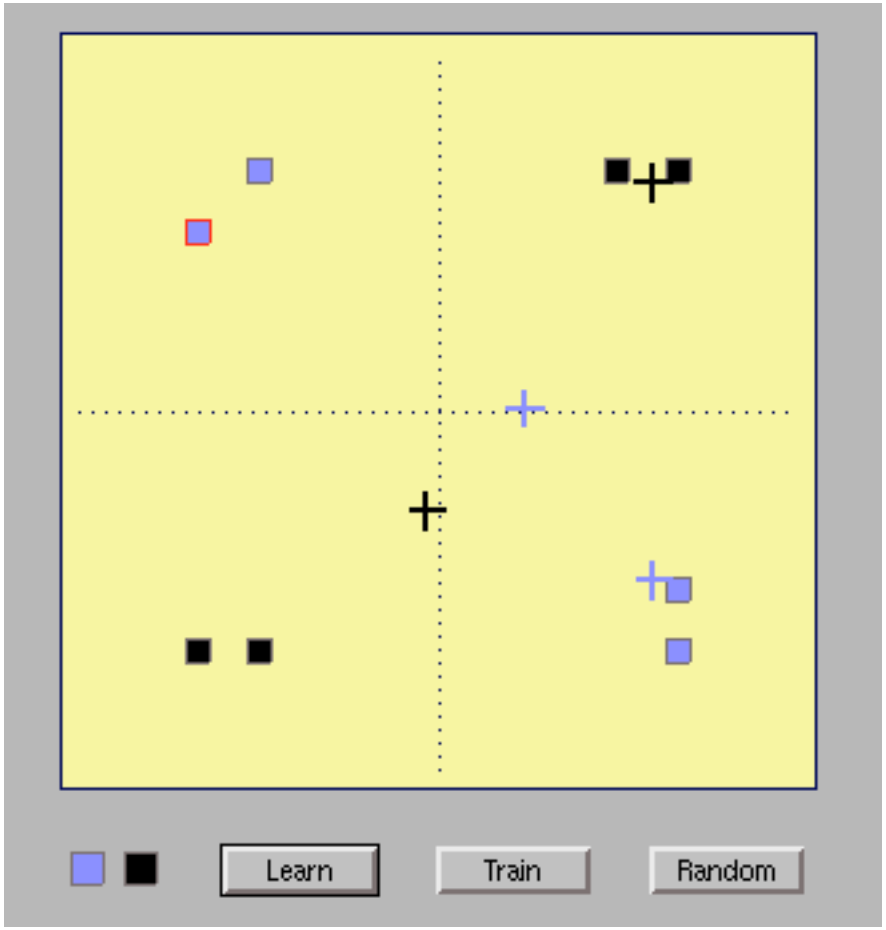
- If the winning neuron in the hidden layer **correctly** classifies the current input, the response is the same as in LVQ1.
- If the winning neuron in the hidden layer **incorrectly** classifies the current input, we move its weight vector **away** from the input vector, as in LVQ, **and**
- ***also*** move the weights of the **closest** neuron to the input vector that **does correctly** classify the input **toward** the input vector.

Matlab LVQ demos

- nnd14lv1, nnd14lv2

Blue vs. black represent class of neuron, desired class of input sample.

Selected input sample changes to green if correctly classified, red if not; weights adjusted.



Counterpropagation Networks

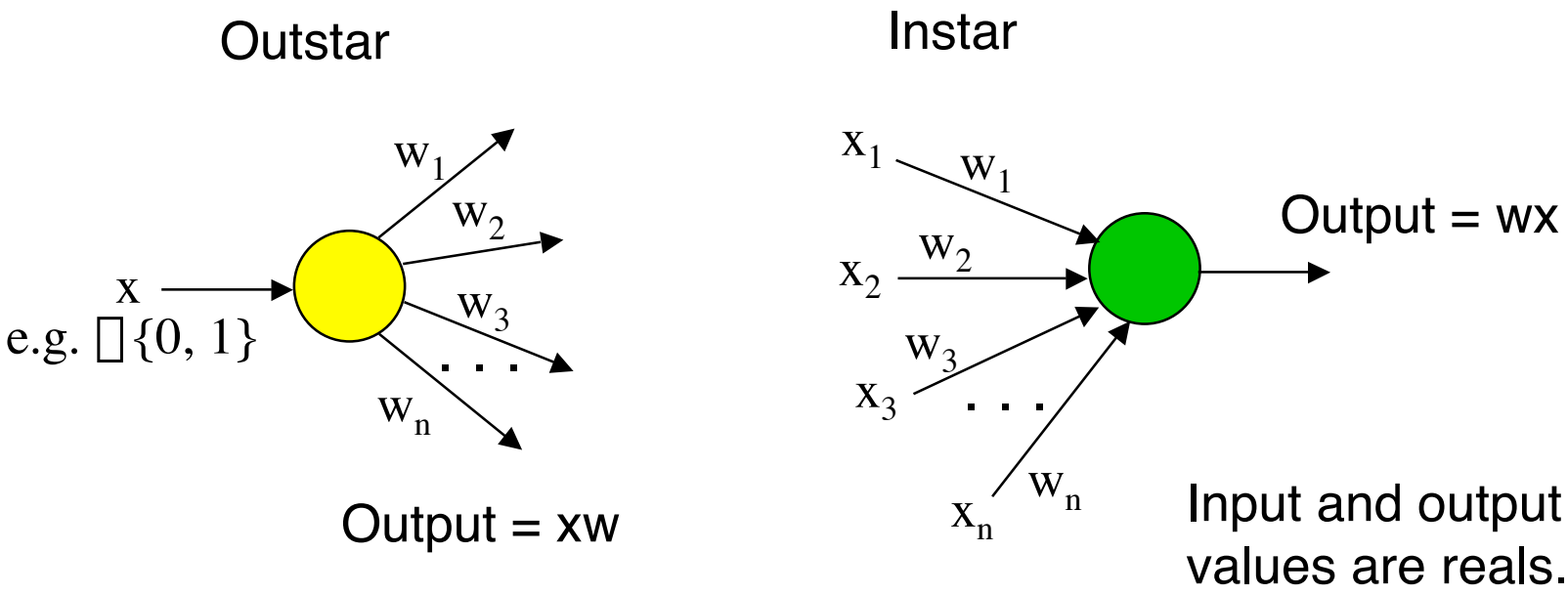
Counterpropagation Networks

- Invented by **Robert Hecht-Nielson**, founder of HNC Inc., recently acquired by Fair Isaac Co., Inc.
- Consists of **two** opposing networks, one for learning a **function**, the other for learning its **inverse**.
- Each network has two layers:
  - A first layer that clusters inputs (usually stated to be a self-organizing map, see later).
  - An “outstar” second layer to provide the output values for each cluster.

Outstar and Instar (due to Stephen Grossberg)

- An **instar** responds to a single input vector.
- An **outstar** produces a single output vector when stimulated with a binary value.

Outstar and Instar



Biologically, outstar would be synaptic weights, while instar would have dendritic ones.
However, it is common to refer to both weights as “synaptic”.

Outstar and Instar

- Outstar learning rule:

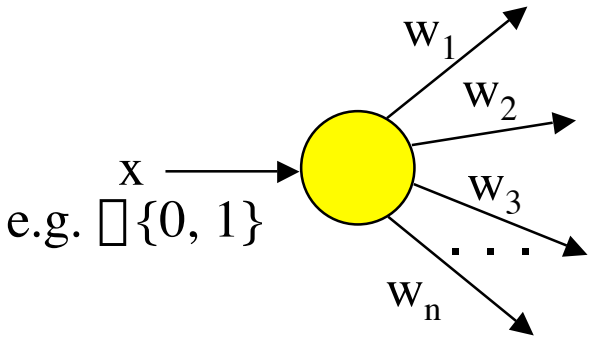
  $\Delta w = \eta x (d - w)$

  (d = desired)

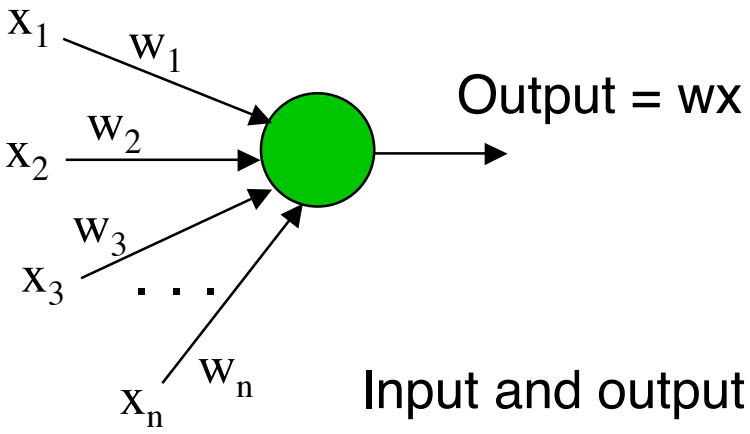
Instar learning rule:

$\Delta w = \eta(x - w) d$

(d = desired)



Output = xw



Input and output values are reals.

Outstar and Instar

- Variations are possible, e.g. adding weight-decay.

Counterpropagation Operation

- An outstar neuron is associated with each cluster representative.
- Given an input, the winner is found.
- The outstar is then stimulated to give the output.

Counterpropagation Network Computing Numeric Reciprocal

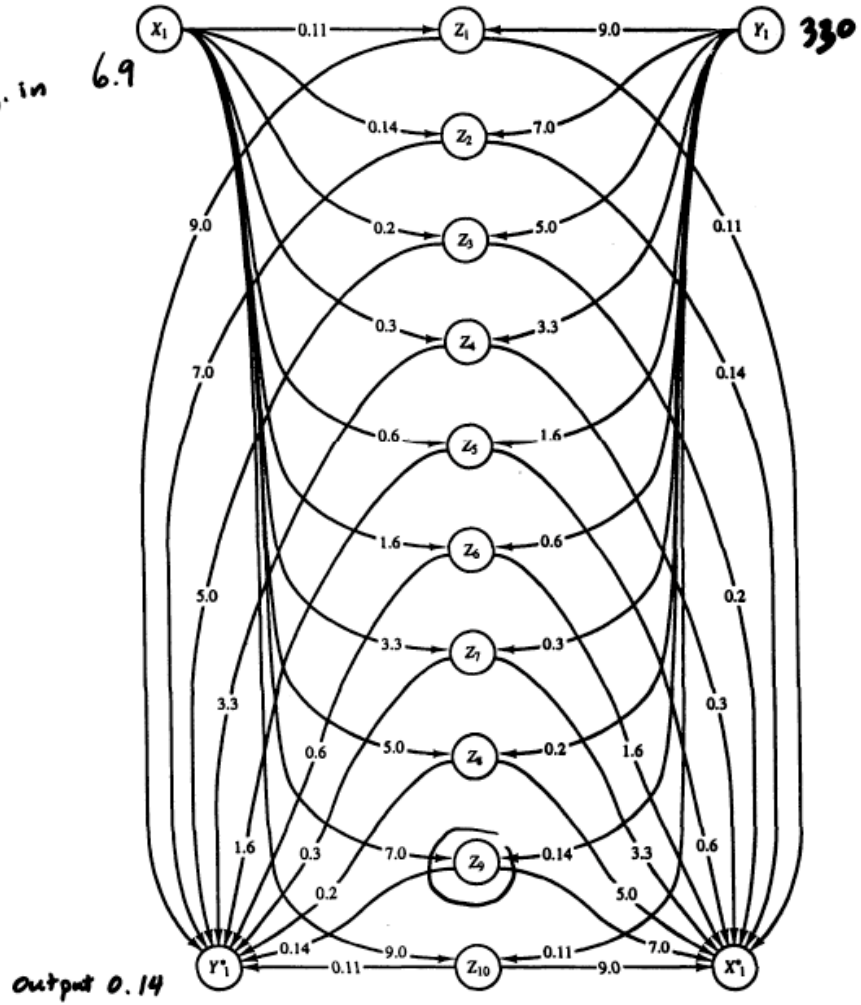


Figure 4.44 Full counterpropagation network for $y = 1/x$.

Competitive layer;
only winner fires.

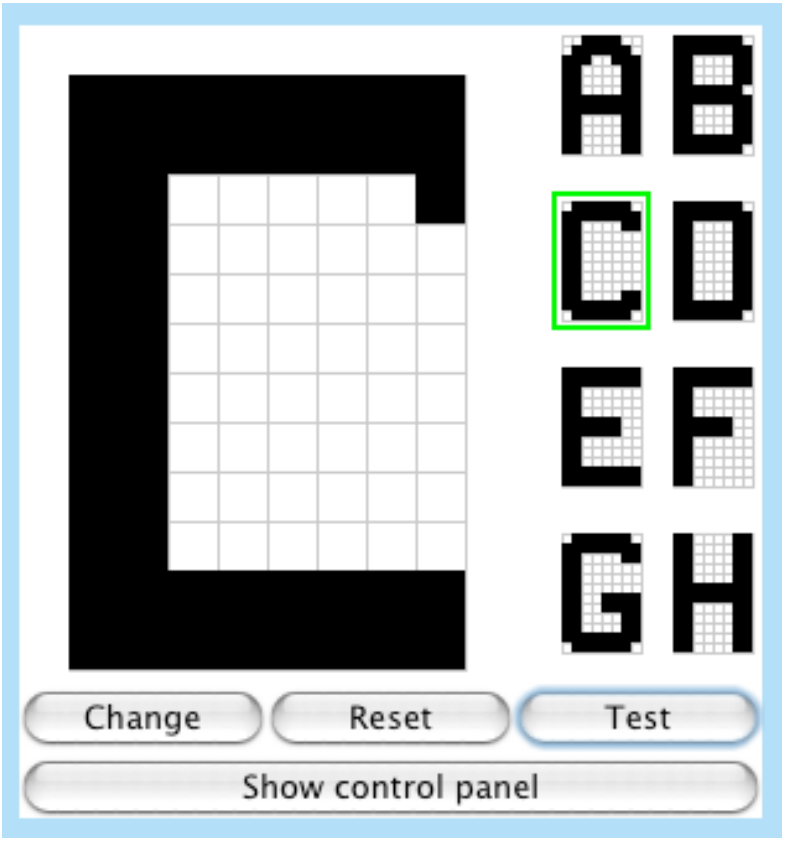
outstar weights determine output

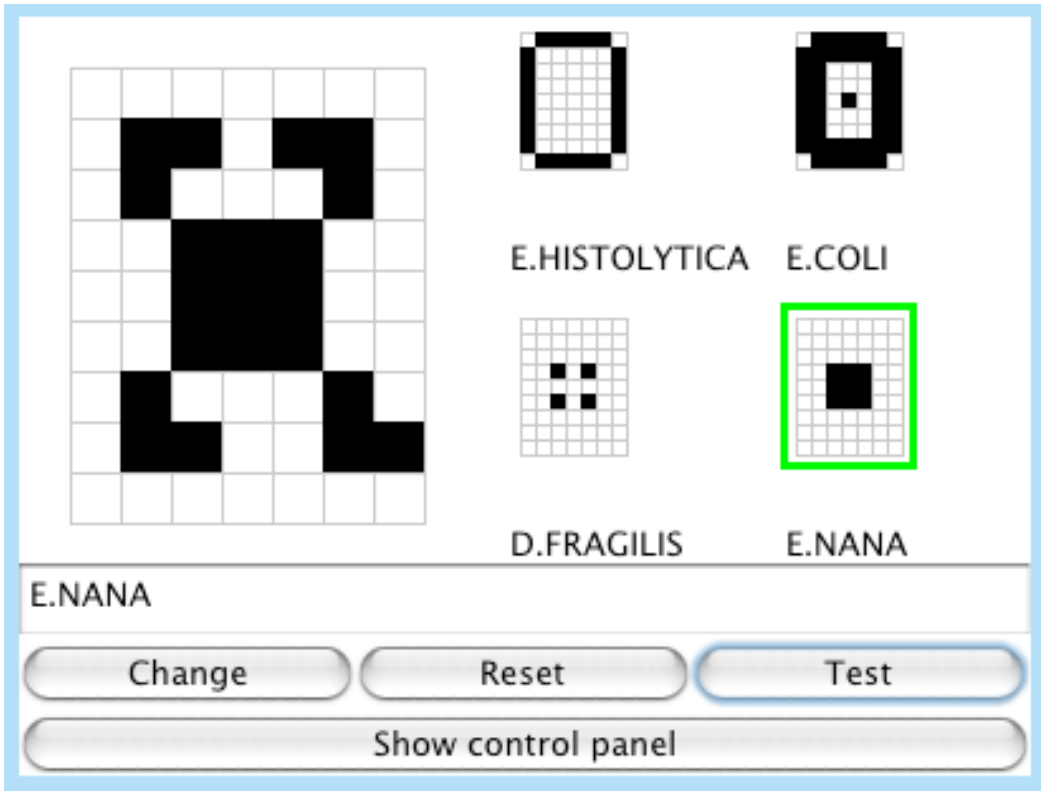
Counterpropagation Notes

- Since these networks operate by **recognizing input patterns** in the first layer, one would generally use ***lots*** of neurons in this layer.

Counterpropagation Applets

http://eple.hib.no/~hib00ssl/BitMapRecognizerCounterpropagation.html



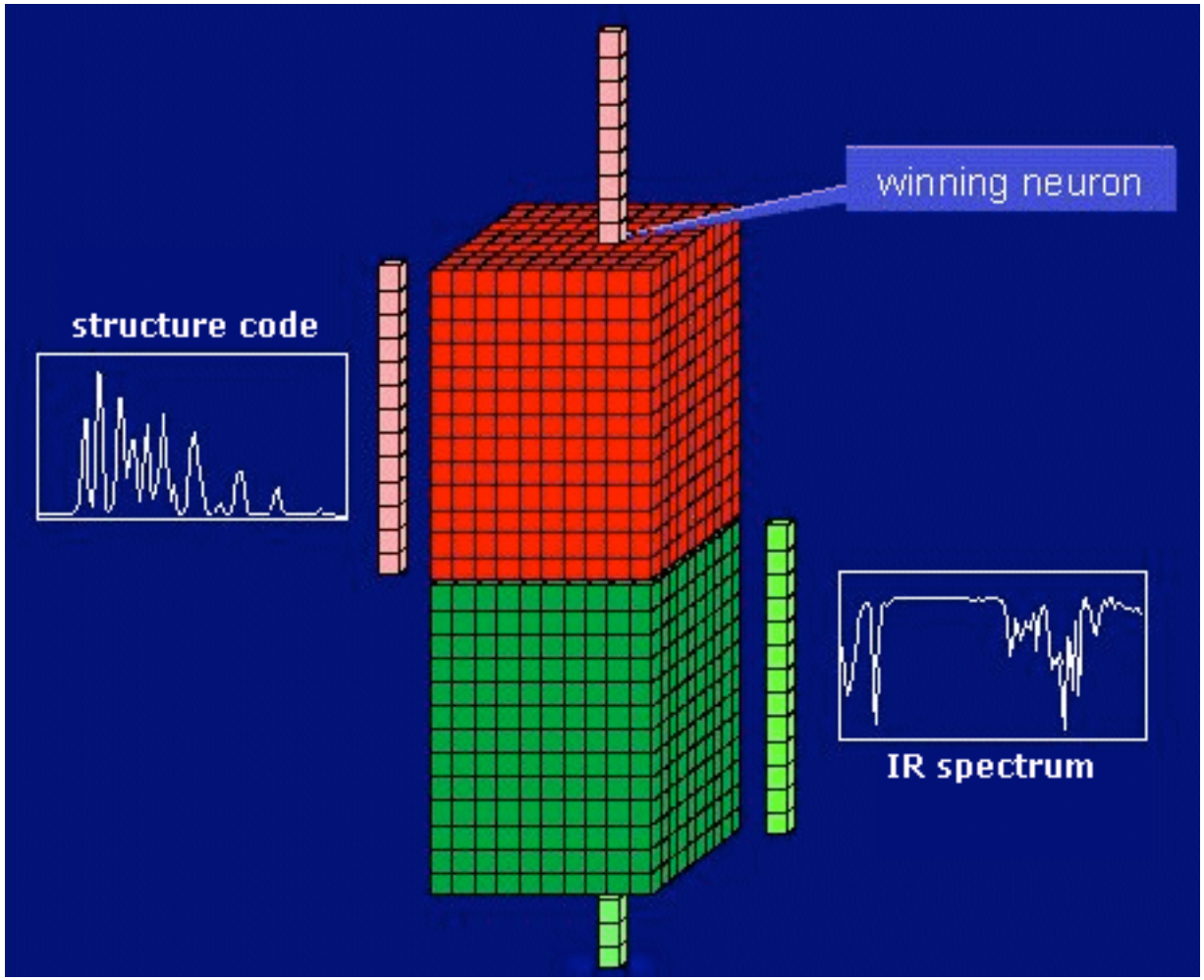


Counterpropagation Applications

- IR Spectrum of Molecular Structures
  www2.chemie.uni-erlangen.de/services/telespec/english/StructureInput.html
- A trained neural network is able to predict an infrared spectrum for a molecule it has not seen before.
- The structure code of the query structure is compared with the structure block of the neural network.
- The similarity criterion is the MSE between the query structure code and the neuron weights.
- The most similar neuron, the winning neuron, acts as a poniter to the corresponding information in the output bloc, The output block provides the simulated spectrum

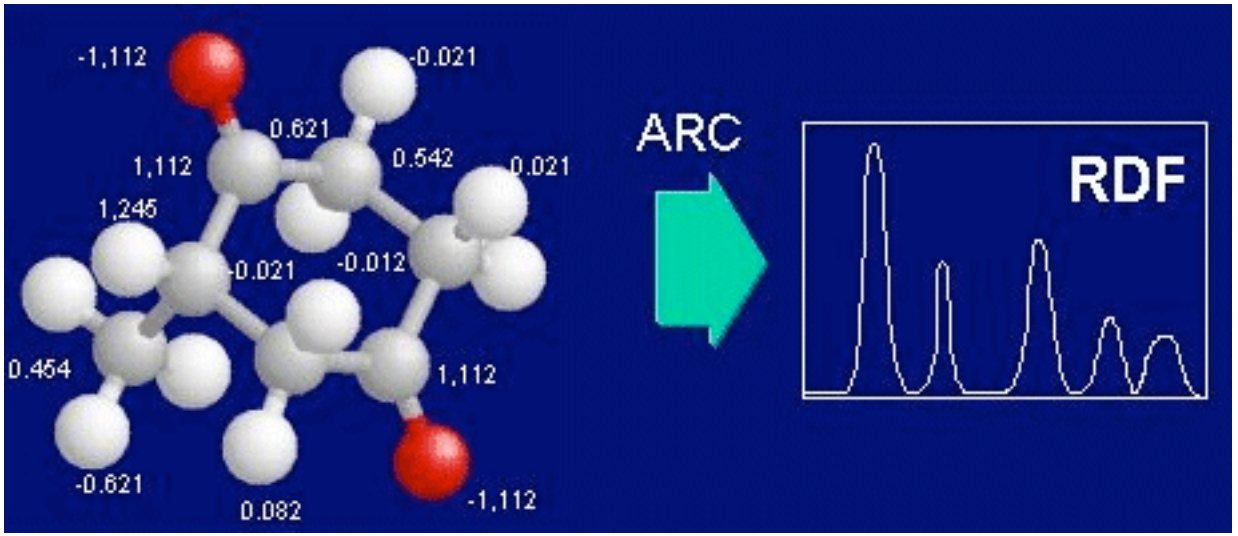
IR Spectrum of Molecular Structures

www2.chemie.uni-erlangen.de/services/telespec/english/StructureInput.html



Structure Coding

www2.chemie.uni-erlangen.de/services/telespec/english/StructureInput.html



Networks use input a data vector of constant length, a representation of a molecular 3D structure by a fixed number of descriptors.

Methods were developed to transform the 3D structure and associated physicochemical atomic properties into a one-dimensional vector. See the web page for more specifics.

Counterpropagation Applications

- Dolphin Echolocation:
  - Roitblat, H. L., P. W. B. Moore, P. E. Nachtigall, R. H. Penner, and W. W. L. Au. 1989. Dolphin Echolocation: Identification of Returning Echoes Using a Counterpropagation Network. International Joint Conf. on Neural Networks, Vol. 1, IEEE and International Neural Network Society, Piscataway, NJ, pp. 295-30.
  - Roitblat, H.L., P. W. B. Moore, P. E. Nachtigall, R. H. Penner, and W. W. L. Au. 1989. Natural Echolocation With an Artificial Neural Network. *International Journal of Neural Networks: Research and Applications* **1**(4): 239-248.
  - Roitblat, H. L., P. W. B. Moore, P. E. Nachtigall, and R. H. Penner. 1991. Biomimetic Sonar Processing: From Dolphin Echolocation to Artificial Neural Networks. In: *From Animals to Animats* , J. A. Meyer and S. Wilson (eds.). MIT Press, Cambridge, MA, pp. 66-76.

Counterpropagation vs. MLP

Robocup soccer application “a4ty”
(dssg.cs.rtu.lv/download/robocup/a4ty2003_long.pdf)

“We applied [counterpropagation] for **pass selection**. A trained, with offline learning, agent needs to decide whether it should give a pass to a partner or not. If it makes an incorrect pass (the ball is intercepted by an opponent) in online mode in the given situation, the network updates weights so as next time the probability of choosing this action for the same (or similar) situation would be lower.

However, this approach has not been implemented in the final version of a4ty team since the overall performance (approximation capabilities) of counterpropagation network is worse than, for example, multilayer perceptron has. Thus, counterpropagation neural network has been replaced by multilayer perceptron (MLP)”.

Counterpropagation vs. LVQ

- Both combine Supervise and Unsupervised learning.
- LVQ is used for classification; Counterprop can be used for function computation
- LVQ has one training phase; Counterprop has two.
- LVQ output weights are fixed throughout; Counterprop weights are trained in a second phase.