How to find proper architecture?

- Predict theoretically:
  - Reduce number of inputs using PCA
- Use evolutionary computation technique
- Try and error approach
- Start with large system and prune it.
  - Eliminate (combine) neurons with the same similar (complement) responses for all patterns
- Start with smaller system and add neurons as needed
  - Look for oscillatory weight changes during training process.

Principle Component Analysis

Hebb’s rule: \( w_{ij} = \alpha x_i y_j \) Connection between two neurons become stronger if neurons are responding the same way

By training a simple neuron where \( x \) are inputs and \( y \) are outputs will lead to a solution with weights specifying a most probable direction of patterns.

Oja’s modification:

\[
\dot{w}_i = \xi y_i \left( x_i - \sum_j \xi w_{ij} y_j \right)
\]

This leads to a solution where length of the weight vector is normalized to one and direction is set by the principle component.

In order to extract all \( M \) principle components then network with \( M \) neurons has to be used and

\[
W_{ij} = \gamma y_i \left( x_i - \sum_{k=1}^{M} y_k w_{ik} \right)
\]

Deficiency of statistical approaches (correlation or PC analysis)

![Deficiency of statistical approaches](image)

Polynomial Networks

- Fourier or other series?
- Nonlinear regression?

Functional Link Networks

- Nilsen-Pao
- Unipolar neuron
- Bipolar neuron

Functional link networks for solution of the XOR problem: (a) using unipolar signals, (b) using bipolar signals.

Genetic algorithms
The counterpropagation networks

Kohonen layer

Hamming layer

unipolar neurons

summing circuits

inputs

outputs

1 -1 -1 -1
-1 -1 -1 -1
-1 -1 -1 -1
1 1 -1 -1
1 -1 -1 -1
1 -1 -1 -1
1 -1 -1 -1
1 1 -1 -1
1 -1 -1 -1
1 -1 -1 -1

The counterpropagation networks (ROM)

inputs

outputs

1 -1 -1 -1
-1 1 -1 -1
-1 1 -1 -1
-1 1 -1 -1
1 1 -1 -1
1 -1 -1 -1
1 -1 -1 -1
1 1 -1 -1
1 -1 -1 -1
1 -1 -1 -1

The counterpropagation networks (analog memory)

Hamming layer

binary input

unipolar neurons

summing circuits

inputs

outputs

1 -1 -1 -1
-1 1 -1 -1
-1 1 -1 -1
1 1 -1 -1
1 -1 -1 -1
1 -1 -1 -1
1 -1 -1 -1
1 1 -1 -1
1 -1 -1 -1
1 -1 -1 -1

analog memory with analog address

inputs

outputs

1 1 2 1
2 3 1 -3
3 -1 -2 4
1 -1 -1 -1
2 3 1 -3
1 -1 -1 -1
1 -1 -1 -1
1 1 -1 -1
1 -1 -1 -1
1 -1 -1 -1

analog memory with analog address

inputs

output

1 1 2.236
1 2 3.162
1 3 -2.236
2 1 2.828
2 2

Normalization of outputs of unipolar neurons

\[ z = \sqrt{x^2 + y^2} \]

RBF - Radial Basis Function networks

Output function

\[ \text{out} = \exp \left( -\frac{\| \mathbf{x} - \mathbf{s} \|}{\sigma} \right) \]
RBF - Radial Basis Function networks

\[
\text{out} = \exp\left(-\frac{(x-s)^2}{\sigma}\right)
\]

LVQ Learning Vector Quantization

First layer computes Euclidean distances between input pattern and stored patterns.
Wining “neuron” is with the minimum distance

The cascade correlation architecture

Step 1 - output neuron is trained

Step 2 - hidden neuron is inserted and trained to the errors from the Step 1
at this step hidden neuron #1 is not connected to the network and it is trained so its output values matches errors from the first step
Notice that this is one neuron training and an efficient algorithm can be used such as pseudo inversion

Step 3 - output neuron is trained to the desired outputs
at this step hidden neuron #1 is connected to the network and its weights are frozen
The cascade correlation architecture

Step 4 - new hidden neuron is inserted and trained to the errors from the previous step.

Step 5 - output neuron is trained to the desired outputs.

Step 6 - new hidden neuron is inserted and trained to the errors from the previous step.

Step 7 - output neuron is trained to the desired outputs.

MAX net

\[ W = \begin{bmatrix} 1 & -\epsilon & -\epsilon & -\epsilon \\ -\epsilon & 1 & -\epsilon & -\epsilon \\ -\epsilon & -\epsilon & 1 & -\epsilon \\ -\epsilon & -\epsilon & -\epsilon & 1 \end{bmatrix} \]

Activation function

Kohonen Network

If inputs are binaries, for example \( X = [1, -1, 1, -1, -1] \) then the maximum value of net

\[ net = \sum_{j=1}^{n} x_jw_j = XW^T = n - 2HD \]

where \( n \) is the number of inputs and \( HD \) is the Hamming distance between input vector \( X \) and weight vector \( W \).
**Kohonen Network**

The unsupervised training process

1. All patterns are normalized (the lengths of the pattern vectors are normalized to unity)

\[ z_j = \frac{x_j}{\sqrt{\sum_{i=1}^{n} x_i^2}} \]

\[ z_n = \frac{x_n}{\sqrt{\sum_{i=1}^{n} x_i^2}} \]

2. Weights are chosen randomly for all neurons

3. Lengths of the weight vectors are normalized to unity

\[ w_j = \frac{w_j}{\sqrt{\sum_{i=1}^{n} w_i^2}} \]

\[ w_n = \frac{w_n}{\sqrt{\sum_{i=1}^{n} w_i^2}} \]

4. A pattern is applied to input and net values are calculated for all neurons

\[ net = \sum_{i=1}^{5} z_i v_i = ZV^T \]

5. A winning neuron is chosen (neuron with largest net value)

6. Weights for the winner \( k \) are modified using a weighted average:

\[ W_k = V_k + \alpha Z \]

\( \alpha \) is the learning.

Weights of other neurons are not modified.

7. Weights for the winning neuron are normalized.

\[ v_j = \frac{w_j}{\sqrt{\sum_{i=1}^{n} w_i^2}} \]

\[ v_n = \frac{w_n}{\sqrt{\sum_{i=1}^{n} w_i^2}} \]

8. Another pattern is applied (go to step 4)

During pattern applications some neurons are frequent winners and other never take part in the process. The latter ones are eliminated and the number of recognized clusters is equal to the number of surviving neurons.
Kohonen Network
The unsupervised training process

One disadvantage of this method is that the classification is strongly dependent on the initial set of randomly chosen weights.

Another disadvantage is that during the normalization process important information about the length of input patterns is lost.
Kohonen Network
The unsupervised training process example

- Normalized patterns
- Initial weights
- Current weights

Weights are representing center of clusters

Sarajedini and Hecht-Nielsen network
Let us consider stored vector \( \mathbf{w} \) and input pattern \( \mathbf{x} \). Both input and stored patterns have the same dimension \( n \). The square Euclidean distance between \( \mathbf{x} \) and \( \mathbf{w} \) is:

\[
| \mathbf{x} - \mathbf{w} |^2 = (x_1 - w_1)^2 + (x_2 - w_2)^2 + \cdots + (x_n - w_n)^2
\]

After defactorization

\[
| \mathbf{x} - \mathbf{w} |^2 = x_1^2 + x_2^2 + \cdots + x_n^2 + w_1^2 + \cdots + w_n^2 - 2(x_1w_1 + x_2w_2 + \cdots + x_nw_n)
\]

Finally

\[
| \mathbf{x} - \mathbf{w} |^2 = x_1^2 + x_2^2 + \cdots + x_n^2 + w_1^2 + \cdots + w_n^2 - 2\mathbf{x} \cdot \mathbf{w}
\]

Input pattern transformation on a sphere
Fix to Kohonen network deficiency

\[
R - x^2
\]

Network with two neurons capable of separating crescent shape of patterns (a) input-output mapping, (b) network diagram

Input pattern transformation on a sphere 3

Spiral problem solved with sigmoidal type neurons (a) network diagram, (b) input-output mapping.

ART
Adaptive Resonance Theory

Step 1: The resonance threshold \( \epsilon \) is set and the weight vector \( \mathbf{w} \) for each neuron \( \mathbf{w} \) is initialized randomly in weight space. The mean \( \mathbf{W} \) is set to \( \mathbf{W} = 0 \) and each \( \mathbf{w} \) is associated with identical neuron.

\[
\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_n \end{bmatrix}, \quad \mathbf{W} = 0
\]

Step 2: Binary encoding: Output neuron \( \mathbf{y} \) is presented as input signal.

\[
\mathbf{y} = \begin{cases} 1 & \text{if } \mathbf{w} \text{ has a match} \\ 0 & \text{otherwise} \end{cases}
\]

Step 3: All matching neurons are compared to \( \mathbf{y} \). If the best match has energy more than a single neuron, each unit in the winner takes energy from the winner neuron and units with energy less than \( \mathbf{y} \) receive energy from the units with energy greater than \( \mathbf{y} \).

\[
\mathbf{W}_{\text{new}} = \mathbf{W} + \frac{1}{\text{energy}} (\mathbf{y} - \mathbf{W})
\]

Step 4: Neurons which are not matched to \( \mathbf{y} \) are updated by setting their energy to zero. The algorithm goes to Step 4.

Step 5: The output is halted if the energy of each neuron is zero or the number of cycles exceeds a pre-defined limit. The algorithm goes to Step 4.

In this case, selection of the best matching incoming neuron is performed according to the maximum resonance (7.3.4) as follows:

\[
q = \max_q | \mathbf{w}^T \mathbf{x} |^2
\]

A practical way is to use the resonance value found. This can be used to find the closest match. The algorithm then examines a matrix of all neuron outputs and with all output values. This is a very effective way of finding the closest match.
Mountain clustering

Superposition of Gaussian hills

1. For each pattern a small hill of a bell (Gaussian) shape is formed.
2. These hills are forming mountains.
3. The highest mountains are the clusters

This is very computationally intensive process.

Forming clusters as needed using minimum distance concept

Much simpler and more efficient then ART

1. First pattern is applied and the cluster is formed
2. Next pattern is applied and then:
   a) If distance form all existing clusters is larger then threshold then a new cluster is formed
   b) Else weights of the closest cluster are updated

\[ W_i = \frac{mW_j + \alpha X}{m+1} \]

where \( m \)  is the number of previous patterns of a given set which were used to update this particular neuron and \( \alpha \)  is the learning constant

Parity problems

Unipolar

(FF networks)

Parity problems

Bipolar

(FF fully connected)
Parity problems (cascade)