Unsupervised Networks

Learning without a teacher

Dr J. Charles Sullivan

What is this lecture about?

- So far we have looked at several examples of supervised learning
- This lecture concentrates on **unsupervised learning**
- The best known example of an ANN architecture for unsupervised learning is the Self Organising Map or “Kohonen network”
- Before looking at this type of network in more detail, let us think a bit more about the meaning of the word ***learning***

What is Learning?

- *"the process which leads to the modification of behavior or the acquisition of new abilities or responses, and which is additional to natural development by growth or maturation"*

  (Oxford English Dictionary)

- *"Learning is a process by which the free parameters of neural networks are adapted through a process of simulation by the environment in which the network is embedded. The type of learning is determined by the manner in which the parameter changes take place."* (Haykin, 1999)

Types of learning

- The minimisation of an error function, which involves target values for the network outputs is called ***supervised learning*** since, for each input pattern, the target value of the desired output is specified

- A second form of learning in neural networks, called ***unsupervised*** learning, does not involve the use of target data.
  - Instead of learning an input-output mapping, the goal may be to model the probability distribution of the input data or to discover clusters or other structure in the data

- There is a third form of learning, called ***reinforcement learning*** in which information is supplied as to whether the network outputs are good or bad, but again no actual desired values are given.
  - This is mainly used for control applications, and we will look more closely at this in a subsequent lecture

- It is informative to reflect on how **we** learn things

Hebbian Learning

Hebbian learning is the most common rule for unsupervised or self-organized learning.

"When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic changes take place in one or both cells such that As

efficiency as one of the cells firing B, is increased." (Hebb, 1949)

Mathematically, the Hebbian learning rule can be directly interpreted as

$$\frac{\partial w_{ij}(t)}{\partial t} = \alpha x_i(t) y_i(t)$$

where α is a positive learning rate (0 < α < 1), and x and y are the input and output of the neural system, respectively (or can also be regarded as the outputs of the two neurons).

the change of the synaptic weight is proportional to the correlation between an input and its associated output. If the input and output are coherent, the weight connecting them is strengthened (xy is positive), otherwise, weakened (xy is either negative or zero).

Competitive Learning

- basic architecture consists of a set of hierarchically layered units in which each layer connects, via excitatory connections, with the layer immediately above it, and has inhibitory connections to units in its own layer.
- each unit in a layer receives an input from each unit in the layer immediately below it and projects to each unit in the layer immediately above it.
- within a layer, the units are broken into a set of inhibitory clusters in which all elements within a cluster inhibit all other elements in the cluster.
- Thus elements within a cluster at one level **compete** with one another to respond to the pattern appearing on the layer below. The more strongly any particular unit responds to an incoming stimulus, the more it shuts down the other members of its cluster

Self-Organizing Map

- A simple modification to the competitive learning model gives rise to a powerful new class of models: the Self-Organizing Map (SOM).

- These models were pioneered by Kohonen (1982) and are also referred to as Kohonen Maps

- The SOM can be thought of as the simple competitive learning model with a ***neighborhood constraint*** on the output units

Self-organisation

- Self-organisation in networks is one of the most fascinating topics in the neural network field.
- Such networks can learn to detect regularities and correlations in their input and adapt their future responses to that input accordingly.
- The neurons of competitive networks learn to recognize groups of similar input vectors.
- Self-organising maps learn to recognise groups of similar input vectors in such a way that neurons physically “near” each other in the neuron layer respond to similar input vectors.
  - This is what “topology preserving” means
- Self-organising maps do not have target vectors, since their purpose is to divide the input vectors into clusters of similar vectors.
- There is no “*desired output”* for these types of networks.

Self-organisation in the brain

- The idea of self-organization was proposed in 1973 by von der Malsburg and was based on close studies of the topology of the brain's cortex region.
- The development of SOM as a neural model is motivated by the **topographical** nature of cortical maps.
- Visual, tactile, and acoustic inputs are mapped in a topographical manner.
  - Visual: retinotopy (position in visual field), orientation, spatial frequency, direction, ocular dominance, etc.
  - Tactile: somatotopy (position on skin)
  - Acoustic: tonotopy (frequency)

Cortical Maps

- The main structures (primary sensory areas) of the cortical maps are established before birth in a predetermined topographically ordered fashion.
- Other more detailed areas (associative areas) are developed through self-organization gradually during life and in a topographically meaningful order
- studying such topographically ordered projections is undoubtedly important for understanding and constructing dimension-reduction mapping
- And for the effective representation of sensory information and feature extraction.

Biological Inspiration

Unlike other network paradigms, for SOMs it is not necessary to ask what the neurons *calculate*

We only ask which neuron is *"active"* at a given moment

Consider biological neurons connected to muscles: it is less interesting to know how strongly a certain muscle is contracted than which muscle is activated

In other words: We are not interested in the exact output of the neuron but in knowing which neuron provides output

Thus, SOMs are considerably more related to biology than, for example, the feedforward networks, which are increasingly used for calculations

Feature Mapping

- Typically, SOMs have (like our brain) the task of mapping a high-dimensional input (N dimensions) onto areas in a low-dimensional grid of cells (G dimensions) to draw a “map” of the high dimensional space

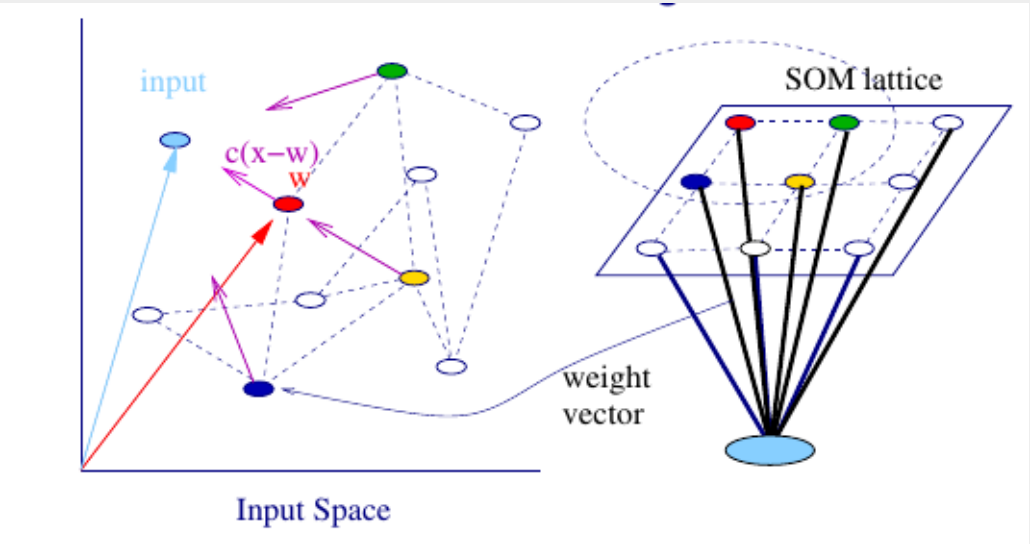
- To generate this map, the SOM simply obtains arbitrarily many points of the input space

- the SOM will try to achieve as good a coverage as possible by its neurons of the positions on which the points appear

- In particular, this means,that every neuron can be assigned to a unique position in the input space.

Kohonen SOM

- “The principal goal of the self-organizing feature-mapping algorithm developed by Kohonen (1982) is to transform an incoming signal pattern of arbitrary dimension into a one or two-dimensional discrete map, and to perform this transformation adaptively in a topological ordered fashion”. (Haykin, 1999)



Input Space and Grid Space

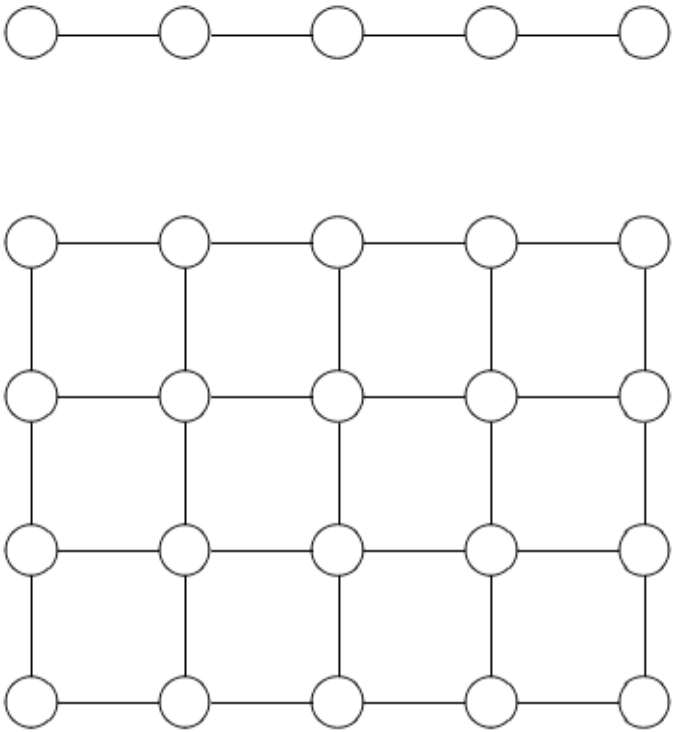
There are two spaces in which SOMs work:

The N-dimensional input space and the G-dimensional grid on which the neurons lie and which indicates the neighborhood relationships between the neurons and therefore the network topology.

Even if N = G, the two spaces are not the same and have to be distinguished.

In a one-dimensional grid, the neurons could be, for instance, like pearls on a string. Every neuron would have exactly two neighbors (except for the two end neurons). A two-dimensional grid could be a square array of neurons. Another possible array in two-dimensional space would be some kind of honeycomb shape. Irregular topologies are possible,too, but not used very often.

Topologies with more dimensions and considerably more neighborhood relationships would also be possible, but due to their lack of visualisation capability they are not employed very often.



How it works

The neurons are interconnected by neighborhood relationships which define the topology. The training of a SOM is highly influenced by the topology.

Like many other neural networks, the SOM has to be trained before it can be used.

At the start of the learning, all the weights are initialized to small random numbers

Learning consists of repeating the following sequence of steps until the map converges

1. Input an arbitrary value p of the input space $\mathbb{R}^N$
2. Calculate the distance between every neuron k and p by means of a norm, $\|\vec{p} - \vec{w}_k\|$
3. One neuron becomes active, namely the one with the shortest calculated distance to the input.
4. All other neurons remain inactive. This paradigm of activity is also called the “winner-takes-all” scheme
5. Adapt the centres

Adaptation of Neuron Centres

The neuron centres are moved within the input space according to the rule

$$\Delta w_k = \alpha(t)\eta(\nu, k, t)[\vec{p}(t) - \vec{w}_\nu(t)]$$

where the values Δw_k are simply added to the existing centres

the change in position of the neurons k is proportional to the distance to the input pattern p and, as usual, to a time-dependent learning rate $\alpha(t)$

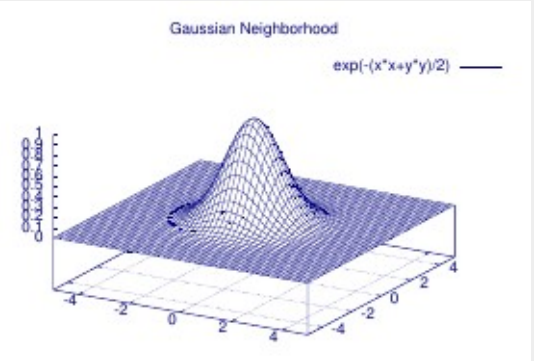
The above-mentioned network topology exerts its influence by means of the function $\eta(\nu, k, t)$

The topology function $\eta(\nu, k, t)$ describes the neighborhood relationships in the topology. It can be any unimodal function that reaches its maximum when i = k

Time-dependence is optional, but often used.

Neighbourhood Function

A common distance function is the familiar Gaussian bell.



It is unimodal with a maximum at 0.

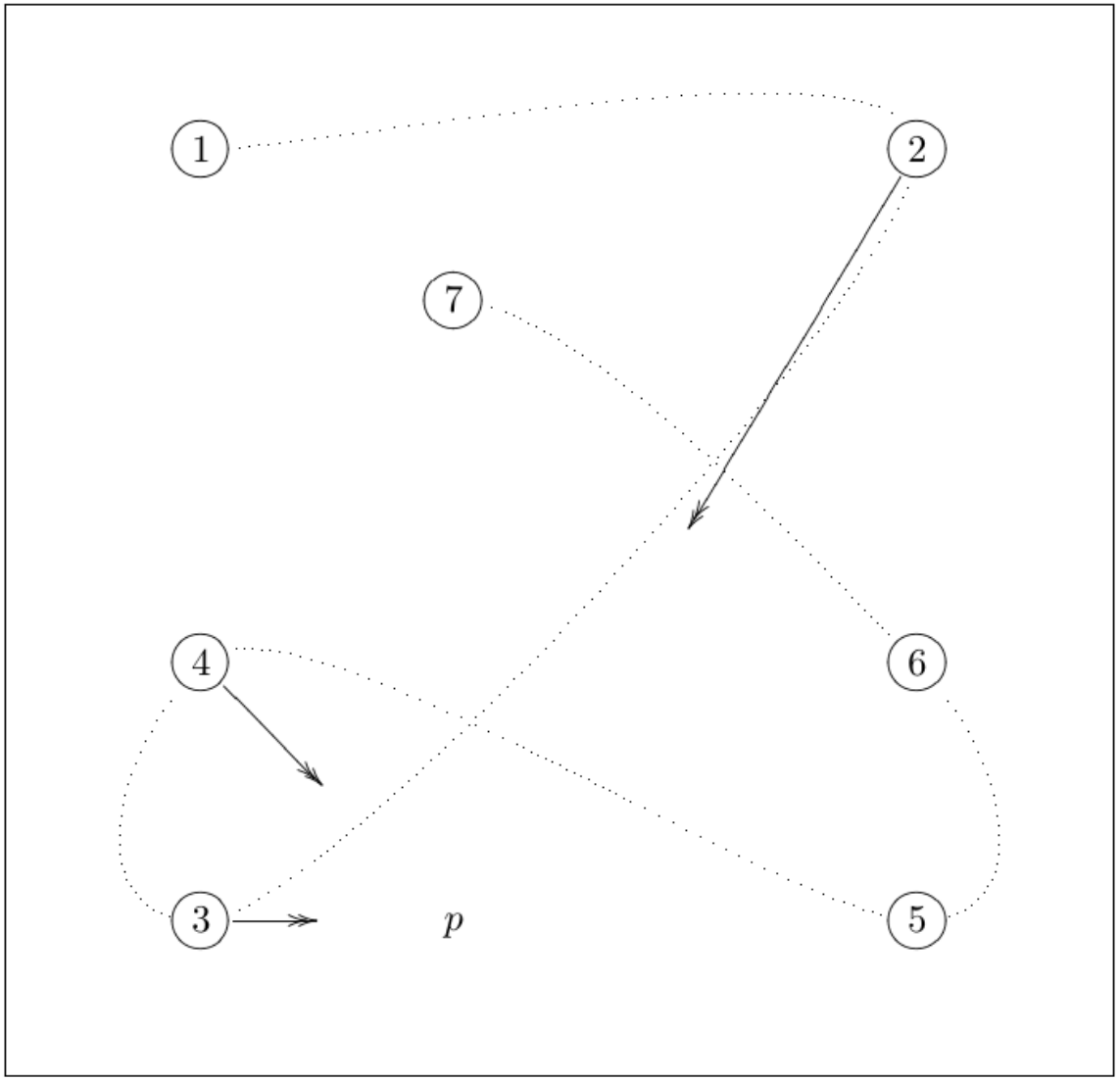
Additionally, its width can be changed by applying its parameter σ, which can be used to reduce the neighborhood with time: We simply relate the time-dependence to the σ and the result is a monotonically decreasing σ(t).

$$\eta(i,k,t)=\exp\left(\frac{-\|g_i-g_k\|^2}{2\sigma(t)^2}\right)$$

Other functions that can be used instead of the Gaussian function are, for instance, the cone function, the cylinder function or the Mexican hat function

Simple Example

From : D. Kriesel – A Brief Introduction to Neural Networks (ZETA2-EN) chapter 10.



The Learning Process Summarised

The learning process is roughly as follows:

initialise the weights for each output unit

loop until weight changes are negligible

    for each input pattern

        present the input pattern

        find the winning output unit

        find all units in the neighbourhood of the winner

        update the weight vectors for all those units

    reduce the size of neighbourhoods if required

Proof of Convergence?

- Although the SOM algorithm has a simple computational form, a formal analysis of it and the associated learning processes and mathematical properties is not easily realized.

- Some important issues still remain unanswered

- Self-organization has been studied in some depth
  - however a universal conclusion has been difficult, if not impossible, to obtain

- Convergence and ordering has only been formally proven in the trivial one-dimensional case

SOFMs in MATLAB

- The neurons in the single layer of an SOFM are arranged originally in physical positions according to a **topology function**

- The functions *gridtop, hextop,* or *randtop* can arrange the neurons in a grid, hexagonal, or random topology

- Distances between neurons are calculated from their positions with a distance function. There are four distance functions, *dist, boxdist, linkdist,* and *mandist.*
  - *link* distance is the most common.

Matlab *newsom* function

- net = **newsom** (PR,[D1,D2,...],TFCN,DFCN,OLR,OSTEPS,TLR,TND)

  takes

  PR - R x 2 matrix of min and max values for R input elements.

  I - Size of ith layer dimension, defaults = [5 8].

  TFCN - Topology function, default ='hextop'.

  DFCN - Distance function, default ='linkdist'.

  OLR - Ordering phase learning rate, default = 0.9.

  OSTEPS - Ordering phase steps, default = 1000.

  TLR - Tuning phase learning rate, default = 0.02;

  TND - Tuning phase neighborhood distance, default = 1.

  and returns a new self-organizing map.

- The topology function TFCN can be hextop, gridtop, or randtop.
- The distance function can be linkdist, dist, or mandist.

Kohonen Learning Rule in MATLAB (function learnk)

- The weights of the winning neuron (a row of the input weight matrix) are adjusted with the Kohonen learning rule
- Supposing that the i^{th} neuron wins, the elements of the i^{th} row of the input weight matrix are adjusted
- the neuron whose weight vector was closest to the input vector is updated to be even closer
- The result is that the winning neuron is more likely to win the competition the next time a similar vector is presented, and less likely to win when a very different input vector is presented
- Eventually, if there are enough neurons, every cluster of similar input vectors will have a neuron that outputs 1 when a vector in the cluster is presented, while outputting a 0 at all other times
- Thus, the competitive network learns to categorize the input vectors it sees.

Applications

- Data clustering and visualization.
- Optimization problems:
  - Traveling salesman problem.
- Semantic maps:
  - Natural language processing.
- Preprocessing for signal and image-processing.
  - Hand-written character recognition
  - Phonetic map for speech recognition
- Machine Vision
- Telecommunications

An Early Application:The “Phonetic Typewriter”

from Beale and Jackson, “Neural Computing, an Introduction”

perhaps easier in his native tongue of Finnish than it would be in other languages (Finnish being a phonetic language), but it was still quite a complex task. Kohonen has approached the problem applying a mixture of the best of many techniques—he is quick to point out that neural networks are not a universal panacea for all aspects of a data processing problem. The system that he devised is shown schematically in the following figure, figure 5.5.

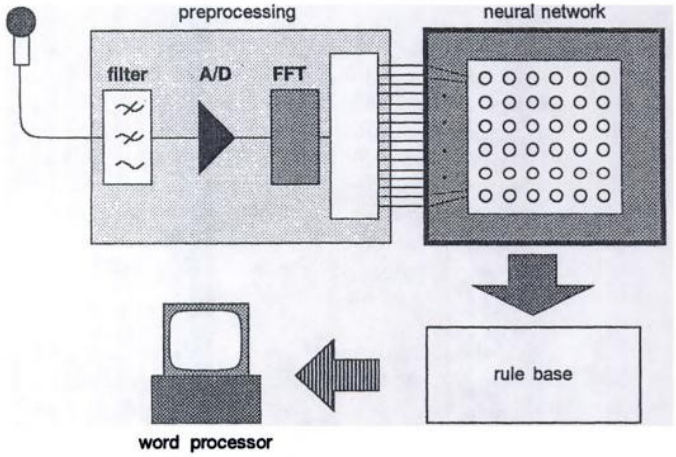


Figure 5.5 A schematic circuit of Kohonen's neural based phonetic typewriter.

The neural network is only dealing with one part of the total task. The system is not totally “neural”—in fact the neural network

An Early Application in Robotics

- Ritter et al.(1989) Used a Kohonen network to learn inverse kinematics of a 3 link arm

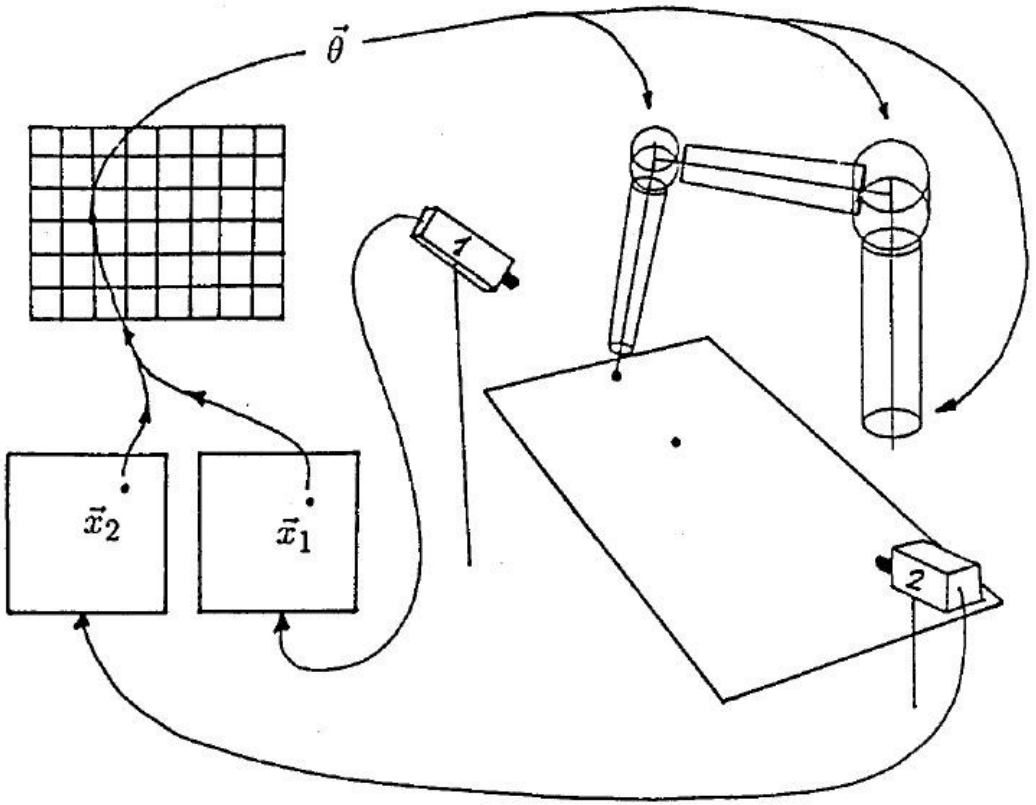
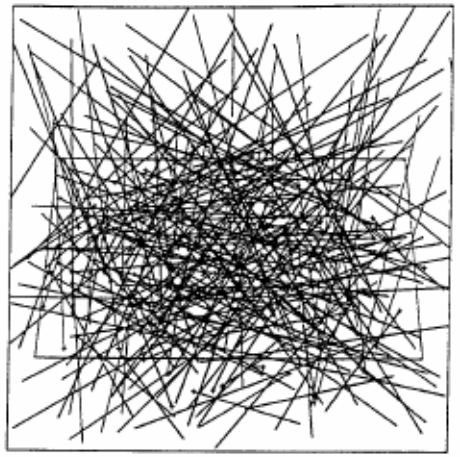
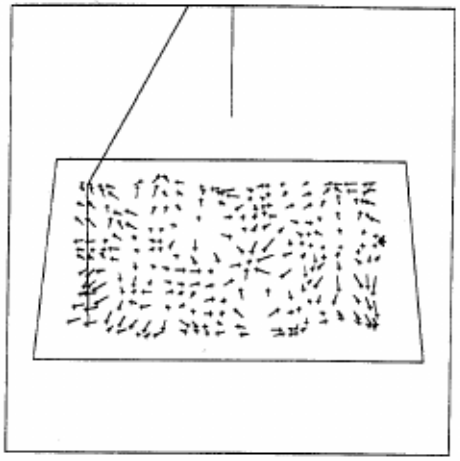


Fig.1 The simulated system. Two cameras observe the robot arm behind the table. Each camera has a quadratic image plane and maps a specified location from the scene to a pair of coordinates $\vec{x}_i$ $(i = 1, 2)$. The four-component vector $\mathbf{x} = (\vec{x}_1, \vec{x}_2)$ is fed as input to the array of neurons. The output of the array is determined by the neuron s whose vector $\mathbf{w}_s^{in}$ matches the "sensory input" $\mathbf{x}$ best.

Learning of End-effector Positions





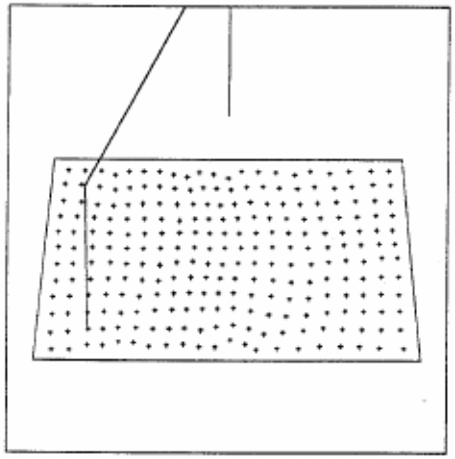
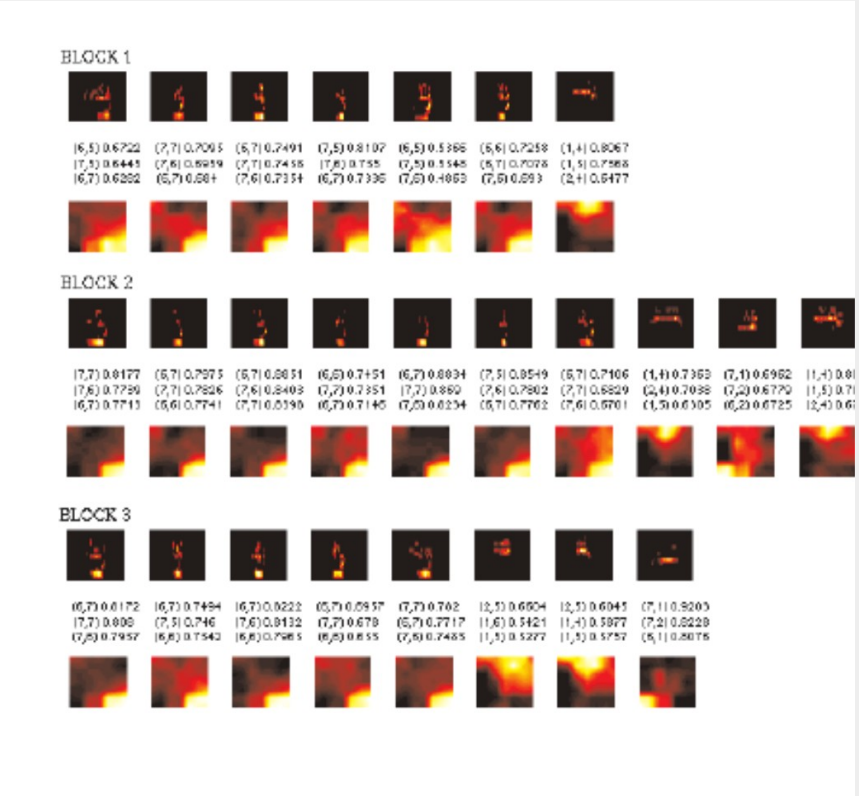


Fig.2a-c Error of end effector positions as seen from camera 1. For each of the 12 × 21 target positions associated with $\mathbf{w}_r^{in}$ the resulting end effector location is indicated by a cross mark. The mismatch to the intended target position is indicated by an appended line segment. From top left to bottom left: (a) Large initial errors, reflecting random initialization of the system. (b) After 4000 trial movement only small errors remain. Furthermore, target positions outside the table surface, being never required during training, are no longer represented by neurons. (c) After 20000 iterations, no visible deviations between target and achieved end effector locations remain.

Comparison with MLP and RBF

- Gorinevsky and Connolly (1994)
- compared Kohonen Map, a MLP, a RBF and a Local Polynomial Fit
- on the problem of learning the inverse kinematics of a 3 link manipulator
  - The RBF and Local Polynomial methods had lowest error
  - but RBF was worse with added random noise
  - KM was about equal with MLP for accuracy

Whale Song Classification



- A widespread problem in the study of humpback whale song vocalizations involves evaluating the similarity of song elements within a whale's repertoire, between individuals of a social group, and between social groups separated by time and space.
- Whilst humpback whale songs demonstrate a remarkable amount of regular high level structure, they are composed of a variety of complex and transient elemental phonological units.
- Reliable classification of song structure requires robust unit classification - a feature which has made this process difficult to automate.
- This work presents a fully automated technique for performing multiple-resolution unit classification.
- In this scheme, units are simultaneous assigned membership to a series of increasingly general acoustic classes such that degrees of song structural similarities (and differences) emerge from analysis of units classified at different resolutions.

Walker, A. Fisher, R. & Mitsakakis, N. (1996), 'Singing maps: Classification of whalesong units using a self-organizing feature mapping algorithm' , Journal of the Acoustical Society of America

Classification of Acoustic Emissions Data

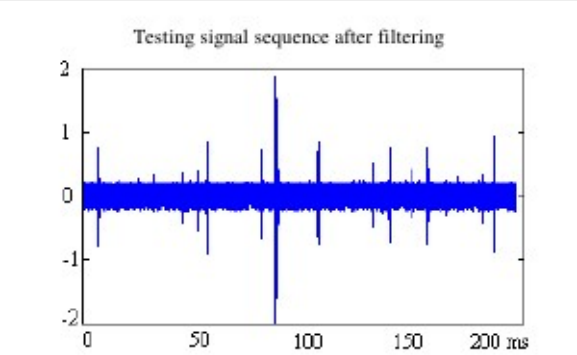


Fig. 3 A segment of time domain signal from sensor 2

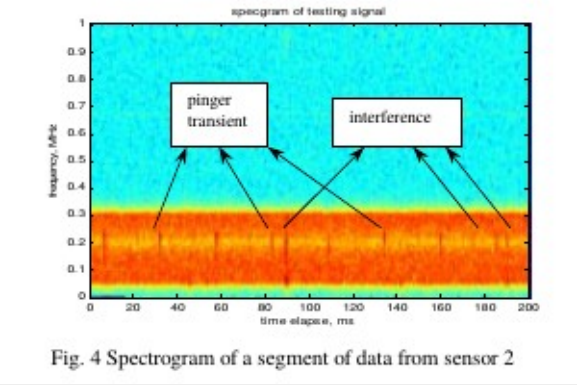


Fig. 4 Spectrogram of a segment of data from sensor 2

- Acoustic Emissions (AE), generated by the formation and growth of micro-cracks in metal components, provide a promising mechanical fault detection technique in monitoring complex-shaped components in helicopters and aircraft
- A major challenge for an AE-based fault detection algorithm is to distinguish crack related AE signals from other interfering transient signals, such as fretting related AE signals and electromagnetic transients.
- this paper presents a classifier, which makes its decision based on the features extracted from joint time-frequency distribution data by Self-Organizing Map (SOM) neural network
- In-flight data are used to test the performance of this classification system, with promising results

H.Sun,M Kaveh,A.H.Tewfik, “SelfOrganizing Map Neural Network for Transient Signal Classification in Mechanical Diagnostics” IEEEIEURASIP Workshop on Nonlinear Signal Processing, June 1999, Antalia, Turkey

Learning Vector Quantization

- The previous methods are un-supervised, meaning that the user doesn't have any preconceived idea of which data points belong to what cluster
- If the user supplies a target class for each data point and the network uses this information when training, we have **supervised learning**
- Kohonen (1989) suggested a supervised version of the clustering algorithm for competitive networks called learning vector quantization (LVQ)
- First an un-supervised competitive learning run is made.
- Then each output neuron in the competitive network is assigned to one of the target classes (usually the class to which most of "its" data points belong).
- There are usually more output neurons than classes, which means that several output neurons can belong the the same class
- Finally a LVQ run is made, where the idea is to "unlearn" the bad clusters and learn the "good" ones (i.e. the ones where the winning class agrees with the target class)

References and Further Reading

- Kohonen (1982) Biol.Cybern. 43,59-69 “Self-Organized Formation of Topologically Correct Feature Maps”

- D. Kriesel – A Brief Introduction to Neural Networks (ZETA2-EN). Available from:

  http://www.dkriesel.com/en/science/neural_networks

- D Gorinevsky and TH Connolly (1994), Neural computation 6 (3), 521-542. Comparison of some neural network and scattered data approximations: The inverse manipulator kinematics example

- H Ritter, T Martinetz and K Schulten (1991), Neural computation and self-organizing maps [revised English edition], Reading, MA: Addison Wesley

- S. Haykin, Neural Networks,1999, Prentice Hall (recommended reading)

- Beale and Jackson, Neural Computing: An Introduction, Adam Hilger, 1991