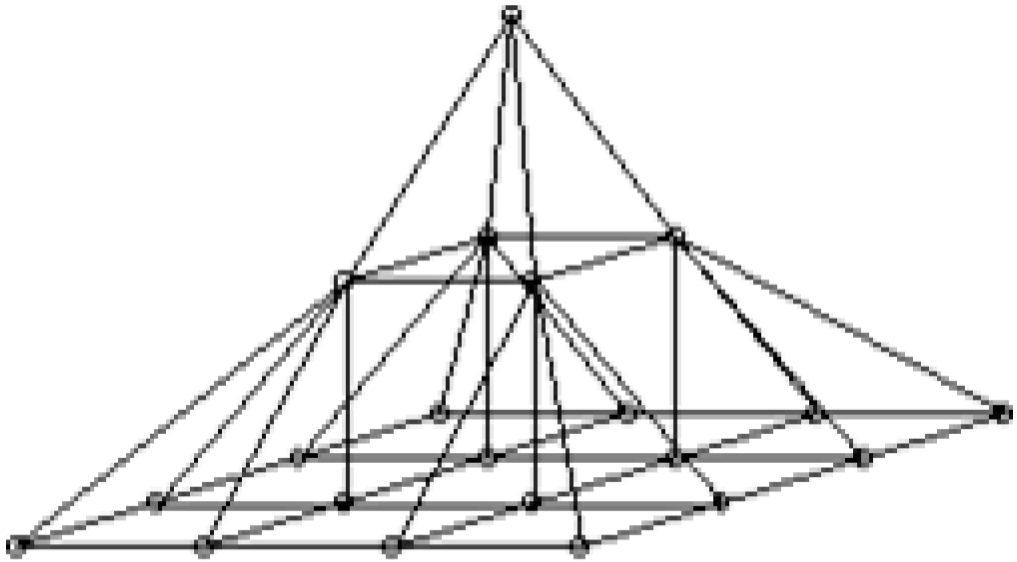
Tree-Structured SOM (TS-SOM)



9.1.2009

Mikko Kolehmainen

Tree-Structured SOM (TS-SOM)

- The learning rule of the TS-SOM has been modified
- The neighbourhood function has been reduced to a fixed form where the neighbourhood id always defined to be the four adjacent neurons of the BMU
- The role of the altering neighbourhood radius is handled by the previous (more coarse) levels of the pyramid-like structure of several SOMs

$$\mathbf{W}_m(t+1) = \mathbf{W}_m(t) + a(t)\left[\mathbf{X}_i(t) - \mathbf{W}_m(t)\right]$$

Comparable traditional methods

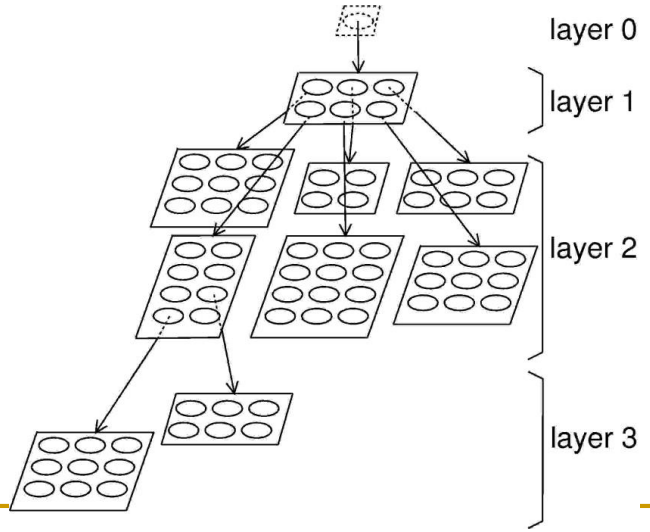
- Linear regression
  - a good analog to MLP neural networks
  - can be used to extract seasonal variations
- Principal Components Analysis (PCA)
  - works similarly to SOM
  - can be used for data pre-processing

In spite of the stability and popularity of the *Self-Organizing Map (SOM)*, at least two limitations have to be noted, which are related, on the one hand, to the

- static architecture of this model, as well as, on the other hand, to the
- limited capabilities for the representation of hierarchical relations of the data.

- With our novel ***Growing Hierarchical Self-Organizing Map (GHSOM)*** we address both limitations.

- The growing hierarchical som is an artificial neural network model with hierarchical architecture composed of independent growing self-organizing maps.
- By providing a global orientation of the independently growing maps in the individual layers of the hierarchy, navigation across branches is facilitated



Automatic face recognition

• The Gabor Filters have received considerable attention because the characteristics of certain cells in the visual cortex of some mammals can be approximated by these filters.

• In addition these filters have been shown to posses optimal localization properties in both spatial and frequency domain

The complex 2-D Gabor function is written as [Daug88]

$$G(x,y) = \exp\left(-\pi\left\{(x-x_0)^2\alpha^2 + (y-y_0)^2\beta^2\right\}\right) \cdot \exp\left(-2\pi i\left\{u_0(x-x_0) + v_0(y-y_0)\right\}\right) \tag{4.2}$$

with Fourier transform $F(u,v)$

$$F(u,v) = \exp\left(-\pi\left\{\frac{(u-u_0)^2}{\alpha^2} + \frac{(v-v_0)^2}{\beta^2}\right\}\right) \cdot \exp\left(-2\pi i\left\{x_0(u-u_0) + y_0(v-v_0)\right\}\right) \tag{4.3}$$

where $i = \sqrt{-1}$ is the imaginary unit, α and β are the variances of the Gaussian frequency band, x_0 and y_0 define the location of the function in the image and u_0 and v_0 define the mean frequency in F-domain. Note, that the latter term in Eq.(4.3) encodes the phase shifts for translating the function to (x_0, y_0) and vanish from the equation for the Fourier amplitude of the function (i.e. the frequency response of the filter).

Two limiting cases of the Gabor functions are:

- $\alpha, \beta \to \infty$:

$$\begin{aligned} G(x,y) &\to \delta(x-x_0, y-y_0) \\ |F(u,v)| &\to 1 \end{aligned} \tag{4.4}$$

  This corresponds to sampling the image by delta function, while no information from the frequency domain is acquired.

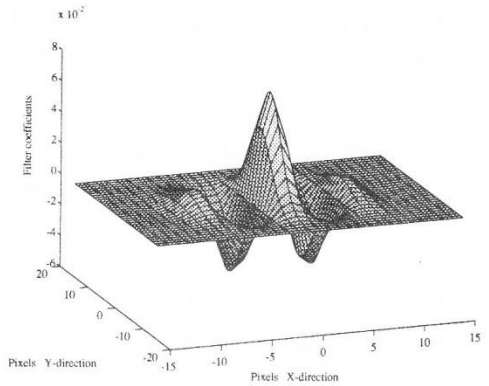


Figure 4.1: The real part of a Gabor filter

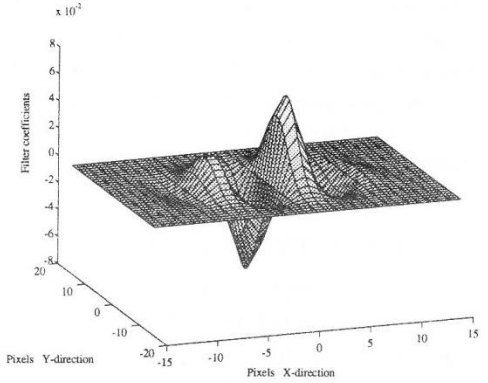


Figure 4.2: The imaginary part of a Gabor filter

- $\alpha, \beta \to 0$:

$$\begin{aligned} G(x,y) &\to \exp\left(-2\pi i \left\{u_0(x-x_0) + v_0(y-y_0)\right\}\right) \\ |F(u,v)| &\to \delta(u-u_0, v-v_0) \end{aligned} \tag{4.5}$$

This corresponds to sampling the frequency domain, and the spatial filters are infinite sinusoids, i.e. the base functions of the Fourier transformation.

The Gabor functions provide a complete basis for R^2, so that any signal can be composed of the elementary Gabor filters. They are not orthogonal, since the inner product of two functions $G_i(x,y)$ and $G_j(x,y)$ is

$$< G_i(x,y), G_j(x,y) >= \exp\left(-\pi \left\{ \frac{(u_i-u_j)^2}{\alpha_i^2+\alpha_j^2} + \frac{(v_i-v_j)^2}{\beta_i^2+\beta_j^2} \right\}\right) \tag{4.6}$$

σ_x, σ_y are the standard deviations of the Gaussian envelope along the x- and y-dimensions, f is the central frequency of the sinusoidal plane wave, and θ_n the orientation. The rotation of the x-y plane by an angle θ_n will result in a Gabor filter at the orientation θ_n. The angle θ_n is defined by:

$$\theta_n = \frac{\pi}{p}(n-1), \tag{7}$$

for n=1,2,….p and $p \in \mathbf{N}$, where p denotes the number of orientations.

Design of Gabor filters is accomplished by tuning the filter with a specific band of spatial frequency and orientation by appropriately selecting the filter parameters; the spread of the filter σ_x, σ_y, radial frequency f, and the orientation of the filter θ_n. The important issue in the design of Gabor filters for face recognition is the choice of filter parameters. This research organizes 15 Gabor channels consisting of five

$$\left\| g_{f,\theta}(x,y) \right\| = \sqrt{\Re^2\left\{ g_{f,\theta}(x,y) \right\} + \Im^2\left\{ g_{f,\theta}(x,y) \right\}} \tag{9}$$

This research uses the magnitude response $\left\| g_{f,\theta}(x,y) \right\|$ to represent the features. To reduce the influence of the lighting conditions, the output of Gabor filter about each direction has been normalized. In the sequel, a transformation $Q_{f,\theta}(x,y)$ to which $\left\| g_{f,\theta}(x,y) \right\|$ is subjected is given by:

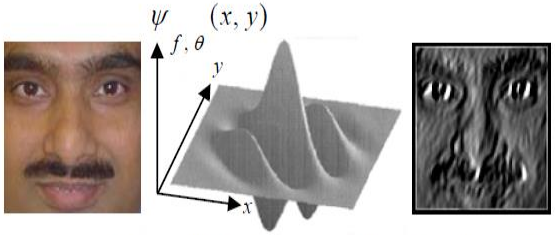


Fig. 3 Convolution result of a face image with a Gabor filter. (a) Face image, (b) Gabor filter (f=0.19,θ=3π/4,σ_x=σ_y=3), (c) Output of Gabor filter

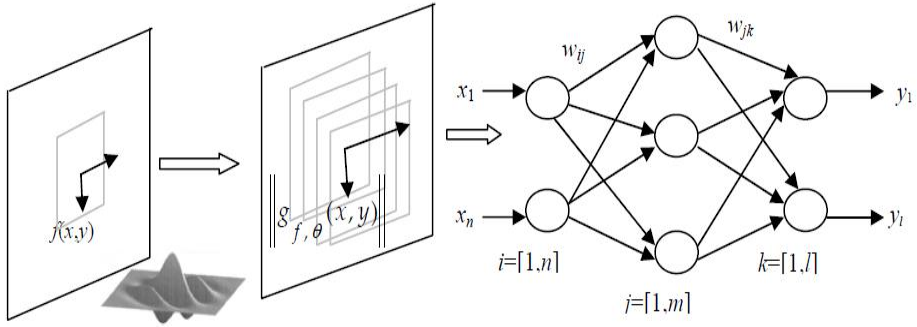


Fig. 4 Network architecture of Gabor based multi-layer perceptron

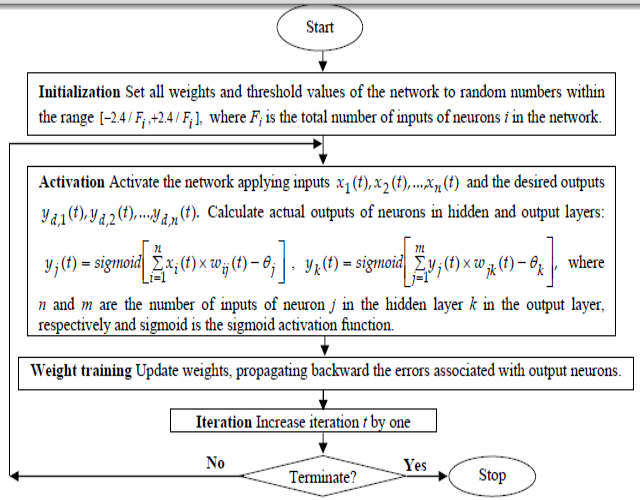


Fig. 5 Flowchart for backpropagation algorithm

$$Q_{f,\theta}(x,y) = \frac{\left\| g_{f,\theta}(x,y) \right\|}{\sum_{\theta \in \Theta} \left\| g_{f,\theta}(x,y) \right\|}. \quad (10)$$

The function $Q_{f,\theta}(x,y)$ takes the value in the range [0,1] and the parameter θ is in $\Theta \equiv \left\{0, \frac{\pi}{5}, \frac{2\pi}{5}, \frac{3\pi}{5}, \frac{4\pi}{5}\right\}$.

IV. Gabor Filter Based Neural Network

Face recognition is achieved by employing a multilayer perceptron with backpropagation algorithm. The architecture of the neural network is illustrated in Fig. 4. The first layer, called Gabor layer, receives the Gabor features. The number of nodes in this layer is, obviously equal to the dimension of the feature vector incorporating the Gabor features. The number of nodes in the output layer equals to the number of individual faces the network is required to recognize. The number of epochs for this experiment was 10,000 and the goal was 0.01. The back-propagation training algorithm is illustrated in a flowchart as shown in Fig. 5.

V. Experimental Results and Performance

In order to evaluate the effectiveness of the proposed

200 images of 40 individual subjects with 2 different poses and 5 different illumination conditions. Half of the images in the database were used as a training dataset and the remaining images were used as probe images in the recognition test. All images were subjected to Gabor filters and were convolved with 15 Gabor filters. To each face image, the outputs were 15 images which record the magnitudes of the Gabor filter responses. Fig. 6 shows the results for a typical face image. The output of the Gabor filters was used to train the neural network. The training curve, indicating the gradual reduction in error over several epochs due to backpropagation learning algorithm, is shown in Fig. 7.

```
function gb=gabor_fn(sigma,theta,lambda,psi,gamma)
sigma_x = sigma;
sigma_y = sigma/gamma;

% Bounding box nstds = 3;

xmax = max(abs(nstds*sigma_x*cos(theta)),abs(nstds*sigma_y*sin(theta)));
xmax = ceil(max(1,xmax));
ymax = max(abs(nstds*sigma_x*sin(theta)),abs(nstds*sigma_y*cos(theta)));
ymax = ceil(max(1,ymax));
xmin = -xmax;
ymin = -ymax;
[x,y] = meshgrid(xmin:xmax,ymin:ymax);
% Rotation x_theta=x*cos(theta)+y*sin(theta);
y_theta=-x*sin(theta)+y*cos(theta);
gb= 1/(2*pi*sigma_x *sigma_y) *
exp(.5*(x_theta.^2/sigma_x^2+y_theta.^2/sigma_y^2)).*cos(2*pi/lambda*x_theta+
psi);
```

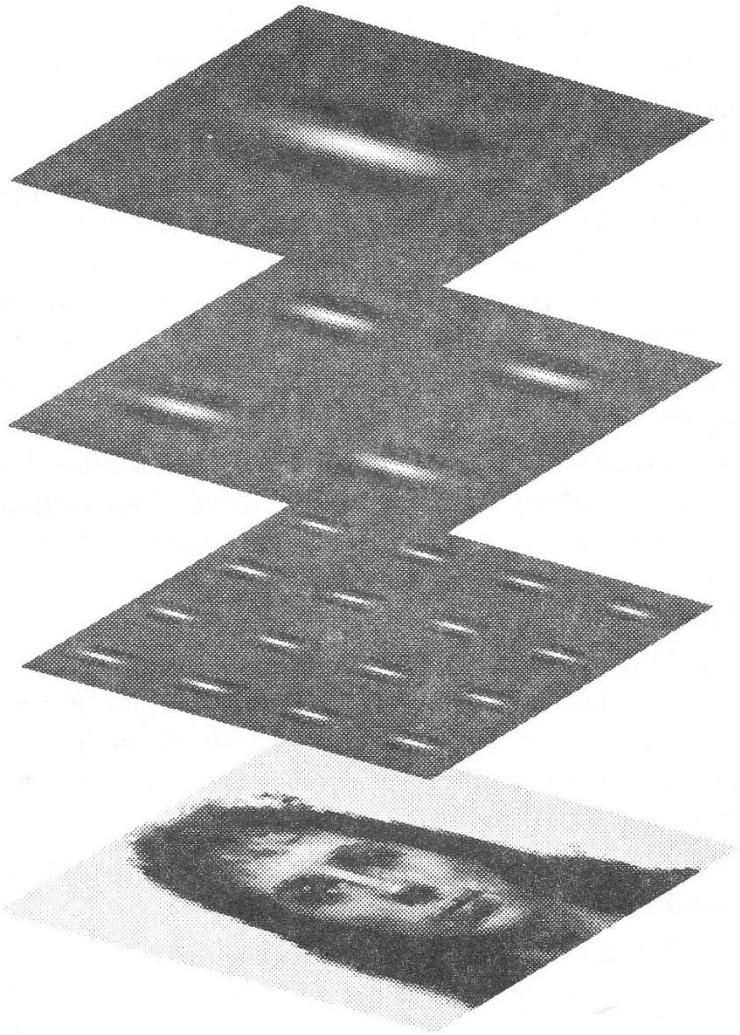


Figure 4.4: A schematic figure showing the Gabor pyramid over an image. The mean frequencies of the filter planes are from down to upwards $\pi/2$, $\pi/4$ and $\pi/8$, respectively. To capture all the information from the image the filters should sample the image about 4 times more densely.

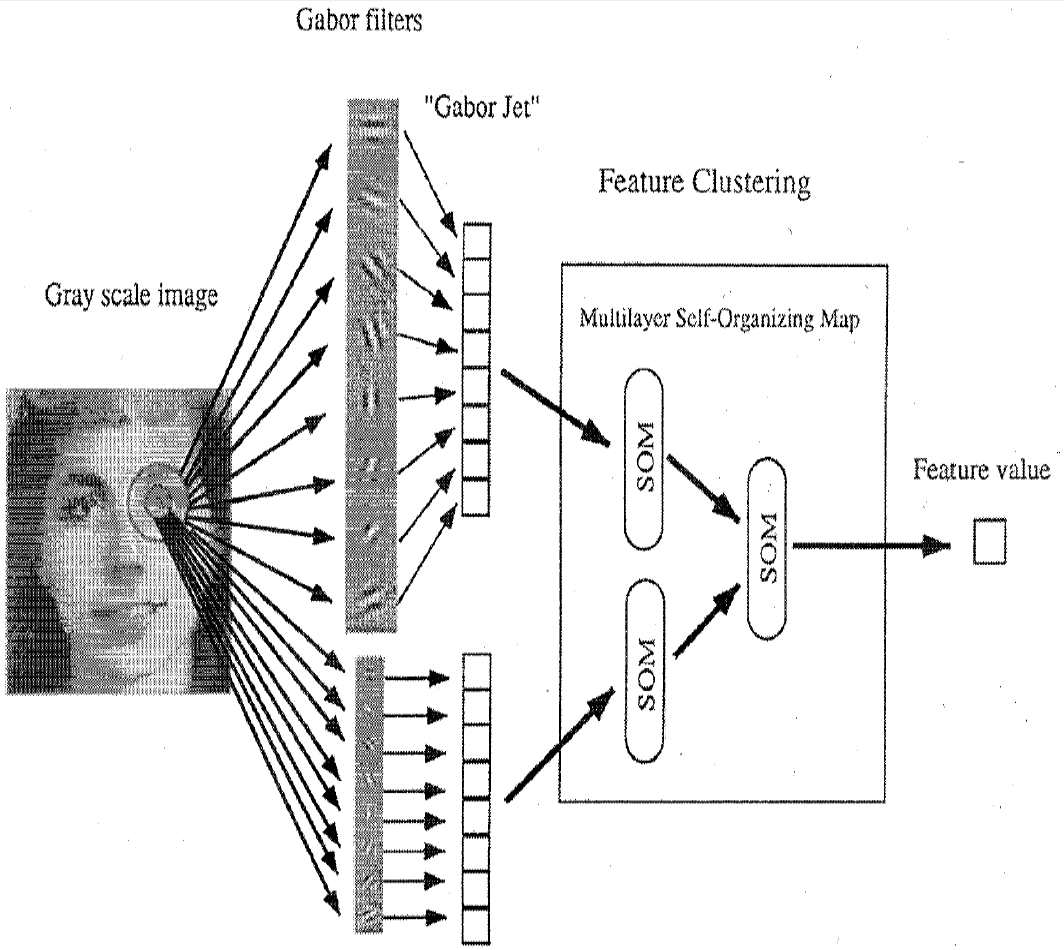


Fig. 7. Schematic drawing of feature extraction and compression in object recognition. First, a bank of 16 Gabor filters is applied to all image windows. Gabor filters are orientation- and spatial-frequency-selective masks that detect characteristic microfeatures of the face images. Each small region in the image is represented as a "Gabor jet" (set of outputs from the 16 local Gabor filters). These vectors are clustered by a two-layer SOM that has been trained by a number of face images and other natural objects. The first-layer SOM's divide their input space into convex clusters centered around the map weight vectors, and the second SOM combines these clusters into 100 nonconvex clusters that are able to capture small variations in the images. The cluster number is used as a feature code for that image region.

The Growing Hierarchical Self-Organizing Map: Exploratory Analysis of High-Dimensional Data

Advisor：Dr. Hsu

Graduate：Sheng-Shuan Wang

Authors：Andreas Rauber

Dieter Merkl

Michael Dittenbach

Outline

- Motivation
- Objective
- Introduction
- SOM
- Related Architectures
- GHSOM
- Experiments
- Conclusions
- Personal Opinion
- Review

Motivation

- The SOM, however, at least two limitations have to be noted.
  - The static architecture of the model.
  - The limited capabilities for the representation of hierarchical relations of the data.

Objective

- To provide a model, growing hierarchical SOM(GHSOM), to overcome those problems.

Introduction

- Two limitations of the SOM
  - First, the SOM uses a fixed network architecture.
  - Second, hierarchical relations between the input data are not mirrord in a straightforward fashion.

SOM

- Find the winner from (1).
- Update the winner and neighborhood according to (2).

$$c(t) = \arg\min_i \{ \| x(t) - m_i(t) \| \} \quad (1)$$

$$m_i(t+1) = m_i(t) + \alpha(t) h_{ci}(t) [x(t) - m_i(t)] \quad (2)$$

$$h_{ci}(t) = \exp\left(-\frac{\| r_c - r_i \|^2}{2 \cdot \delta(t)^2}\right) \quad (3)$$

Related Architectures

- The identification of inter- and intra- cluster similarity has been addressed.
  - U-Matrix, adaptive coordinates, and cluster connections.
- LabelSOM, grouping units that have the same characteristics allows to identify clusters within the output space of the SOM.
- The hierarchical feature map, a tree-structured SOM, a hierarchical modification of the SOM, is introduced.

Related Architectures

- The shortcoming of having to define the size of the SOM in advance has been addressed by a number of different models.
  - Incremental grid growing.
  - Growing grid.
  - Growing SOM.
  - Hypercubical SOM.

GHSOM

- A. Principles
  - The GHSOM has a hierarchical structure of multiple layers, where each layer consists of several independent growing SOMs.

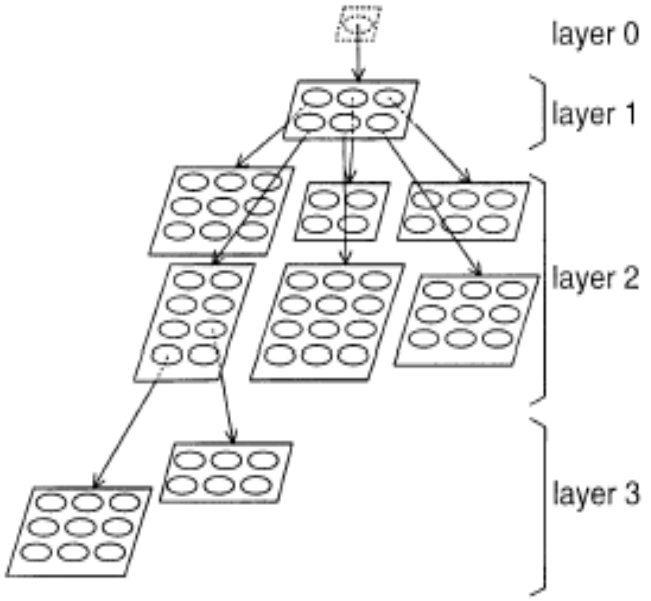


Fig. 1. GHSOM: The GHSOM evolves to a structure of SOMs reflecting the hierarchical structure of the input data.

GHSOM

- B. Training Algorithm.
- *1) Initial Setup and Global Network Control:*
  - The principle of the GHSOM architecture is its adaptation to the training data.
  - Two different strategies can be used for the control of the growth process.
    - The mean quantization error (mqe) of a unit.
    - Or the absolute value, i.e., the quantization error (qe) of a unit.

GHSOM

- The mqe of a unit i is calculated according to (4).
- The starting point for the GHSOM training process is the calculation of an mqe0 of the unit forming the layer 0 map as provided in (5).

$$mqe_i = \frac{1}{n_C} \cdot \sum_{x_j \in C_i} \| m_i - x_j \|, \quad n_C = | C_i |, \; C_i \neq \phi \qquad (4)$$

$$mqe_0 = \frac{1}{n_I} \cdot \sum_{x_i \in I} \| m_0 - x_i \|, \quad n_I = | I | \qquad (5)$$

GHSOM

- The minimum quality of data representation of each unit will be specified as a fraction, indicated by a parameter τ_2, of mqe_0 .
- In other words, all units must represent their respective subsets of data, satisfy the global termination criterion specified in (6).

$$mqe_i < \tau_2 \cdot mqe_0 \qquad (6)$$

GHSOM

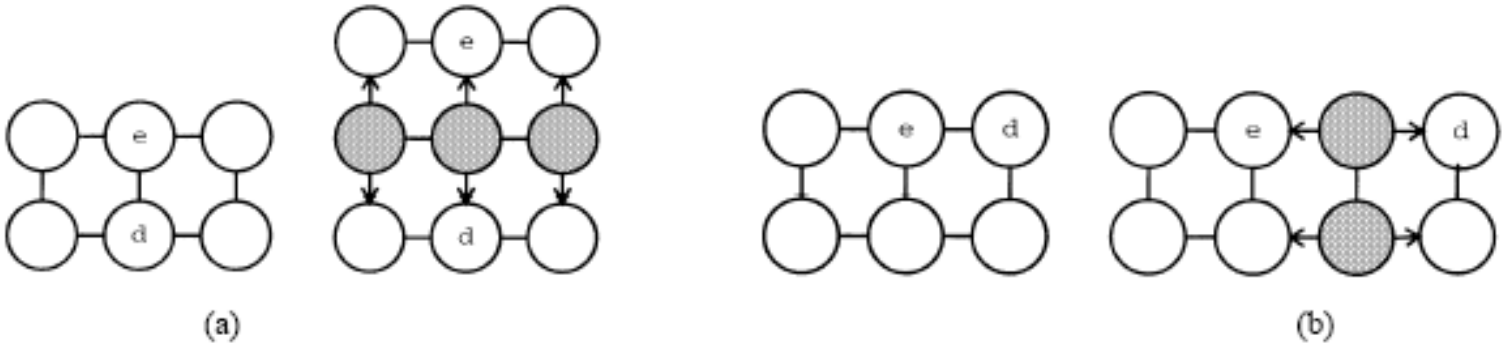
- Alternatively, the quantization error (qe) of the unit as given in (7). And the global termination criterion as given in (8).

$$qe_i = \sum_{x_j \in C_i} \| m_i - x_j \| \qquad (7)$$

$$qe_i < \tau_2 \cdot qe_0 \qquad (8)$$

GHSOM

- *2) Training and Growth Process of a Growing SOM:*
  - The unit with the highest qe is selected and denoted as the *error unit*. We will refer to the error unit as *e*.
  - Next, the most dissimilar neighboring unit *d* in terms of input space distance is selected.
  - A new row or column of units is inserted between *e* and its most dissimilar neighbor *d*.



GHSOM

- More formally, the growth process of a growing SOM can be described as follows.

$$e = \arg\max_i \Big(\sum_{x_j \in C_i} \| m_i - x_j \| \Big), \quad n_C = | C_i |, \; C_i \neq \phi \qquad (9)$$

$$d = \arg\max_i \big(\| m_e - m_i \| \big), \quad m_i \in N_e \qquad (10)$$

GHSOM

- *3) Termination of Growth Process:*
  - The MQE of a map is computed as the mean of all units' quantization errors qe_i of the subset u of the maps' units onto which data is mapped

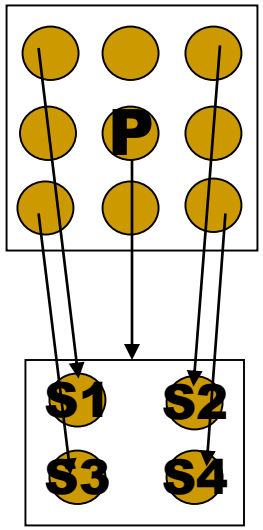
$$MQE_m = \frac{1}{n_u} \cdot \sum_{i \in u} qe_i, \qquad n_u = |u| \qquad (11)$$

In general terms, the stopping criterion for the growth of a single map m is defined as follows:

$$MQE_m < \tau_1 \cdot qe_u \qquad (12)$$

GHSOM

- *4) Hierarchical Growth With Global Map Orientation:*



GHSOM

- *5) Analysis of GHSOM Characteristics:*
  - The hierarchical structure of data can be represented in different forms.
    - Lower hierarchies with rather detailed refinements presented at each subsequent layer.
    - Deeper hierarchies, which provide a stricter separation of the various subclusters by assigning separate maps.
    - Parameter τ_1 is used to control this tradeoff between shallow or deep hierarchies.

Experiments

- *A. Experiment 1: TIME Magazine*
  - 1) Data Representation
  - 2) SOM of the TIME Magazine Collection
  - 3) Hierarchical Archive of the TIME Magazine Collection:
  - 4) Comparison of SOM and GHSOM Representation

Experiments

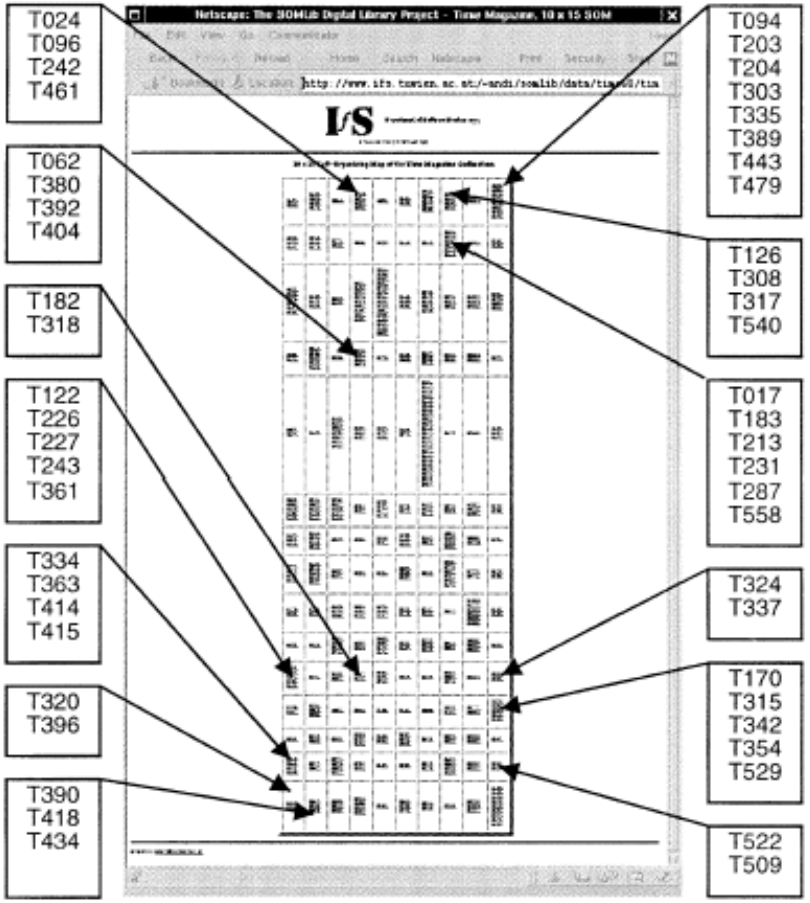


Fig. 4. *TIME* Magazine SOM: 10 × 15 map of the *TIME* Magazine collection.

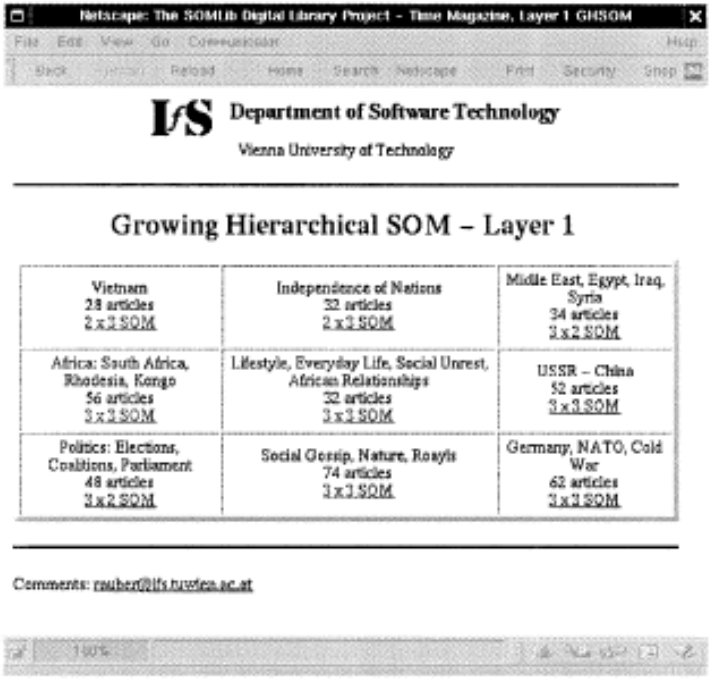


Fig. 5. *TIME* Magazine GHSOM Layer 1: 3 × 3 map of the *TIME* Magazine collection.

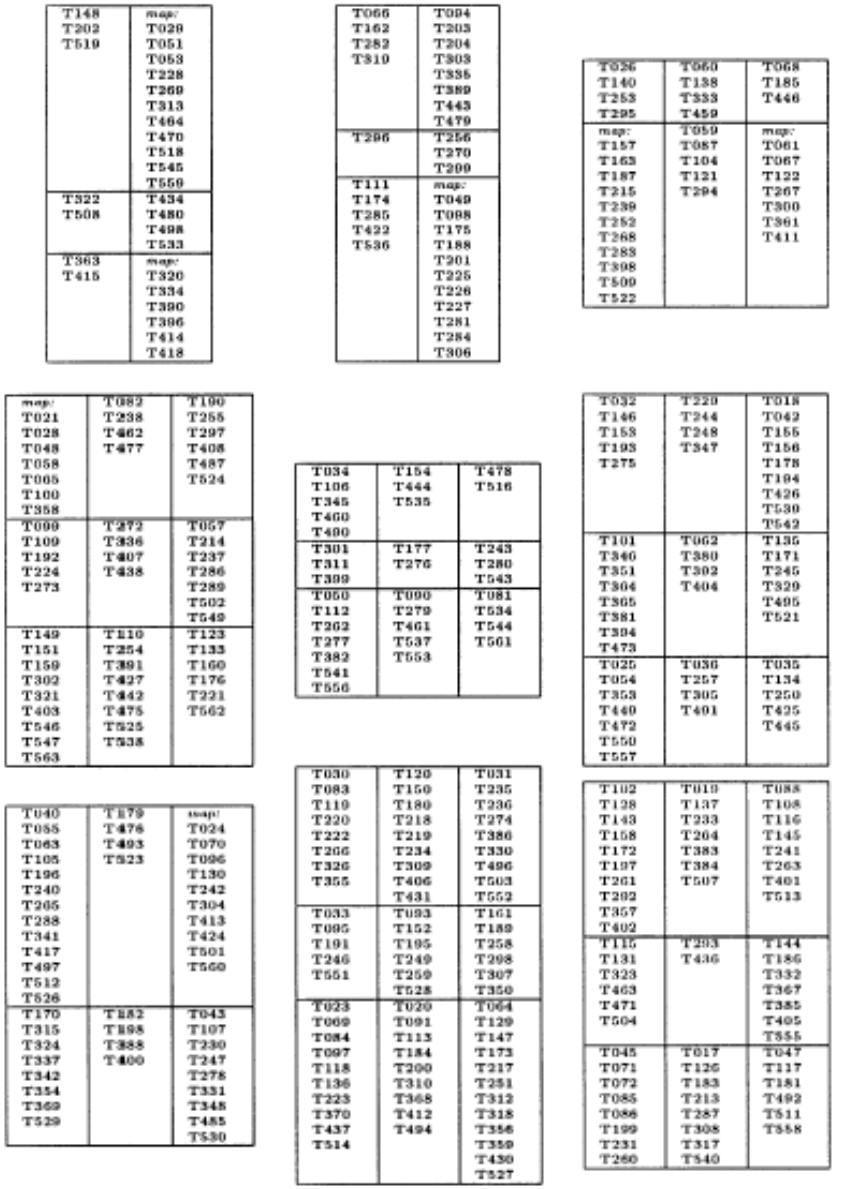
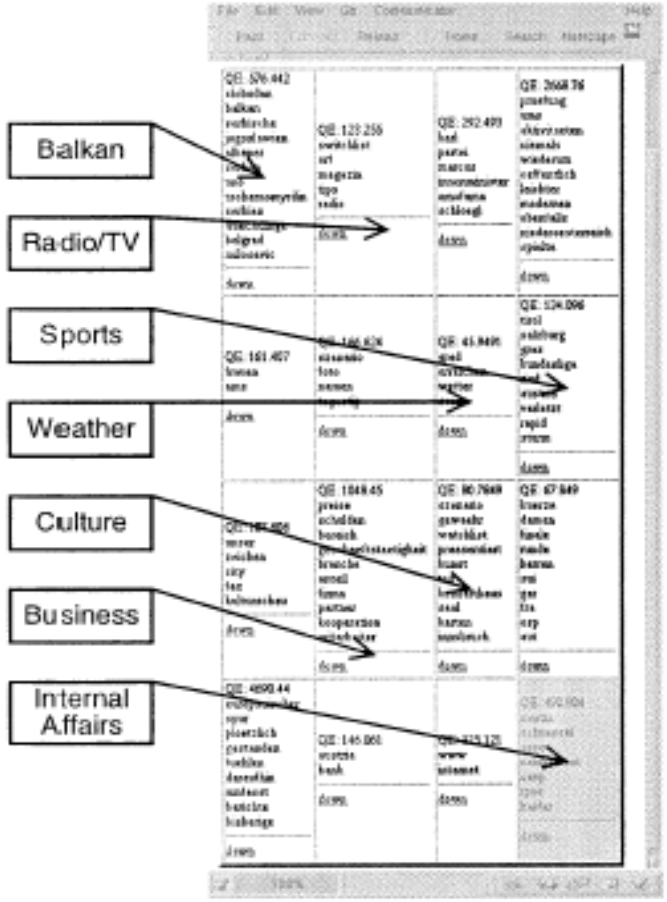


Fig. 6. *TIME* Magazine GHSOM Layer 2: 9 layer-2 maps of the *TIME* Magazine collection: 1 SOM per unit of the layer-1 map.

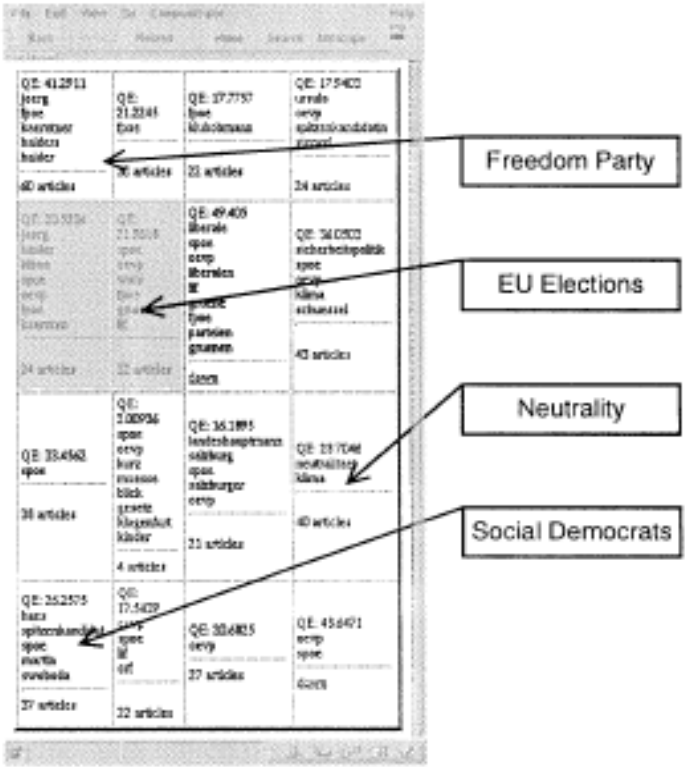
Experiments

- *B. Experiment 2: Two Hierarchies of Newspaper Articles From "Der Standard"*
  - 1) Data Representation
    - 11627 articles, 3799 D, weighted by tf * idf, mqe0=12180.3.
  - 2) Deep Hierarchy
    - τ_1 =0.07, 13 layers, layer1=4*4.
  - 3) Shallow Hierarchy
    - τ_1 =0.035, 7 layers, layer1=7*4.

Experiments

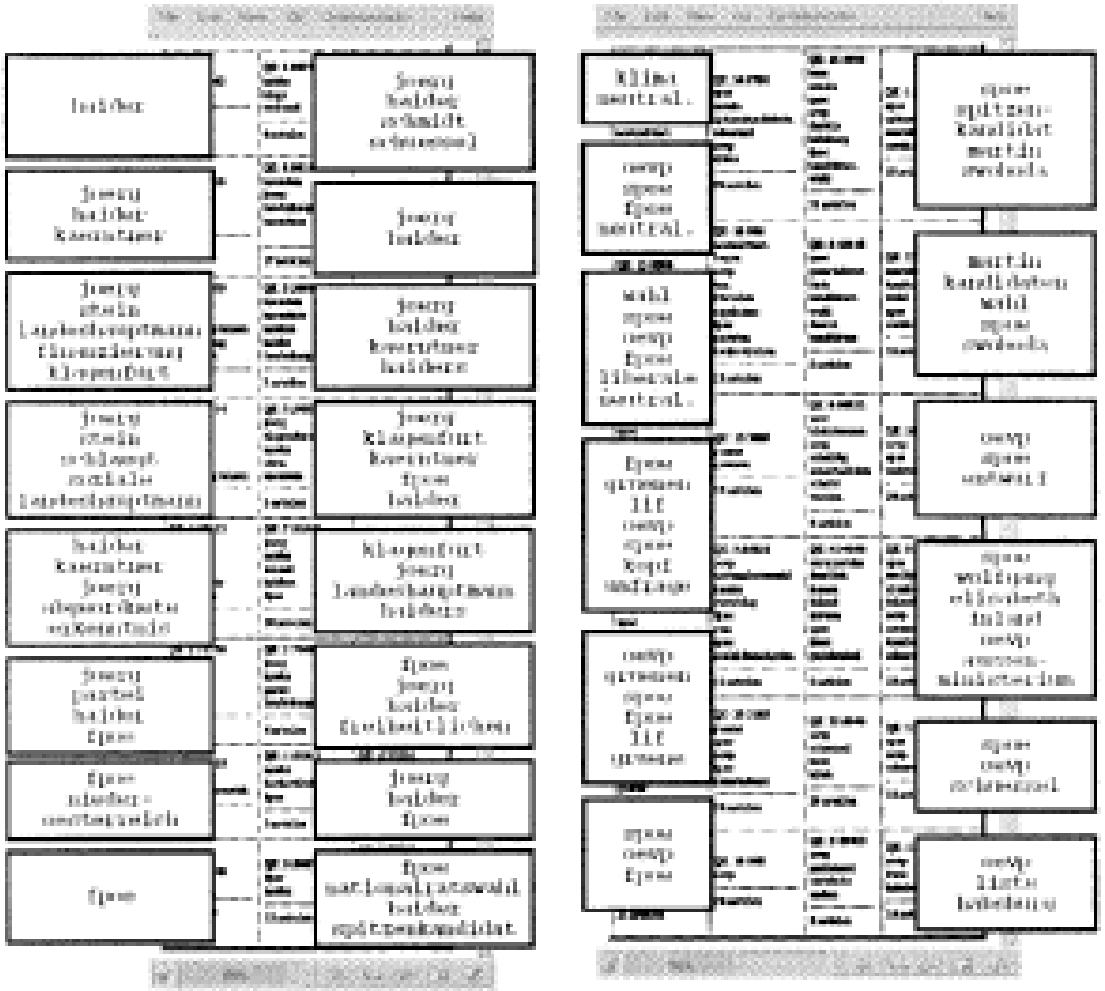


(a)



(b)

Experiments



Conclusions

- The major benefits of our GHSOM model compared with the standard SOM are the following.
  - The overall training time.
  - The hierarchical structure of the data.
  - With the various emerging maps at each layer of the hierarchy being rather small in size.
  - By ensuring a consistent global orientation of the individual maps in the respective layers.
  - The GHSOM with an information retrieval application.

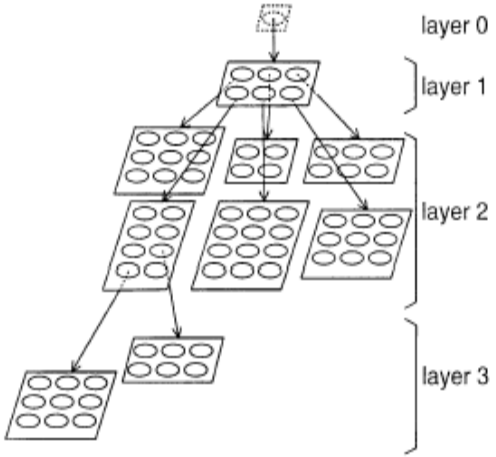


Fig. 1. GHSOM: The GHSOM evolves to a structure of SOMs reflecting the hierarchical structure of the input data.

Algorithm 7 GHSOM($\mathcal{X}$,η,r,θ_1,θ_2)

Input:

i) $\mathcal{X}$, the training set.

ii) η, the learning factor.

iii) r, the neighborhood radius.

iv) θ_1, Fraction of the MQE of the parent neuron, used for hierarchical expansion.

v) θ_2, Fraction of the QE of the virtual neuron, used for inserting rows (or columns).

Method:

1: Create a 2×2 SOM associated to the parent layer. Initialize the weights of its neurons by selecting stimuli from the subset of stimuli represented by the parent.
2: **repeat**
3: Train the map using the SOM algorithm.
4: **if** *lambda* iterations has been performed **then**
5: **if** quality of the SOM is not good **then**
6: Insert new row (or column) between the neuron with largest QE and its most dissimilar neighbor.
7: **end if**
8: **else**
9: Expand neurons possessing a high QE by creating a new 2×2 SOM as in step 2.
10: For training this new SOM, consider only the stimuli represented by its parent neuron.
11: **end if**
12: **until** A terminating criterion is reached

End Algorithm

5.7 Growing Hierarchical Tree SOM (GHTSOM)

The Growing Hierarchical Tree SOM (GHTSOM) [28], is a tree-based SOM where each neuron of the tree is a SOM-layered strategy (similar to the GHSOM [24]). During the training process, the GHTSOM dynamically creates a hierarchy of hyper-tetrahedrons which correspond to a hierarchical adaptation of the model utilized in the GCS [30].

The algorithm also incorporates a growing mechanism that appropriately "splits" neurons, gener-

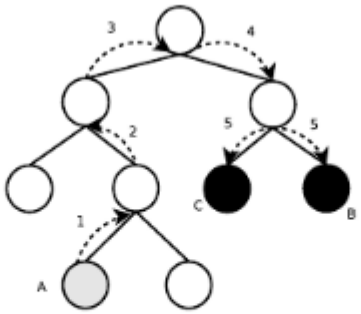


Figure 12: An example of the "neighborhood" for the ET, where neurons B and C are equidistant to A. In the ET, the definition of the distance pertains only to leaf neurons.

ating a new level of descendants. This growing process requires the definition of a "activity" function that returns the number of patterns associated with a particular neuron analogous to the BMU counter (see Section 3.1). The strategy for growing neurons is similar to one used by the ET [57], with $\theta = 3$.

After the construction of the hierarchy, the algorithm incorporates a post-processing module which refines the neural structure. At this juncture, only the BMUs are involved for any further computations, implying that it only utilizes *hard* CL. Additionally, during this phase the method processes and updates the so-called class-links that relate different SOMs and which are further utilized for triggering delete operators that finally establish the final cluster configuration.

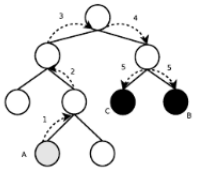


Figure 12: An example of the "neighborhood" for the ET, where neurons B and C are equidistant to A. In the ET, the definition of the distance pertains only to leaf neurons.

Algorithm 9 ET($\mathcal{X}$,η,r,θ,k)

Input:

i) $\mathcal{X}$, the training set.

ii) η, the learning factor.

iii) r, the neighborhood radius.

iv) θ, the split threshold.

v) k, the number of children to be created once a neuron is split.

Method:

1: Initialize the root of the tree by, for example, locating it in the center of mass of $\mathcal{X}$.
2: **repeat**
3: Find the BMU as per Algorithm 2
4: Update all the leaves within a radius r within the BMU using Equation 4
5: Increment the BMU counter of the current BMU according to Equation 5
6: **if** the BMU counter of the current BMU exceeds the threshold θ **then**
7: Split the winner neuron, generating k children and initialize their respective weights to that of the parent neuron.
8: **end if**
9: **until** An end criterion is reached, such as a total number of training epochs

The Tree-Based Topology-Oriented SOM (TTO-SOM)

- Static tree.
- Arbitrary number of children.
- Neighborhood based on tree space.
- Deterministic winner search.
- Winner search based on feature space.
- Infers both the data distribution and its structured topology.
- Generalization of the 1D SOM.

Input and Parameters of the TTO-SOM

- Input
  1. Training sample set.
  2. Configuration of the k-ary tree.
- Parameters
  1. Radius of the Bubble of Activity.
  2. The SOM learning rate.

The Radius

- The Size of the Bubble of Activity.
- Between 0 and the number of neurons.
- Its value should decrease as time proceeds.

The Learning Rate

- Neurons should be moved toward the input signal.
- Learning Rate: The Factor of such a movement.
- Should decrease as convergence take place.

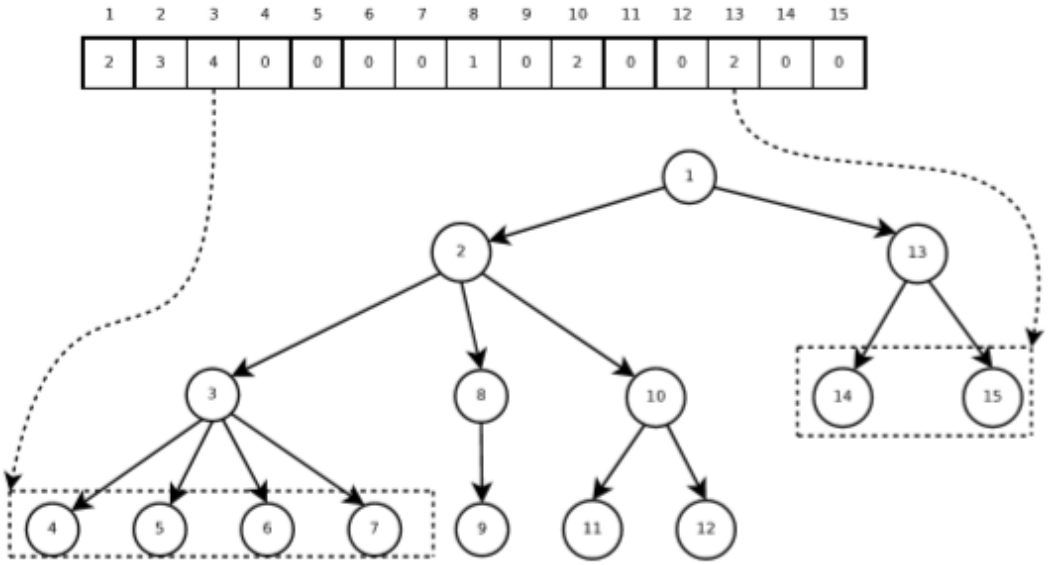
Declaration of the User-Defined Tree

- The user describe the tree.
- Arbitrary number of children.
- Reflects *a priori* knowledge about the data distribution.
- Recursive definition.

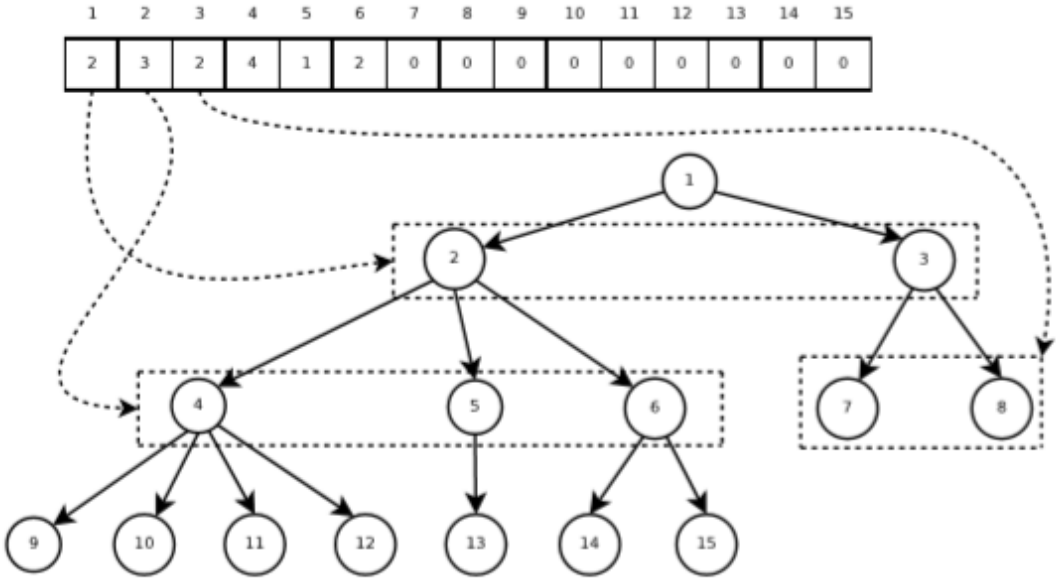
Declaration of the User-Defined Tree

- Array specifying the number of children for each node.
- Depth-First/Breadth-First traversal.
- Index of the array: Node in the respective traversal.
- Content of the array: Number of children of each node.

Example 1: Depth-First traversal



Example 2: Breadth-First traversal



Neural Distance Between Two Neurons...

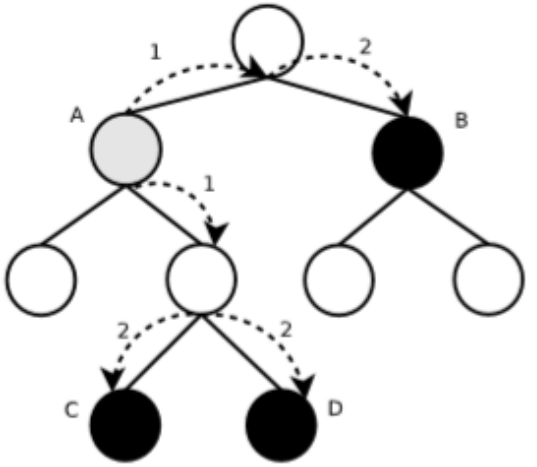
Definition

No. of edges in the shortest path connecting two given nodes.

- Depends on connections.
- Minimum Path.
- Includes non-leaf nodes.

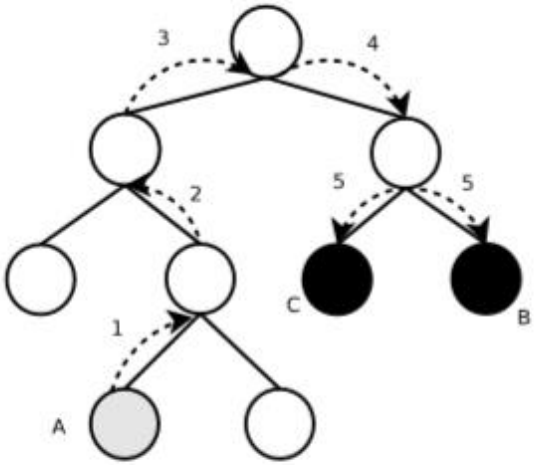
Neural Distance: First example

- *B*, *C* and *D* are equidistant to *A*.
- *B* and *D* are at different levels.
- *B* is a non-leaf node.



Neural Distance: Another example

- *B* and *C* are at distance 5 to *A*.
- Distance from *A* to *A* is zero.



The Bubble of Activity.

Definition

Subset of nodes: Distance r away from the node examined.

- Intricately related to the notion of neural distance.
- Subset of nodes "close" to a given neuron.
- The Radius determines the size of the bubble.

The Bubble of Activity.

Formal Definition

$$B(v_i; T, r) = \{v | d_N(v_i, v; T) \leq r\}$$

Where,

- v_i The node currently being examined.
- v An arbitrary node in the tree T.
- d_N The neural distance.
- r The radius.

The Bubble of Activity.

Formal Definition

$$B(v_i; T, r) = \{v | d_N(v_i, v; T) \leq r\}$$

The Bubble of Activity also present the following properties:

1. $B(v_i, T, 0) = \{v_i\}$
2. $B(v_i, T, i) \supseteq B(v_i, T, i-1)$
3. $B(v_i, T, |V|) = V$

TTOSOM properties

- Static tree.
- Distance in the Feature space for winner search.
- Distance in the Tree space for bubble of activity.
- Deterministic winner search.

TTOSOM Training Algorithm

1. Describe Topology
   - 1.1 Read next item in the array of children.
   - 1.2 Create children.
   - 1.3 Call Descibe Topology recursively.
2. Main Loop (Repeat until Convergence)
   - 2.1 Receive input.
   - 2.2 Find BMU.
   - 2.3 Calculate Neighbors using neural distance.
   - 2.4 Move Neurons within the bubble of activity.
   - 2.5 Decrease radius/learning rate.

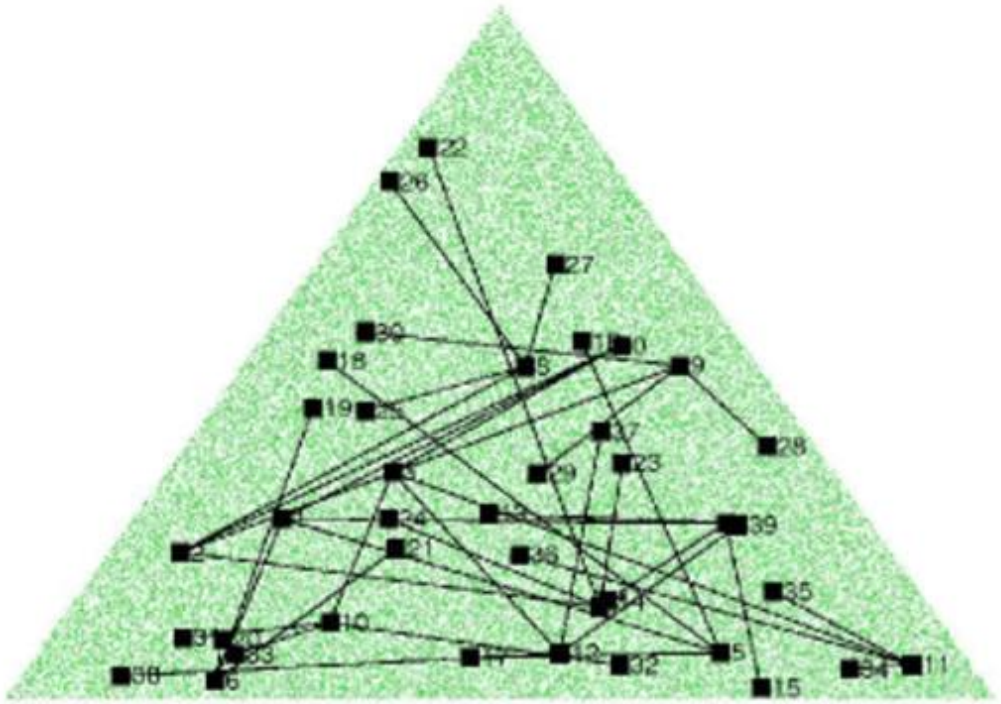
TTOSOM Mapping Algorithm

1. Receive input.
2. Find BMU (in feature space).
3. Return BMU.

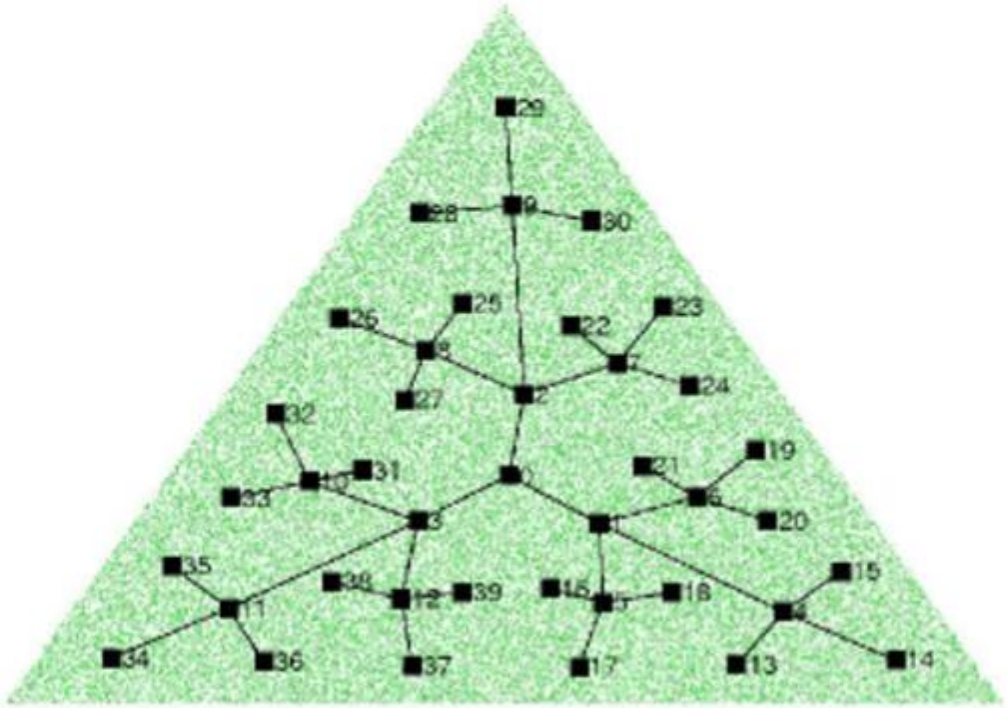
Experiment 1: Triangular-Spaced Distribution

- Complete tree of 4 levels.
- Triangular shape.
- Capture essence of distribution.
- Define 3 children per node.

Experiment 1: Triangular-Spaced Distribution

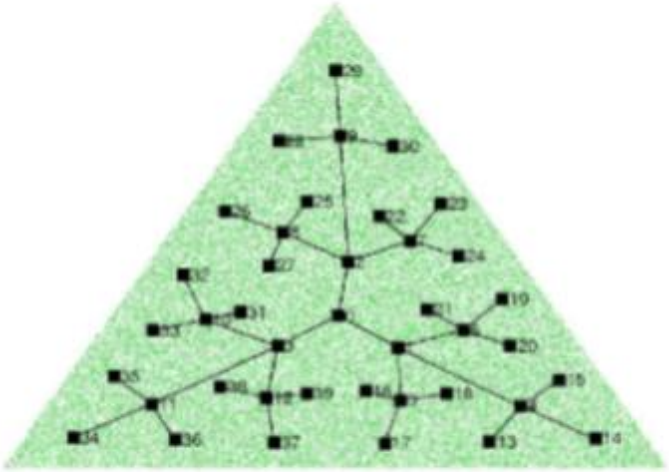


Experiment 1: Triangular-Spaced Distribution



Experiment 1: Triangular-Spaced Distribution

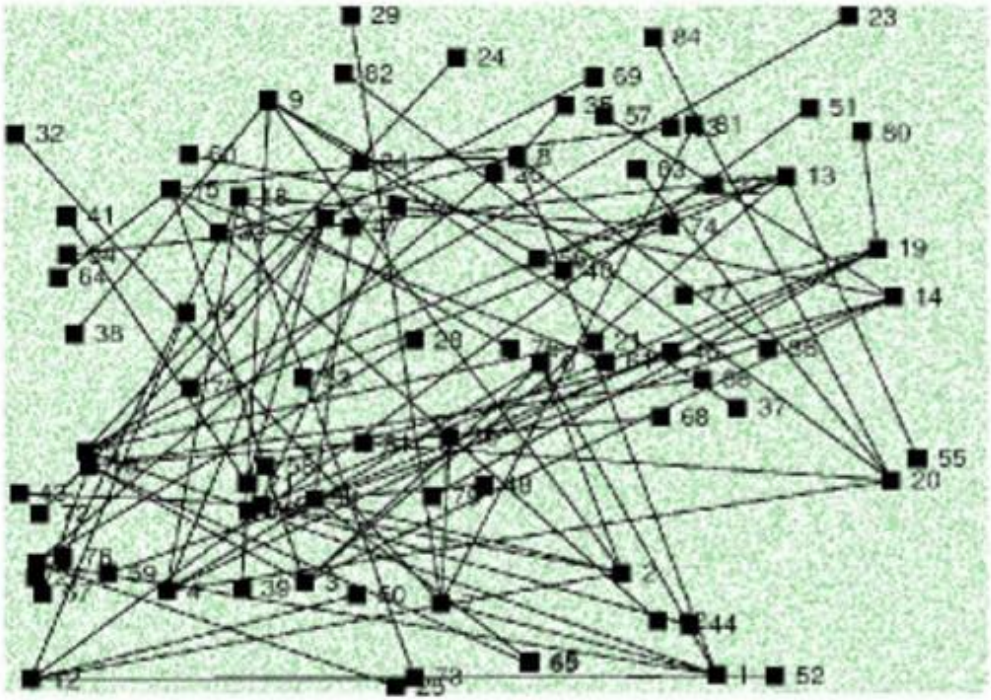
- The root roughly in the center.
- Each main branch towards a vertex.
- Sub-branches uniformly fill space around main branches.



Experiment 2: “Square-Shaped” Distribution

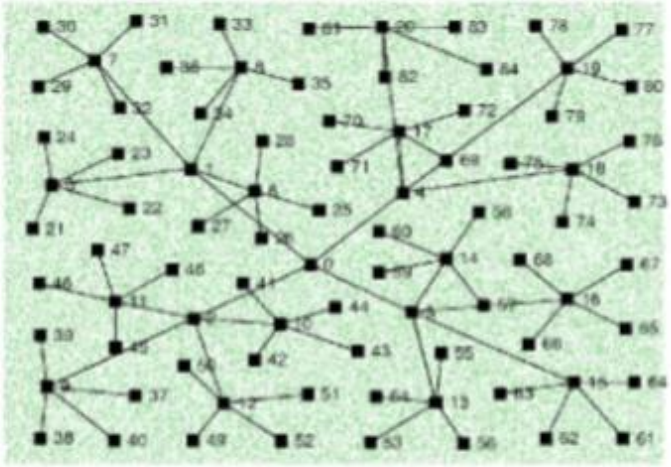
- Complete tree of depth 4.
- Rectangular shape.
- Capture essence of distribution.
- Define 4 children per node.

Experiment 2: “Square-Shaped” Distribution



Experiment 2: “Square-Shaped” Distribution

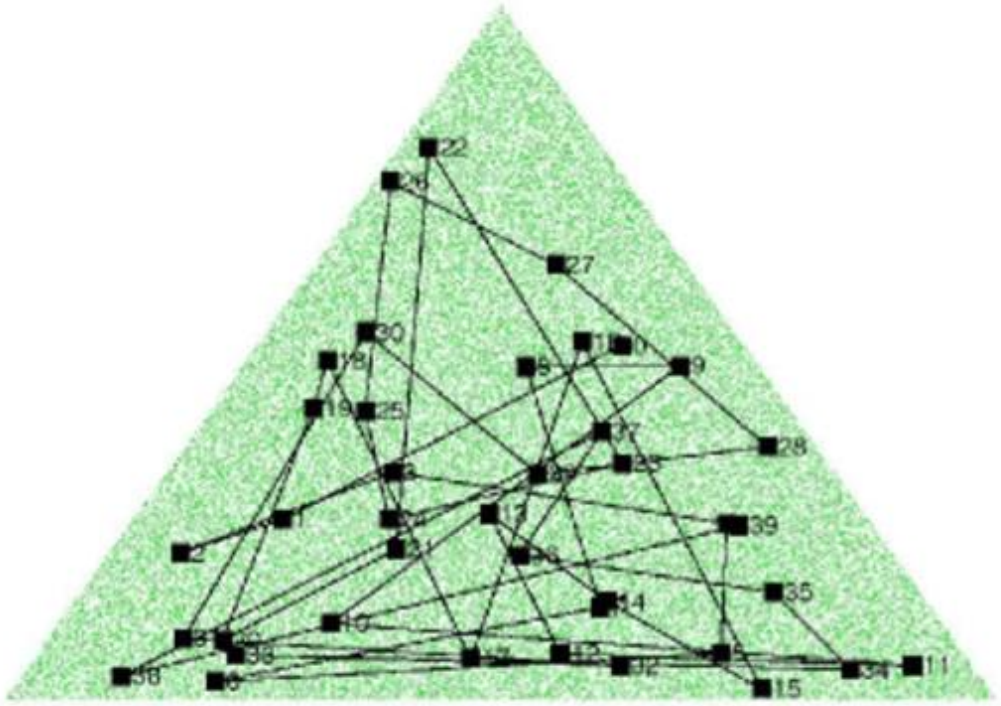
- Root near the center of mass.
- Main branches cover the principal diagonals.
- Sub-branches uniformly fill space around main branches.



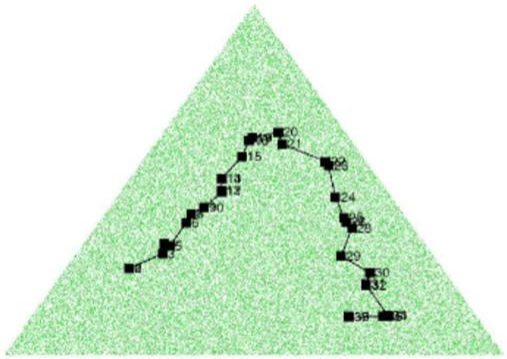
Experiment 3: 1-ary tree

- Triangular shape.
- A list as the imposed topology.
- 1 children per node.

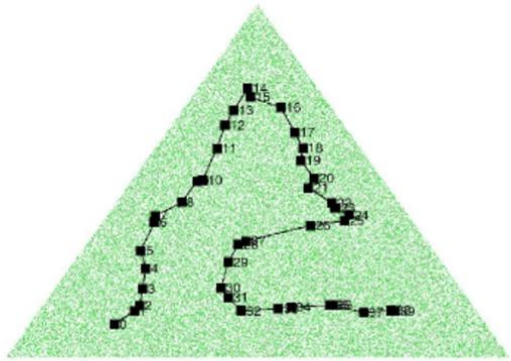
Experiment 3: 1-ary tree



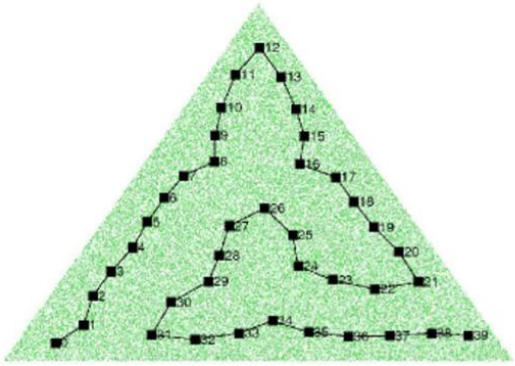
Experiment 3: 1-ary tree



Experiment 3: 1-ary tree

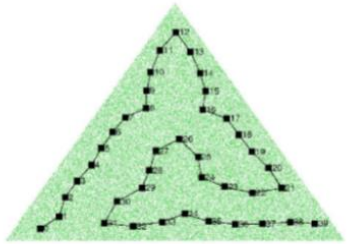


Experiment 3: 1-ary tree



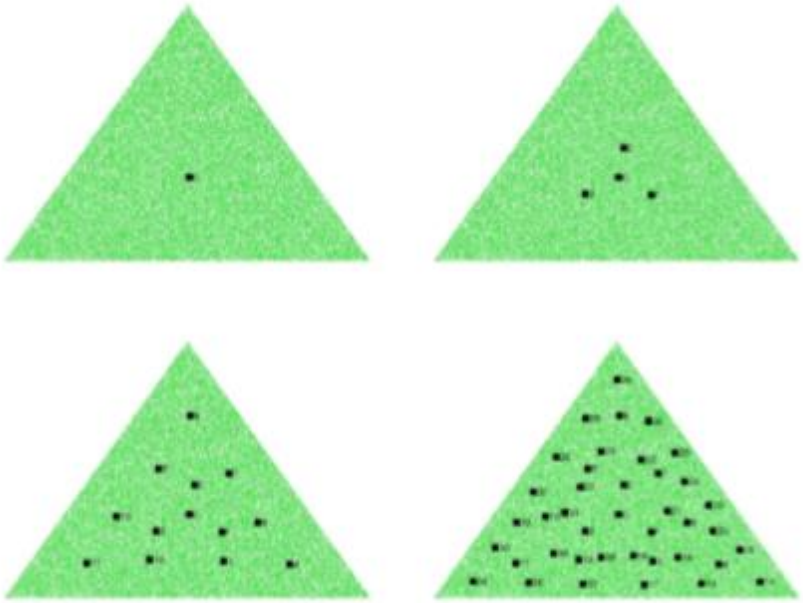
Experiment 3: 1-ary tree

- The list represents the triangle effectively.
- Preserves “tree-like” topology.
- Generalization of 1D SOM.



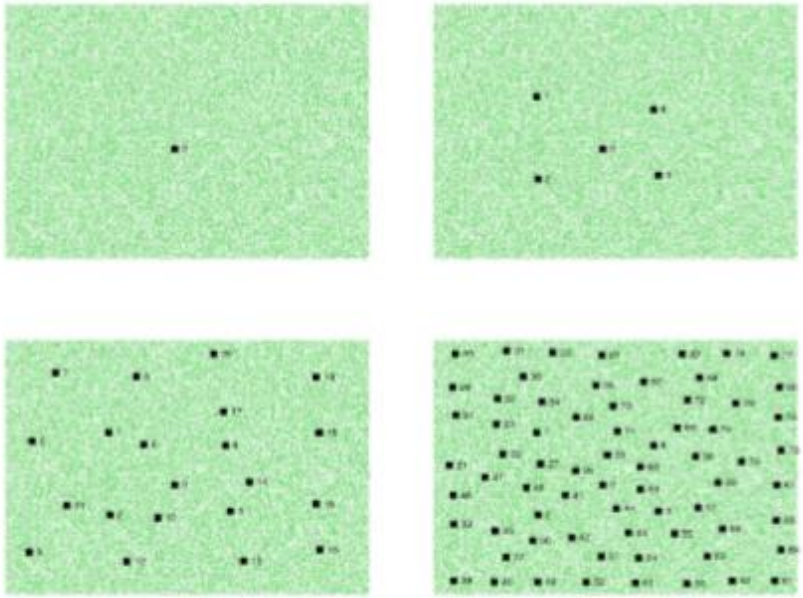
Experiment 4: Multi-Resolution

- Hologram-like properties
- Not possessed by other SOM-based networks.
- Still represent the distribution and the structure.



Experiment 5: Multi-Resolution

- Few level of the tree $\rightarrow$ lower resolution.
- More levels of the tree $\rightarrow$ finer resolution.



Extracting the Skeleton of a Data Set

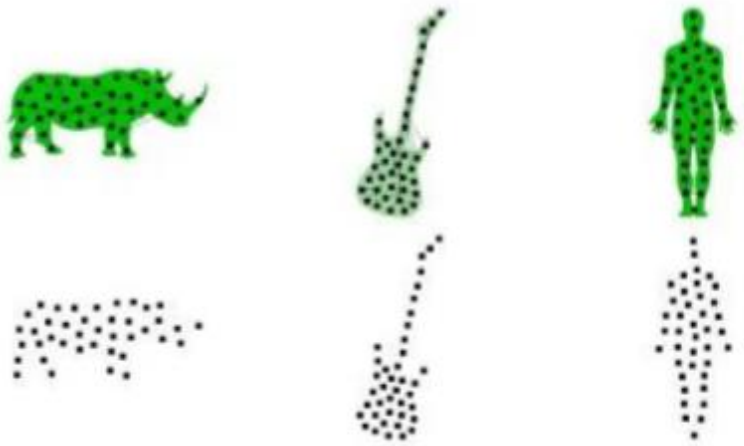
Definition: Skeletonization

Process by which a 2D shape is transformed into 1D.

- Dimensionality reduction technique.
- Traditional skeletonization: Assumes pixel connectivity.
- SOM variations have been used to tackle this situation.
  - GNG-like approach.
  - MST calculated over neurons.

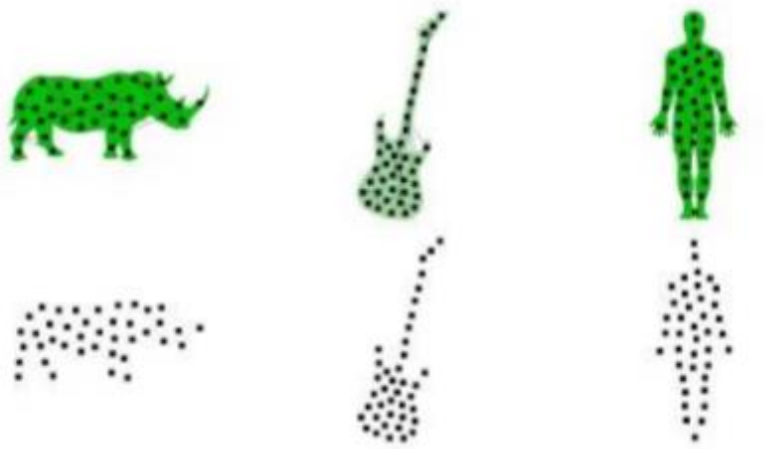
Experiment 6: Skeletonization

- 3 objects processed using the same tree structure.
- Same schedule for parameters.
- No post processing of edges.



Experiment 6: Skeletonization

- Effectively represents 2D objects in 1D.
- Without any specific adaption.

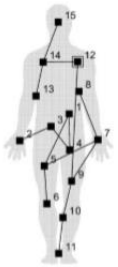


Experiment 8: Human Shape – TTOSOM-CONROT

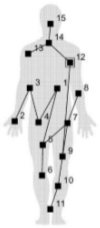
- Human Shape.
- Originaly a list topology .
- Most accessed nodes are moved to the root conditionally.

Experiment 8: Human Shape – TTOSOM-CONROT

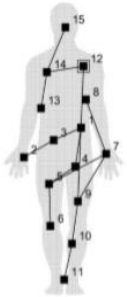
Experiment 8: Human Shape – TTOSOM-CONROT



Experiment 8: Human Shape – TTOSOM-CONROT



Experiment 8: Human Shape – TTOSOM-CONROT



5.9 TTOSOM with Conditional Rotations (TTOCONROT)

In [3], the authors merged the fields of SOMs and Adaptive Data Structure (ADS). The adaptive nature of the strategy presented, namely the TTOSOM with Conditional Rotations (TTOCONROT), is unique because adaptation is perceived in two forms: The migration of the weight vectors in the feature space is a consequence of the SOM update rule, and the rearrangement of the neurons *within* the tree is a result of the rotations. This reorganization can be perceived to be both automatic and adaptive, such that on convergence, the Data Structure (DS) tends towards an optimal configuration with a minimum average access time. In most cases, the most probable element will be positioned at the root (head) of the tree (DS), while the rest of the tree is recursively positioned in the same manner.

The TTOCONROT is a further enhancement of the TTOSOM [4] which considers how the underlying tree itself can be rendered dynamic and adaptively transformed. To do this, the authors of [3] present methods by which a SOM with an underlying Binary Search Tree (BST) structure can be adaptively re-structured using Conditional Rotations [18]. These rotations on the neurons of the tree are local, can be done in constant time, and performed so as to decrease the Weighted Path Length (WPL) of the entire tree. In [3], the authors also introduced the concept referred to as *Neural Promotion*, where neurons gain prominence in the NN as their significance increases. The advantages of such a scheme is that the leaned tree learns the topological peculiarities of the stochastic data distribution and at the same time recursively positions neurons accessed more often close to the root. As a result, the TTOCONROT converges in such a manner that the neurons are ultimately placed in the input space so as to represent its stochastic distribution, and additionally, the neighborhood properties of the neurons suit the best BST that represents the data.

Summary

- TTO-SOM is able to determine:
  - Distribution of the data.
  - Structured topology.
- TTO-SOM can represent data in multiple granularity levels.
- TTO-SOM can extract a skeleton from the shape.

- TTOSOM-CONROT is able to determine:
  - Distribution of the data.
  - Adaptive Structure of topology.
- TTOSOM-CONROT merges Neural Nets and Adaptive DS.
- TTOSOM-CONROT has huge potential; Many open areas.

Personal Opinion

- This method can be used in information retrieval that have hierarchic architecture and can be gained good performance.

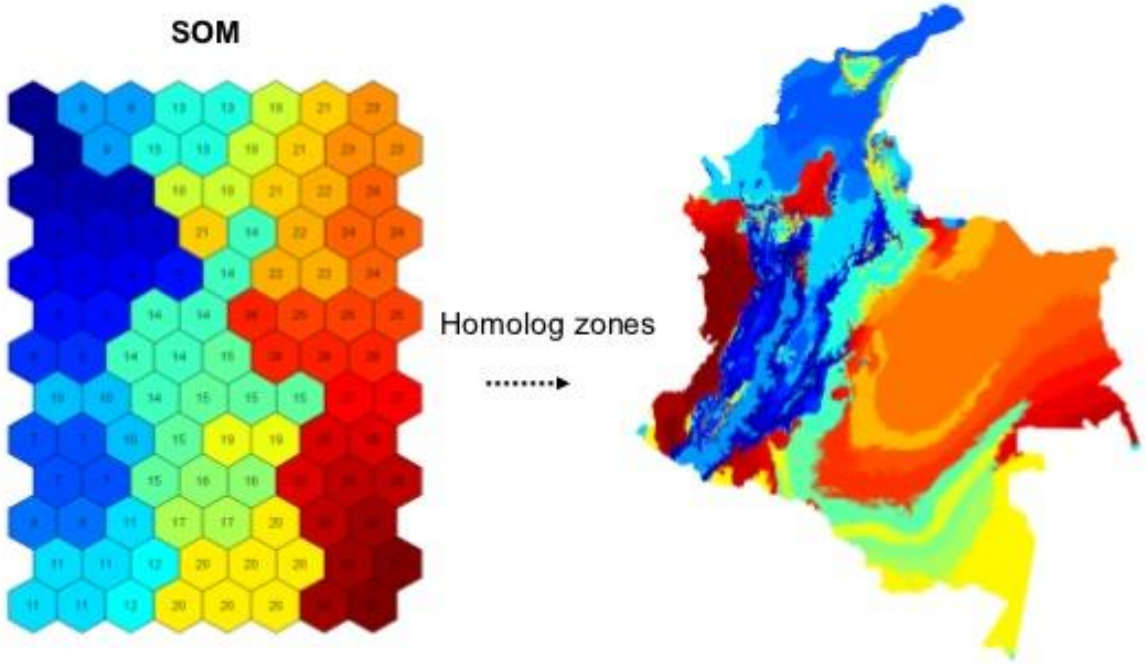
Review

- The limitation of the standard SOM.
- Training and Growth Process of a Growing SOM.
- Hierarchical Growth With Global Map Orientation.
- Two experiments of the information retrieval application.

Introduction

First solution: Self-Organizing Maps

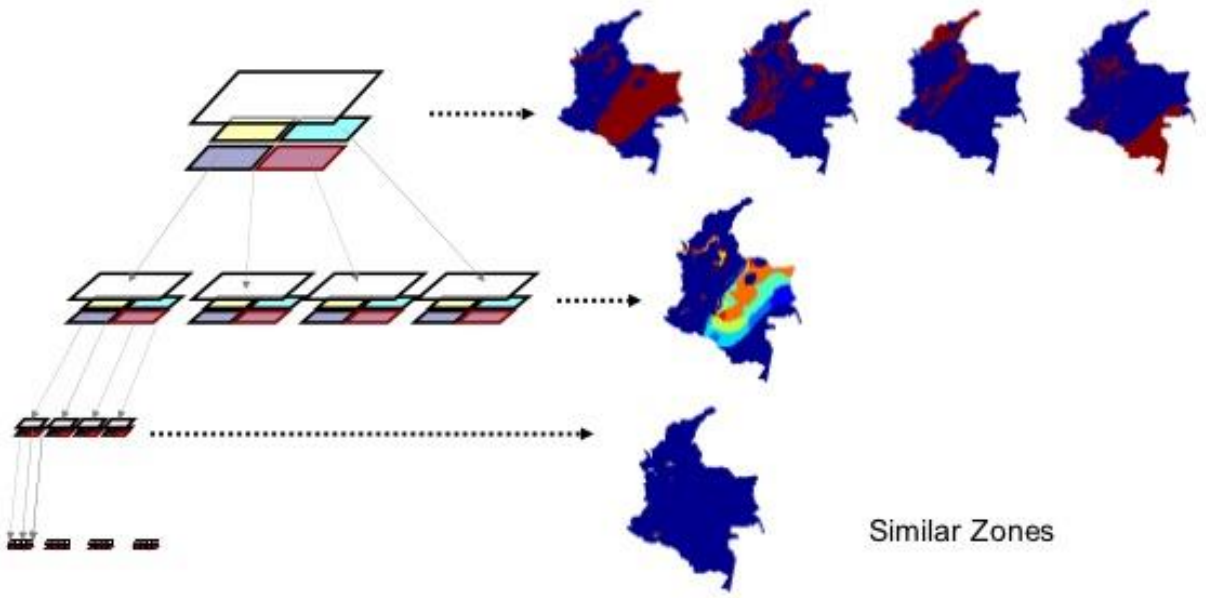
Advantages : It is possible to obtain prototypes
Disadvantages : It is not possible to obtain different resolutions (fix size of the Kohonen map)



Introduction

Second solution: Growing hierarchical SOM (GHSOM)

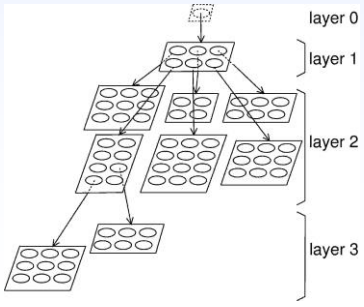
Advantages : It is possible to obtain different resolutions
Disadvantages : It is not possible to represent fuzzy relationships



In spite of the stability and popularity of the *Self-Organizing Map (SOM)*, at least two limitations have to be noted, which are related, on the one hand, to the

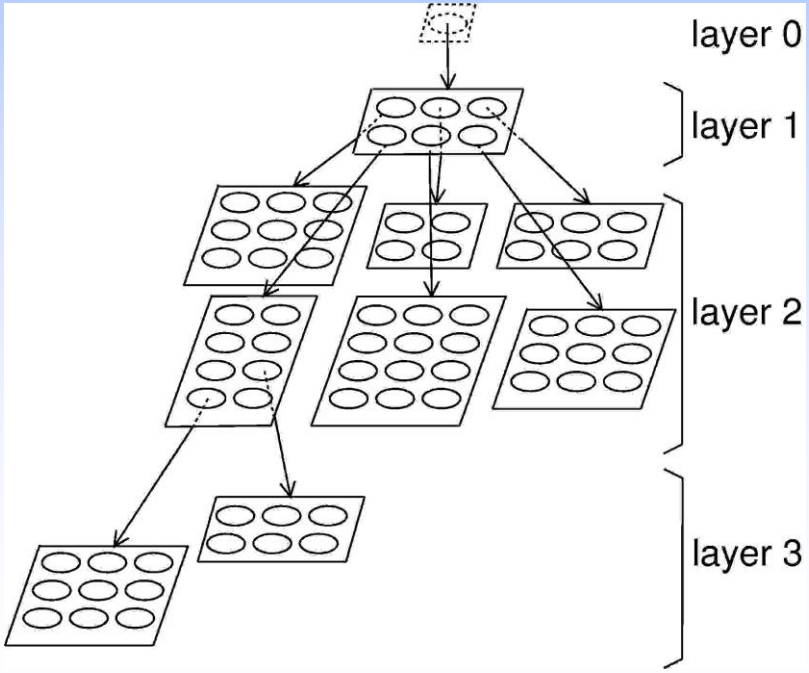
- static architecture of this model, as well as, on the other hand, to the
- limited capabilities for the representation of hierarchical relations of the data.

- **One of the shortcommings of the SOM lies with its fixed architecture that has to be defined a-priori.**
- Dynamically growing variants of the SOM, on the other hand, tend to produce huge maps that are hard to handle.
- This lead to the development of the GHSOM , a new architecture which grows both in a hierarchical way according to the data distribution, allowing a hierarchical decomposition and navigation in sub-parts of the data, and in a horizontal way, meaning that the size of each individual map adapts itself to the requirements of the input space.
- This provides a convenient interface for navigating large digital libraries, as it closely follows the model of conventional libraries, which are also structured by, e.g. different floors, sections and shelves.



•With our novel ***Growing Hierarchical Self-Organizing Map (GHSOM)*** we address both limitations.

• The growing hierarchical som is an artificial neural network model with hierarchical architecture composed of independent growing self-organizing maps.
• By providing a global orientation of the independently growing maps in the individual layers of the hierarchy, navigation across branches is facilitated



The starting point for the growth process is the overall deviation of the input data as measured with the single-unit *SOM* at layer 0. This unit is assigned a weight vector m_0, $m_0 = [\mu_{0_1}, \mu_{0_2}, \ldots, \mu_{0_n}]^T$, computed as the average of all input data. The deviation of the input data, i.e. the *mean quantization error* of this single unit, is computed as given in Expression (1) with d representing the number of input data x. We will refer to the *mean quantization error* of a unit as **mqe** in lower case letters.

$$\mathbf{mqe}_0 = \frac{1}{d} \cdot ||m_0 - x|| \tag{1}$$

After the computation of $\mathbf{mqe}_0$, training of the *GHSOM* starts with its first layer *SOM*. This first layer map initially consists of a rather small number of units, e.g. a grid of 2×2 units. Each of these units i is assigned an n-dimensional weight vector m_i, $m_i = [\mu_{i_1}, \mu_{i_2}, \ldots, \mu_{i_n}]^T$, $m_i \in \Re^n$, which is initialized with random values. It is important to note that the weight vectors have the same dimensionality as the input patterns.

The learning process of *SOMs* may be described as a competition among the units to represent the input patterns. The unit with the weight vector being closest to the presented input pattern in terms of the input space wins the competition. The weight vector of the winner as well as units in the vicinity of the winner are adapted in such a way as to resemble more closely the input pattern.

The degree of adaptation is guided by means of a learning-rate parameter α, decreasing in time. The number of units that are subject to adaptation also decreases in time such that at the beginning of the learning process a large number of units around the winner is adapted, whereas towards the end only the winner is adapted. These units are chosen by means of a neighborhood function h_{ci} which is based on the units' distances to the winner as measured in the two-dimensional grid formed by the neural network. In combining these principles of *SOM* training, we may write the learning rule as given in Expression (2), where x represents the current input pattern, and c refers to the winner at iteration t.

$$m_i(t+1) = m_i(t) + \alpha(t) \cdot h_{ci}(t) \cdot [x(t) - m_i(t)] \tag{2}$$

In order to adapt the size of this first layer *SOM*, the *mean quantization error* of the map is computed ever after a fixed number λ of training iterations as given in Expression (3). In this formula, u refers to the number of units i contained in the *SOM* m. In analogy to Expression (1), $\mathbf{mqe}_i$ is computed as the average distance between weight vector m_i and the input patterns mapped onto unit i. We will refer to the *mean quantization error* of a map as **MQE** in upper case letters.

$$\mathbf{MQE}_m = \frac{1}{u} \cdot \sum_i \mathbf{mqe}_i \tag{3}$$

The basic idea is that each layer of the *GHSOM* is responsible for explaining some portion of the deviation of the input data as present in its preceding layer. This is done by adding units to the *SOMs* on each layer until a suitable size of the map is reached. More precisely, the *SOMs* on each layer are allowed to grow until the deviation present in the unit of its preceding layer is reduced to at least a fixed percentage τ_m.

Obviously, the smaller the parameter τ_m is chosen the larger will be the size of the emerging *SOM*. Thus, as long as $\mathbf{MQE}_m \geq \tau_m \cdot \mathbf{mqe}_0$ holds true for the first layer map m, either a new row or a new column of units is added to this *SOM*. This insertion is performed neighboring the unit e with the highest *mean quantization error*, $\mathbf{mqe}_e$, after λ training iterations. We will refer to this unit as the *error unit*. The distinction whether a new row or a new column is inserted is guided by the location of the most dissimilar neighboring unit to the *error unit*. Similarity is measured in the input space. Hence, we insert a new row or a new column depending on the position of the neighbor with the most dissimilar weight vector. The initialization of the weight vectors of the new units is simply performed as the average of the weight vectors of the existing neighbors. After the insertion, the learning-rate parameter α and the neighborhood function h_{ci} are reset to their initial values and training continues according to the standard training process of *SOMs*. Note that we currently use the same value of the parameter τ_m for each map in each layer of the *GHSOM*. It might be subject to further research, however, to search for alternative strategies, where layer or even map-dependent quantization error reduction parameters are utilized.

Consider the following figure for a graphical representation of the insertion of units. In this figure the architecture of the *SOM* prior to insertion is shown on the left-hand side where we find a map of 2×3 units with the *error unit* labeled by *e* and its most dissimilar neighbor signified by *d*. Since the most dissimilar neighbor belongs to another row within the grid, a new row is inserted between units *e* and *d*. The resulting architecture is shown on the right-hand side of the figure as a map of now 3×3 units.

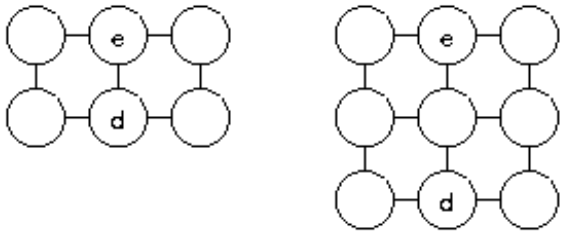


Figure 1: Insertion of units to a self-organizing map

As soon as the growth process of the first layer map is finished, i.e. $\mathbf{MQE}_m < \tau_m \cdot \mathbf{mqe}_0$, the units of this map are examined for expansion on the second layer. In particular, those units that have a large *mean quantization error* will add a new *SOM* to the second layer of the *GHSOM*. The selection of these units is based on the *mean quantization error* of layer 0. A parameter τ_u is used to describe the desired level of granularity in input data discrimination in the final maps. More precisely, each unit i fulfilling the criterion given in Expression (4) will be subject to hierarchical expansion.

$$\mathbf{mqe}_i > \tau_u \cdot \mathbf{mqe}_0 \tag{4}$$

The training process and unit insertion procedure now continues with these newly established *SOMs*. The major difference to the training process of the second layer map is that now only that fraction of the input data is selected for training which is represented by the corresponding first layer unit. The strategy for row or column insertion as well as the termination criterion is essentially the same as used for the first layer map. The same procedure is applied for any subsequent layers of the *GHSOM*.

The training process of the *GHSOM* is terminated when no more units require further expansion. Note that this training process does not necessarily lead to a balanced hierarchy, i.e. a hierarchy with equal depth in each branch. Rather, the specific requirements of the input data is mirrored in that clusters might exist that are more structured than others and thus need deeper branching. Consider Figure 2 for a graphical representation of a trained *GHSOM*. In particular, the neural network depicted in this figure consists of a single-unit *SOM* at layer 0, a *SOM* of 2×2 units in layer 1, three *SOMs* in layer 2, i.e. one for each unit in the layer 1 map. Note that each of these maps might have a different number and different arrangements of units as shown in the figure. Finally, we have several *SOMs* in layer 3 which were expanded from one of the layer 2 units, just indicated by dotted arrows.

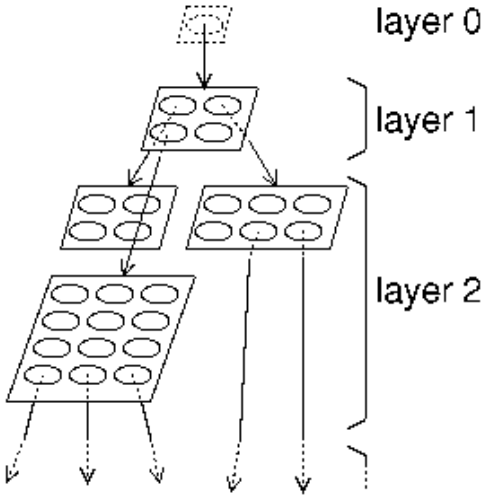


Figure 2: Architecture of a trained *GHSOM*

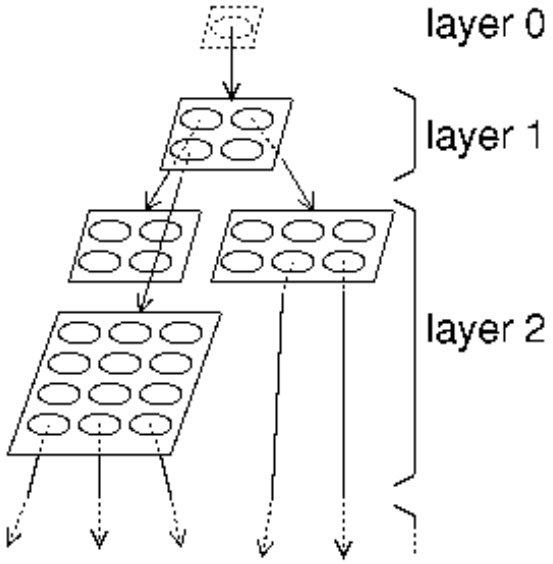


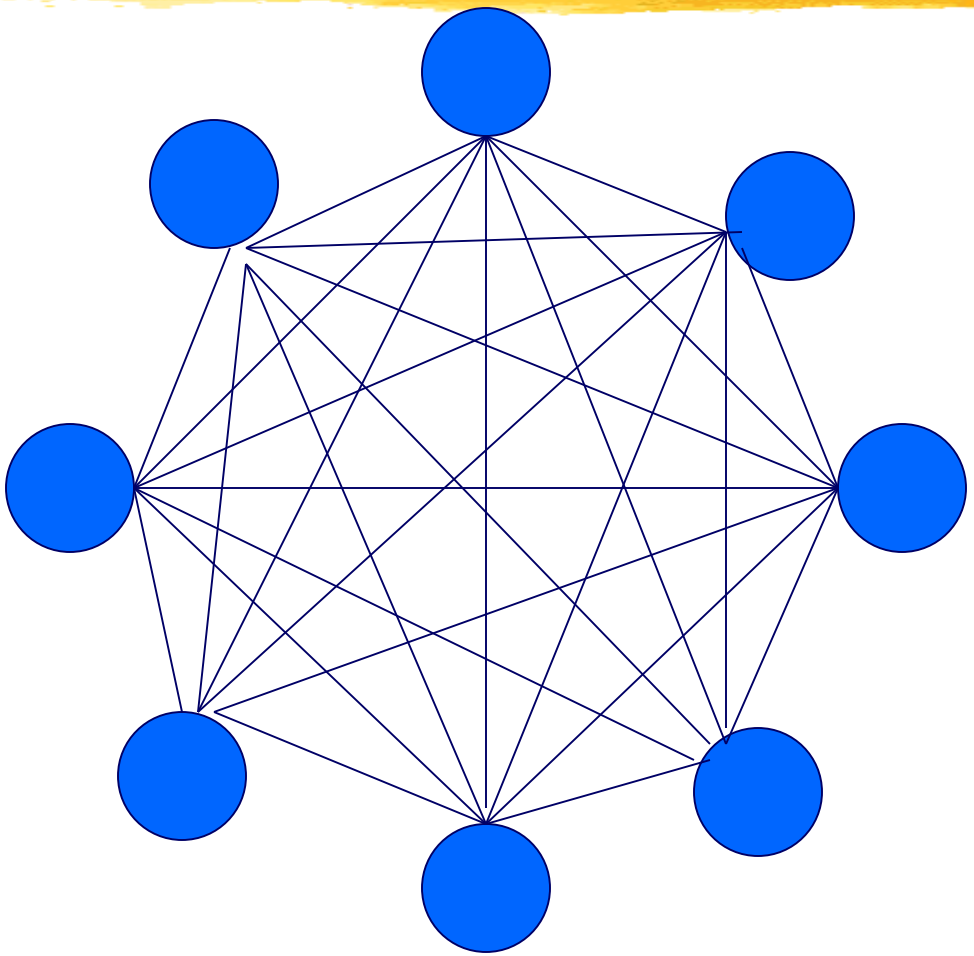
Figure 2: Architecture of a trained *GHSOM*

To summarize, the growth process of the *GHSOM* is guided by two parameters τ_u and τ_m. The parameter τ_u specifies the desired quality of input data representation at the end of the training process. Each unit i with $\mathbf{mqe}_i > \tau_u \cdot \mathbf{mqe}_0$ will be expanded, i.e. a map is added to the next layer of the hierarchy, in order to explain the input data in more detail. Contrary to that, the parameter τ_m specifies the desired level of detail that is to be shown in a particular *SOM*. In other words, new units are added to a *SOM* until the **MQE** of the map is a certain fraction, τ_m, of the **mqe** of its preceding unit. Hence, the smaller τ_m the larger will be the emerging maps. Conversely, the larger τ_m the deeper will be the hierarchy.

		Reference		Topology					Dynamism				BoA		Distance		Counters			Neuron State		Search	
		Citation	Section	(Unconnected) Graph	Tree (acyclic graph)	Grid	Irregular Grid	SOM-layers	Static	ADS	Add connections	Delete connections	Add neurons	Delete neurons	Feature space	Neural space	BMU	QE	Age of connection	Frozen	Boundary	Heuristic	Deterministic
Non Tree-Based	SOM	[46]	2			✓			✓			✓											
	GCS	[30]	4.1	✓	✓	✓	✓	✓	✓	✓													
	TRN	[51]	4.2	✓	✓	✓	✓	✓															
	GNG	[32]	4.3	✓	✓	✓	✓	✓	✓	✓													
	GG	[31]	4.4	✓	✓	✓	✓	✓															
	IGG	[14]	4.5	✓	✓	✓	✓	✓	✓	✓	✓												
	GSOM	[1]	4.6		✓		✓		✓		✓		✓		✓								
Tree-Based	TSVQ	[47]	5.1		✓		✓		✓		✓		✓										
	TSSOM	[47]	5.1	✓	✓	✓	✓	✓	✓														
	HFM	[54]	5.2	✓	✓	✓	✓																
	GHSOM	[61]	5.3		✓		✓		✓		✓		✓										
	SOTM	[36]	5.4		✓		✓		✓		✓												
	SOTA	[25]	5.5		✓		✓		✓		✓												
	ET	[57]	5.6	✓	✓	✓	✓	✓	✓														
	GHTSOM	[28]	5.7	✓	✓	✓	✓	✓	✓	✓													
	TTOSOM	[4]	5.8	✓				✓				✓				✓							
	TTOCONROT	[3]	5.9	✓	✓	✓	✓	✓															

Table 1: A comparison between different SOM strategies based on the criteria specified in Section 6.

Hopfield Network



Hopfield Network

- The state of the system is given by the activation values $y = (y_k)$.
- The net input $s_k(t+1)$ of a neuron k at cycle $(t+1)$ is a weighted sum

$$s(t+1) = \sum_{j \neq k} y_j(t) w_{jk} + b_k$$

Hopfield Network

- A threshold function is applied to obtain the output

$$y_k(t+1) = \mathrm{sgn}\left(s_k(t+1)\right)$$

Hopfield Network

- A neuron k in the net is stable at time t
  I.e.

$$y_k(t) = \mathrm{sgn}\left(s_k(t-1)\right)$$

- A state is state if all the neurons are stable

Hopfield Networks

- If $w_{jk} = w_{kj}$ the behavior of the system can be described with an energy function

$$\varepsilon = -\frac{1}{2}\sum_{j \neq k}\sum y_j\, y_k w_{jk} - \sum_k b_k y_k$$

- This kind of network has stable limit points

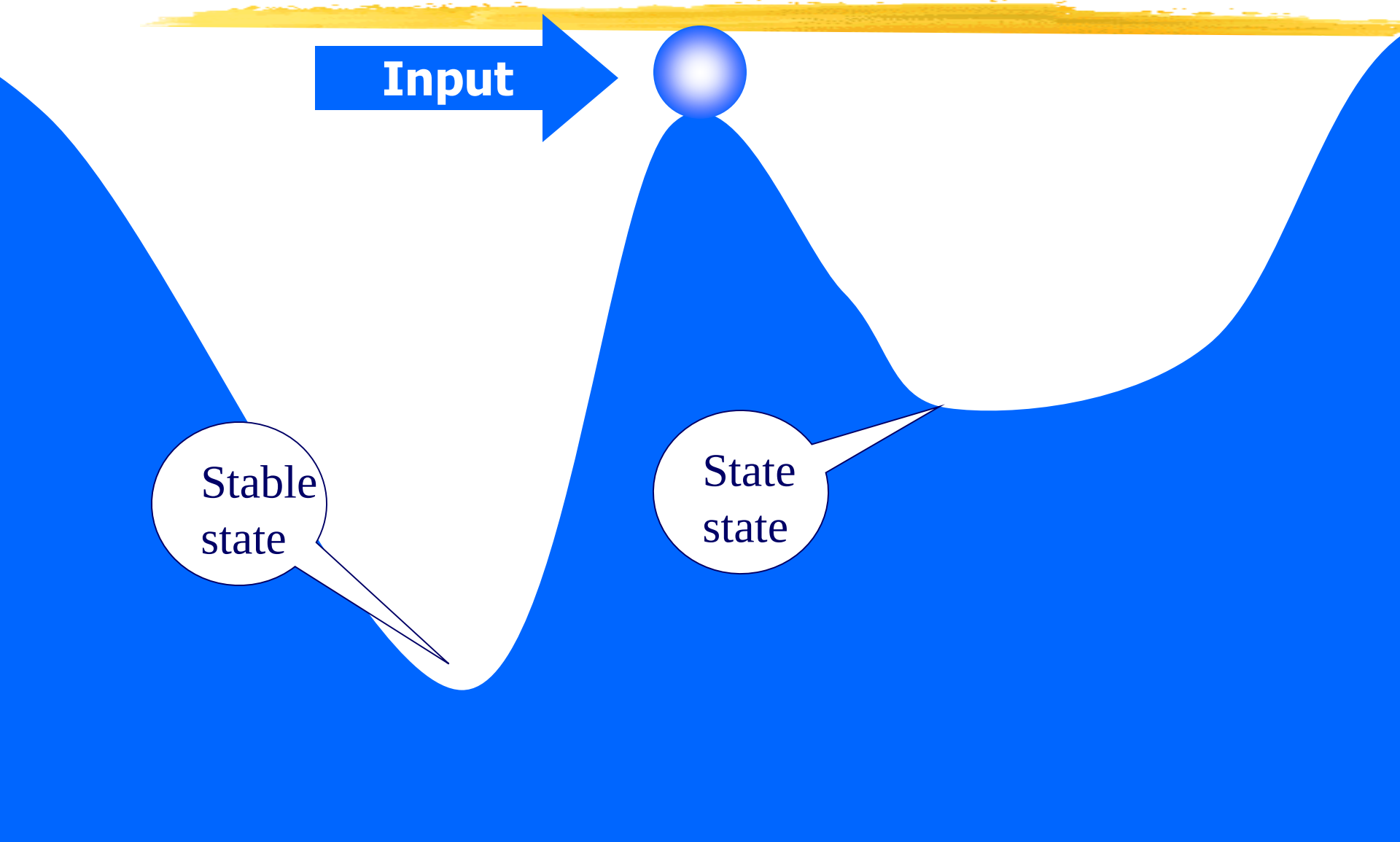
Hopfield net. applications

- A primary application of the Hopfield network is an associative memory.
- The states of the system corresponding with the patterns which are to be stored in the network are stable.
- These states can be seen as `dips' in energy space.

Hopfield Networks

- It appears, however, that the network gets saturated very quickly, and that about 0.15N memories can be stored before recall errors become severe.

Hopfield Networks



Hopfield Networks

- Used in
  - Chung, Y.-M., Pottenger, W. M., and Schatz, B. R. (1998) Automatic subject indexing using an associative neural network. In: I. Witten, R. Akscyn and F. M. Shipman III (eds.) Proceedings of The Third ACM Conference on Digital Libraries (Digital Libraries '98), Pittsburgh, USA, June 23-26, 1998, ACM Press, pp. 59-6

Self Organization

- The unsupervised weight adapting algorithms are usually based on some form of global competition between the neurons.
- Applications of self-organizing networks are:

S.O. Applications

- **clustering:** the input data may be grouped in `clusters' and the data processing system has to find these inherent clusters in the input data.

S.O. Applications

- **vector quantisation**: this problem occurs when a continuous space has to be discretised. The input of the system is the n-dimensional vector x, the output is a discrete representation of the input space. The system has to find optimal discretisation of the input space.

S.O. Applications

- **dimensionality reduction**: the input data are grouped in a subspace which has lower dimensionality than the dimensionality of the data. The system has to learn an “optimal” mapping.

S.O. Applications

- **feature extraction:** the system has to extract features from the input signal. This often means a dimensionality reduction as described above.