Growing Hierarchical Self-Organizing Maps for Web Mining

Joseph P. Herbert and JingTao Yao

Department of Computer Science,
University or Regina
CANADA S4S 0A2

herbertj@cs.uregina.ca jtyao@cs.uregina.ca
http://www2.cs.uregina.ca/~herbertj http://www2.cs.uregina.ca/~jtyao

Introduction

- Many information retrieval and machine learning techniques have not evolved to survive the Web environment.

- There are two major problems in applying some machine learning techniques for Web Mining:
    1. The dynamic and ever-changing nature of Web data.
    2. Dimensionality and sheer size of Web data.

Introduction

- Three domains of application: Web content mining, Web usage mining, and Web structure mining.

- Self-Organizing Maps (SOM) have been used for:
  - Web page clustering
  - Document retrieval
  - Recommendation systems

Growing Hierarchical SOMs

- Growing Hierarchical SOMs are a hybridization of Growing SOMs and Hierarchical SOMs:
  - Growing SOMs have a dynamic topology of neurons to help solve the dynamic nature of data on the Web.
  - Hierarchical SOMs are multi-level systems designed to minimize the high dimensionality problem of data.

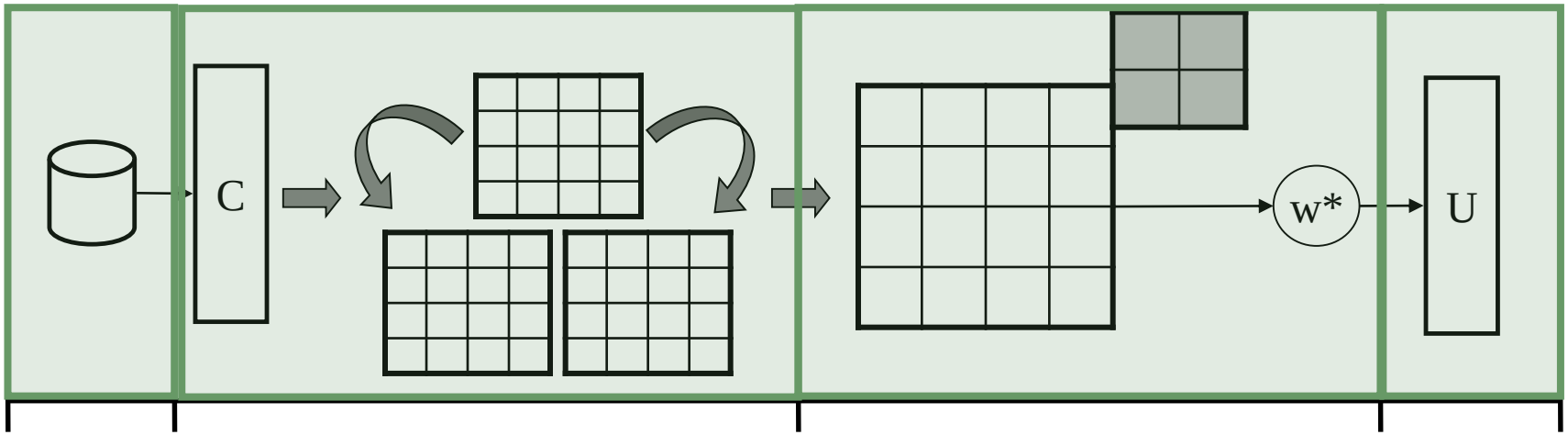
- Together, the hybrid system provides a logical solution when considering the combined problem of dynamic, high-dimensional data sources.

The Consistency Problem

- The growing hierarchical SOM model suffers from a new problem:
  - Maintaining consistency of hierarchical relationships between levels.
  - Training is done locally, without consideration of how changes effect other SOMs that have connection to the local focus.

- The Web Mining model for Self-Organizing Maps solve this problem through **bidirectional update propagation**.

The Web Mining Model for Self-Organizing Maps



Update Layer:

- This layer updates the winning neuron and the neighborhood associated with it.
- Bidirectional Update Propagation updates parent neurons and children feature maps that are associated with the winning neuron.

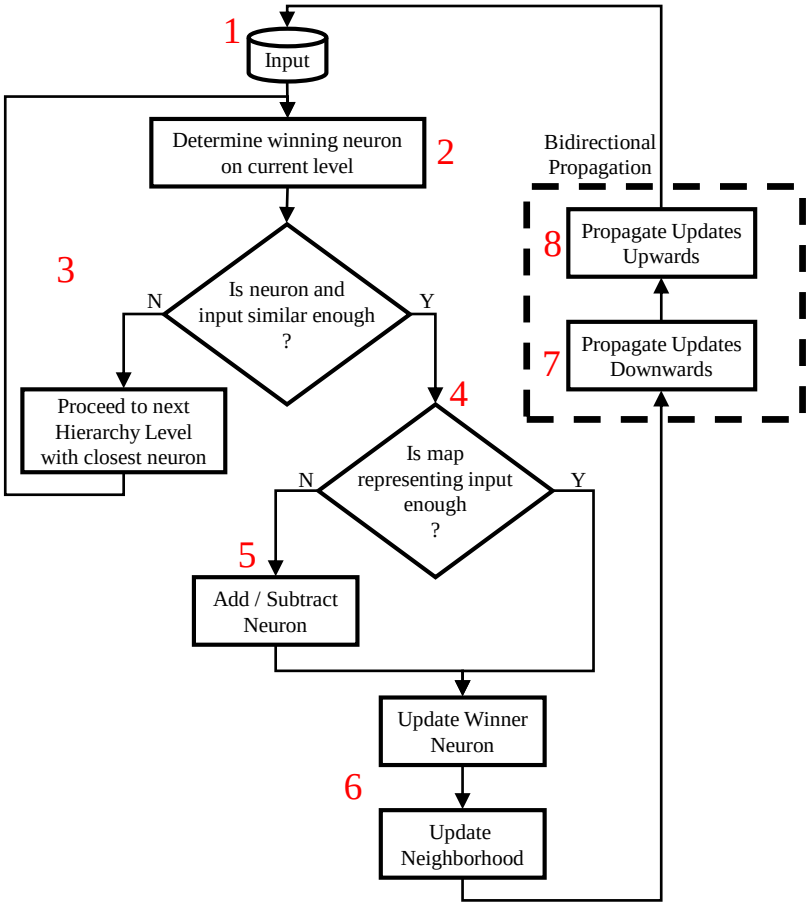
Formal Definition

- $A = \{A_1, \ldots, A_t\}$
  - A set of hierarchy levels.
- $A_i = \{W_{i,1}, \ldots, W_{i,m}\}$
  - A set of individual SOMs.
- $W_{i,j} = \{w_1, \ldots, w_n\}$
  - A SOM of n neurons.
  - Each neuron contains a storage unit s_k and a weight vector $\mathbf{v}_k$.

Three Basic Functions

- Three functions are introduced for actions on the system:
- *Lev()*
  - Returns the hierarchy level that a SOM currently resides on.
- *Chd()*
  - Returns a set of SOMs that have child relationship to a particular neuron.
- *Par()*
  - Returns the parent SOM of a particular neuron.

Process Flow for Training



1. Input is inserted into network
2. Neuron that is most similar is selected.
3. Descend through hierarchy until similarity is maximal.
4. Determine whether correct number of neurons represent pattern.
5. Add / Subtract neurons accordingly.
6. Update neuron and neighbourhood.
7. Update children SOMs.
8. Update parent SOM.

Conceptual View

- At the top-most hierarchy level (A_1), only one feature map would exist.
  - This map contains the absolute highest conceptual view of the entire hierarchical structure.

- Additional SOMs on subsequent levels offer more precise pattern abstraction.
  - SOMs are denoted by the sequence of their parents.
  - $W_{3,6,4}$ denotes the feature map is the fourth map on the third level derived from the sixth map on the previous level.

Learning of Features

- Once a winning neuron wi*has been identified (denoted by an asterisk), its weight vector vi* is updated according to a learning rate α.
  - The value α decays over time according to the current training iteration.
  - $\boldsymbol{v}_i^*(q) = \boldsymbol{v}_i^*(q\text{-}1) + \alpha(\boldsymbol{p}_k(q) - \boldsymbol{v}_i^*(q\text{-}1))$

- The neighbourhood must also be updated with a modified learning rate α'.
  - $\boldsymbol{v}_{Ni*(d)}(q) = \boldsymbol{v}_{Ni*(d)}(q\text{-}1) + \alpha(\boldsymbol{p}_k(q) - \boldsymbol{v}_{Ni*(d)}(q\text{-}1))$

Bidirectional Update Propagation

- Let w_i^* be the winning neuron in SOM $W_{j,k}$ for input k.

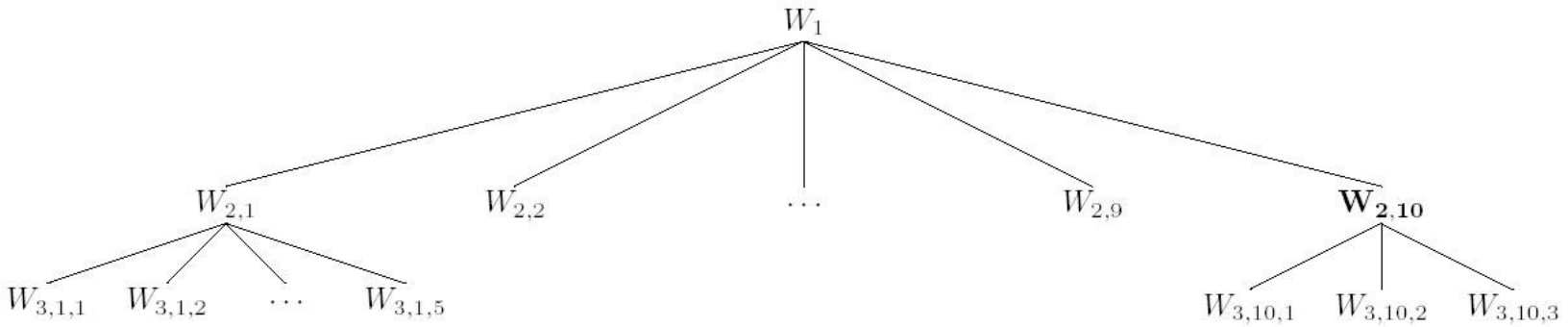
- To propagate upwards:
  - Calculate $Par(w_i^*) = W_{j-1,m}$, where $Lev(W_{j-1,m}) < Lev(W_{j,k})$.
  - Update all neurons w_a contained in $W_{j-1,m}$ that are similar to w_i^*.
  - $\boldsymbol{v}_a^*(q) = \boldsymbol{v}_a^*(q\text{-}1) + \beta(\boldsymbol{p}_k(q) - \boldsymbol{v}_a^*(q\text{-}1))$

Bidirectional Update Propagation

- To propagate downwards:
  - Calculate $Chd(w_i^*) = A^*_{j+1}$, where $j+1$ is the next level in the hierarchy succeeding level j.
  - Update the corresponding weight vectors for all neurons w_b in SOM $W_{j+1,t}$, where $W_{j+1,t}$ is on the lower level A^*_{j+1}.
  - $\boldsymbol{v}_b^*(q) = \boldsymbol{v}_b^*(q\text{-}1) + \gamma(\boldsymbol{p}_k(q) - \boldsymbol{v}_b^*(q\text{-}1))$

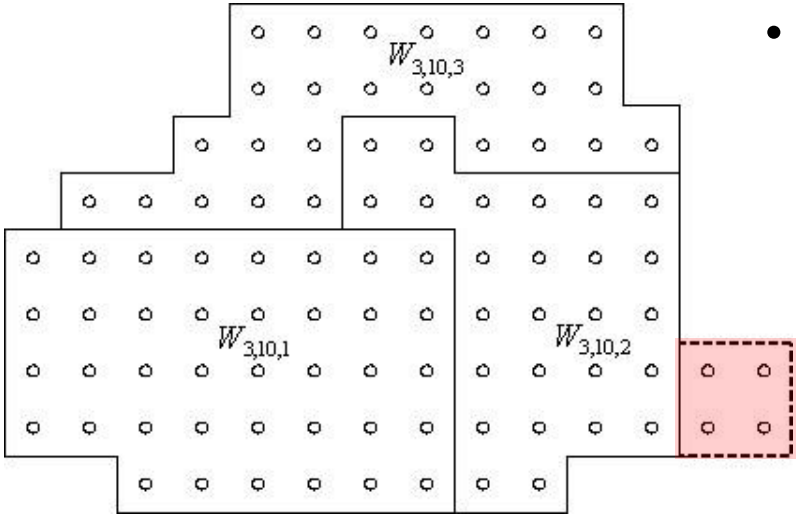
- The learning rates β and γ are derived from a value of α.
- Generally, updates to a parent neuron are not as strong as updates to children neurons.

Web-based News Coverage Example



- The top-most level of the hierarchy contains news articles pertaining to high-level concepts and are arranged according to their features.
  - The entire collection of Web documents on the online news site are presented through feature maps that abstract their similarities.
- Individual maps $W_{2,1}$, ..., $W_{2,10}$ are Web documents pertaining to Global, Local, Political, Business, Weather, Entertainment, Technology, Sports, Opinion, and Health news respectively.

Web-based News Coverage Example



- Feature map $W_{2,10}$ with neurons linking to three children maps: $W_{3,10,1}$, $W_{3,10,2}$, $W_{3,10,3}$.
  - Articles in $W_{2,10}$ relate to Health News.
  - $W_{3,10,1}$ relates to Health Research Funding.
  - $W_{3,10,2}$ relates to Health Outbreak Crises.
  - $W_{3,10,3}$ relates to Doctor shortages.

- New Health-related articles are coming in rapidly relating to a recent international outbreak.
- Neurons are added to $W_{2,10}$ in the Health Outbreak Crises cluster, that point to the SOM $W_{3,10,2}$.

Conclusion

- The Web mining model of growing hierarchical self-organizing maps minimizes the effect of the dynamic data and high-dimensionality problems.
- Bidirection Update Propagation allows for changes in pattern abstractions to be reflect on multiple levels in the hierarchy.
- The Web-based News Coverage example demonstrates the effectiveness of growing hierarchical self-organizing maps when used in conjunction with bidirectional update propagation.

Growing Hierarchical Self-Organizing Maps for Web Mining

Joseph P. Herbert and JingTao Yao

Department of Computer Science,
University or Regina
CANADA S4S 0A2

herbertj@cs.uregina.ca
http://www2.cs.uregina.ca/~herbertj

jtyao@cs.uregina.ca
http://www2.cs.uregina.ca/~jtyao

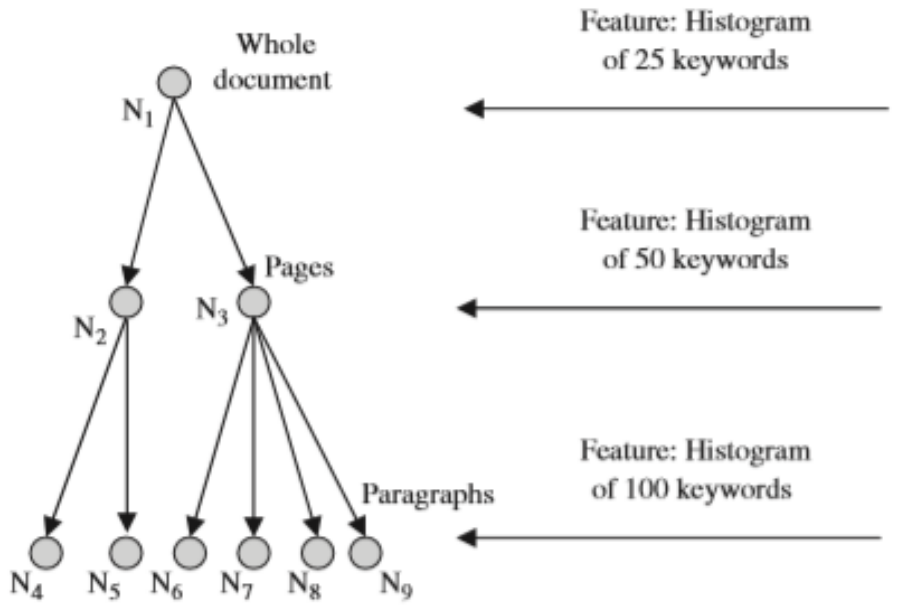


Fig. 7. Illustration for the tree-structured feature representation of a document.

Fig. 10. Samples from 12 different species of flower images.

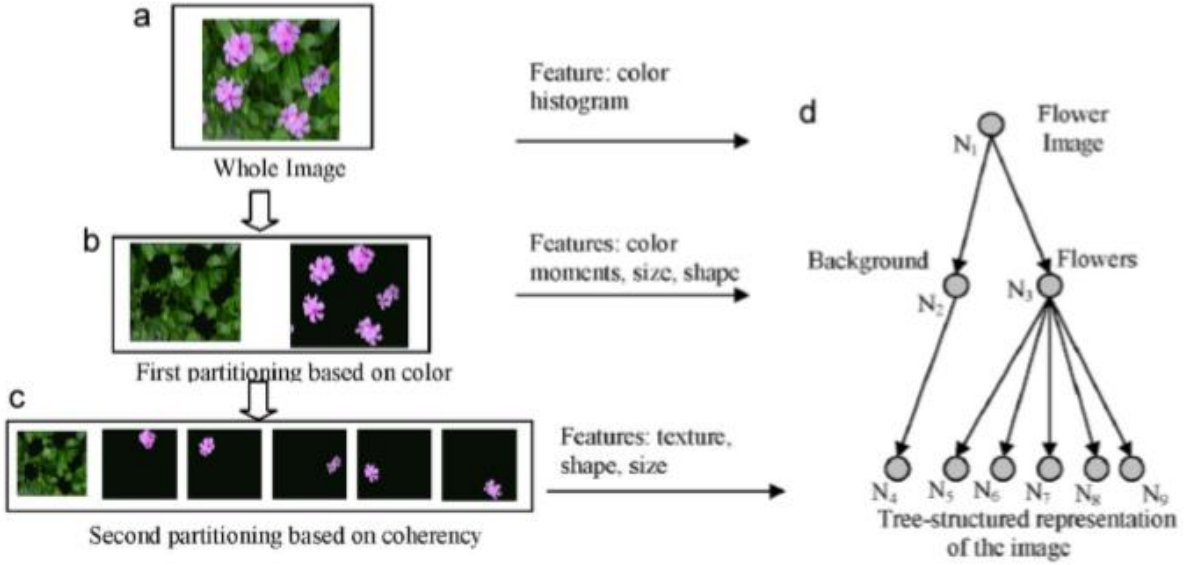
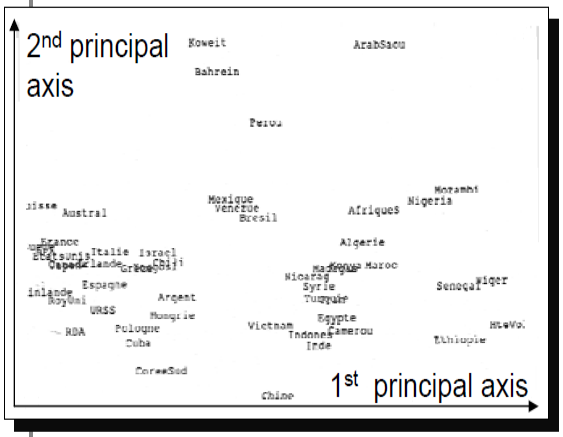


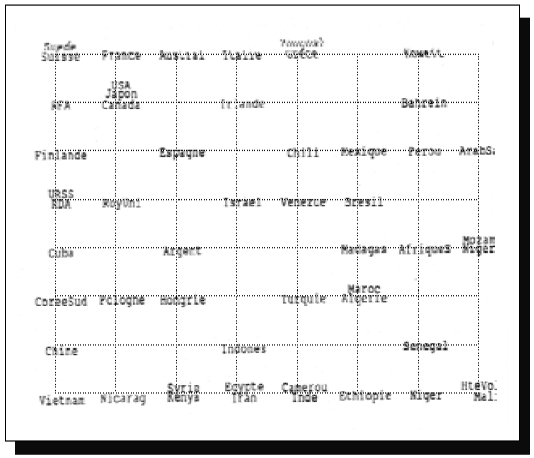
Fig. 11. Demonstration of a tree-structure from a flower image. (a)–(c) partitioning flower images; and (d) tree-structured image representation.

Macroeconomical data (2/2)

PCA



Kohonen



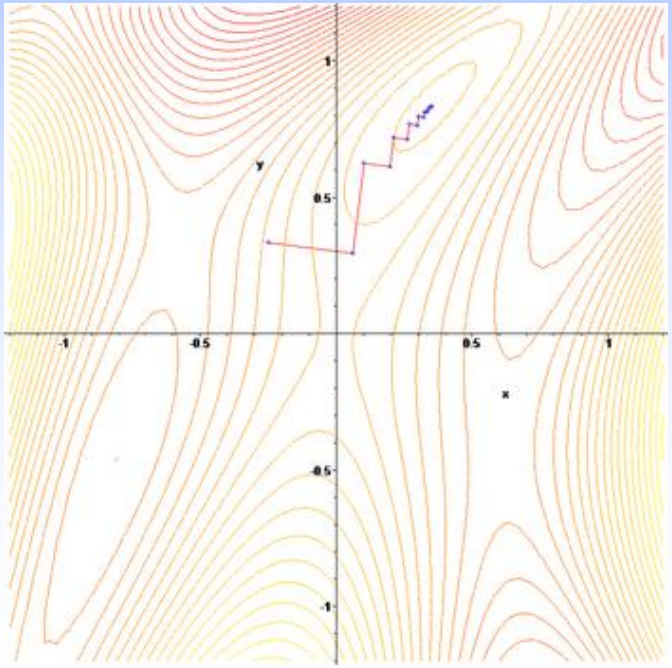
M. Verleysen
& G. Simon
UCL

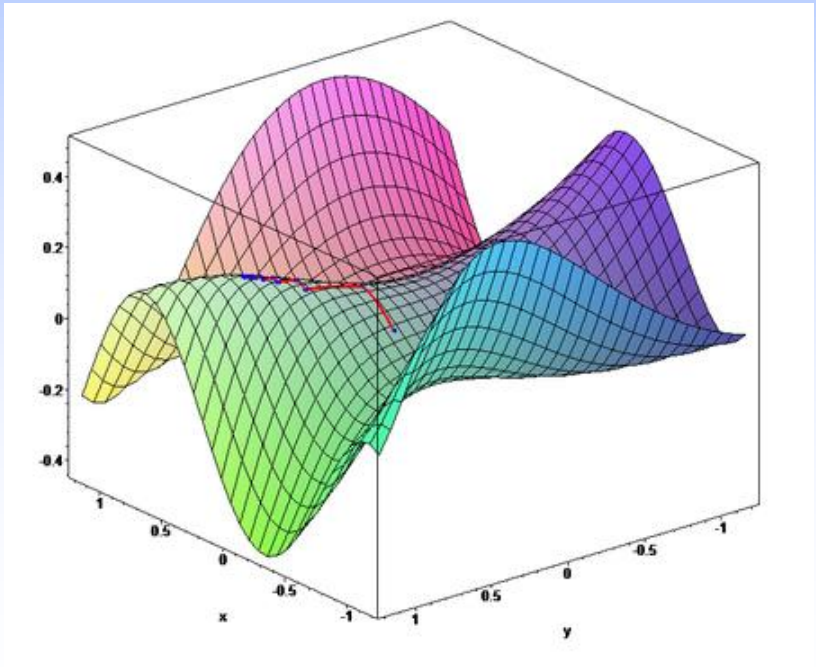
Sammon's projection, or **Sammon's mapping** is an algorithm that maps a high-dimensional space to a space of lower dimensionality (see multidimensional scaling).
Denote the distance between **i**th and **j**th objects in the original space by d^*_{ij}, and the distance between their projections by d_{ij}

Sammon's projection aims to minimize the following error function, which is often referred to as **Sammon's stress**:

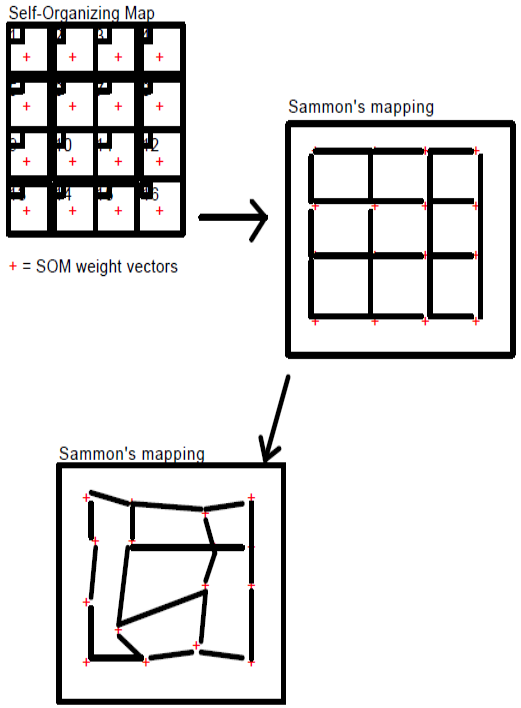
$$E = \frac{1}{\sum_{i<j} d^*_{ij}} \sum \sum_{i<j} \frac{(d^*_{ij} - d_{ij})^2}{d^*_{ij}}.$$

The minimization can be performed either by gradient descent, as proposed initially, or by other means.





Combining SOM and Sammon's mapping



1) Initialize the N-dimensional array with SOM weight vectors
2) Calculate the 2-dimensional Sammon's mapping as usua

BENEFIT: The calculation is less time consuming because
SOM algorithm has already made substantial data reductior

9.1.2009

Mikko Kolehmainen

WEBSOM

• Searching for relevant documents from a very large collection has traditionally been based on keywords and their Boolean expressions.
• Often, however, the search results show high recall and low precision, or vice versa

• WEBSOM is a full-text information retrieval and exploration method for large document collections.
• Self-Organizing Map (SOM) is used to statistically analyse relations between the words, and then, based on this analysis, to create a document map.
• Similar documents become positioned close to each other on the document map.
• Therefore, this document landscape provides a good basis for search and exploration

• The WEBSOM method thus has a two-level information processing architecture.
• On the first level, a "semantic SOM" categorizes the words of the source text into clusters.
• The second level uses these clusters of the *word category map* and creates an ordered display of the documents, a *document map*.

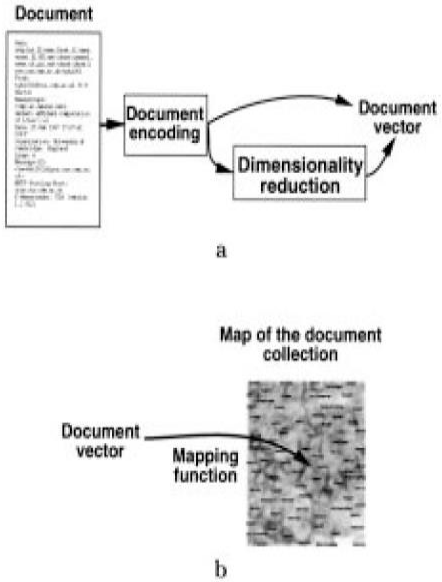
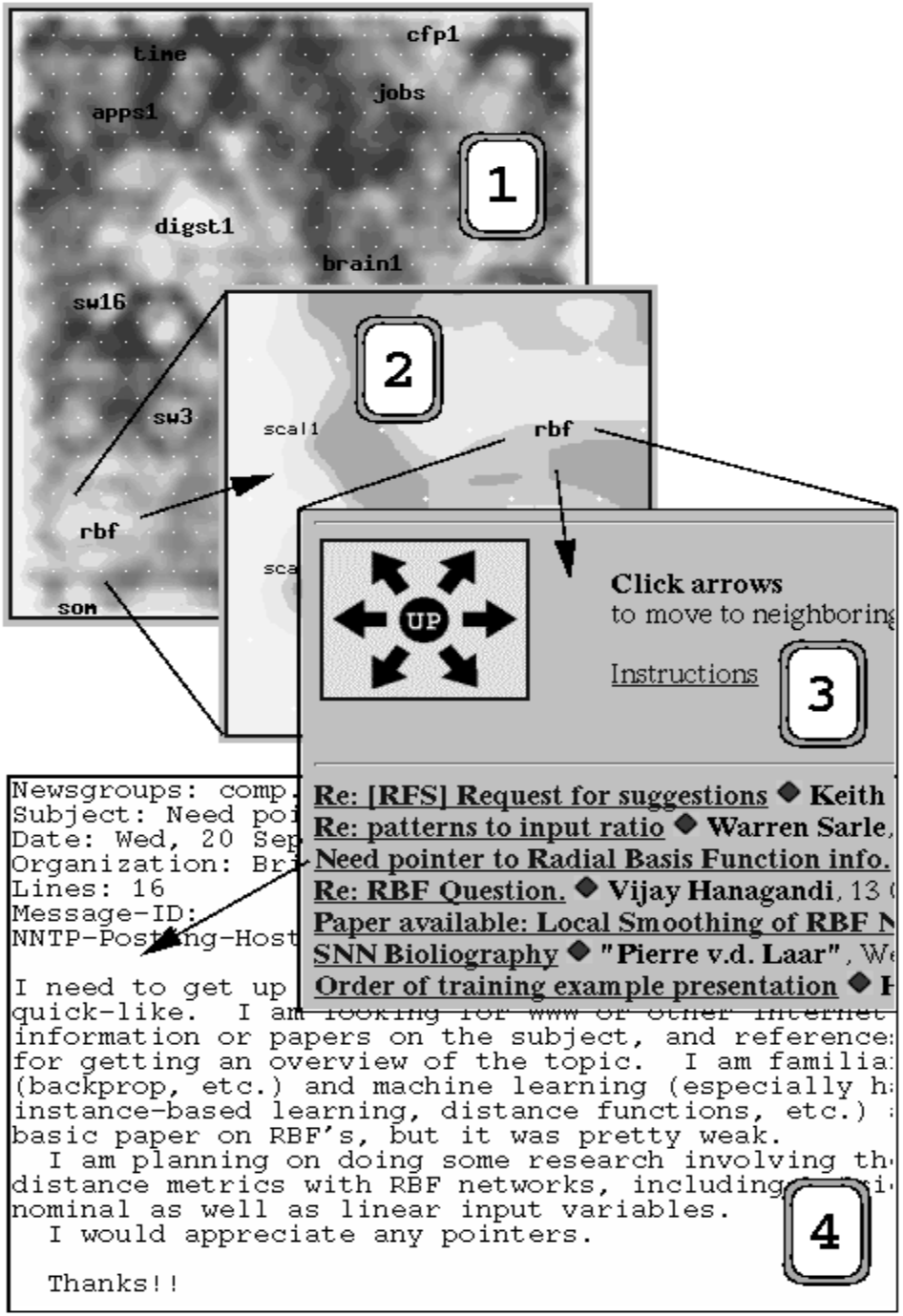
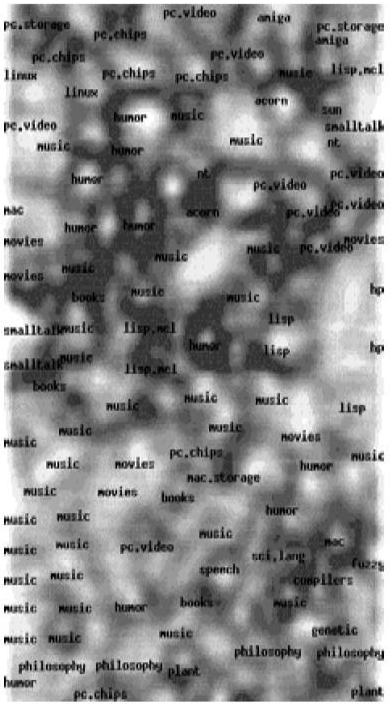


Figure 1. The basic building blocks of the WEBSOM system. (a) First the document is encoded into a numerical vector. The dimensionality of the vector can be reduced for computational reasons. (b) The numerical representation of the document is mapped onto a display of the document collection that has been generated using the SOM algorithm.


cfp1
time
jobs
apps1
1
digst1
brain1
sw16
2
sw3
scal1
rbf
rbf
som
UP
Click arrows
to move to neighborin
Instructions
3
Re: [RFS] Request for suggestions ◆ Keith
Re: patterns to input ratio ◆ Warren Sarle,
Need pointer to Radial Basis Function info.
Re: RBF Question. ◆ Vijay Hanagandi, 13
Paper available: Local Smoothing of RBF N
SNN Bioliography ◆ "Pierre v.d. Laar", W
Order of training example presentation ◆ H
Newsgroups: comp.
Subject: Need poi
Date: Wed, 20 Sep
Organization: Bri
Lines: 16
Message-ID:
NNTP-Posting-Host
I need to get up
quick-like. I am looking for www or other internet
information or papers on the subject, and reference
for getting an overview of the topic. I am familia
(backprop, etc.) and machine learning (especially h
instance-based learning, distance functions, etc.)
basic paper on RBF's, but it was pretty weak.
I am planning on doing some research involving th
distance metrics with RBF networks, including
nominal as well as linear input variables.
I would appreciate any pointers.
Thanks!!
4

Extensions of SOM (2/3)

- WEBSOM: SOM for collections of documents: from text to web pages as well as pictures
- Data preprocessing: specific encoding before running SOM



S. Kaski, T. Honkela, K. Lagus, T. Kohonen, "WEBSOM - Self-Organizing Maps of document collections", *Neurocomputing*, 21, pp 101-117, 1998.

Available demo online at: http://websom.hut.fi/websom/

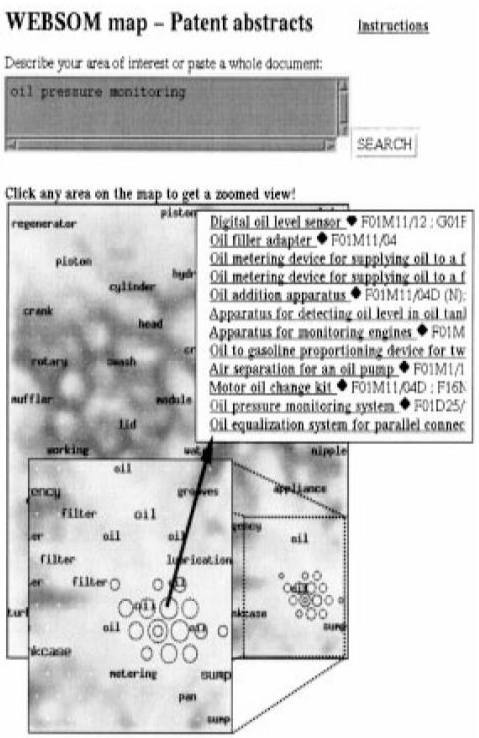


Figure 2. Exploration of a collection of 10,000 patent abstracts by mapping a new document or query to provide starting points for exploration. The user was interested in finding patents related to oil pressure monitoring. The system encoded this description of interest as a document and found best matching locations for it on the map, marked with circles. A detailed closeup of the region was revealed by clicking on the map and after the second click titles of individual abstracts were shown. An individual document may be viewed by clicking one of the titles. The labels which have been automatically selected and positioned on the map help in gaining an overview of the patent collection and also guide the exploration of the map.

Visualizing the image database

Strong & Gong, 2008

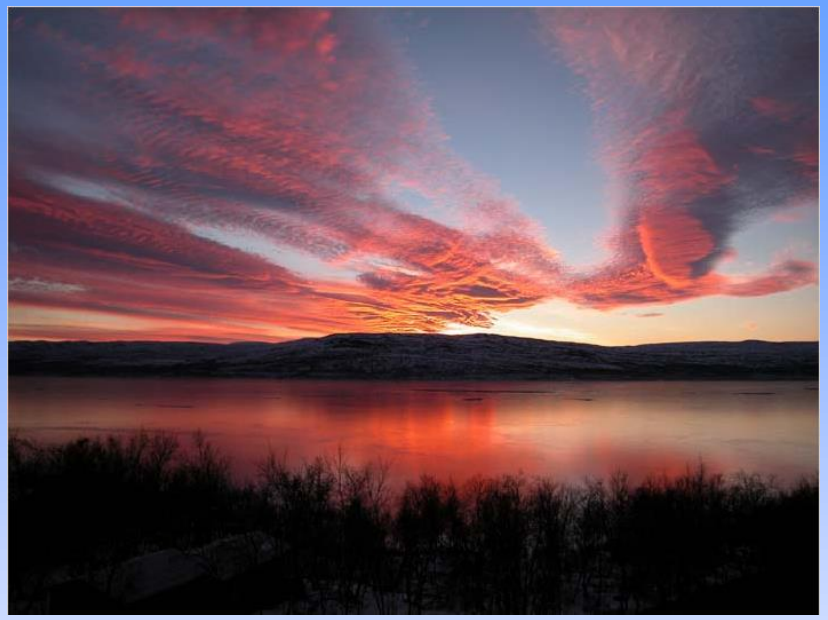
Similar colors as in the example image

Images with different colors related to the texture of the example image

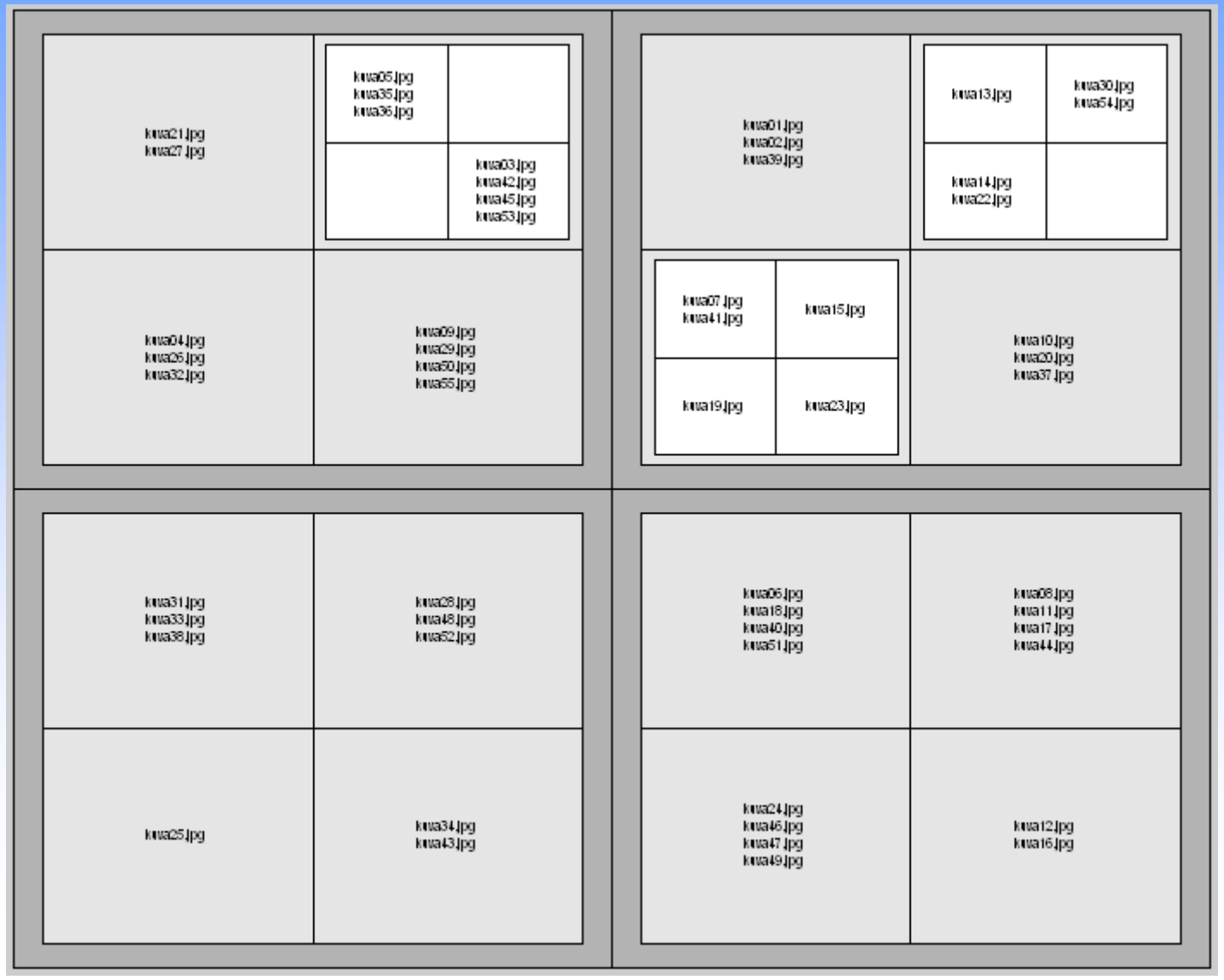
Example image

Similar texture as in the example image

Images with various colors related to the texture of the example image



This way the color SOM classified the images of the test set

































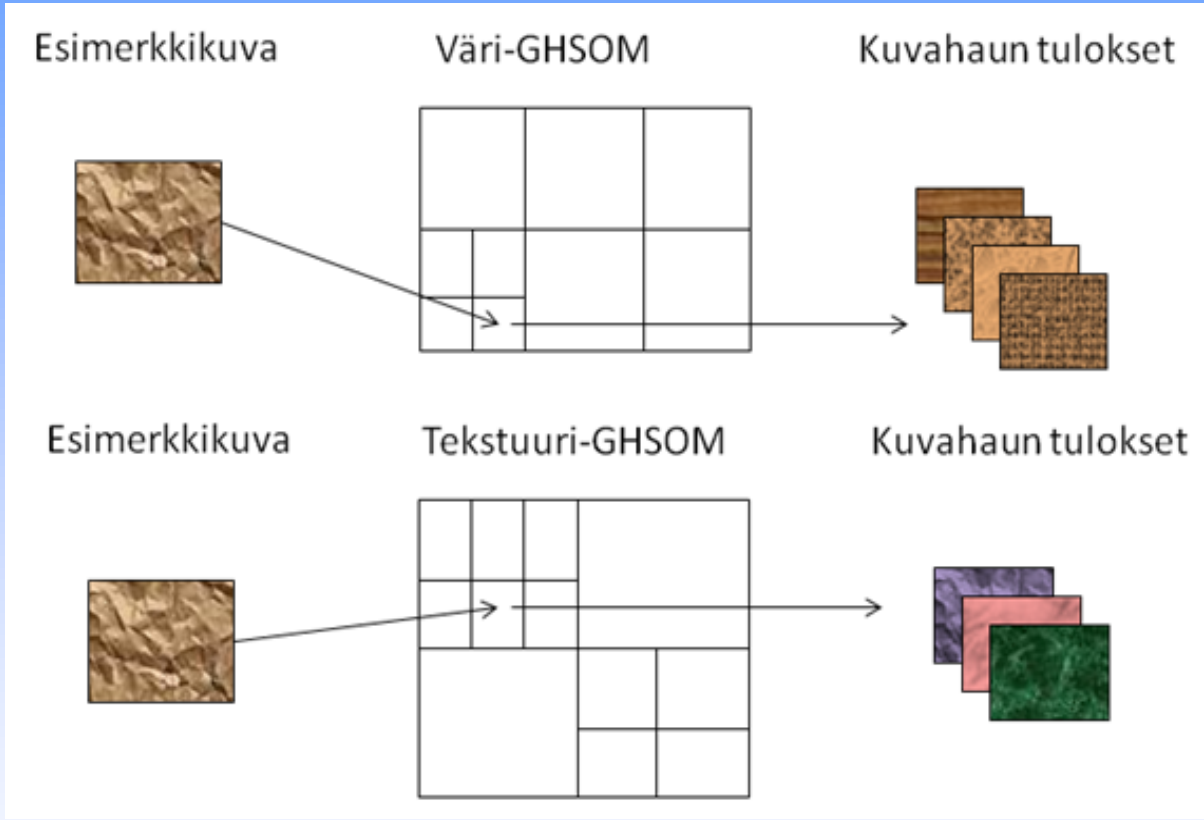


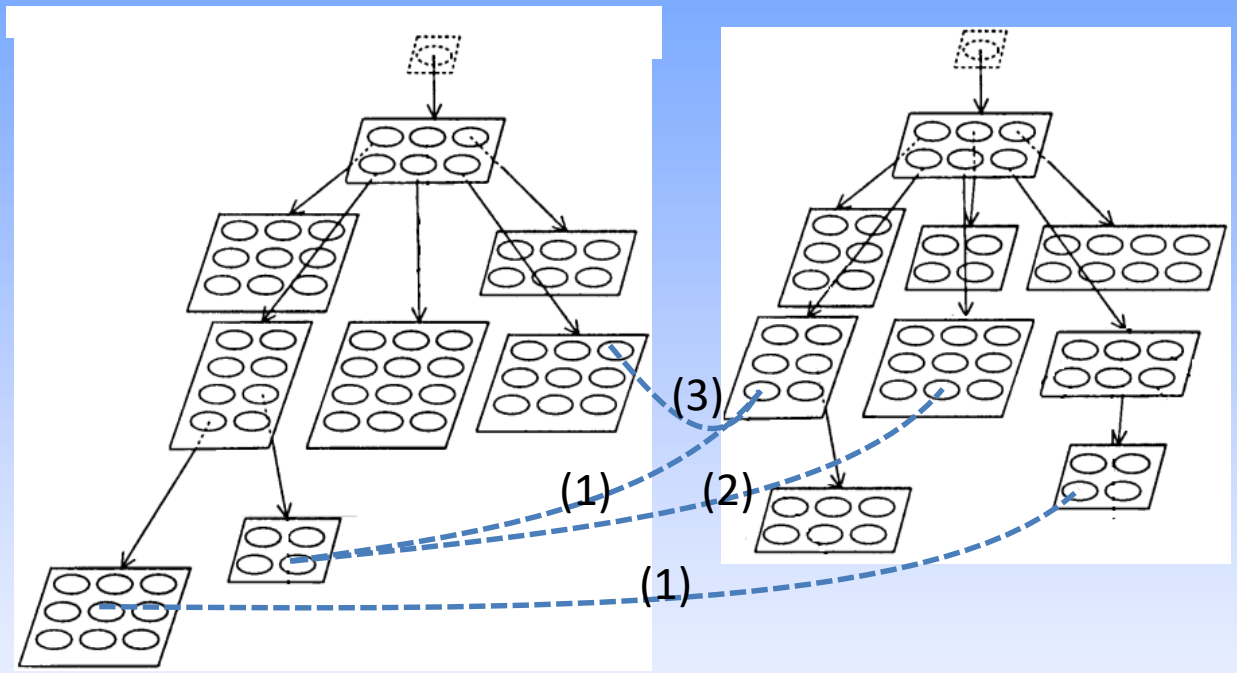


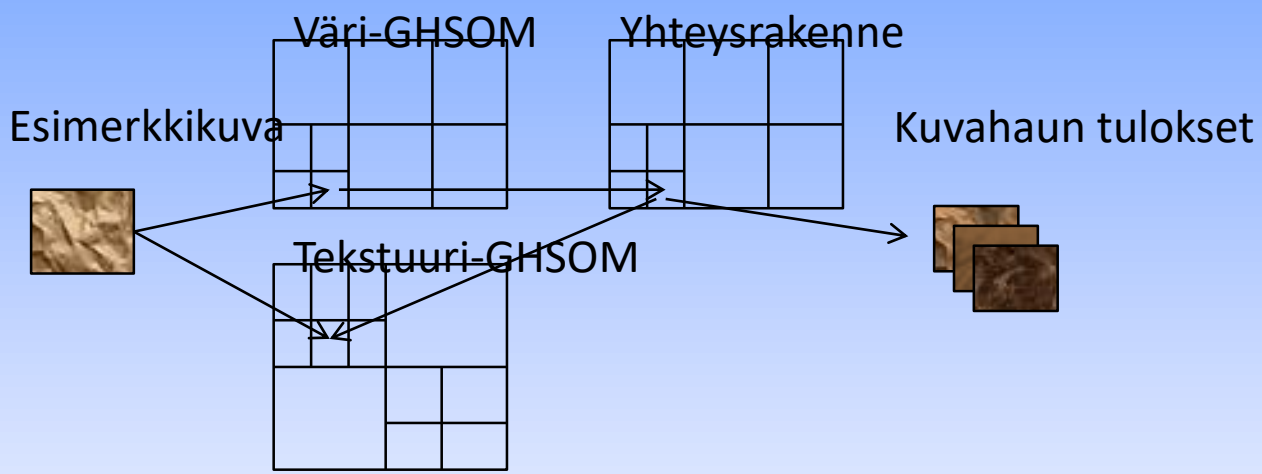


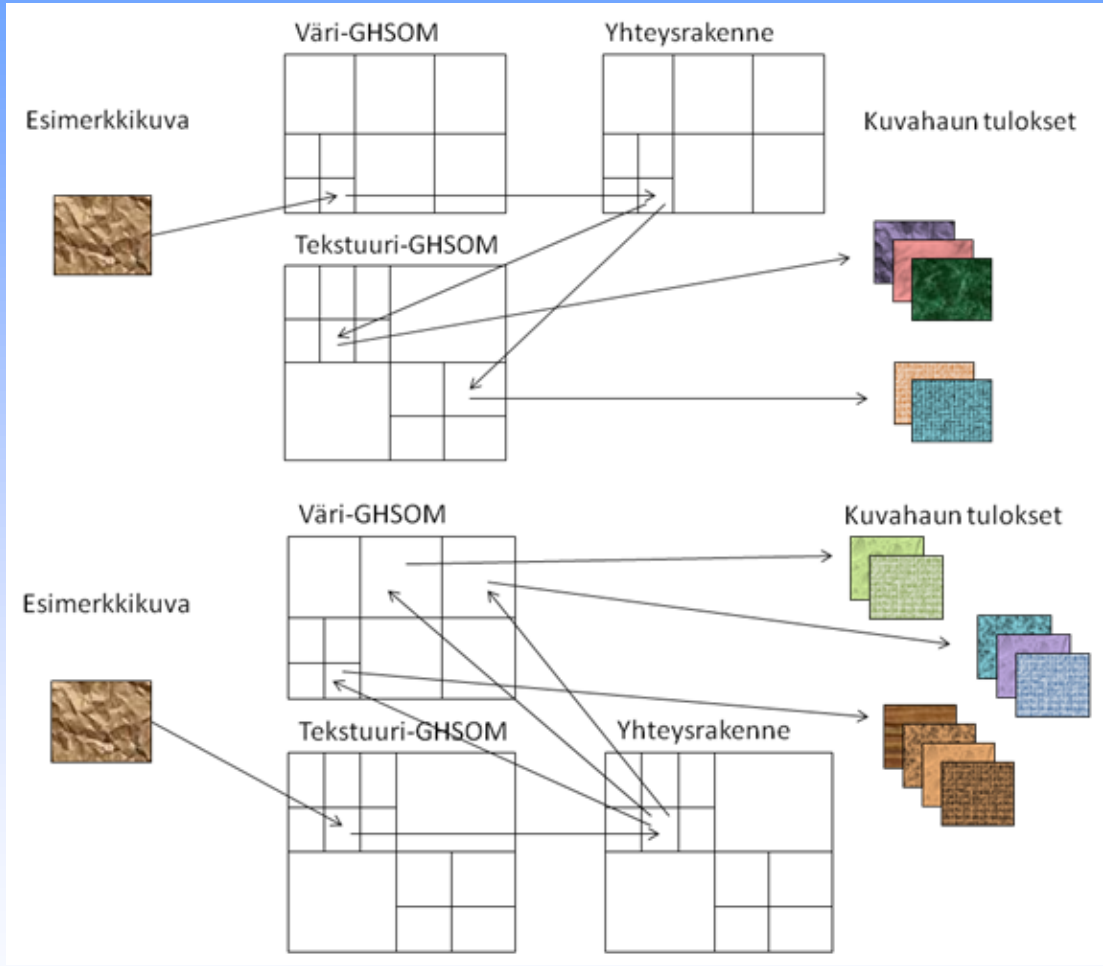


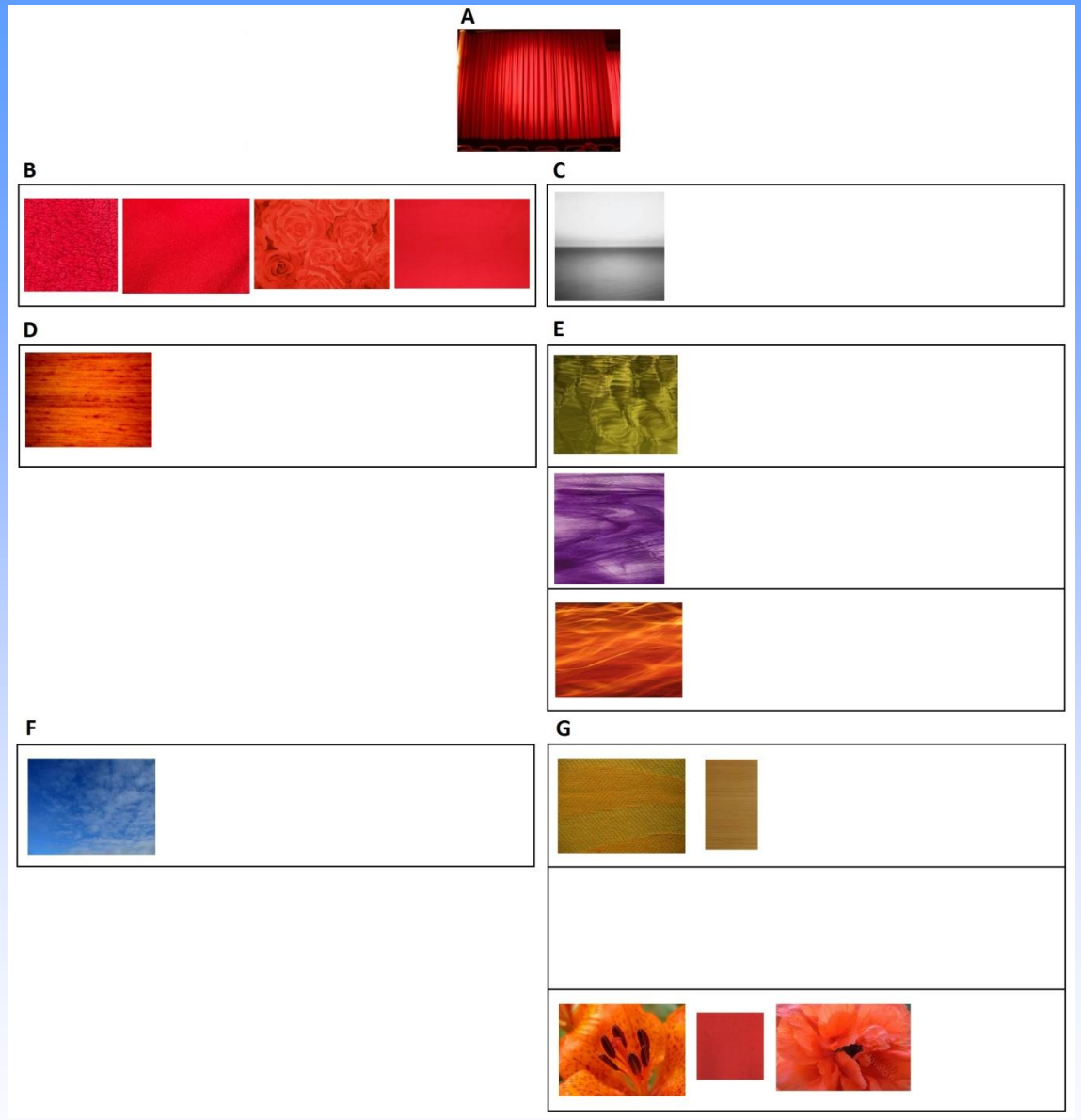

Näin tekstuuri-SOM luokitteli koulutuskuvajoukon kuvat


Esimerkkikuva
Väri-GHSOM
Kuvahaun tulokset
Esimerkkikuva
Tekstuuri-GHSOM
Kuvahaun tulokset

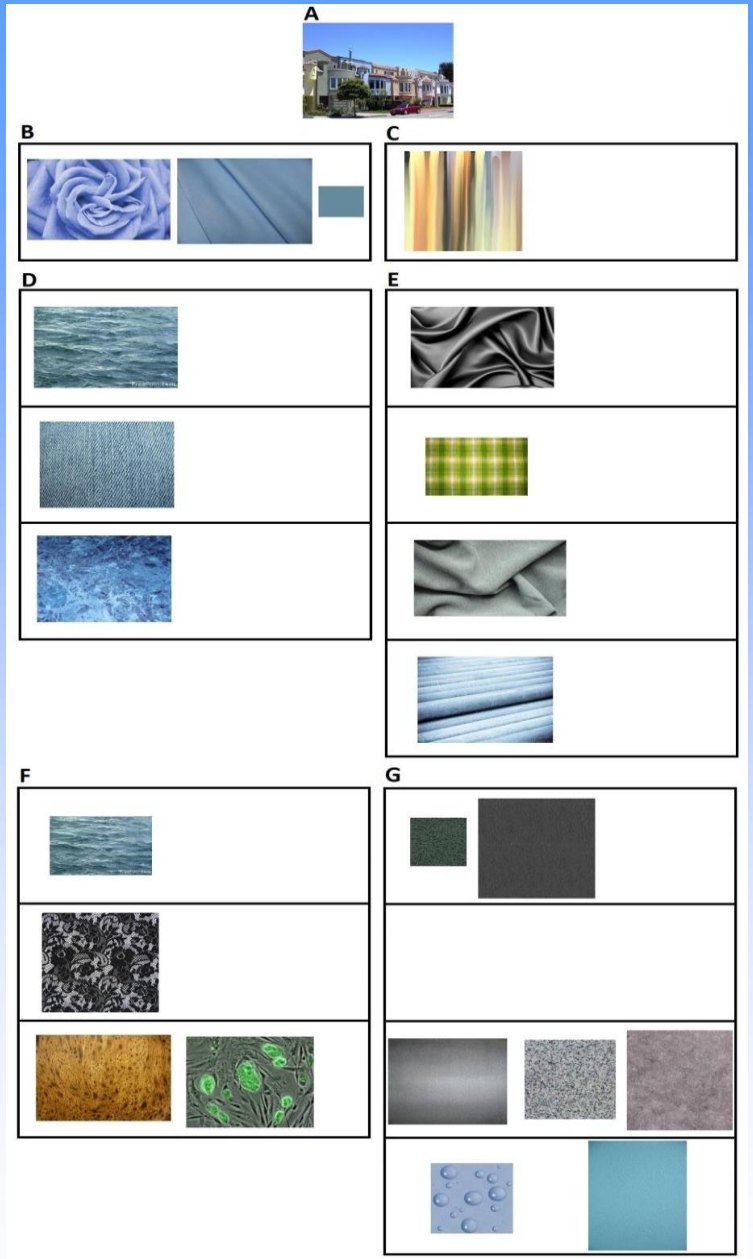

(3)
(1)
(2)
(1)


Väri-GHSOM
Yhteysrakenne
Esimerkkikuva
Kuvahaun tulokset
Tekstuuri-GHSOM

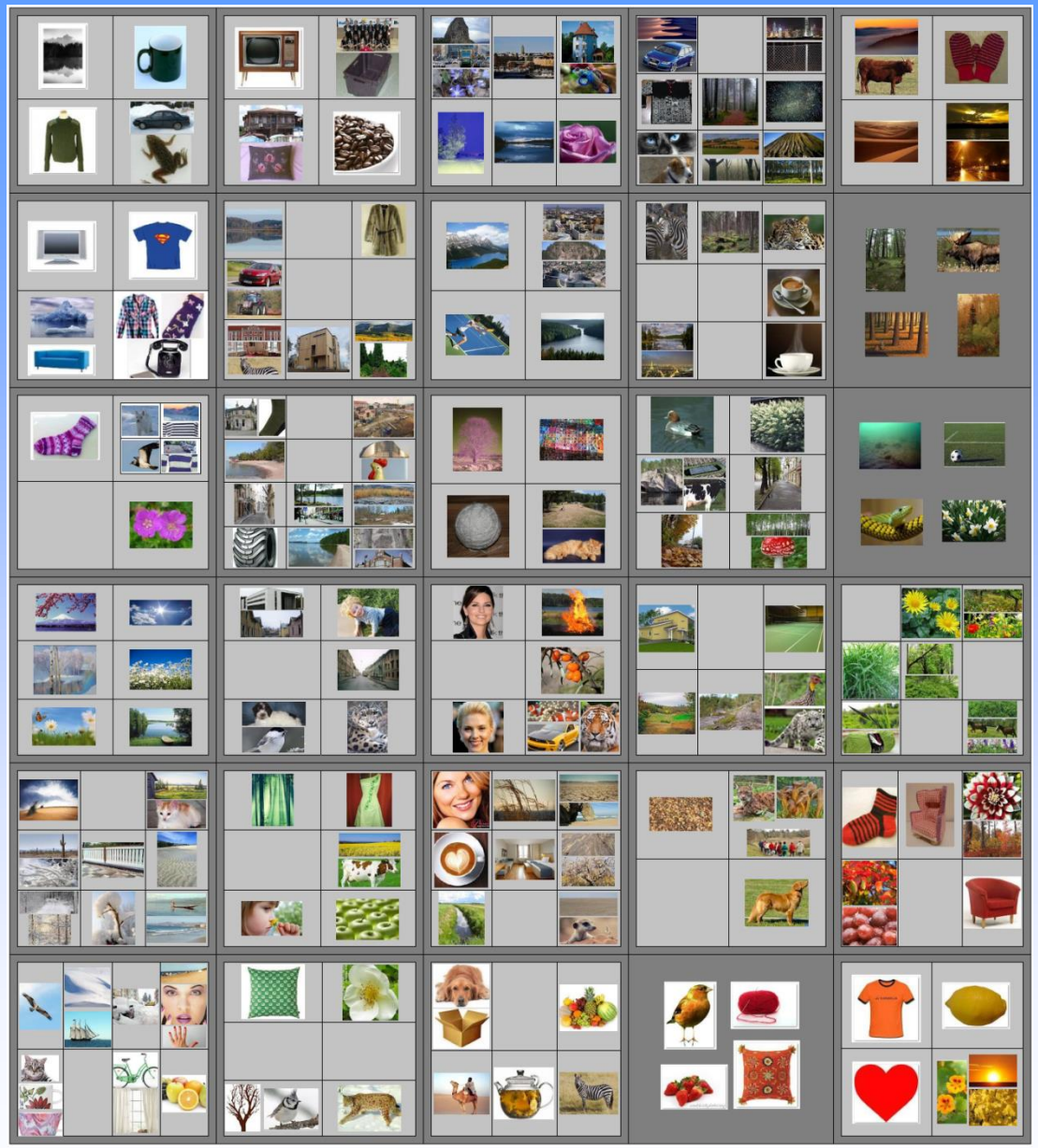

Väri-GHSOM
Yhteysrakenne
Esimerkkikuva
Kuvahaun tulokset
Tekstuuri-GHSOM
Väri-GHSOM
Kuvahaun tulokset
Esimerkkikuva
Tekstuuri-GHSOM
Yhteysrakenne

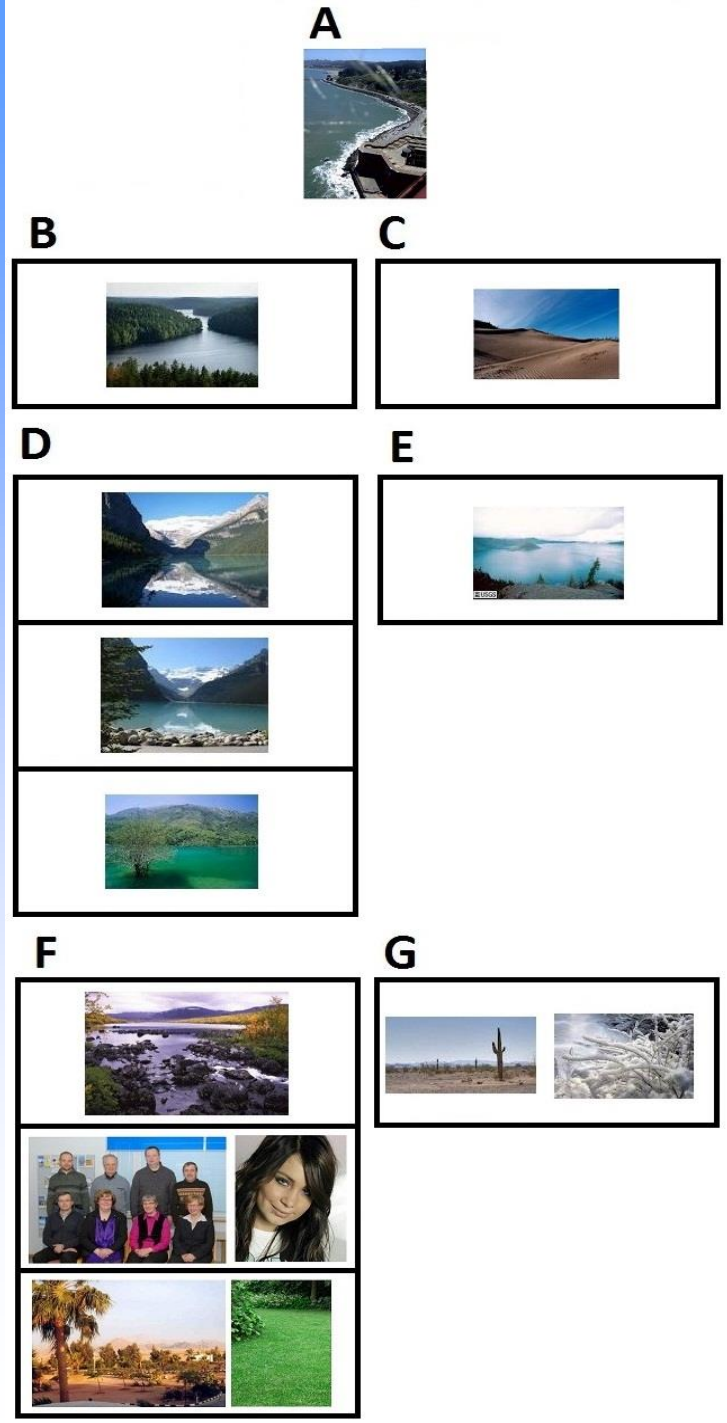


Hakutulokset eri hakufunktioilla yhdelle esimerkkikuvalle (A). B) searchColor, C) searchTexture, D) matchColor, E) matchTexture, F) searchColor2Texture, G) searchTexture2Color. Hakufunktiolla matchBoth ei saatu tuloksia.



Hakutulokset eri hakufunktioilla toiselle esimerkkikuvalle (A). B) searchColor, C) searchTexture, D) matchColor, E) matchTexture, F) searchColor2Texture, G) searchTexture2Color. Hakufunktiolla matchBoth ei saatu tuloksia.

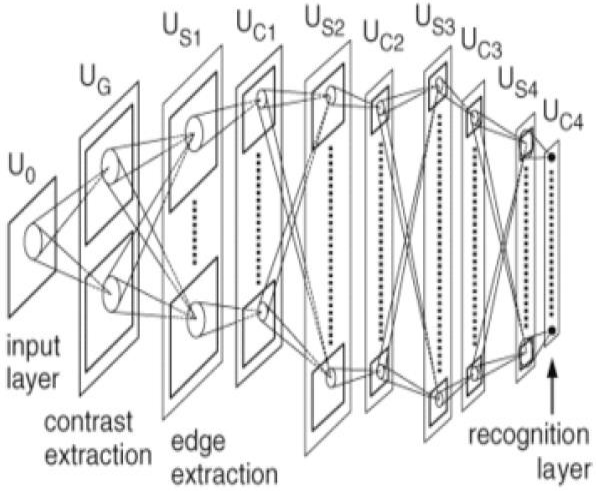




Hakutulokset eri hakufunktioilla yhdelle esimerkkikuvalle (A). B) searchColor, C) searchTexture, D) matchColor, E) matchTexture, F) searchColor2Texture, G) searchTexture2Color. Hakufunktiolla matchBoth ei saatu tuloksia.

Fukushima's Neocognitron (1980)

A hierarchical multi-layered neural network for visual pattern recognition.



Neocognitron

Fukushima, K. and Miyake, S. 1981

presented by Sam Thomson
Oct. 3, 2013

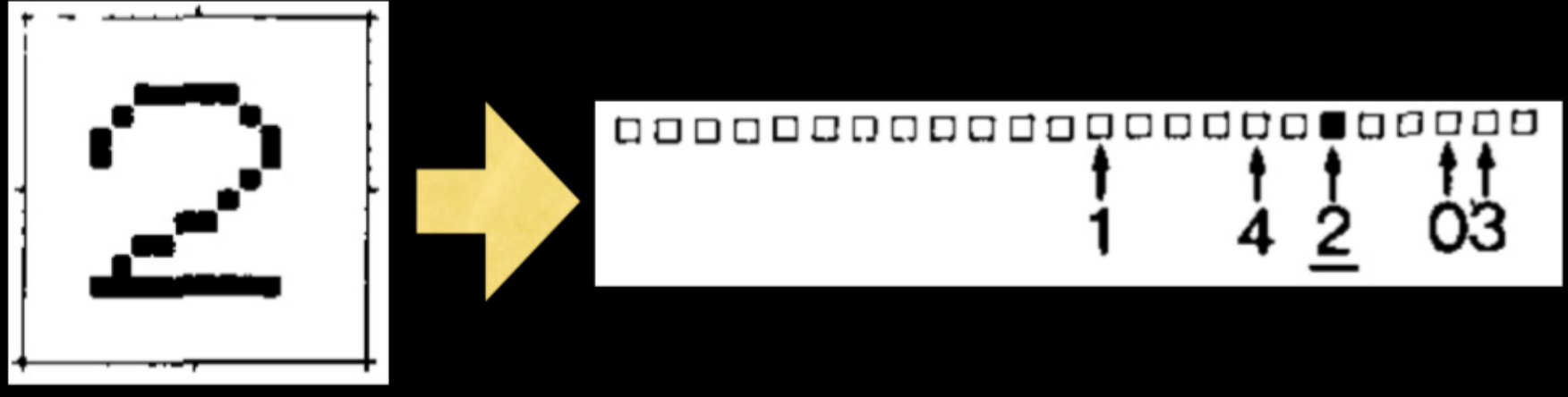
Overview

- multilayer neural network inspired by the mammalian visual system
- unsupervised image classification, tolerant to shifts and deformations
- improvement on the cognitron

Kunihiko Fukushima

Task

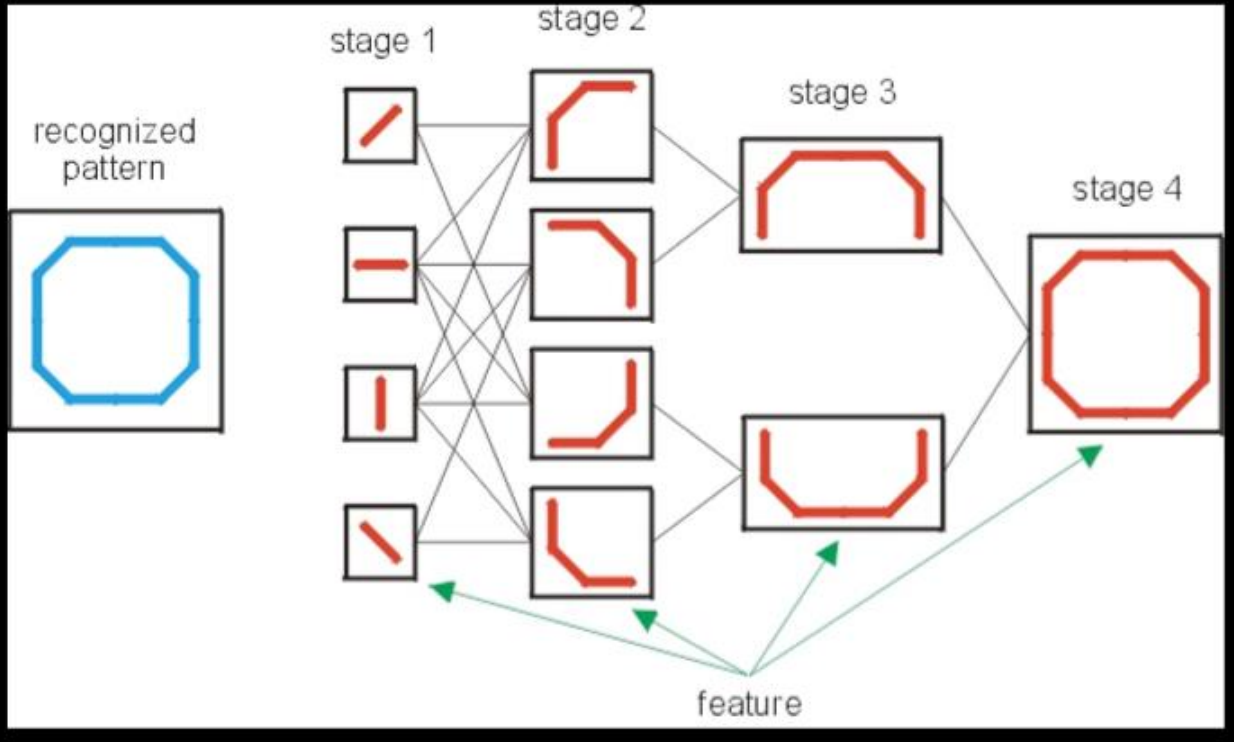
- Unsupervised handwritten character recognition
  - input - unlabeled images
  - output - vector, with each bit hopefully encoding a distinct class of images



Design

Design - High Level

- multiple (usually about 3 hidden) *layers*
- Successive layers recognize higher-level patterns

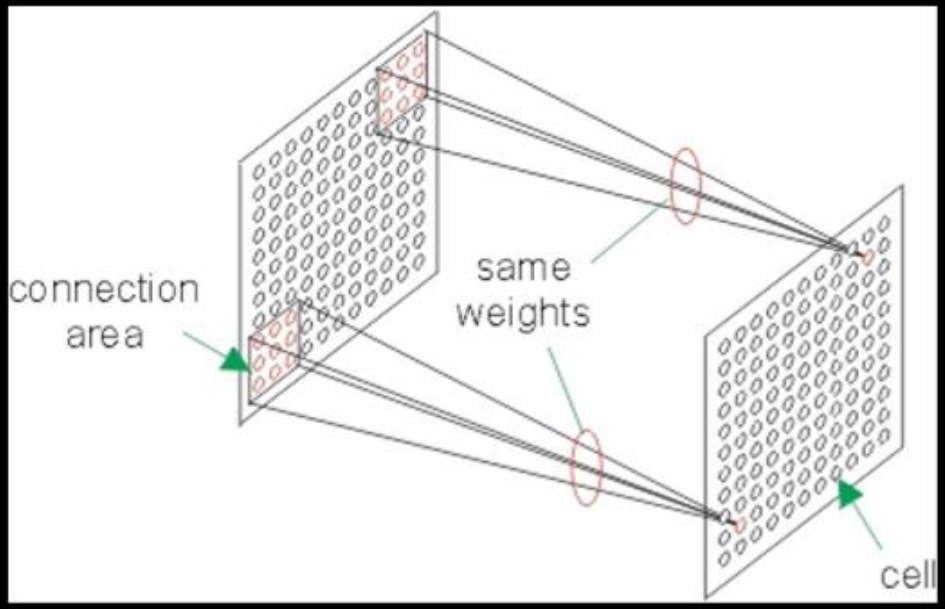


Design - Makeup of a Layer

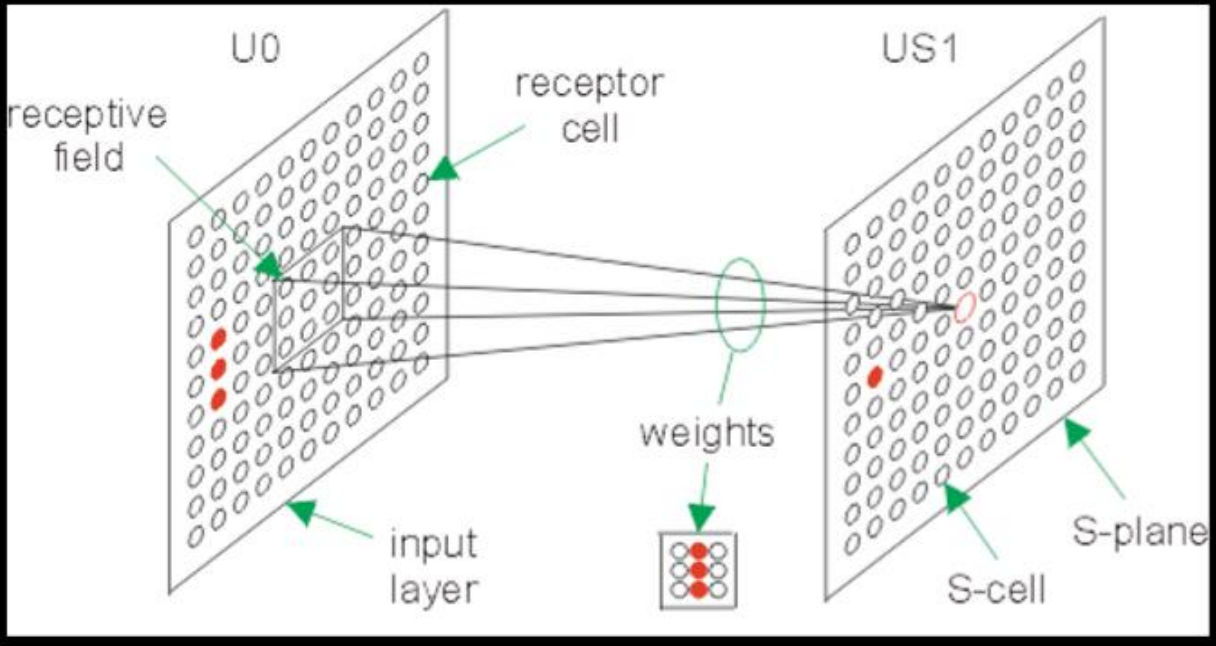
- each layer has k *S-planes*
- each S-plane feeds into its own *C-plane*
- V_S*-planes* and V_C*-planes* inhibit S-planes and C-planes, respectively

Design - S-plane

- cells in each plane are arranged in a 2-d grid
- each S-cell looks at a sliding 2-d window in the previous layer
- S-cells in a plane all have the same coefficients (i.e. they are convoluted), but look at a different window



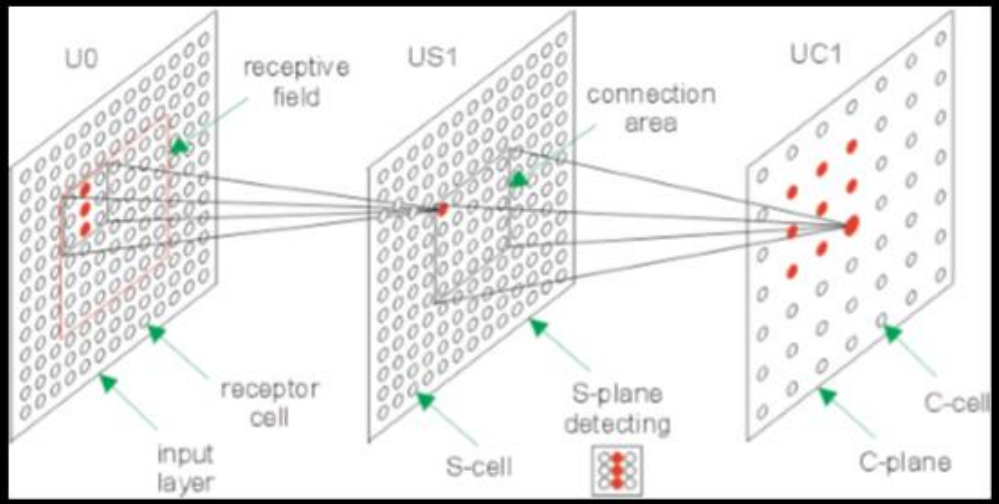
Design - S-plane



Design - S-plane

Design - C-plane

- an S-plane learns to recognize one feature no matter where it is
- the corresponding C-plane ORs a region of S-cells to recognize that feature anywhere in that region (achieving a level of shift invariance)
- C-cell input weights are not learned

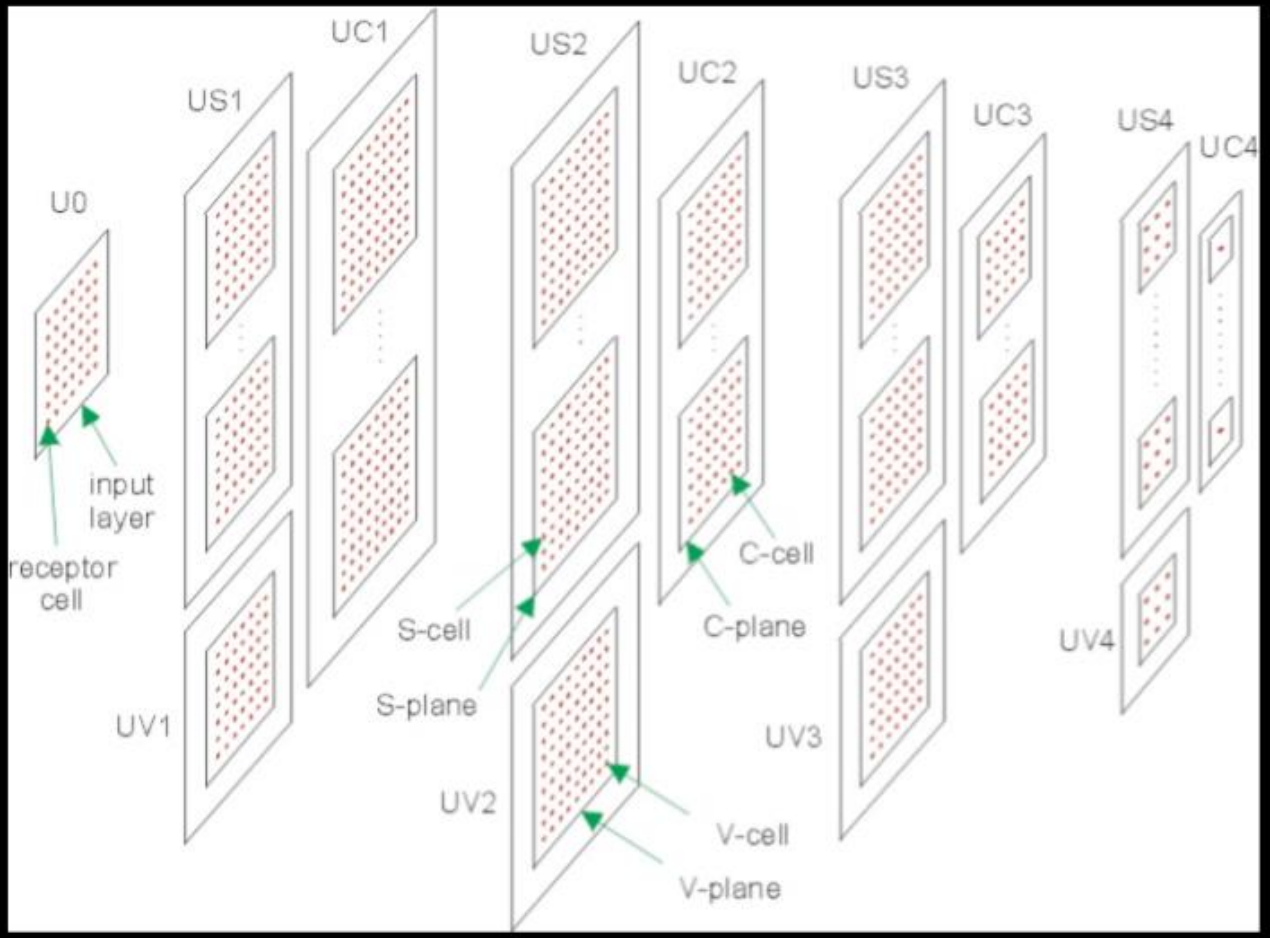


Design - C-plane

Design - Output

- In the final layer, each C-plane has only one cell, which effectively looks at the entire image

Design - All Together Now



Learning - Cognitron

- weights get initialized with small positive values
- for each training instance, if a cell is the most active in its region *and* in its plane, then its active weights get reinforced
- show the same few training instances over and over again

Learning - Cognitron

- similar to Hebbian learning ("fire together, wire together"), but only one cell maximum per layer and region gets reinforced
- note: we're not doing gradient descent, and not minimizing any objective

Learning

- mostly glossing over inhibitor cells and and mathematical formulas. refer to paper
- math works out so that an S-cell's weights directly correspond to the feature it is recognizing, and activation = cosine similarity

Learning - Example Weights

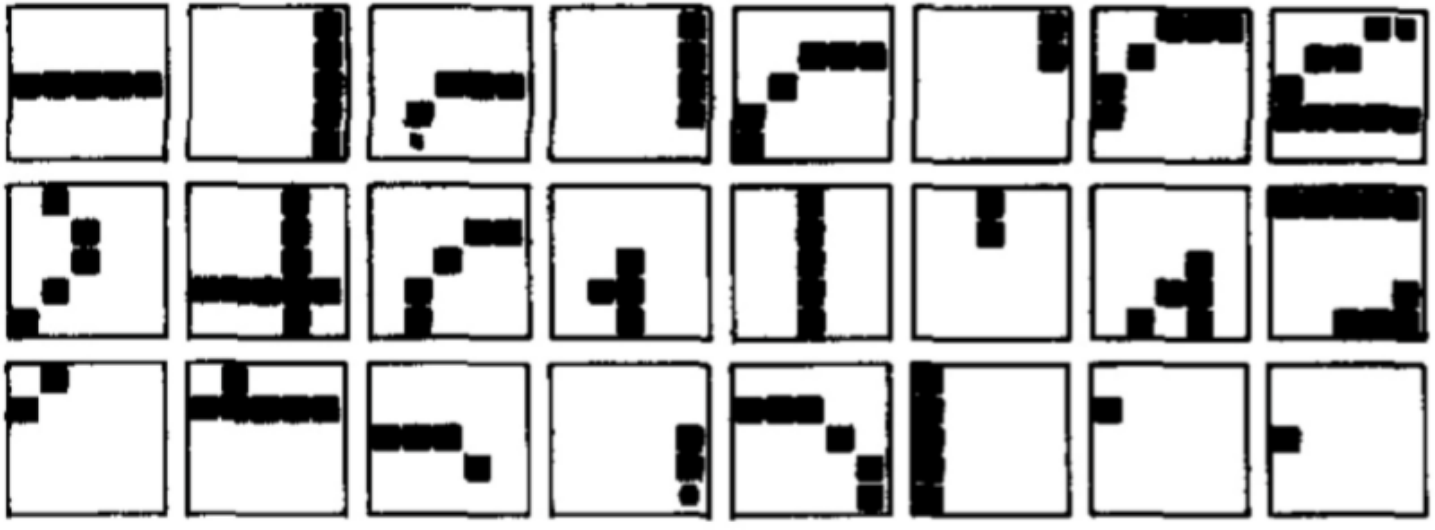


Fig 12 Receptive fields of the cells of each of the 24 S-planes of layer U_{S1}, which has finished learning

Learning - Example Activations

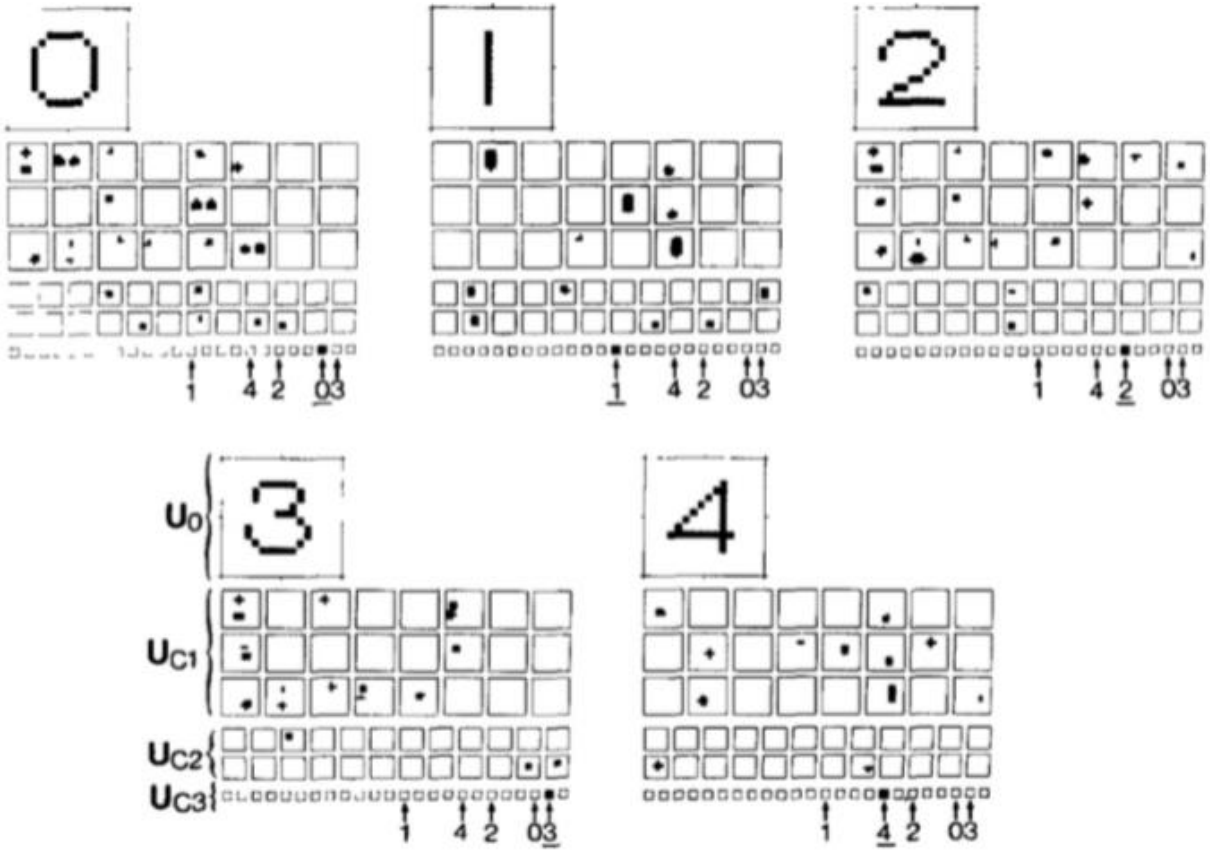


Fig 10 Response of the cells of layers U_0, U_{C1}, U_{C2} and U_{C3} to each of the five stimulus patterns

Problems - Not Really Scale-Invariant

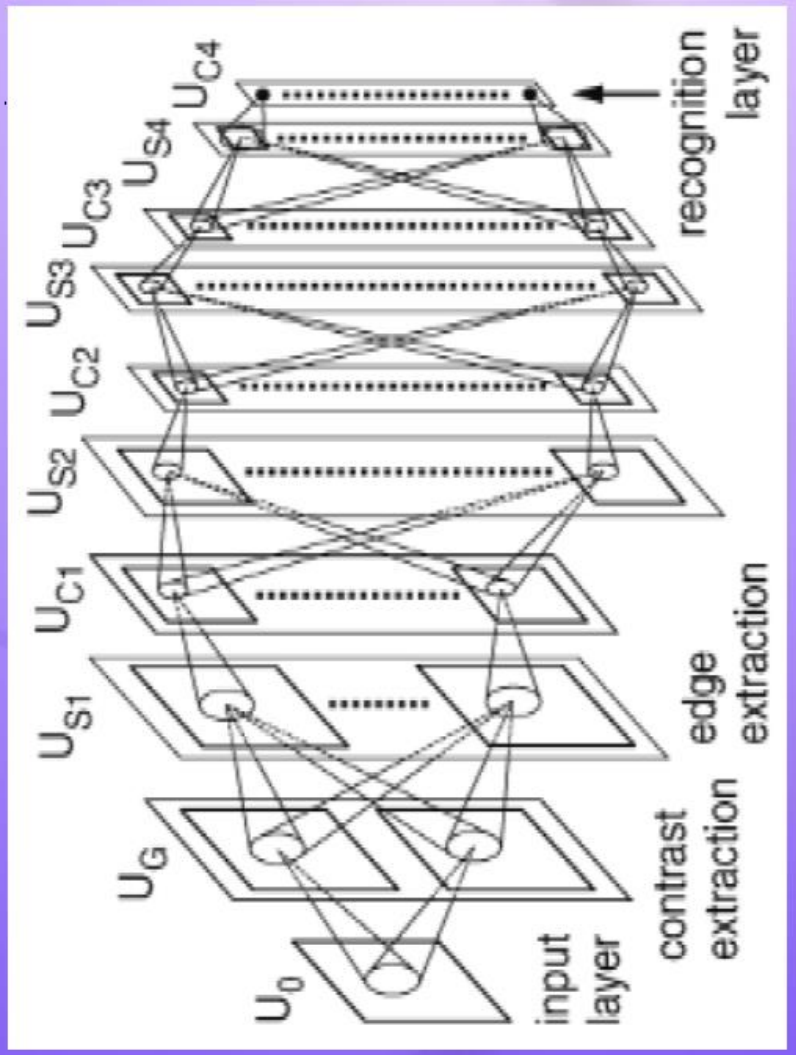
- the amount of shift/deformation-invariance is hardcoded into the structure, by how big a region each C-cell covers e.g.
- intuitively: only one training example is used for each digit; how could it possibly be learning what kinds of deformations to allow?
- empirically demonstrated by Barnard and Casasent, 1990

References

- Barnard, E., and Casasent, D. "Shift Invariance and the Neocognitron." Neural Networks 3, no. 4 (1990): 403–410.
- Fukushima, K., and Miyake, S. "Neocognitron: A New Algorithm for Pattern Recognition Tolerant of Deformations and Shifts in Position." Pattern Recognition 15, no. 6 (1982): 455–469.
- figures from http://www.kiv.zcu.cz/studies/predmety/uir/NS/Neocognitron/en/index.html

How the Brain does IT

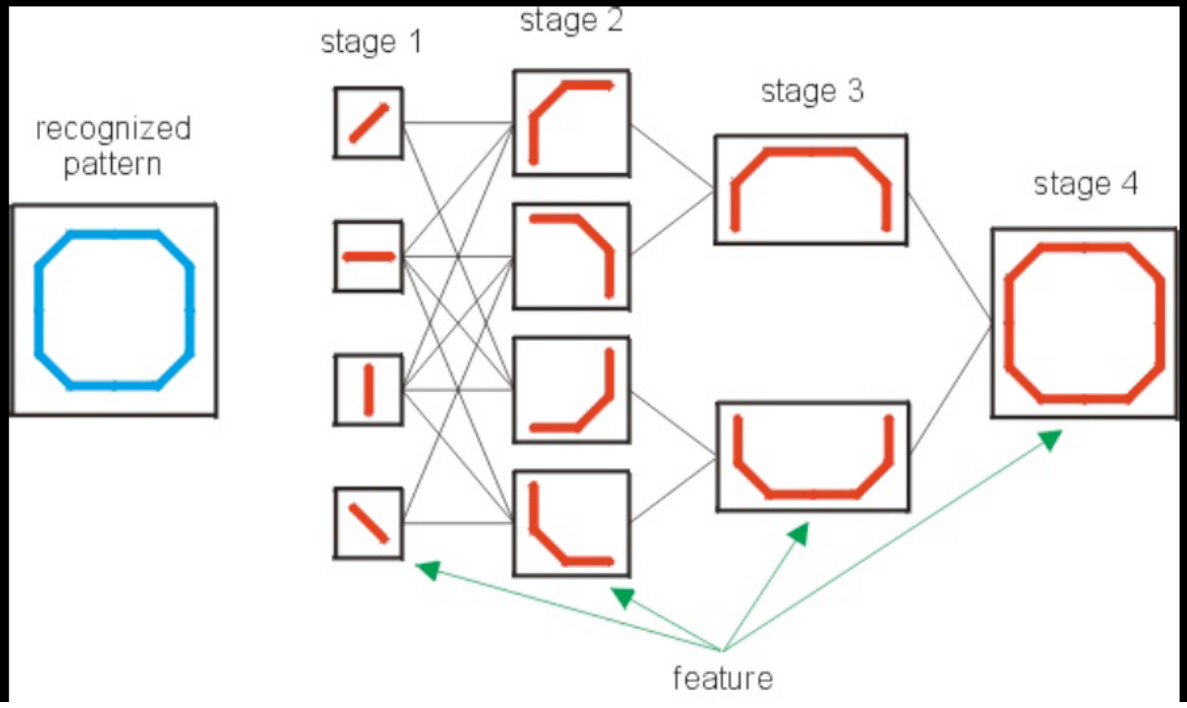
- Fukushima (Neocognitron, 1980) got it right on first try!
- Hierarchical extraction of increasing:
  - **spatial invariance** (S)
  - **featural complexity** (C)
- Most biological models since then are variations on the same theme (Mozer, Mel, Riesenhuber & Poggio, Deco & Rolls, etc)



http://www.youtube.com/watch?v=Qil4kmvm2Sw
http://www.youtube.com/watch?v=oVYCjL54qoY

Design - High Level

- multiple (usually about 3 hidden) *layers*
- Successive layers recognize higher-level patterns



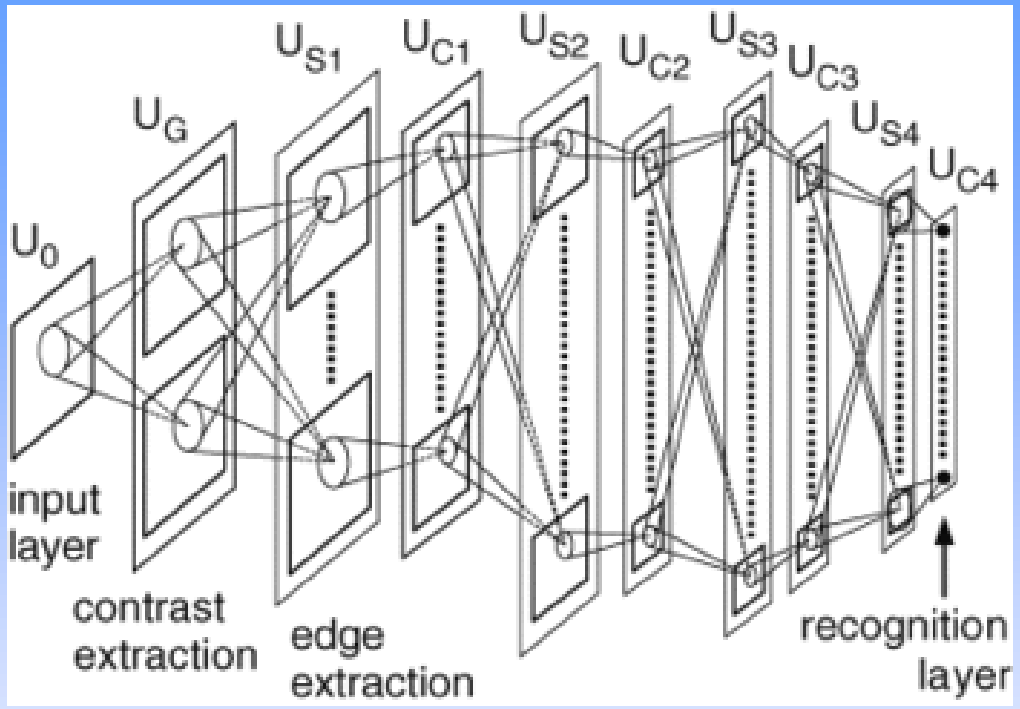
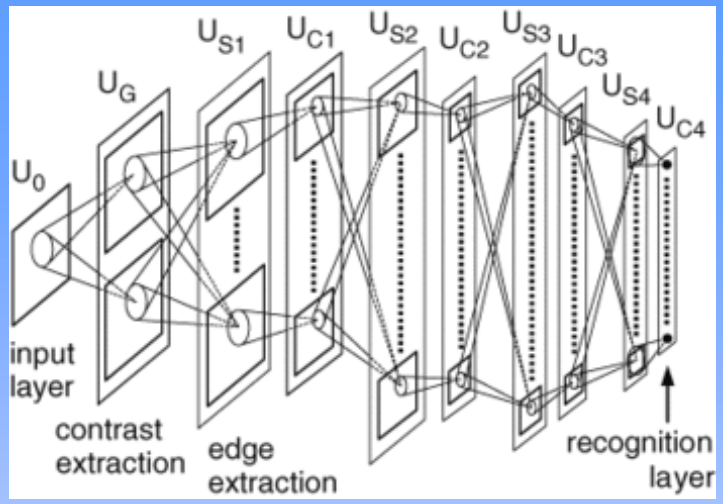
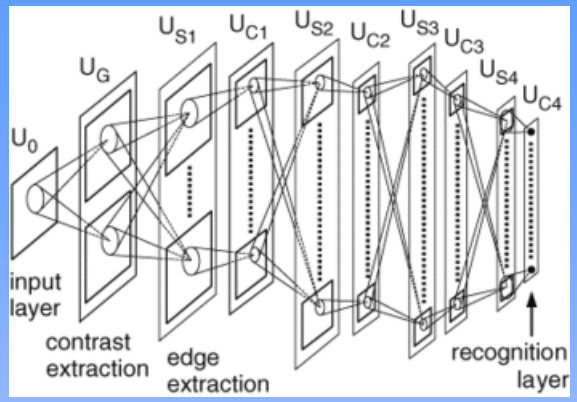


Figure 1: A typical architecture of the neocognitron

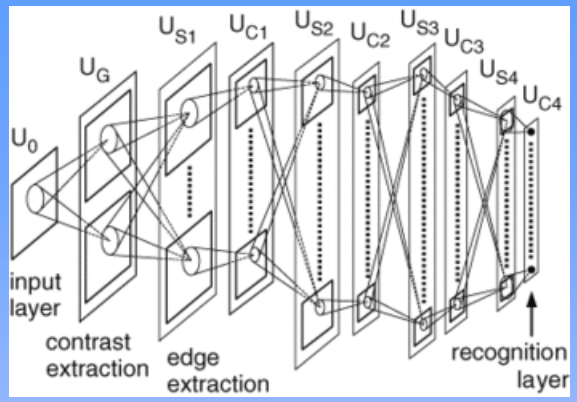
• The lowest stage is the input layer consisting of two-dimensional array of cells, which correspond to photoreceptors of the **retina**.
• There are retinotopically ordered connections between cells of adjoining layers. Each cell receives input connections that lead from cells situated in a limited area on the preceding layer.
• Layers of "S-cells" and "C-cells" are arranged alternately in the hierarchical network. (In the network shown in Fig.1, a contrast-extracting layer is inserted between the input layer and the S-cell layer of the first stage).



• **S-cells** work as feature-extracting cells. They resemble simple cells of the primary visual cortex in their response. Their input connections are variable and are modified through learning.
• After having finished learning, each S-cell come to respond selectively to a particular feature presented in its receptive field.
• The features extracted by S-cells are determined during the learning process.
• Generally speaking, *local* features, such as edges or lines in particular orientations, are extracted in lower stages.
• More *global* features, such as parts of learning patterns, are extracted in higher stages.



• **C-cells**, which resembles complex cells in the visual cortex, are inserted in the network to allow for positional errors in the features of the stimulus.
• The input connections of C-cells, which come from S-cells of the preceding layer, are fixed and invariable.
• Each C-cell receives excitatory input connections from a group of S-cells that extract the same feature, but from slightly different positions.
• The C-cell responds if at least one of these S-cells yield an output. Even if the stimulus feature shifts in position and another S-cell comes to respond instead of the first one, the same C-cell keeps responding.
• Thus, the C-cell's response is less sensitive to shift in position of the input pattern.
• We can also express that C-cells make a blurring operation, because the response of a layer of S-cells is spatially blurred in the response of the succeeding layer of C-cells.



• Each layer of S-cells or C-cells is divided into sub-layers, called "cell-planes", according to the features to which the cells responds.
• The cells in each cell-plane are arranged in a two-dimensional array.
• A cell-plane is a group of cells that are arranged retinotopically and share the same set of input connections.
• In other words, the connections to a cell-plane have a translational symmetry.
• As a result, all the cells in a cell-plane have receptive fields of an identical characteristic, but the locations of the receptive fields differ from cell to cell.

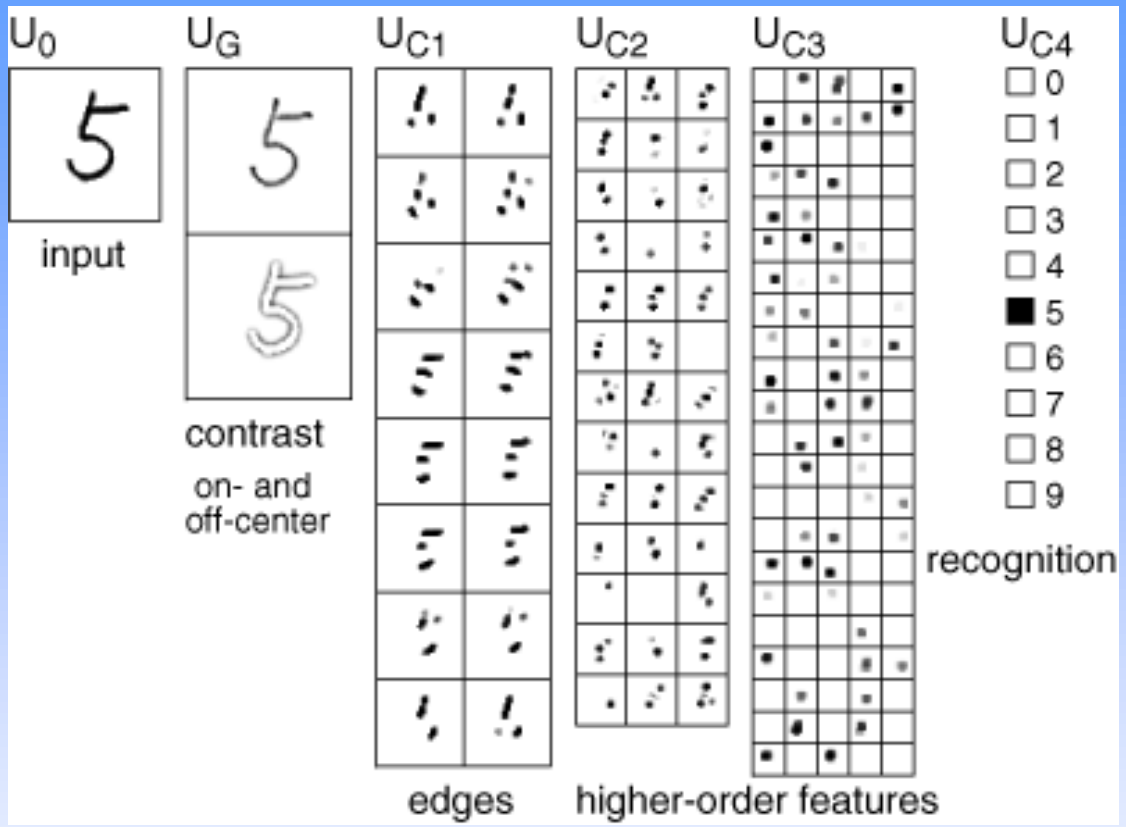


Figure 4: An example of the response of a **neocognitron** that has been trained to recognize handwritten digits. The input pattern is recognized correctly as '5'

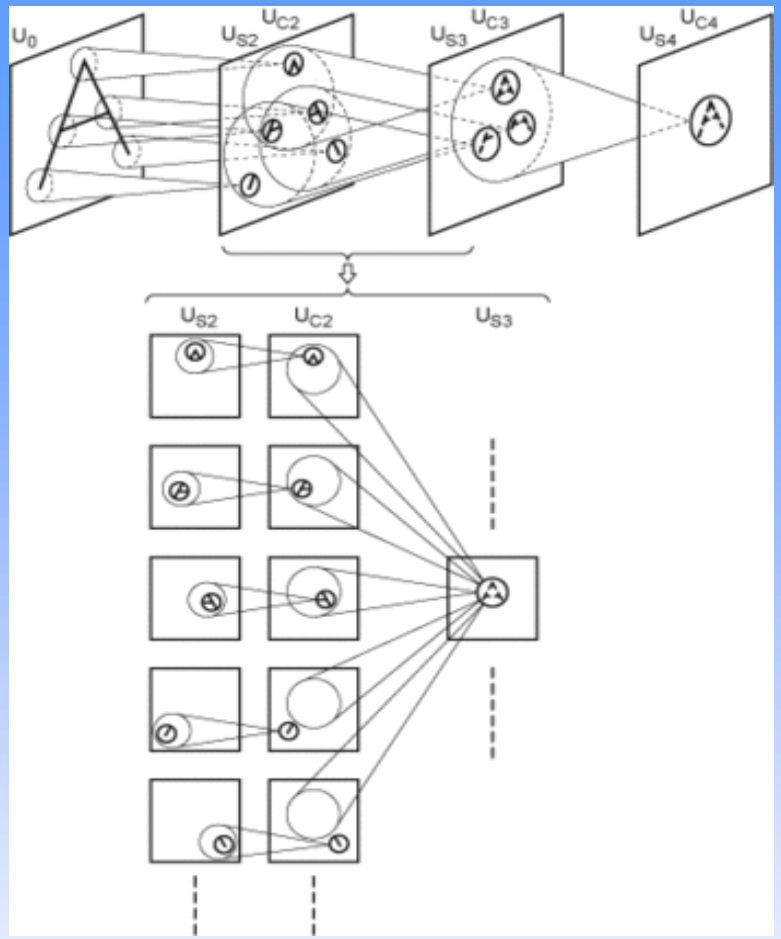


Figure 2: The process of pattern recognition in the **neocognitron**. The lower half of the figure is an enlarged illustration of a part of the network

• In the whole network, with its alternate layers of S-cells and C-cells, the process of feature-extraction by S-cells and toleration of positional shift by C-cells is repeated.

• During this process, local features extracted in lower stages are gradually integrated into more *global* features, as illustrated in Fig.2

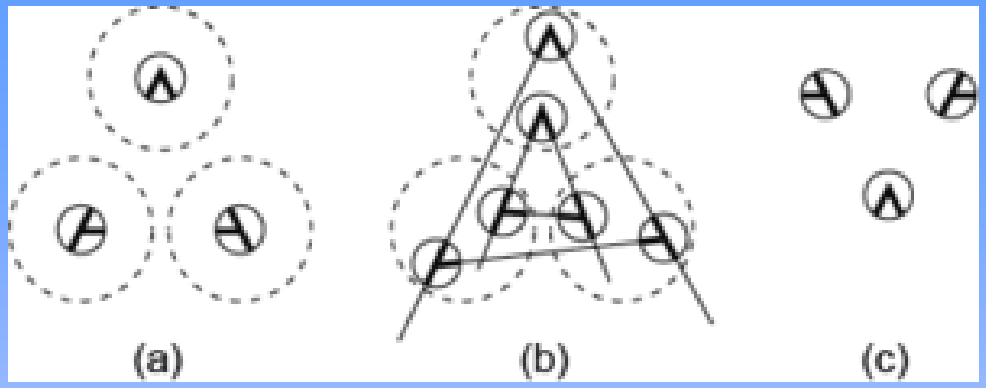
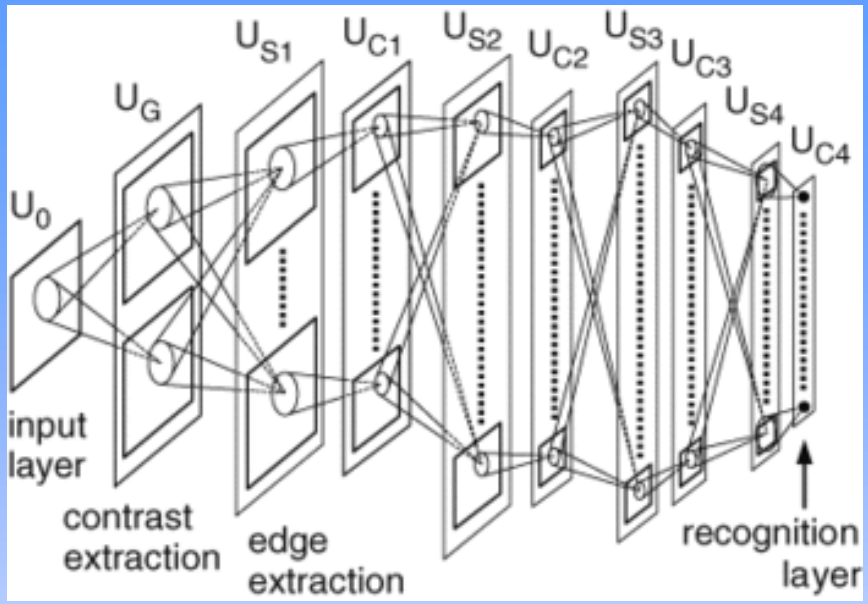


Figure 3: The principle for recognizing deformed patterns.

Since small amounts of positional errors of local features are absorbed by the blurring operation by C-cells, an S-cell in a higher stage comes to respond robustly to a specific feature even if the feature is slightly deformed or shifted.

• Fig.3 illustrates this situation. Let an S-cell in an intermediate stage of the network have already been trained to extract a global feature consisting of three local features of a training pattern ‘A’ as illustrated in Fig.3(a).
• The cell tolerates a positional error of each local feature if the deviation falls within the dotted circle.
• The S-cell responds to any of the deformed patterns shown in Fig.3(b).
• The toleration of positional errors should not be too large at this stage. If large errors are tolerated at any one step, the network may come to respond erroneously, such as by recognizing a stimulus like Fig.3(c) as an 'A' pattern.
• Thus, tolerating positional error a little at a time at each stage, rather than all in one step, plays an important role in endowing the network with the ability to recognize even distorted patterns.



• The C-cells in the highest stage work as recognition cells, which indicate the result of the pattern recognition.

• Each C-cell of the recognition layer at the highest stage integrates all the information of the input pattern, and responds only to one specific pattern.

• Since errors in the relative position of local features are tolerated in the process of extracting and integrating features, the same C-cell responds in the recognition layer at the highest stage, even if the input pattern is deformed, changed in size, or shifted in position.

• In other words, after having finished learning, the **neocognitron** can recognize input patterns robustly, with little effect from deformation, change in size, or shift in position.

• The neocognitron can be trained to recognize patterns through learning.
• Only S-cells in the network have their input connections modified through learning.
• Various training methods, including unsupervised learning and supervised learning, have been proposed so far.
• This section introduces a process of unsupervised learning.
• In the case of unsupervised learning, the self-organization of the network is performed using two principles.
• The first principle is a kind of *winner-take-all rule*: among the cells situated in a certain small area, which is called a *hypercolumn*, only the one responding most strongly becomes the *winner*.
• The winner has its input connections strengthened. The amount of strengthening of each input connection to the winner is proportional to the intensity of the response of the cell from which the relevant connection leads.

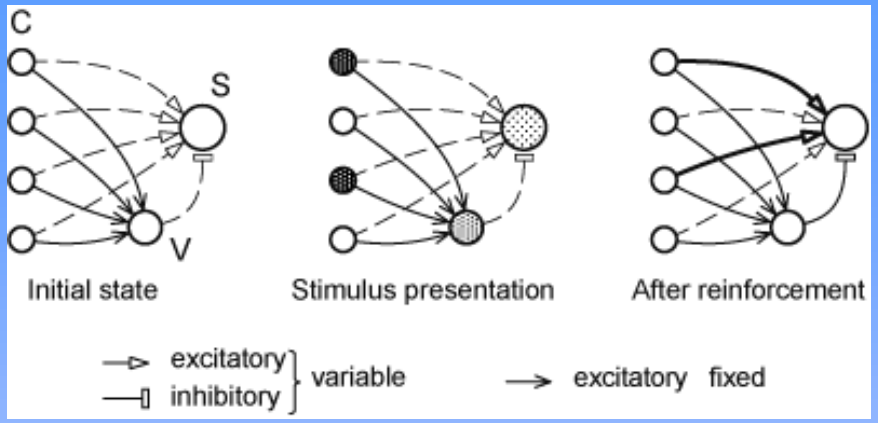
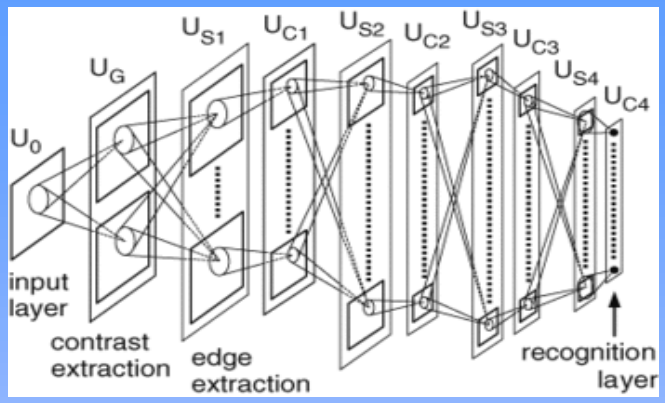
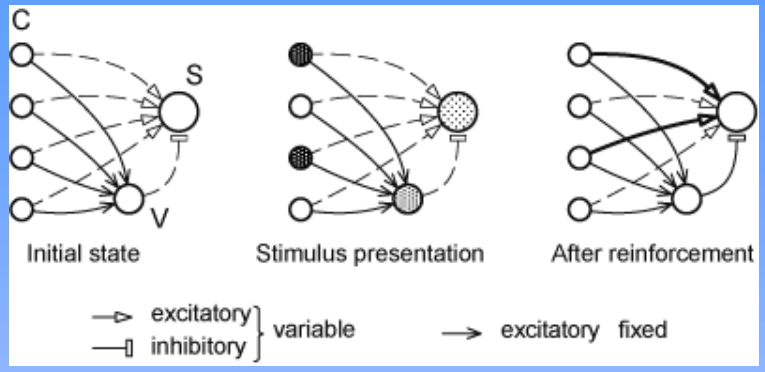


Figure 5: Connections converging to an S-cell in the learning phase

• S-cell receives variable excitatory connections from a group of C-cells of the preceding stage as illustrated in Fig.5.
• Each S-cell is accompanied with an inhibitory cell, called a V-cell.
• The S-cell also receives a variable inhibitory connection from the V-cell.
• The V-cell receives fixed excitatory connections from the same group of C-cells as does the S-cell, and always responds with the average intensity of the output of the C-cells.
• The initial strength of the variable connections is very weak and nearly zero.
• Suppose the S-cell responds most strongly among the S-cells in its vicinity when a training stimulus is presented.
• According to the winner-take-all rule described above, variable connections leading from activated C-cells are strengthened.
• The variable excitatory connections to the S-cell grow into a *template* that exactly matches the spatial distribution of the response of the cells in the preceding layer.
• The inhibitory variable connection from the V-cell is also strengthened at the same time to the average strength of the excitatory connections.



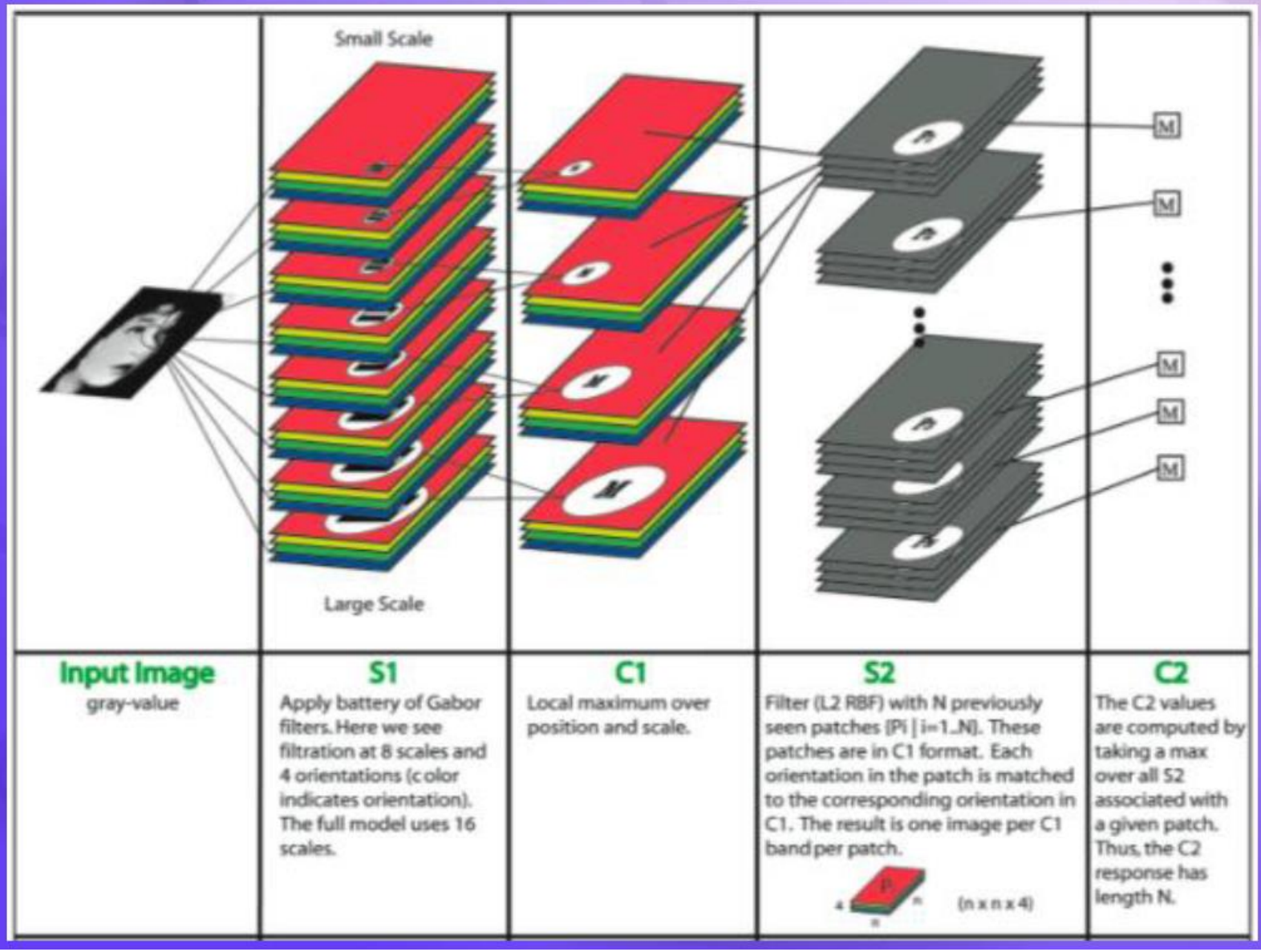


• After the learning, the S-cell acquires the ability to extract a feature of the stimulus presented during the learning period.
• Through the excitatory connections, the S-cell receives signals indicating the existence of the relevant feature to be extracted.
• If an irrelevant feature is presented, the inhibitory signal from the V-cell becomes stronger than the direct excitatory signals from the **C-cells**, and the response of the **S-cell** is suppressed.
• Once an **S-cell** is thus selected and has learned to respond to a feature, the cell usually loses its responsiveness to other features.
• When a different feature is presented, a different cell usually yields the maximum output and learns the second feature. Thus, a "division of labor" among the cells occurs automatically.
• The second principle for the learning is introduced in order that the connections being strengthened always preserving translational symmetry, or the condition of shared connections.
• The maximum-output cell not only grows by itself, but also controls the growth of neighboring cells, working, so to speak, like a seed in crystal growth.
• To be more specific, all of the other S-cells in the cell-plane, from which the “seed cell” is selected, follow the seed cell, and have their input connections strengthened by having the same spatial distribution as those of the seed cell.

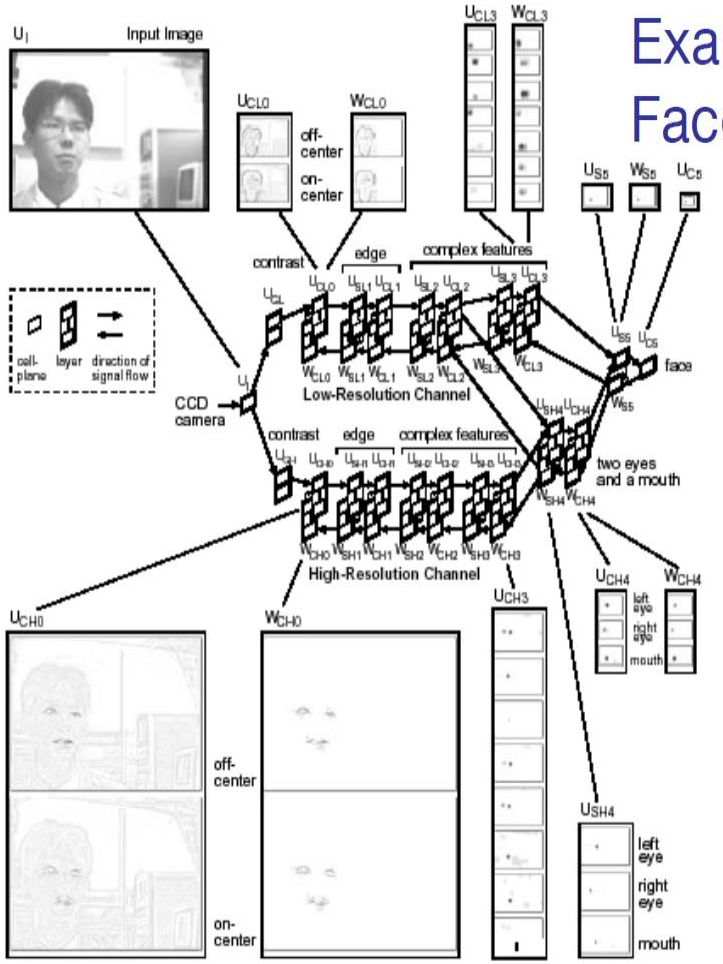
http://www.youtube.com/watch?v=Qil4kmvm2Sw

http://www.youtube.com/watch?v=oVYCjL54qoY

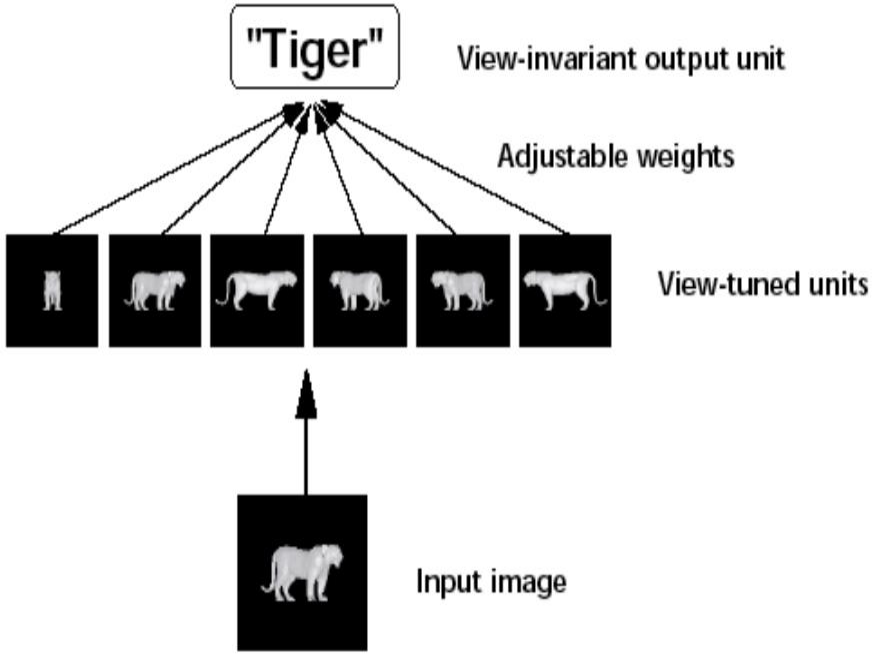
Riesenhuber, Poggio et al Model
(successful current version of Neocognitron)

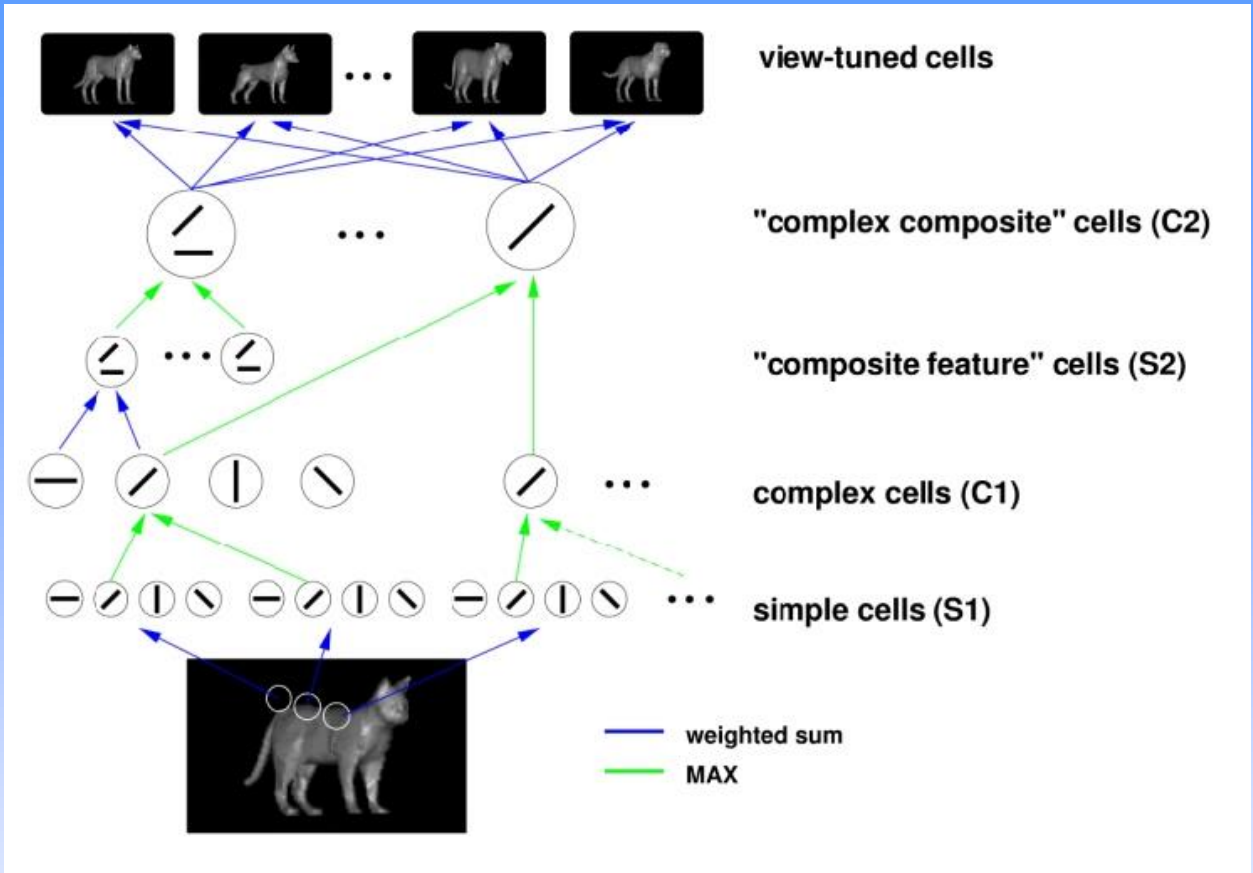


Example: Face Recognition



Viewpoint-dependent recognition





• In the HMAX model of object recognition in the ventral visual stream of primates, input images (we used 128 x 128 or 160 x 160 greyscale pixel images) are densely sampled by arrays of two-dimensional Gaussian filters, the so-called S1 units (second derivative of Gaussian, orientations 0°, 45°, 90°, and 135°, sizes from 7 x 7 to 29 x 29 pixels in two-pixel steps) sensitive to bars of different orientations, thus roughly resembling properties of simple cells in striate cortex.

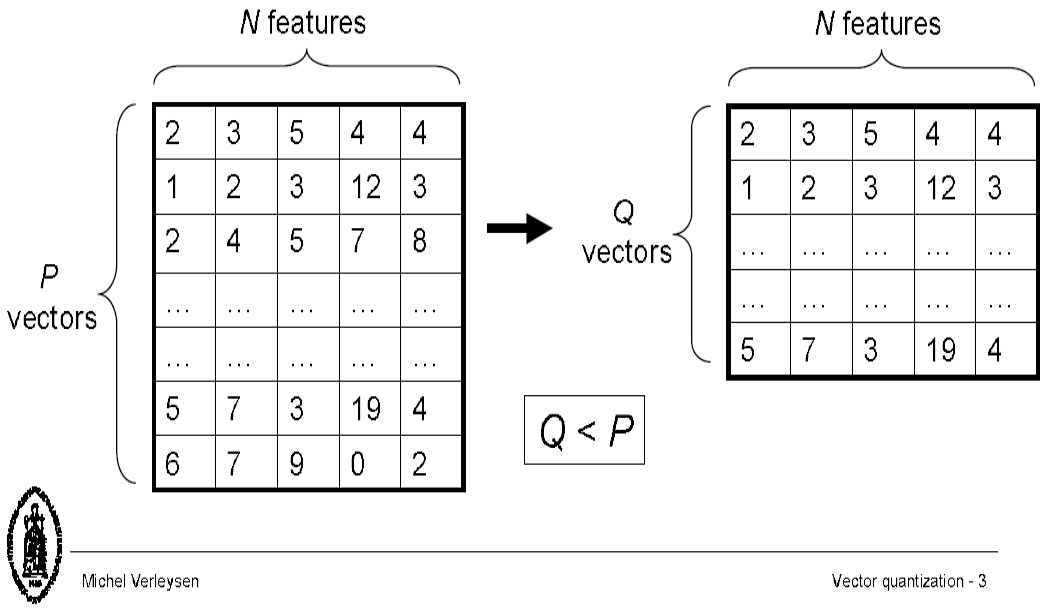
• At each pixel of the input image, filters of each size and orientation are centered. The filters are sum-normalized to zero and square-normalized to 1, and the result of the convolution of an image patch with a filter is divided by the power (sum of squares) of the image patch. This yields an S1 activity between −1 and 1.

Convolutional neural networks

https://www.youtube.com/watch?v=bEUX_56Lojc

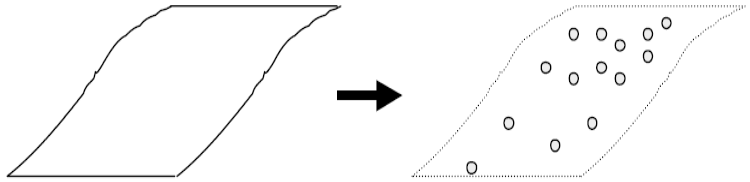
Aim of vector quantization

- To reduce the *size* of a database

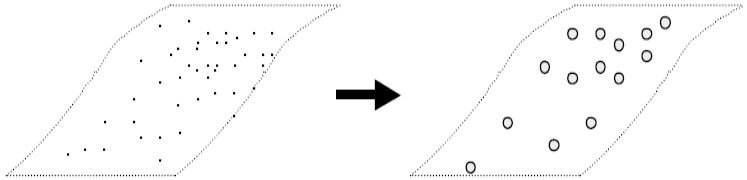


Principle of vector quantization

- To project a continuous input space on a discrete output space, while minimizing the *loss of information*



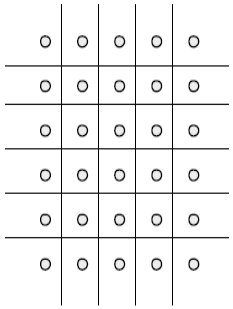
Q vectors

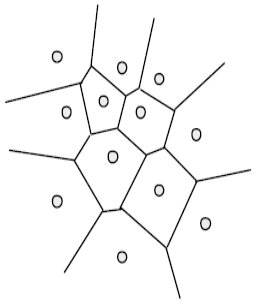


P points *Q* vectors

Principle of vector quantization

- To define zones in the space, the set of points contained in each zone being projected on a representative vector (*centroid*)
- Example: 2-dimensional spaces





What is a vector quantizer ?

- Vector quantizer =

1. A codebook (set of centroids, or codewords)

$$\boldsymbol{m} = \left\{ \boldsymbol{y}^j, 1 \le j \le Q \right\}$$

2. A quantization function q:

$$q \equiv \boldsymbol{x}^i \rightarrow q\left(\boldsymbol{x}^i\right) = \boldsymbol{y}^j$$

Usually: q is defined by the *nearest neighbour* rule (according to some distance measure)

Distance measures

- Least square error

$$d_2\left(\boldsymbol{x}^i, \boldsymbol{y}^j\right) = \sqrt{\sum_{k=1}^{D}\left(x_k^i - y_k^j\right)^2} = \left\|\boldsymbol{x}^i - \boldsymbol{y}^j\right\| = \left(\boldsymbol{x}^i - \boldsymbol{y}^j\right)^T\left(\boldsymbol{x}^i - \boldsymbol{y}^j\right)$$

- r-norm error

$$d_r\left(\boldsymbol{x}^i, \boldsymbol{y}^j\right) = \left(\sum_{k=1}^{D}\left(x_k^i - y_k^j\right)^r\right)^{1/r}$$

Distance measures

- Weighted least square error

$$d_W(\boldsymbol{x}^i, \boldsymbol{y}^j) = (\boldsymbol{x}^i - \boldsymbol{y}^j)^T \boldsymbol{W} (\boldsymbol{x}^i - \boldsymbol{y}^j)$$

- if $\boldsymbol{W} = \boldsymbol{I}$ $\rightarrow$ least square error

- if $\boldsymbol{W} = \boldsymbol{\Gamma}^{-1}$ where Γ is the covariance matrix of the inputs:

$$\Gamma = E\left[(\boldsymbol{x} - \overline{\boldsymbol{x}})(\boldsymbol{x} - \overline{\boldsymbol{x}})^T\right] \quad \text{and} \quad \overline{\boldsymbol{x}} = E[\boldsymbol{x}]$$

$\rightarrow$ Mahalanobis distance

- if $\boldsymbol{W}$ is symmetrical: $\boldsymbol{W} = \boldsymbol{P}^T \boldsymbol{P}$

$$\tilde{\boldsymbol{x}}^i = \boldsymbol{P}\boldsymbol{x}^i, \ \tilde{\boldsymbol{y}}^j = \boldsymbol{P}\boldsymbol{y}^j \quad \rightarrow \quad d_W(\boldsymbol{x}^i, \boldsymbol{y}^j) = d_W(\tilde{\boldsymbol{x}}^i, \tilde{\boldsymbol{y}}^j)$$

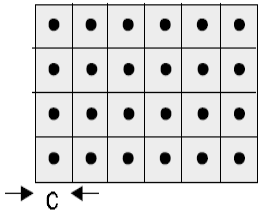
Vector ↔ scalar quantization

- Shannon: *a vector quantizer always gives better results than the product of scalar quantizers, even if the probability densities are independent*

- Example #2:
  uniform 2-D distribution

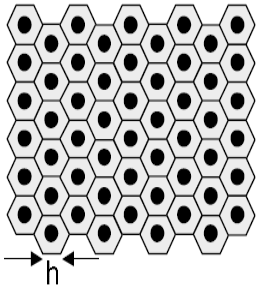
scalar quantization

$$E_{SQ}\left[d(\boldsymbol{x}-\boldsymbol{y})^2\right]=\frac{c^4}{16}$$



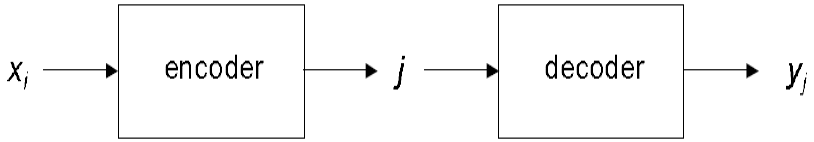
vector quantization

$$E_{VQ}\left[d(\boldsymbol{x}-\boldsymbol{y})^2\right]=1.08h^4$$

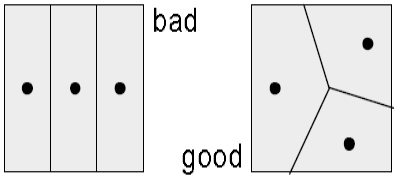


with equal surfaces: $E_{VQ} = 0.962\ E_{SQ}$

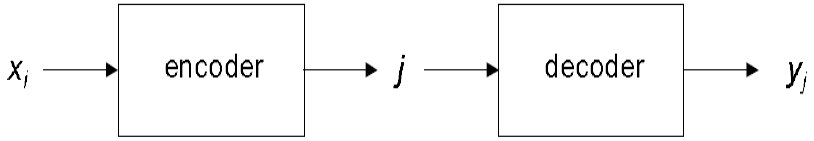
Lloyd's principle



- 3 properties:
  1. The first one gives the best encoder, once the decoder is known
  2. The second one gives the best decoder, once the encoder is known
  3. There is no point on the borders between Voronoï regions (probability = 0)
- Optimal quantizer: properties are necessary, but not sufficient:



Lloyd: property #1

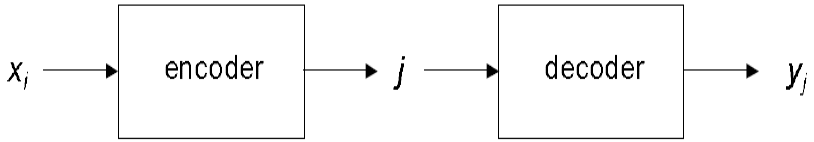


- For a given decoder β, the best encoder is given by:

$$\alpha(\boldsymbol{x}^i) = \underset{j}{\operatorname{argmin}} \left[d(\boldsymbol{x}^i, \boldsymbol{y}^j) \right] \quad \text{where } \boldsymbol{y}^j = \beta(j)$$

nearest-neighbor rule !

Lloyd: property #2



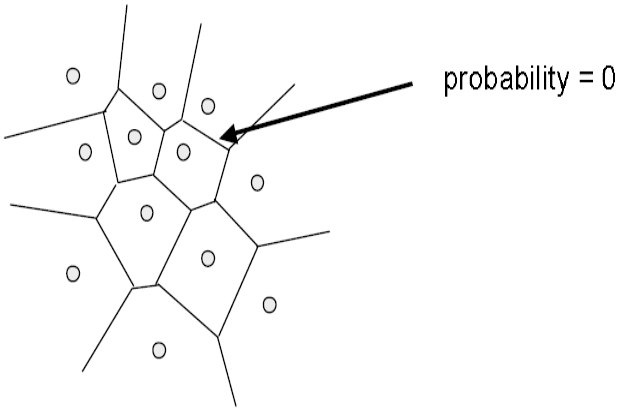
- For a given encoder α, the best decoder is given by:

$$\beta(j) = \underset{y^j}{\operatorname{argmin}} \left[E\left[d\left(x^i, y^j\right) \middle| \alpha\left(x^i\right) = j \right] \right]$$

center-of-gravity rule!

Lloyd: property #3

- The probability to find a point xi on a border (between Voronoï regions) is zero !

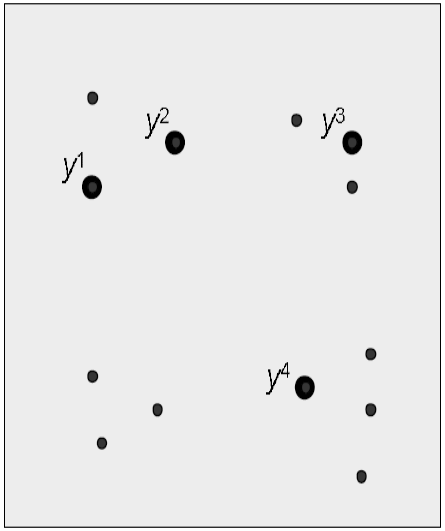


Lloyd's algorithm

1. Choice of an initial codebook.
2. All points x^i are encoded; E_{VQ} is evaluated.
3. If E_{VQ} is small enough, then stop.
4. All centroids y^j are replaced by the center-of-gravity of the data x^i associated to y^j in step 2.
5. Back to step 2.

Lloyd: example

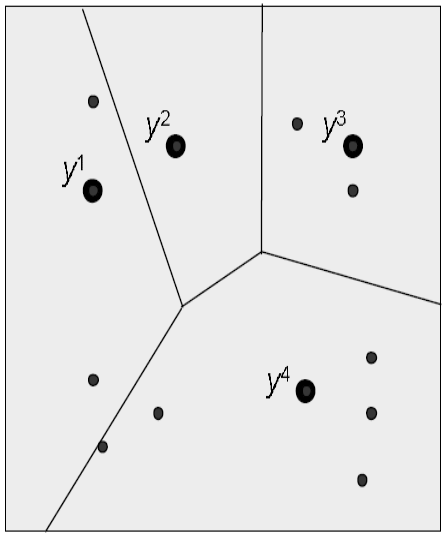
1. Initialization of the codeboook



- x^i: data
- y^j: centroids

Lloyd: example

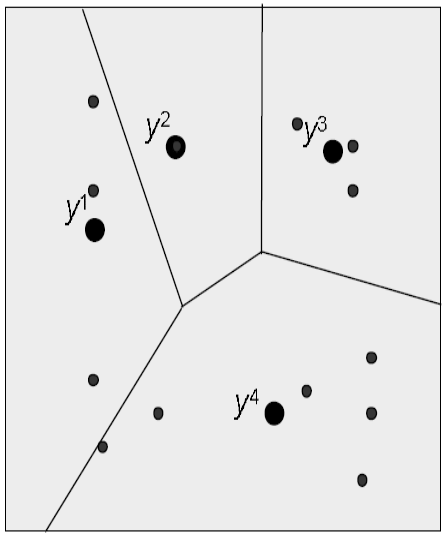
2. Encoding (nearest-neighbor)



- x^i: data
- y^j: centroids

Lloyd: example

4. Decoding
   (center-of-gravity)

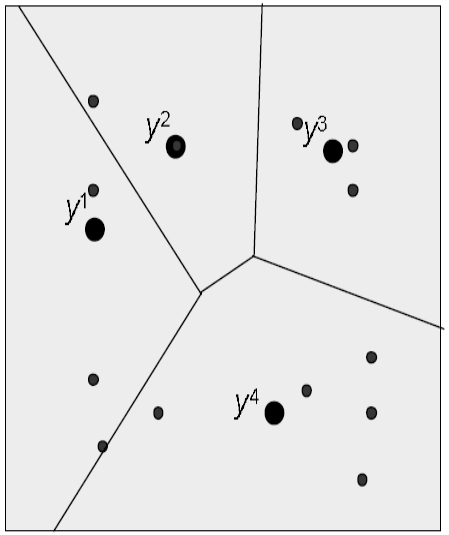


- x^i: data
- y^j: centroids

Lloyd: example

2. Encoding (nearest-neighbor)

→ new borders

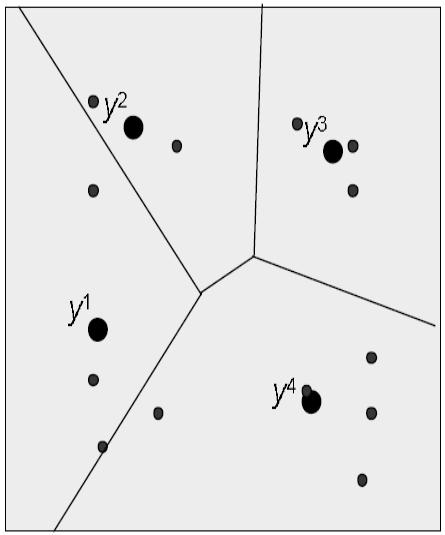


- x^i: data
- y^j: centroids

Lloyd: example

4. Decoding (center-of-gravity)

   → new positions of centroids

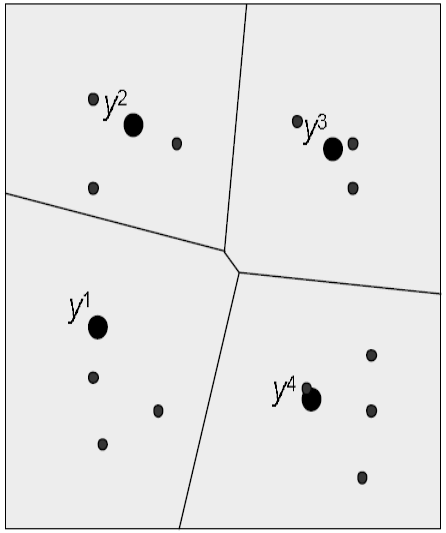


- x^i: data
- y^j: centroids

Lloyd: example

2. Encoding (nearest-neighbor)

   → new borders

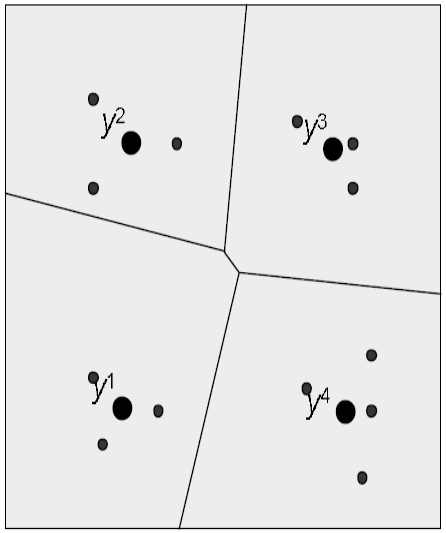


- x^i: data
- y^j: centroids

Lloyd: example

4. Decoding (center-of-gravity)

   → new positions of centroids

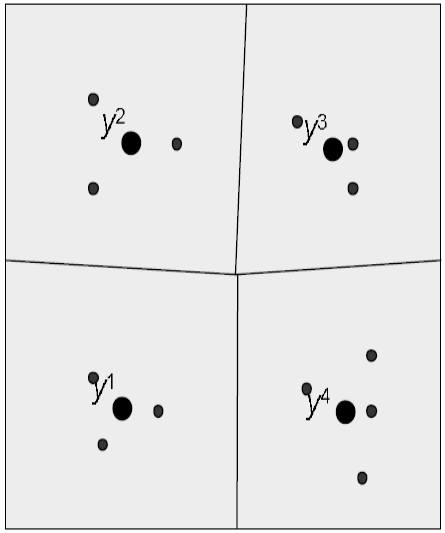


- x^i: data
- y^j: centroids

Lloyd: example

2. Encoding (nearest-neighbor)

→ final borders (convergence)



- x^i: data
- y^j: centroids

Lloyd's algorithm: the names

- Lloyd's algorithm
- Generalized Lloyd's algorithm
- Linde-Buzzo-Gray (LBG) algorithm
- K-means
- ISODATA
- ...

All based on the same principle!

Lloyd's algorithm: properties

- The codebook is modified only after the presentation of the whole dataset
- The mean square error (E_{VQ}) decreases at each iteration
- The risk of getting trapped in local minima is high
- The final quantizer depends on the initial one

How to initialize Lloyd's algorithm ?

1. randomly in the input space
2. the Q first data points x^i
3. Q randomly chosen data points x^i
4. 'Product codes': the product of scalar quantizers
5. growing initial set:
   - a first centroid y^1 is randomly chosen (in the data set)
   - a second centroid y^2 is randomly chosen (in the data set); if $d(y^1,y^2) > threshold$, y^2 is kept
   - a third centroid y^3 is randomly chosen (in the data set); if $d(y^1,y^3) > threshold$ AND $d(y^2,y^3) > threshold$, y^3 is kept
   - …

How to initialize Lloyd's algorithm ?

6. 'pairwise nearest neighbor':
   - a first codebook is built with all data points x^i
   - the two centroids y^i nearest one from another are merged (center-of-gravity)
   - …

   Variant:
   - the increase of distortion (E_{VQ}) is evaluated for the merge of each pair of centroids y^i; the pair gicing the lowest increase is merged.

How to initialize Lloyd's algorithm ?

7. 'Splitting'
   - a first centroid y^1 is randomly chosen (in the data set)
   - a second centroid $y^1+\varepsilon$ is created; Lloyd's algorithm is applied to the new codebook
   - two new centroids are created by perturbing the two existing ones; Lloyd's algorithm is applied to this 4-centroids codebook
   - …

Vector quantization: « neural » algorithms

- Principle: the codebook is (partly or fully) modified at each presentation of one data vector $\boldsymbol{x}^i$
- Advantages:
  - simplicity
  - adaptive algorithm (with varying data)
  - possible parallelisation
  - speed ?
  - avoids local minima ?

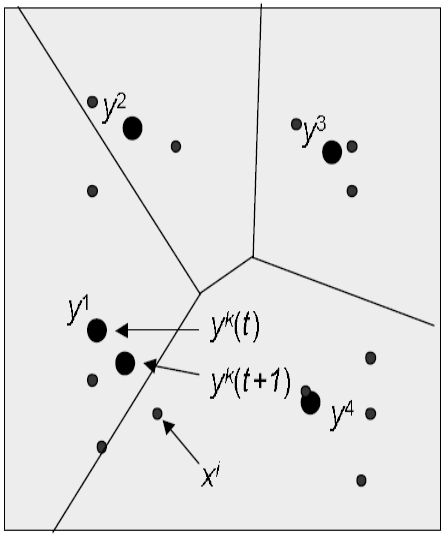
Competitive learning

- Algorithm:
  - For each input vector $\boldsymbol{x}^i$, the « winner » is selected:

$$d(\boldsymbol{x}^i, \boldsymbol{y}^k) \leq d(\boldsymbol{x}^i, \boldsymbol{y}^j), \quad 1 \leq j,k \leq Q$$

  - Adaptation rule: the winner is moved towards the input vector:

$$\boldsymbol{y}^k(t+1) = \boldsymbol{y}^k(t) + \alpha(\boldsymbol{x}^i - \boldsymbol{y}^k)$$



- x^i: data
- y^j: centroids

Competitive learning

- Adaptation rule: stochastic gradient descent on

$$E = \int \left(\boldsymbol{x} - \boldsymbol{y}^{j(x)} \right)^2 p(\boldsymbol{x}) d\boldsymbol{x}$$

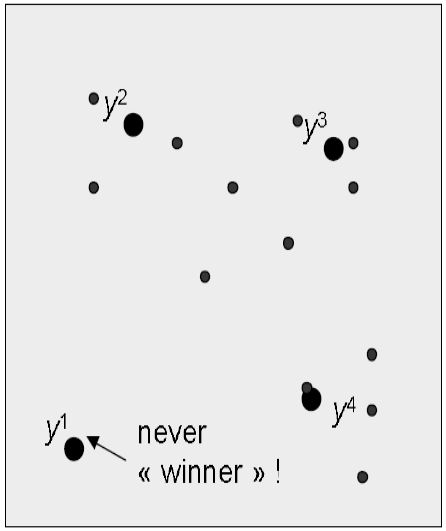
→ convergence to (local) minimum

- Robbins-Monro conditions on α:

$$\sum_{t=0}^{\infty} \alpha(t) = \infty \quad \text{and} \quad \sum_{t=0}^{\infty} \alpha^2(t) < \infty$$

- Local minima !

- Some centroids may be « lost »!



Frequency sensitive learning

- Competitive learning; some centroids may be lost during learning

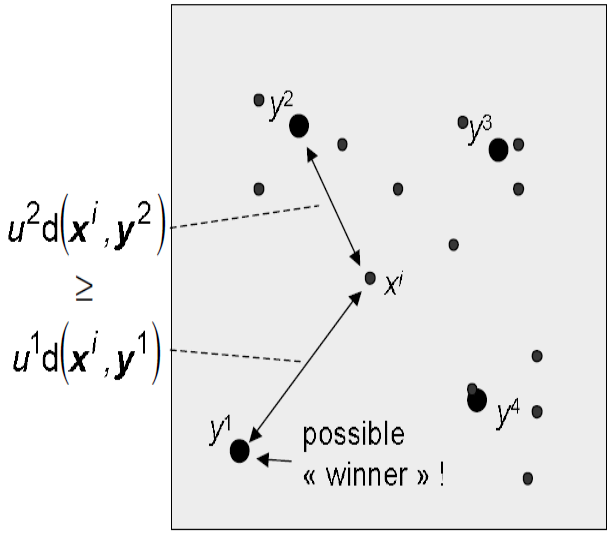
  → centroids often chosen (as winners) are penalized!

- Choice of winner is replaced by

$$u^k \mathrm{d}\left(\boldsymbol{x}^i, \boldsymbol{y}^k\right) \le u^j \mathrm{d}\left(\boldsymbol{x}^i, \boldsymbol{y}^j\right), \quad 1 \le j,k \le Q$$

  where u^j, u^k are incremented each time they are chosen as winner (starting at 1)

Frequency sensitive learning



(with u^1=1 and u^2=3 for example)