# Self-Organizing Maps

Kohonen Nets
Feature Maps

## Teuovo Kohonen

Dr. Eng., Professor of the Academy of Finland;
Head, Neural Networks Research Centre,
Helsinki University of Technology, Finland

## Self-Organizing Maps (SOMs)

Competitve: update weight vectors in a ***neighborhood*** of the winning neuron.

**Kohonen Rule:** $_i\mathbf{w}(q) = {}_i\mathbf{w}(q-1) + \alpha(\mathbf{p}(q) - {}_i\mathbf{w}(q-1))$ $\quad i \in N_i(d)$

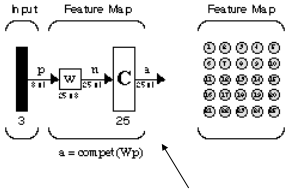
$$_i\mathbf{w}(q) = (1-\alpha){}_i\mathbf{w}(q-1) + \alpha\mathbf{p}(q)$$

$$N_i(d) = \{j, d_{i,j} \le d\}$$

$N_{13}(1) = \{8, 12, 13, 14, 18\}$ ← Two possible neighborhoods of neuron 13

$N_{13}(2) = \{3, 7, 8, 9, 11, 12, 13, 14, 15, 17, 18, 19, 23\}$

## SOM as a matlab Competitive Network




Does not show neighborhood computation. Neighborhood starts large, gradually decreases, along with learning rate.

## Maps in Neurobiology

- Related neural functions ***map*** onto identifiable regions of the brain
  - retinotopic map: vision, *superior colliculus*
  - phonotopic map: hearing, auditory cortex
  - somatotopic map: touch, somatosensory cortex

## Human Brain

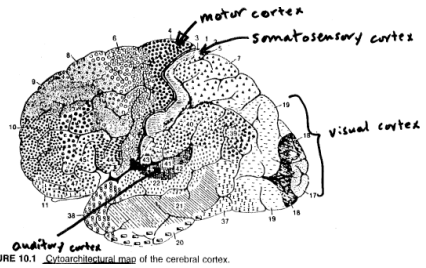
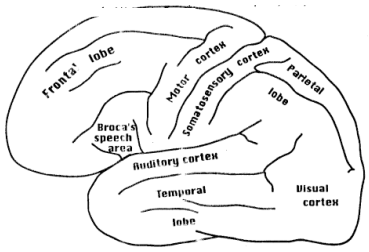



FIGURE 10.1 Cytoarchitectural map of the cerebral cortex.

## Regions of your Brain




## Desired Properties of Maps

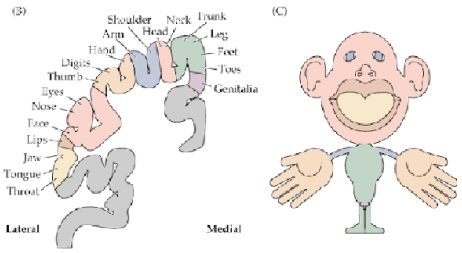
- **Approximation of input space**: generally a many-one mapping of input space into weight space
- **Topology-preserving**: points close together in input space should map to points close together in weight space.
- **Density-preserving**: regions of similar density should map to regions of proportional density.

## Somatotopic Map Illustration: The "Homunculus"

Cartoon map of the relationship between body surfaces and the regions of the brain that control them

(somewhat different from the original "little person inside" meaning).

## Another Depiction of the Homunculus




## Project: Map your own homunculus:

http://www.woodrow.org/teachers/biology/institutes/1991/homunculus.html

**Procedure II: Experiment.**

Your team should consist of an experimental subject (blindfolded) and an experimenter/recorder. Starting from the head and working down to the feet, measure the distance between touch receptor fields in specific parts of the right hand side of the body using the following method:

1. Spread the pins apart and press the points lightly on the skin of the subject. The subject should detect two points of contact. If he feels only one, repeat the process , moving the pins farther apart.
2. Move the 2 pins closer together, 0.5 cm at a time, until the subject will no longer be able to distinguish 2 separate pins. Measure this distance in cm. At this point, both pins are within the same receptive field of one sensory receptor, and so the 2 points cannot be identified separately.
3. Repeat this measurement 2 times in the same general area . Average these measurements, and record the mean in the data table.
4. Measure as many parts of the body as possible. See the data table for suggested areas.

**Procedure III: Calculations.**

1. The number recorded for each body part represents the distance between each sensory receptor field, so the distance measured is inversely proportional to the cortical area dedicated to that body part. That is: the closer the receptor fields, the larger the area on the cortex. To calculate the inverses, divide each mean into the number 1. For example: Distance = 0.25 cm. 1/0.25 cm divides out to be 4.0. Calculate the inverses for each body part and record on the data sheet.
2. Draw a proportional picture of the homunculus on graph paper. If the inverse is 4.0, then the body part occupies 4 boxes on the graph paper, approximating the normal body shape. To enlarge the scale, for example, just multiply all values by the enlarging factor. For example, to make the drawing 5 times larger, multiply the inverses by 5.

## "Imposed Dimensionality" of SOM

- In a general SOM, an n-dimensional graph, n ≥ 1, can be ***overlaid*** with the neurons as nodes.
- The edges connecting the neurons that ***constrain the neighborhood*** *for updating* (rather than using Euclidean distance).
- Only nodes a specified ***graphical*** (not Euclidean) ***diameter*** away are in the neighborhood.

## Uses of "Imposed Dimensionality" in Data Analysis

- Data points may have an *unknown* dimensionality, e.g. the underlying phenomenon or process that has several dimensions of attributes (such as color dimensions, various size or rate dimensions, etc.)
- By training a network with an ***imposed*** dimensionality, the relationships among the data may become visualizable, especially if the imposed number of dimensions is $\leq$ the actual dimensions in the data.

## Uses of "Imposed Dimensionality" in Data Analysis

- In other words, training the map "organizes" or "sorts" the data according to similarity in each dimension.

## Demo Applets

- /cs/cs152/kohonen/ demo1, demo2, demo3
- Legend:
  - black = input data point (random over a 2-D region; 2-dimensional data)
  - red = neuron in winner neighborhood; learns by Kohonen rule
  - blue = other neuron
- Learning rate and neighborhood both decrease with time; demo speeds up over time.

## Competition Code

```
// compete finds the neuron with weights closest to the input.
// It sets winpoint to the indices of that neuron.

  public void compete(Input input)
    {
    // initialize min with an distance to an arbitrary neuron
    winpoint = 0;
    double min = neuron[winpoint].distance(input);

    // find the min distance among all neurons

    for( int point = 0; point < points; point++ )
      {
      double dist = neuron[point].distance(input);
      if( min > dist )
        {
        min = dist;              // update the min distance
        winpoint = point;
        }
      }
    }
```

## Competition Code

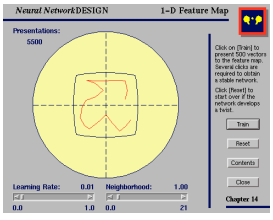
```
// learn updates all neurons within current neighborhood
// by applying the Kohonen learning rule against the current
// input.

  public void learn(Input input, double learningRate)
    {
    for( int point = 0; point < points; point++ )
      {
      if( inWinnersNeighborhood(point) )
        {
        neuron[point].learn(input, learningRate);
        }
      }
    }
```

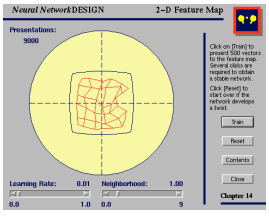
## Similar matlab demos

Same input spaces, different imposed dimensions

1D

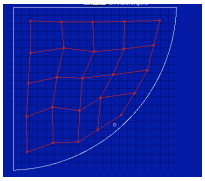



2D



## Related Demos on the Web

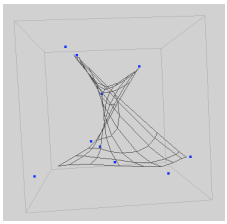
- Non-rectangular data distribution



http://www.patol.com/java/fill/index.html

## Related Demos on the Web

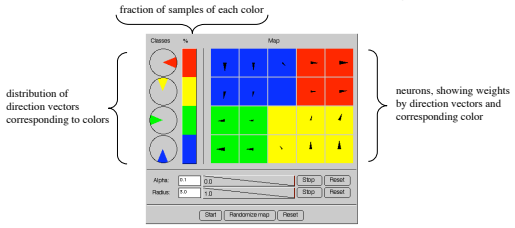
- rotating 3D data & neurons with 2D overlay



http://rfhs8012.fh-regensburg.de/~saj39122/jfroehl/diplom/e-index.html

## Related Demos on the Web

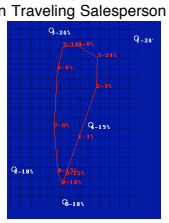
- 2D data & neurons -> 2D overlay




http://www.cis.hut.fi/research/javasomdemo/demo2.html

## Related Demos on the Web

- Finding a heuristic (not necessarily optimal) solution to the Euclidean Traveling Salesperson Problem using Kohonen net



red points = neurons,
red lines = overlay
circles = cities
neurons travel toward cities

http://www.patol.com/java/TSP/index.html

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

## Related Demos on the Web

- Eight competitive models in one applet, including:
  - Kohonen
  - Neural Gas
  - Growing Neural Gas (GNG): Number of neurons grows
- Thirteen choices of input distribution
- Worth visiting

http://www.sund.de/netze/applets/gng/full/GNG_0.html

## Applications of Kohonen Nets

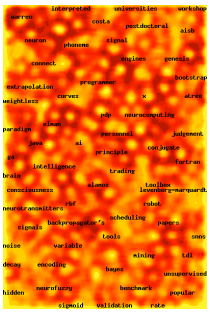
- Most applications that can be treated with MLP's can also be treated in some way by variants of Kohonen nets.
- Example: Learning a function f:D R is done by treating the sample space as a set of (d, r) pairs.

## Applications of Kohonen Nets

- Some applications more suited to Kohonen nets
- Data-mining applications, discovering similarities in data.

## Example: Classifying news messages on Neural Nets by terms occurring within

(seehttp://websom.hut.fi/websom/comp.ai.neural-nets-new/html/root.html)

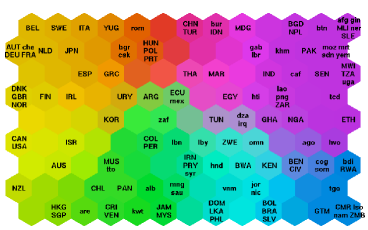



Automatically generated labels and examples of titles in which the labels have occurred

ai - SubSymbolic AI :-O
aisb - AISB and ECCS 1997 - Call for Registrations
alamos - Graduate Research Assistantship in Los Alamos National Laboratory
atree - Atree 3.0 EK mirror sites
backpropagator's - Backpropagator's review, by Donald R. Tveter
bayes - nn and stat decisions for med: Bayes vs neurotic nets
benchmark - Benchmark data for signal processing
bootstrap - Bootstrapping vs. Bayesian
brain - Brain usage Was:Re: Function of sleep
conjugate - Moller's Scaled conjugate gradienconnect - Direct Connect Problems
consciousness - electromagnetics, brain waves and consciousness
costa - New paper available "The Application of Fuzzy Logic in
curves - Predicting Learning Curves (Ph.D. thesis, online)
decay - weight decay
elman - Elman (or Edelman?) nn, references please.
encoding - Question on Encoding Input Data
engines - IC ENGINES & NEURAL NETS
extrapolation - Generalization (Interpolation & Extrapolation)
fortran - Neural Net Fortran Code
ga - GA for learning in ANN
genesis - GENESIS 2.0
hidden - Hidden layer heuristics
intelligence - Commercial Intelligence?
interpreted - Lisp is not an interpreted language
java - NN in Java?

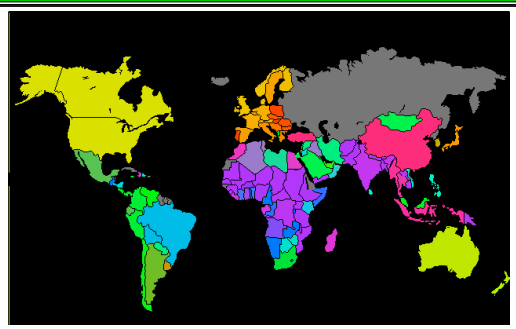
## Example: Classifying World Poverty from a 39-dimension indicator

(http://www.cis.hut.fi/research/som-research/worldmap.html)




The colors were automatically assigned after 2-D clustering using a Kohonen map.


## Colors transferred to a world map



## Other Interesting Applications

- Tree experiment
- Animal characterization experiment
- Phonetic typewriter
- Function approximation
  - Robot Hand-Eye coordination
  - Master/Slave learning technique

## Additional Applications of Kohonen Networks

## Color Quantization Problem (discussed earlier)

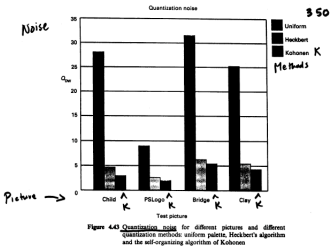



Figure 4.43 Quantization noise for different pictures and different quantization methods: uniform palette, Heckbert's algorithm and the self-organizing algorithm of Kohonen

Heckbert's median cut algorithm "Color Quantization by Dynamic Programming and Principal Analysis" by Xiaolin Wu in ACM Transactions on Graphics, Vol. 11, No. 4, October 1992, 348-372.

## Animal Similarity Expt.

**Table 3.4.** Animal names and their attributes

| | | dove | hen | duck | goose | owl | hawk | eagle | fox | dog | wolf | cat | tiger | lion | horse | zebra | cow |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| is | small | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| | medium | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| | big | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| has | 2 legs | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 4 legs | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | hair | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | hooves | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| | mane | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| | feathers | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| likes to | hunt | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| | run | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| | fly | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| | swim | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## Animal Similarity Expt.

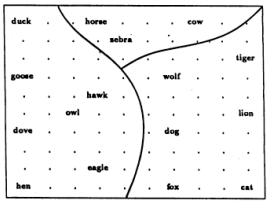



**Fig. 3.22.** After the network had been trained with inputs describing attribute sets from Table 3.4, the map was calibrated by the columns of Table 3.4 and labeled correspondingly. A grouping according to similarity has emerged

## Semantic Map Expt.

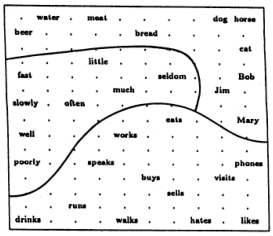
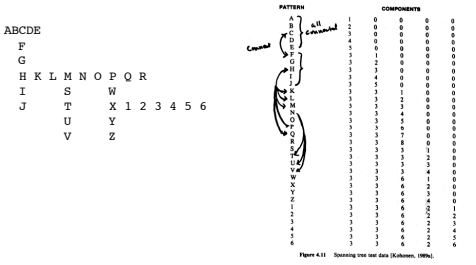



**Fig. 7.11.** "Semantic map" obtained on a network of 10 × 15 cells after 2000 presentations of word-context-pairs derived from 10,000 random sentences of the kind shown in Fig. 7.8. Nouns, verbs and adverbs are segregated into different domains. Within each domain a further grouping according to aspects of meaning is discernible

## Tree Data Structure Expt.

- Encode a 5-dimensional tree as shown:




## Tree Data Structure Expt.

- Use a 2D imposed structure, with 10 x 7 neurons
- Results:
  Branches of tree tend to align in rows or columns of the array.

  5D  2D mapping

## Tree Data Structure Expt.

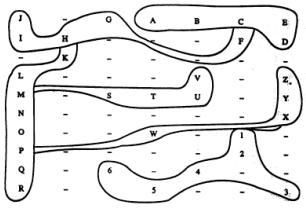


Figure 4.16 Results after another 75 iterations with $R = 0$.

Compare Original 5DTree

```
ABCDE
  F
  G
  H K L M N O P Q R
  I     S       W
  J     T       X 1 2 3 4 5 6
        U       Y
        V       Z
```

## Tree Data Structure Expt. Using Hexagonal Array

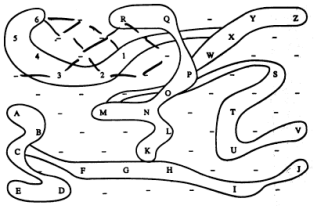


Figure 4.17 Results of spanning tree example using hexagonal array.

Compare Original 5DTree

```
ABCDE
  F
  G
  H K L M N O P Q R
  I     S       W
  J     T       X 1 2 3 4 5 6
        U       Y
        V       Z
```

## Function Approximation Expt.

- Assume 1-dimensional inputs for simplicity (not required in general).
- Represent pairs y = f(x) as (x, y).
- Learn pairs by a 2-D Kohonen net.

## A Possible Aberration

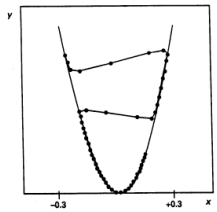



Figure 4.63 Representation of 1000 pairs of argument and function values of the function $y = 10x^2$ by two-dimensional weight vectors (dots) in a one-dimensional neural net of fifty neurons. Input argument values in $[-0.3, +0.3]$

## Master-Slave Technique

- A technique known as “master-slave” can reduce the likelihood of an aberration as shown previously.
  - Select **winners** based only on x, not on the pair (x, y). The weight components corresponding to x are called the **master weights**, and others are called **slave weights.**
  - Adapt both master and slave.

## No Aberration

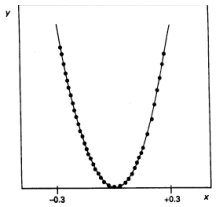



Figure 4.66 The master-slave method. The representation of 1000 pairs of argument and function values of the function $y = 10x^2$ by two-dimensional master-slave weight vectors (dots) in a one-dimensional neural net of fifty neurons. Input argument values in $[-0.3, +0.3]$

## Example: Robot Hand-Eye Coordination

- Want robot to coordinate its hand with what it sees on the monitor.
- Robot arm should be able to place an object given position identified in **view**.
- The problem is how to set the arm's angles (called "inverse kinematics").

## Robot Hand-Eye

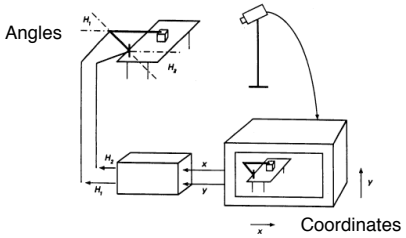



Figure 4.68 Simple outline of robot arm control

## Master-Slave

- The view is the master.
- The hand is the slave.
- Winner is determined w.r.t. the view; both view and hand are adapted.
- Example:
  - 20x20 neuron map, 2D overlay
  - 4 weights per neuron (2 for master/view and 2 for slave/angles)

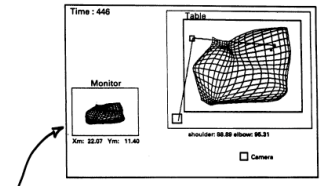



Figure 4.70 Representation of table coordinates by the two-dimensional master–slave weight vectors (dots in the right-hand figure) in a two-dimensional net of 20 × 20 neurons after 446 training examples. Weight vectors are connected with a line if they belong to neighbouring neurons

Positions on the monitor, not what the monitor is seeing.

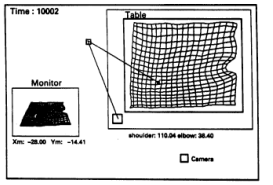
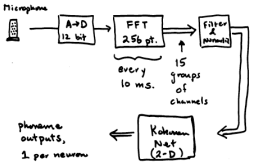



Figure 4.71 Representation of table coordinates by the two-dimensional master–slave weight vectors (dots in the right-hand figure) in a two-dimensional net of 20 × 20 neurons after 10002 training examples. Weight vectors are connected with a line if they belong to neighbouring neurons

## Phonetic Typewriter

- Objective: Spoken words  phonemes
- Languages: Finnish, Japanese
- Claimed ≥ 92% accuracy in 10 mins. of training.



## Resulting Phonotopic Map

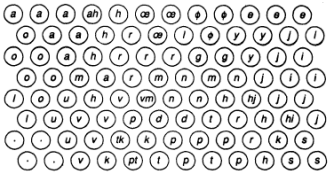


**Figure 7.9** This figure is a phonotopic map with the neurons, shown as circles, and the phonemes to which they learned to respond. A double label means that the node responds to more than one phoneme. Some phonemes—such as the plosives represented by *k*, *p*, and *t*—are difficult for the network to distinguish and are most prone to misclassification by the network. *Source: Reprinted with permission from Teuvo Kohonen, "The neural phonetic typewriter." IEEE Computer, March 1988. ©1988 IEEE.*

## Resulting Phonotopic Map

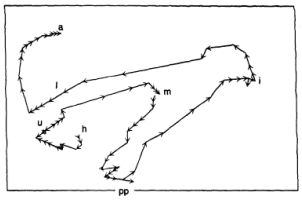



**Figure 7.10** This illustration shows the sequence of responses from tl phonotopic map resulting from the spoken Finnish wo *humppila*. (Do not bother to look up the meaning of this wo in your Finnish–English dictionary: *humppila* is the name of place.) *Source: Reprinted with permission from Teuvo Kohone "The neural phonetic typewriter." IEEE Computer, March 198 ©1988 IEEE.*

## Problems Unsuitable for Kohonen Nets

- Kohonen nets work by clustering
- Nearby data points are expected to behave similarly, e.g. have similar outputs.
- Parity-like problems such as the XOR do not have this property. The would be unstable for solution by a Kohonen net.