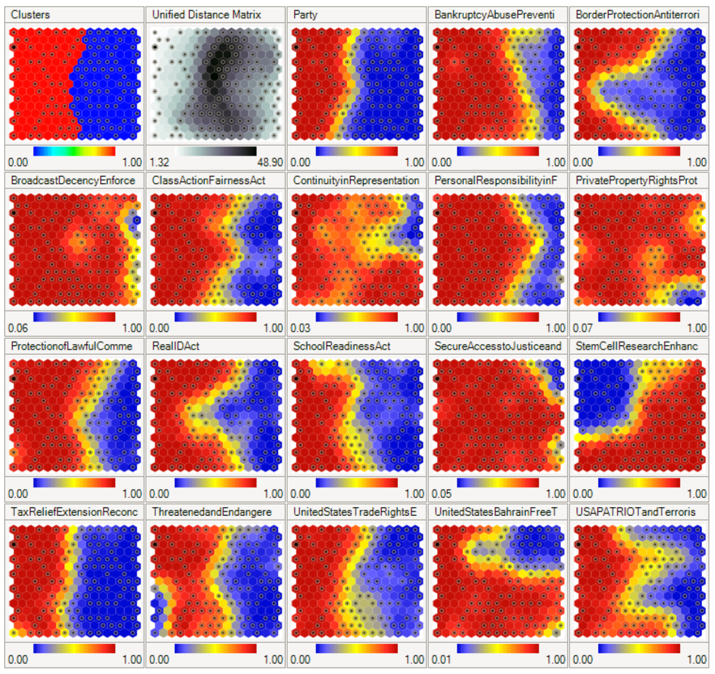
Self-organizing map

A **self-organizing map** (**SOM**) or **self-organizing feature map** (**SOFM**) is a type of artificial neural network that is trained using unsupervised learning to produce a low-dimensional (typically two-dimensional), discretized representation of the input space of the training samples, called a **map**. Self-organizing maps are different from other artificial neural networks in the sense that they use a neighborhood function to preserve the topological properties of the input space.

This makes SOMs useful for visualizing low-dimensional views of high-dimensional data, akin to multidimensional scaling. The model was first described as an artificial neural network by the Finnish professor Teuvo Kohonen, and is sometimes called a **Kohonen map**.[1]



A self-organizing map showing U.S. Congress voting patterns visualized in Synapse. The first two boxes show clustering and distances while the remaining ones show the component planes. Red means a yes vote while blue means a no vote in the component planes (except the party component where red is Republican and blue is Democrat).

Like most artificial neural networks, SOMs operate in two modes: training and mapping. Training builds the map using input examples. It is a competitive process, also called vector quantization. Mapping automatically classifies a new input vector.

A self-organizing map consists of components called nodes or neurons. Associated with each node is a weight vector of the same dimension as the input data vectors and a position in the map space. The usual arrangement of nodes is a regular spacing in a hexagonal or rectangular grid. The self-organizing map describes a mapping from a higher dimensional input space to a lower dimensional map space. The procedure for placing a vector from data space onto the map is to first find the node with the closest weight vector to the vector taken from data space. Once the closest node is located it is assigned the values from the vector taken from the data space.

While it is typical to consider this type of network structure as related to feedforward networks where the nodes are visualized as being attached, this type of architecture is fundamentally different in arrangement and motivation.

Useful extensions include using toroidal grids where opposite edges are connected and using large numbers of nodes. It has been shown that while self-organizing maps with a small number of nodes behave in a way that is similar to K-means, larger self-organizing maps rearrange data in a way that is fundamentally topological in character.

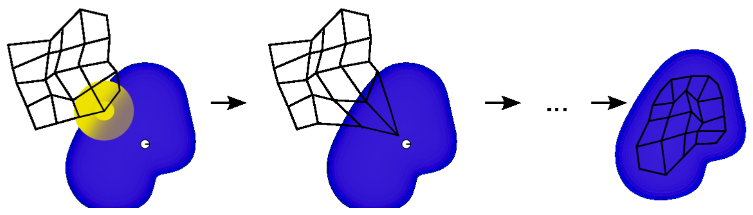
It is also common to use the U-Matrix. The U-Matrix value of a particular node is the average distance between the node and its closest neighbors (ref. 9). In a square grid for instance, we might consider the closest 4 or 8 nodes (the Von Neumann neighborhood and Moore neighborhood respectively), or six nodes in a hexagonal grid.

Large SOMs display properties which are emergent. In maps consisting of thousands of nodes, it is possible to perform cluster operations on the map itself.[2]

Learning algorithm

The goal of learning in the self-organizing map is to cause different parts of the network to respond similarly to certain input patterns. This is partly motivated by how visual, auditory or other sensory information is handled in separate parts of the cerebral cortex in the human brain.[3]

The weights of the neurons are initialized either to small random values or sampled evenly from the subspace spanned by the two largest principal component eigenvectors. With the latter alternative, learning is much faster because the initial weights already give good approximation of SOM weights.[4]


An illustration of the training of a self-organizing map. The blue blob is the distribution of the training data, and the small white disc is the current training sample drawn from that distribution. At first (left) the SOM nodes are arbitrarily positioned in the data space. The node nearest to the training node (highlighted in yellow) is selected, and is moved towards the training datum, as (to a lesser extent) are its neighbors on the grid. After many iterations the grid tends to approximate the data distribution (right).

The network must be fed a large number of example vectors that represent, as close as possible, the kinds of vectors expected during mapping. The examples are usually administered several times as iterations.

The training utilizes competitive learning. When a training example is fed to the network, its Euclidean distance to all weight vectors is computed. The neuron with weight vector most similar to the input is called the best matching unit (BMU). The weights of the BMU and neurons close to it in the SOM lattice are adjusted towards the input vector. The magnitude of the change decreases with time and with distance from the BMU. The update formula for a neuron with weight vector **Wv**(t) is

$$\mathbf{Wv}(t+1) = \mathbf{Wv}(t) + \Theta\,(v, t)\,\alpha(t)(\mathbf{D}(t) - \mathbf{Wv}(t)),$$

where α(t) is a monotonically decreasing learning coefficient and **D**(t) is the input vector. The neighborhood function Θ (v, t) depends on the lattice distance between the BMU and neuron *v*. In the simplest form it is one for all neurons close enough to BMU and zero for others, but a Gaussian function is a common choice, too. Regardless of the functional form, the neighborhood function shrinks with time.[3] At the beginning when the neighborhood is broad, the self-organizing takes place on the global scale. When the neighborhood has shrunk to just a couple of neurons the weights are converging to local estimates.

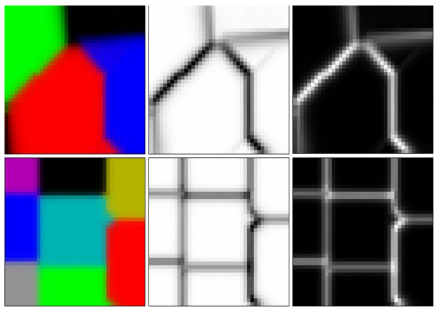
This process is repeated for each input vector for a (usually large) number of cycles **λ**. The network winds up associating output nodes with groups or patterns in the input data set. If these patterns can be named, the names can be attached to the associated nodes in the trained net.

During mapping, there will be one single *winning* neuron: the neuron whose weight vector lies closest to the input vector. This can be simply determined by calculating the Euclidean distance between input vector and weight vector.

While representing input data as vectors has been emphasized in this article, it should be noted that any kind of object which can be represented digitally and which has an appropriate distance measure associated with it and in which the necessary operations for training are possible can be used to construct a self-organizing map. This includes matrices, continuous functions or even other self-organizing maps.

Preliminary definitions

Consider a n×m array of nodes each of which contains a weight vector and is aware of its location in the array. Each weight vector is of the same dimension as the node's input vector. The weights may initially be set to random values.


Self organizing maps (SOM) of three and eight colors with U-Matrix.

Now we need input to feed the map. (The generated map and the given input exist in separate subspaces.) We will create three vectors to represent colors. Colors can be represented by their red, green, and blue components. Consequently our input vectors will have three components, each corresponding to a color space. The input vectors will be:

R = <255, 0, 0>

G = <0, 255, 0>

B = <0, 0, 255>

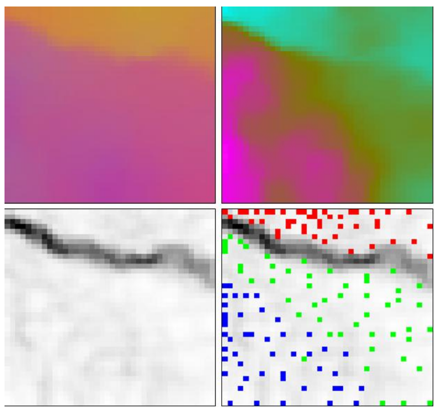
The color training vector data sets used in SOM:

threeColors = [255, 0, 0], [0, 255, 0], [0, 0, 255]

eightColors = [0, 0, 0], [255, 0, 0], [0, 255, 0], [0, 0, 255], [255, 255, 0], [0, 255, 255], [255, 0, 255], [255, 255, 255]

The data vectors should preferably be normalized (vector length is equal to one) before training the SOM.

Neurons (40 x 40 square grid) are trained for 250 iterations with a learning rate of 0.1 using the normalized Iris flower data set which has four dimensional data vectors. A color image formed by first three dimensions of the four dimensional SOM weight vectors (top left), pseudo-color image of the magnitude of the SOM weight vectors (top right), U-Matrix (Euclidean distance between weight vectors of neighboring cells) of the SOM (bottom left) and overlay of data points (red: I. setosa, green: I. versicolor and blue: I. verginica) on the U-Matrix based on the minimum Euclidean distance between data vectors and SOM weight vectors (bottom right).


Self organizing map of Fisher's Iris flower data.

Variables

These are the variables needed, with vectors in bold,

- t denotes current iteration
- λ is the limit on time iteration
- $\mathbf{W_v}$ is the current weight vector
- $\mathbf{D}$ is the target input
- $\Theta(t)$ is restraint due to distance from BMU, usually called the neighborhood function, and
- $\alpha(t)$ is learning restraint due to time.

Algorithm

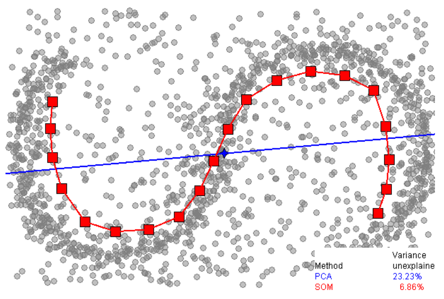
1. Randomize the map's nodes' weight vectors
2. Grab an input vector
3. Traverse each node in the map
    1. Use Euclidean distance formula to find similarity between the input vector and the map's node's weight vector
    2. Track the node that produces the smallest distance (this node is the best matching unit, BMU)
4. Update the nodes in the neighborhood of BMU by pulling them closer to the input vector
    1. $\mathbf{Wv}(t + 1) = \mathbf{Wv}(t) + \Theta(t)\alpha(t)(\mathbf{D}(t) - \mathbf{Wv}(t))$
5. Increase t and repeat from 2 while $t < \lambda$

Interpretation

There are two ways to interpret a SOM. Because in the training phase weights of the whole neighborhood are moved in the same direction, similar items tend to excite adjacent neurons. Therefore, SOM forms a semantic map where similar samples are mapped close together and dissimilar apart. This may be visualized by a U-Matrix (Euclidean distance between weight vectors of neighboring cells) of the SOM. [6]

The other way is to think of neuronal weights as pointers to the input space. They form a discrete approximation of the distribution of training samples. More neurons point to regions with high training sample concentration and fewer where the samples are scarce.

SOM may be considered a nonlinear generalization of Principal components analysis (PCA).[7] It has been shown, using both artificial and real geophysical data, that SOM has many advantages[8][9] over the conventional feature extraction methods such as Empirical Orthogonal Functions (EOF) or PCA.



One-dimensional SOM versus PCA for data approximation. SOM is a red broken line with squares, 20 nodes. The first principal component is presented by a blue line Data points are the small grey circles. For PCA, the Fraction of variance unexplained in this example is 23.23%, for SOM it is 6.86%.[5]

Originally, SOM was not formulated as a solution to an optimisation problem. Nevertheless, there have been several attempts to modify the definition of SOM and to formulate an optimisation problem which gives similar results.[10] For example, Elastic maps use for approximation of principal manifolds[11] the mechanical metaphor of elasticity and analogy of the map with elastic membrane and plate.

Alternatives

- The **generative topographic map** (GTM) is a potential alternative to SOMs. In the sense that a GTM explicitly requires a smooth and continuous mapping from the input space to the map space, it is topology preserving. However, in a practical sense, this measure of topological preservation is lacking.[12]
- The **time adaptive self-organizing map** (TASOM) network is an extension of the basic SOM. The TASOM employs adaptive learning rates and neighborhood functions. It also includes a scaling parameter to make the network invariant to scaling, translation and rotation of the input space. The TASOM and its variants have been used in several applications including adaptive clustering, multilevel thresholding, input space approximation, and active contour modeling.[13]. Moreover, a Binary Tree TASOM or BTASOM, resembling a binary natural tree having nodes composed of TASOM networks has been proposed where the number of its levels and the number of its nodes are adaptive with its environment.[14]
- The **growing self-organizing map** (GSOM) is a growing variant of the self-organizing map. The GSOM was developed to address the issue of identifying a suitable map size in the SOM. It starts with a minimal number of

nodes (usually four) and grows new nodes on the boundary based on a heuristic. By using a value called the *spread factor*, the data analyst has the ability to control the growth of the GSOM.

- The **elastic maps** approach borrows from the spline interpolation the idea of minimization of the elastic energy. In learning, it minimizes the sum of quadratic bending and stretching energy with the least squares approximation error.

References

[1] Kohonen, T. and Honkela, T. (2007). "Kohonen network" (http://www.scholarpedia.org/article/Kohonen_network). *Scholarpedia*. .

[2] Ultsch, Alfred (2007). *Emergence in Self-Organizing Feature Maps, In Proceedings Workshop on Self-Organizing Maps (WSOM '07)*. Bielefeld, Germany. ISBN 978-3-00-022473-7.

[3] Haykin, Simon (1999). "9. Self-organizing maps". *Neural networks - A comprehensive foundation* (2nd ed.). Prentice-Hall. ISBN 0-13-908385-5.

[4] "Intro to SOM by Teuvo Kohonen" (http://www.cis.hut.fi/projects/somtoolbox/theory/somalgorithm.shtml). *SOM Toolbox*. . Retrieved 2006-06-18.

[5] Illustration is prepared using free software: E.M. Mirkes, Principal Component Analysis and Self-Organizing Maps: applet (http://www.math.le.ac.uk/people/ag153/homepage/PCA_SOM/PCA_SOM.html). University of Leicester, 2011

[6] Ultsch A (2003). U*-Matrix: a tool to visualize clusters in high dimensional data. University of Marburg, Department of Computer Science, Technical Report Nr. 36:1-12 (http://www.uni-marburg.de/fb12/datenbionik/pdf/pubs/2003/ultsch03ustar).

[7] *Yin H.* Learning Nonlinear Principal Manifolds by Self-Organising Maps (http://pca.narod.ru/contentsgkwz.htm), In: Gorban A. N. et al. (Eds.), LNCSE 58, Springer, 2007 ISBN 978-3-540-73749-0

[8] Liu, Y., and R.H. Weisberg (2005), Patterns of ocean current variability on the West Florida Shelf using the self-organizing map (http://www.agu.org/pubs/crossref/2005/2004JC002786.shtml). *Journal of Geophysical Research*, 110, C06003, doi:10.1029/2004JC002786.

[9] Liu, Y., R.H. Weisberg, and C.N.K. Mooers (2006), Performance evaluation of the Self-Organizing Map for feature extraction (http://www.agu.org/pubs/crossref/2006/2005JC003117.shtml). *Journal of Geophysical Research*, 111, C05018, doi:10.1029/2005jc003117.

[10] T. Heskes, Energy functions for self-organizing maps, In: Kohonen Maps, E. Oja, S. Kaski (Eds.), Elsevier, 1999.

[11] Gorban A.N., Kegl B., Wunsch D., Zinovyev A. (Eds.), Principal Manifolds for Data Visualisation and Dimension Reduction (http://pca.narod.ru/contentsgkwz.htm), LNCSE 58, Springer: Berlin – Heidelberg – New York, 2007. ISBN 978-3-540-73749-0

[12] Kaski, S.. *Data exploration using self-organizing maps, Acta Polytechnica Scandinavica, Mathematics, Computing and Management in Engineering Series No. 82, Espoo 1997, 57 pp.*.

[13] Hamed Shah-Hosseini and Reza Safabakhsh. *TASOM: A New Time Adaptive Self-Organizing Map, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS, VOL. 33, NO. 2, APRIL 2003, pp. 271-282*. (http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1187438&tag=1)

[14] Hamed Shah-Hosseini. *Binary tree time adaptive self-organizing map* , Neurocomputing, Vol. 74, No. 11, May 2011, pp. 1823-1839. (http://www.sciencedirect.com/science/article/pii/S0925231211000786)

External links

- Self-organizing maps for WEKA (http://jsalatas.ictpro.gr/weka/): Implementation of Self-organizing maps in Java, for the WEKA Machine Learning Workbench.
- Self-organizing maps for Ruby (http://ai4r.rubyforge.org/): Implementation of Self-organizing maps in Ruby, for the AI4R project.
- Spice-SOM (http://www.spice.ci.ritsumei.ac.jp/~thangc/programs/): A free GUI application of Self-Organizing Map
- IFCSoft (http://mathcs.emory.edu/~kthayer/ifcsoft/): An open-source Java platform for generating Self-Organizing Maps

Article Sources and Contributors

Image Sources, Licenses and Contributors

License