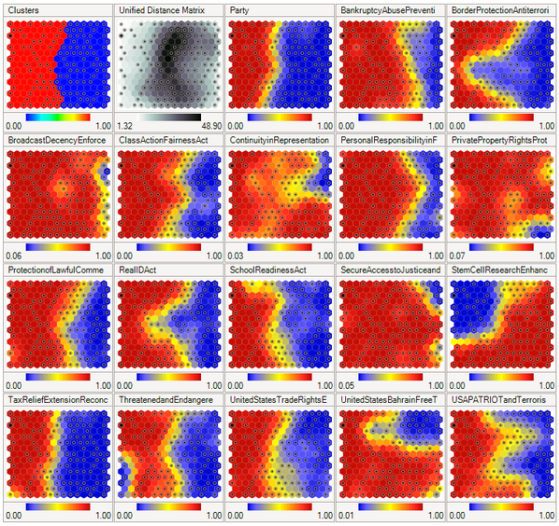
Self-organizing map

A **self-organizing map** (**SOM**) or **self-organising feature map** (**SOFM**) is a type of artificial neural network (ANN) 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 as they apply competitive learning as opposed to error-correction learning (such as backpropagation with gradient descent), and in the sense that they use a neighborhood function to preserve the topological properties of the input space.



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 Democratic).

This makes SOMs useful for visualizing low-dimensional views of high-dimensional data, akin to multidimensional scaling. The artificial neural network introduced by the Finnish professor Teuvo Kohonen in the 1980s is sometimes called a **Kohonen map** or **network**.[1][2] The Kohonen net is a computationally convenient abstraction building on work on biologically neural models from the 1970s[3] and morphogenesis models dating back to Alan Turing in the 1950s[4]

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

A self-organizing map consists of components called nodes or neurons. Associated with each node are 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 two-dimensional 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 find the node with the closest (smallest distance metric) weight vector to the data space vector.

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.

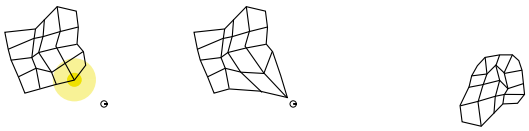
It is also common to use the U-Matrix.[5] The U-Matrix value of a particular node is the average distance between the node's weight vector and that of its closest neighbors.[6] In a square grid, for instance, we might consider the closest 4 or 8 nodes (the Von Neumann and Moore neighborhoods, respectively), or six nodes in a hexagonal grid.

Large SOMs display emergent properties. In maps consisting of thousands of nodes, it is possible to perform cluster operations on the map itself.[7]

1 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.[8]

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 a good ap-



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 datum drawn from that distribution. At first (left) the SOM nodes are arbitrarily positioned in the data space. The node (highlighted in yellow) which is nearest to the training datum is selected. It 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).

proximation of SOM weights.[9]

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 whose weight vector is 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 (within the lattice) from the BMU. The update formula for a neuron v with weight vector $\mathbf{W_v}(s)$ is

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

where s is the step index, t an index into the training sample, u is the index of the BMU for $\mathbf{D}(t)$, $\alpha(s)$ is a monotonically decreasing learning coefficient and $\mathbf{D}(t)$ is the input vector; $\Theta(u, v, s)$ is the neighborhood function which gives the distance between the neuron u and the neuron v in step s.[10] Depending on the implementations, t can scan the training data set systematically (t is 0, 1, 2...T-1, then repeat, T being the training sample's size), be randomly drawn from the data set (bootstrap sampling), or implement some other sampling method (such as jackknifing).

The neighborhood function $\Theta(u, v, s)$ depends on the lattice distance between the BMU (neuron u) and neuron v. In the simplest form it is 1 for all neurons close enough to BMU and 0 for others, but a Gaussian function is a common choice, too. Regardless of the functional form, the neighborhood function shrinks with time.[8] 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. In some implementations the learning coefficient α and the neighborhood function Θ decrease steadily with increasing s, in others (in particular those where t scans the training data set) they decrease in step-wise fashion, once every T steps.

This process is repeated for each input vector for a (usually large) number of cycles $\boldsymbol{\lambda}$. 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, 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.

1.1 Variables

These are the variables needed, with vectors in bold,

- s is the current iteration
- λ is the iteration limit
- t is the index of the target input data vector in the input data set $\mathbf{D}$
- $\mathbf{D}(\mathbf{t})$ is a target input data vector
- v is the index of the node in the map
- $\mathbf{W_v}$ is the current weight vector of node v
- u is the index of the best matching unit (BMU) in the map
- $\Theta(u, v, s)$ is a restraint due to distance from BMU, usually called the neighborhood function, and
- $\alpha(s)$ is a learning restraint due to iteration progress.

1.2 Algorithm

1. Randomize the map's nodes' weight vectors
2. Grab an input vector $\mathbf{D}(\mathbf{t})$
3. Traverse each node in the map
   (a) Use the Euclidean distance formula to find the similarity between the input vector and the map's node's weight vector

   (b) Track the node that produces the smallest distance (this node is the best matching unit, BMU)

4. Update the nodes in the neighborhood of the BMU (including the BMU itself) by pulling them closer to the input vector

   (a) $\mathbf{W_v}(s + 1) = \mathbf{W_v}(s) + \Theta(u, v, s)\ \alpha(s)(\mathbf{D}(t) - \mathbf{W_v}(s))$

5. Increase s and repeat from step 2 while $s < \lambda$

A variant algorithm:

1. Randomize the map's nodes' weight vectors
2. Traverse each input vector in the input data set
   (a) Traverse each node in the map
      i. Use the Euclidean distance formula to find the similarity between the input vector and the map's node's weight vector
      ii. Track the node that produces the smallest distance (this node is the best matching unit, BMU)
   (b) Update the nodes in the neighborhood of the BMU (including the BMU itself) by pulling them closer to the input vector
      i. $\mathbf{W_v}(s + 1) = \mathbf{W_v}(s) + \Theta(u, v, s)\ \alpha(s)(\mathbf{D}(t) - \mathbf{W_v}(s))$
3. Increase s and repeat from step 2 while $s < \lambda$

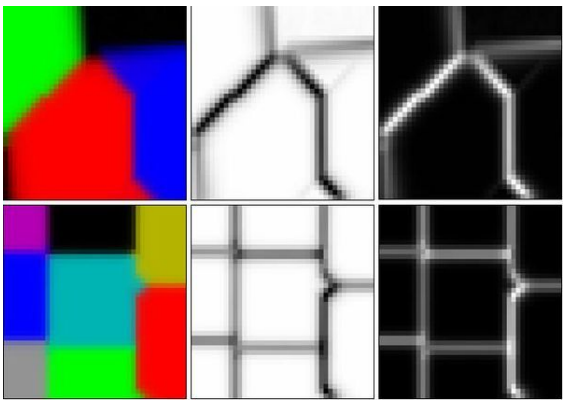
1.3 SOM initiation

Selection of a good initial approximation is a well known problem for all iterative methods of learning neural networks. Kohonen[11] used random initiation of SOM weights but recently the principal component initialization, in which the initial map weights are chosen from the space of the first principal components, became rather popular because exact reproducibility of the results.[12]

Careful comparison of random initiation approach to the principal component initialization for one-dimensional SOM (models of principal curves) demonstrated that the widely accepted presumption about advantages of the principal component SOM initialization is not universal. The best initialization depends on the dataset geometry. Principal component initialization is preferable (in dimension one) if the principal curve approximating the dataset can be univalently and linearly projected on the first principal component (quasilinear sets). For essentially nonlinear datasets the random initiation performs better.[13]

2 Examples

2.1 RGB Color Space



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

Consider an 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.

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]

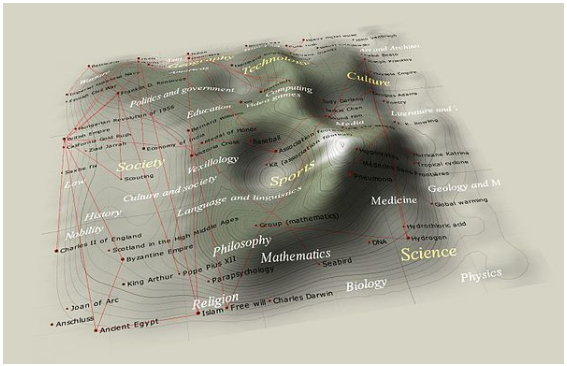
The data vectors should preferably be normalized (vector length is equal to one) before training the SOM.

2.2 Fisher's Iris Flower Data

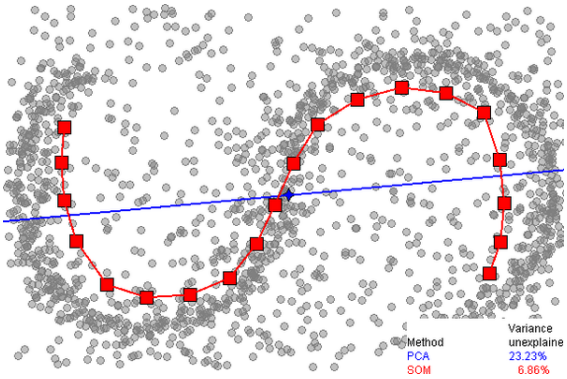
Neurons (40×40 square grid) are trained for 250 iterations with a learning rate of 0.1 using the normalized Iris flower data set which has four-dimensional data vectors. Shown are: a color image formed by the first three dimensions of the four-dimensional SOM weight vectors

(top left), a pseudo-color image of the magnitude of the SOM weight vectors (top right), a U-Matrix (Euclidean distance between weight vectors of neighboring cells) of the SOM (bottom left), and an overlay of data points (red: *I. setosa*, green: *I. versicolor* and blue: *I. virginica*) on the U-Matrix based on the minimum Euclidean distance between data vectors and SOM weight vectors (bottom right).

3 Interpretation



Cartographical representation of a self-organizing map (U-Matrix) based on Wikipedia featured article data (word frequency). Distance is inversely proportional to similarity. The "mountains" are edges between clusters. The red lines are links between articles.



One-dimensional SOM versus principal component analysis (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%.[14]

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 ones apart. This may be visualized by a U-Matrix (Euclidean distance between weight vectors of neighboring cells) of the SOM.[5][6][15]

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).[16] It has been shown, using both artificial and real geophysical data, that SOM has many advantages[17][18] over the conventional feature extraction methods such as Empirical Orthogonal Functions (EOF) or PCA.

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.[19] For example, Elastic maps use the mechanical metaphor of elasticity to approximate principal manifolds:[20] the analogy is an elastic membrane and plate.

4 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.[21]
- 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.[22] 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.[23]
- 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[24] 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.

- The conformal approach [25][26] that uses conformal mapping to interpolate each training sample between grid nodes in a continuous surface. An one-to-one smooth mapping is possible in this approach.

5 Applications

- Meteorology and oceanography[27]
- Project prioritization and selection [28]
- Seismic facies analysis for oil and gas exploration [29]

6 Libraries

- ANNetGPGPU: A lightweight and flexible library written in C++, with Python wrapper for calculation of SOMs on CPU/GPU

7 See also

- Competitive Hebbian Learning
- Neural gas
- Liquid state machine
- Large Memory Storage and Retrieval (LAMSTAR) neural networks (See: Graupe D, Kordylewski H, (1996), "A Large-Memory Storage and Retrieval Neural Network for Browsing and Medical Diagnosis", Proc. 6th ANNIE Conf., St. Louis, Missouri, ASME Press, 711-716; Graupe D, (2013), "Principles of Artificial Neural Networks", 3rd Edition, World Scientific Publishing)
- Hybrid Kohonen SOM
- Sparse coding
- Sparse distributed memory
- Deep learning
- Neocognitron
- Topological data analysis

8 References

[1] Kohonen, Teuvo; Honkela, Timo (2007). "Kohonen Network". *Scholarpedia*.

[2] Kohonen, Teuvo (1982). "Self-Organized Formation of Topologically Correct Feature Maps". *Biological Cybernetics* **43** (1): 59–69. doi:10.1007/bf00337288.

[3] Von der Malsburg, C (1973). "Self-organization of orientation sensitive cells in the striate cortex". *Kybernetik* **14**: 85–100. doi:10.1007/bf00288907.

[4] Turing, Alan (1952). "The chemical basis of morphogenesis". *Phil. Trans. Of the Royal Society* **237**: 5–72.

[5] Ultsch, Alfred; Siemon, H. Peter (1990). "Kohonen's Self Organizing Feature Maps for Exploratory Data Analysis". In Widrow, Bernard; Angeniol, Bernard. *Proceedings of the International Neural Network Conference (INNC-90), Paris, France, July 9–13, 1990* **1**. Dordrecht, Netherlands: Kluwer. pp. 305–308. ISBN 978-0-7923-0831-7.

[6] Ultsch, Alfred (2003); *U*-Matrix: A tool to visualize clusters in high dimensional data*, Department of Computer Science, University of Marburg, Technical Report Nr. 36:1-12

[7] Ultsch, Alfred (2007). "Emergence in Self-Organizing Feature Maps". In Ritter, H.; Haschke, R. *Proceedings of the 6th International Workshop on Self-Organizing Maps (WSOM '07)*. Bielefeld, Germany: Neuroinformatics Group. ISBN 978-3-00-022473-7.

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

[9] Kohonen, Teuvo (2005). "Intro to SOM". *SOM Toolbox*. Retrieved 2006-06-18.

[10] Kohonen, Teuvo; Honkela, Timo (2011). "Kohonen network". *Scholarpedia*. Retrieved 2012-09-24.

[11] T. Kohonen, Self-Organization and Associative Memory. Springer, Berlin, 1984.

[12] A. Ciampi, Y. Lechevallier, Clustering large, multi-level data sets: An approach based on Kohonen self organizing maps, in D.A. Zighed, J. Komorowski, J. Zytkow (Eds.), PKDD 2000, Springer LNCS (LNAI), vol. 1910, pp. 353-358, 2000.

[13] A.A. Akinduko, E.M. Mirkes, A.N. Gorban, SOM: Stochastic initialization versus principal components, *Information Sciences* (2016), Available online 27 October 2015, http://dx.doi.org/10.1016/j.ins.2015.10.013

[14] Illustration is prepared using free software: Mirkes, Evgeny M.; *Principal Component Analysis and Self-Organizing Maps: applet*, University of Leicester, 2011

[15] Saadatdoost, Robab, Alex Tze Hiang Sim, and Jafarkarimi, Hosein. "Application of self organizing map for knowledge discovery based in higher education data." Research and Innovation in Information Systems (ICRIIS), 2011 International Conference on. IEEE, 2011.

[16] Yin, Hujun; *Learning Nonlinear Principal Manifolds by Self-Organising Maps*, in Gorban, Alexander N.; Kégl, Balázs; Wunsch, Donald C.; and Zinovyev, Andrei (Eds.); *Principal Manifolds for Data Visualization and Dimension Reduction*, Lecture Notes in Computer Science and Engineering (LNCSE), vol. 58, Berlin, Germany: Springer, 2008, ISBN 978-3-540-73749-0

[17] Liu, Yonggang; and Weisberg, Robert H. (2005); *Patterns of Ocean Current Variability on the West Florida Shelf Using the Self-Organizing Map*, Journal of Geophysical Research, 110, C06003, doi:10.1029/2004JC002786

[18] Liu, Yonggang; Weisberg, Robert H.; and Mooers, Christopher N. K. (2006); *Performance Evaluation of the Self-Organizing Map for Feature Extraction*, Journal of Geophysical Research, 111, C05018, doi:10.1029/2005jc003117

[19] Heskes, Tom; *Energy Functions for Self-Organizing Maps*, in Oja, Erkki; and Kaski, Samuel (Eds.), *Kohonen Maps*, Elsevier, 1999

[20] Gorban, Alexander N.; Kégl, Balázs; Wunsch, Donald C.; and Zinovyev, Andrei (Eds.); *Principal Manifolds for Data Visualization and Dimension Reduction*, Lecture Notes in Computer Science and Engineering (LNCSE), vol. 58, Berlin, Germany: Springer, 2008, ISBN 978-3-540-73749-0

[21] Kaski, Samuel (1997). "Data Exploration Using Self-Organizing Maps". *Acta Polytechnica Scandinavica*. Mathematics, Computing and Management in Engineering Series No. 82 (Espoo, Finland: Finnish Academy of Technology). ISBN 952-5148-13-0.

[22] Shah-Hosseini, Hamed; Safabakhsh, Reza (April 2003). "TASOM: A New Time Adaptive Self-Organizing Map". *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics* **33** (2): 271–282. doi:10.1109/tsmcb.2003.810442.

[23] Shah-Hosseini, Hamed (May 2011). "Binary Tree Time Adaptive Self-Organizing Map". *Neurocomputing* **74** (11): 1823–1839. doi:10.1016/j.neucom.2010.07.037.

[24] A. N. Gorban, A. Zinovyev, Principal manifolds and graphs in practice: from molecular biology to dynamical systems, International Journal of Neural Systems, Vol. 20, No. 3 (2010) 219–232.

[25] Liou, C.-Y.; Kuo, Y.-T. (2005). "Conformal Self-organizing Map for a Genus Zero Manifold". *The Visual Computer* **21** (5): 340–353. doi:10.1007/s00371-005-0290-6.

[26] Liou, C.-Y.; Tai, W.-P. (2000). "Conformality in the self-organization network". *Artificial Intelligence* **116**: 265–286. doi:10.1016/S0004-3702(99)00093-4.

[27] Liu, Y.,and R.H. Weisberg (2011) A review of self-organizing map applications in meteorology and oceanography. In: Self-Organizing Maps-Applications and Novel Algorithm Design, 253-272.

[28] Zheng, G. and Vaishnavi, V. (2011) "A Multidimensional Perceptual Map Approach to Project Prioritization and Selection," AIS Transactions on Human-Computer Interaction (3) 2, pp. 82-103

[29] Roy, A., Benjamin L. Dowdell., and K. Marfurt (2013) Characterizing a Mississippian tripolitic chert reservoir using 3D unsupervised and supervised multiattribute seismic facies analysis: An example from Osage County, Oklahoma.SEG Interpretation, Vol. 1, No. 2 (November 2013); p. SB109–SB124

Application of self-organising maps and multi-layer perceptron-artificial neural networks for streamflow and water level forecasting in data-poor catchments: the case of the Lower Shire floodplain, Malawi. http://www.iwaponline.com/nh/up/nh2014168.htm

9 External links

- Self-organizing maps for WEKA: Implementation of a self-organizing maps in Java, for the WEKA Machine Learning Workbench.
- Self-organizing maps for Ruby: Implementation of self-organizing maps in Ruby, for the AI4R project.
- Self-organizing map for JavaScript: An open-source implementation of a self-organizing map in JavaScript for node.js from Lucid Technics, LLC.
- Self-organizing map for Python: An open-source implementation of a self-organizing map in python. The SOM structure and training procedure is similar to som toolbox for Matlab
- Self-organizing map for Haskell: An open-source implementation of a self-organising map in Haskell.
- A Self-organizing Map implementation for PHP An open-source implementation of a self-organizing map in PHP.
- Spice-SOM: A free GUI application of self-organizing map
- IFCSoft: An open-source Java platform for generating self-organizing maps
- DemoGNG: Java applet implementing self-organizing maps and other network models (neural gas, growing neural gas, growing grid etc.)
- kohonen An open source Supervised and unsupervised self-organising maps package for R.
- supraHex A supra-hexagonal map for analysing high-dimensional omics data.
- Somoclu A massively parallel implementation of self-organizing maps with interfaces for Python, R, and MATLAB.

10 Text and image sources, contributors, and licenses

10.1 Text

- **Self-organizing map** *Source:* https://en.wikipedia.org/wiki/Self-organizing_map?oldid=709422220 *Contributors:* Ap, Mrwojo, Michael Hardy, Shyamal, Delirium, Pieter Suurmond, Ronz, Glenn, AugPi, Rotem Dan, Hike395, Guaka, Psychonaut, Rholton, Ojigiri~enwiki, Ancheta Wis, Chinasaur, Alensha, Mboverload, Daniel Brockman, Pgan002, Gene s, Urhixidur, JimQ, Thorwald, Rich Farmbrough, Iain-scott, ZeroOne, Alex Kosorukoff, ThruTheLukinGlas, Onay, Janna Isabot, .:Ajvol:., Denoir, Freshraisin, Oleg Alexandrov, Ruud Koot, Male1979, Kbdank71, Rjwilmsi, Miserlou, Itkovian, Hansamurai, Chobot, Wavelength, SpuriousQ, Sanguinity, Aktech, Hakeem.gadi, Ms2ger, Jgzheng, Dq1, Cedar101, Mebden, KnightRider~enwiki, SmackBot, KocjoBot~enwiki, CommodiCast, Mcld, Bluebot, Mge-org~enwiki, Conway71, Miquonranger03, Jasonb05, Goodale, JonHarder, Mitar, Dangraupe, WMod-NS, CRGreathouse, CmdrObot, Zarex, Yarnalgo, Lachambre, Bediako, Galeth, Pihka, JamesBrownJr, Thijs!bot, Jojan, Rasikaa, Lotif, DorisH, Liquid-aim-bot, AnAj, Geo.per, Ninjakannon, Magioladitis, Vernanimalcula, Jqshenker, Schumi555, Andre.holzner, Jorgenumata, KylieTastic, Marllenc, Timo-honkela, Arkadi kagan, Chilti, Anoko moonlight, SieBot, Kylemew, Melcombe, A udachny, Mr. Granger, ClueBot, Rakeshchalasani, Fran-cisco Albani, Agor153, ElectricTypist, Kwantum, FORTRANslinger, Addbot, MrVanBot, Tide rolls, Middayexpress, Tedtoal, Depuarg, Yobot, Mmmcalzones, Citation bot, ArthurBot, Obersachsebot, Xqbot, Ocean518, Breezest, Aultsch, Citation bot 1, Tinton5, Wondigoma, Ismailari, Vp1978, RjwilmsiBot, Ripchip Bot, Helwr, EmausBot, RaoInWiki, Daryakav, MrHedless85, Techteachermayank, Chire, AMan-WithNoPlan, Hfang80, Shinosin, Imprecisekludge, Kleinash, Zackron, Mfemi, Meatsgains, Volkoo, Jsalatas, ÄDA - DÄP, Mansoorexpert, Sevamoo, Hfang bristol, R saadat, Monkbot, Taozaa007, Oyinda8, Atishroy83, Dgdaniel87 and Anonymous: 157

10.2 Images

- **File:Commons-logo.svg** *Source:* https://upload.wikimedia.org/wikipedia/en/4/4a/Commons-logo.svg *License:* CC-BY-SA-3.0 *Contributors:* ? *Original artist:* ?
- **File:Edit-clear.svg** *Source:* https://upload.wikimedia.org/wikipedia/en/f/f2/Edit-clear.svg *License:* Public domain *Contributors:* The *Tango! Desktop Project*. *Original artist:*

  The people from the Tango! project. And according to the meta-data in the file, specifically: “Andreas Nilsson, and Jakub Steiner (although minimally).”
- **File:Question_book-new.svg** *Source:* https://upload.wikimedia.org/wikipedia/en/9/99/Question_book-new.svg *License:* Cc-by-sa-3.0 *Contributors:*

  Created from scratch in Adobe Illustrator. Based on Image:Question book.png created by User:Equazcion *Original artist:*

  Tkgd2007
- **File:SOM_of_RGB_and_eight_colors.JPG** *Source:* https://upload.wikimedia.org/wikipedia/en/1/1b/SOM_of_RGB_and_eight_colors.JPG *License:* PD *Contributors:*

  I (RaoInWiki) created this work entirely by myself.

  Original artist:

  RaoInWiki
- **File:SOMsPCA.PNG** *Source:* https://upload.wikimedia.org/wikipedia/commons/b/bb/SOMsPCA.PNG *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Agor153
- **File:Self_oraganizing_map_cartography.jpg** *Source:* https://upload.wikimedia.org/wikipedia/en/0/07/Self_oraganizing_map_cartography.jpg *License:* CC-BY-SA-3.0 *Contributors:*

  Using my own software

  Original artist:

  Denoir
- **File:Somtraining.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/9/91/Somtraining.svg *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Mcld
- **File:Synapse_Self-Organizing_Map.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/7/70/Synapse_Self-Organizing_Map.png *License:* CC BY-SA 2.5 *Contributors:* Transfered from en.wikipedia Transfer was stated to be made by User:Ddxc. *Original artist:* Original uploader was Denoir at en.wikipedia

10.3 Content license

- Creative Commons Attribution-Share Alike 3.0