J Mol Model (2007) 13:225–228
DOI 10.1007/s00894-006-0140-0

ORIGINAL PAPER

SOMMER: self-organising maps for education and research

Michael Schmuker · Florian Schwarte · André Brück · Ewgenij Proschak · Yusuf Tanrikulu · Alireza Givehchi · Kai Scheiffele · Gisbert Schneider

Received: 1 June 2006 / Accepted: 7 August 2006 / Published online: 22 September 2006

Abstract SOMMER is a publicly available, Java-based toolbox for training and visualizing two- and three-dimensional unsupervised self-organizing maps (SOMs). Various map topologies are implemented for planar rectangular, toroidal, cubic-surface and spherical projections. The software allows for visualization of the training process, which has been shown to be particularly valuable for teaching purposes.

Keywords Chemical space · Drug design · Neural network · SOM · Spherical topology

Abbreviation
SOM self-organizing map

Introduction

The self-organizing map (SOM) concept was introduced as a feature-extraction and data-mapping approach by Kohonen in 1982 [1]. Many variations of Kohonen's original concept have been conceived ever since [2]. In the area of bioinformatics, they have been used primarily for visualizing protein and DNA sequence and structure spaces [3–11], drug-design tasks [12–17], surface and property visualization and prediction [18–22], and binding site analysis [23, 24]—often in conjunction with other clustering and pattern-matching techniques. Typically, the use of SOMs has been restricted to two-dimensional (2D) projections of higher-dimensional data. We developed SOMMER (Self-Organizing Map Maker for Education and Research) as a toolbox for training and visualizing two- and three-dimensional (3D) SOMs. The extra dimension in the SOM grid may allow for a better low-dimensional mapping of complex data manifolds. Moreover, the 3D-grid allows for SOM topologies not available in 2D-space, such as spheres or cubic surfaces. High-quality images can be saved for publication.

The software is written in Java and makes use of the 3D-visualization capabilities of the Java3D-package (https://java3d.dev.java.net/). By displaying the training process of the SOM, it illustratively demonstrates how SOM neurons self-organize to map the data distribution. This feature not only facilitates the understanding of the training process, but can be used to assess the usefulness of a mapping solution. Integrated data-processing tools provide means for data normalization and dimensionality reduction. The SOM topology can also be set to a 2D-scaffold. In this case, the user benefits from the 3D-display of the data distribution, making eventual glitches in the 2D-data mapping obvious.

The use of spherical lattices for self-organizing maps was first proposed by Ritter [25] as an example for the application of SOMs in non-Euclidean spaces. Sangole and Knopf [26] used deformations of spherical SOMs to create three-dimensional representations of numeric data sets that can be used for visual assessment of data set similarity and data classification. In another study, these authors showed how deformable spherical SOMs can be used to create representations of freeform objects with applications to object recognition and shape registration [27]. Liou and Kuo demonstrated the capability of the spherical SOM to

M. Schmuker · F. Schwarte · A. Brück · E. Proschak · Y. Tanrikulu · A. Givehchi · K. Scheiffele · G. Schneider (✉)
Institute of Organic Chemistry and Chemical Biology,
Johann Wolfgang Goethe-University,
Siesmayerstr. 70,
60323 Frankfurt, Germany
e-mail: gisbert.schneider@modlab.de

mimic object surfaces and reconstruct missing points [28]. Wu and Takatsuka [29] showed that spherical SOMs can converge to smaller quantization errors in less training epochs than SOMs with a rectangular, two-dimensional lattice. It has also been shown that spherical SOMs can yield low-dimensional feature maps of data distributed on the surface of a hypersphere with a lower embedding error than planar SOMs can (Nishio et al. (2004), Poster abstract for the Eighth Annual International Conference on Computational Molecular Biology (RECOMB), San Diego, USA). These observations are not surprizing because an SOM-projection is best if the dimensionality of the SOM is identical to the dimension of the data [1, 2]—still, they demonstrate possible primary applications of spherical SOMs.

SOM topologies

An SOM consists of a set of formal neurons and edges linking them. Currently, the following linkage topologies are available in SOMMER:

- Rectangular: a rectangular $X \times Y$ grid, containing $X \cdot Y$ neurons.
- Toroidal: rectangular topology with wrapped edges.
- Cubic: a $X \times Y \times Z$ grid, containing $X \cdot Y \cdot Z$ neurons.
- Spherical: a sphere with the neurons laid out regularly on its surface.

Creating the spherical topology is handled somewhat differently than the other topologies, because distributing a number of points evenly on a spherical surface is a non-trivial task. Ritter [25] proposed tessellating the sphere using a subdivided icosahedron. While this approach yields an almost regular tessellation, it offers a limited choice for the total number of neurons N, which obeys the formula $|N| = 10 \cdot f^2 + 2$, with f the subdivision frequency. So, for f=1,2,3,..., the number of neurons is quantized to $|N|$=12,42,82,162,322,642,1282,... neurons. We adopted the tetrahedron-based tessellation method from Java3D. The number of neurons obeys Eq. 1:

$$|N| = \left(\frac{f + (3 - (f-1) \div 4)}{2} \right)^2 + 2 \quad (1)$$

with ÷ denoting modulo division. It leaves the choice between 6,18,38,66,102,146,198,... neurons. The resulting tessellation is not as regular as that obtained by icosahedron

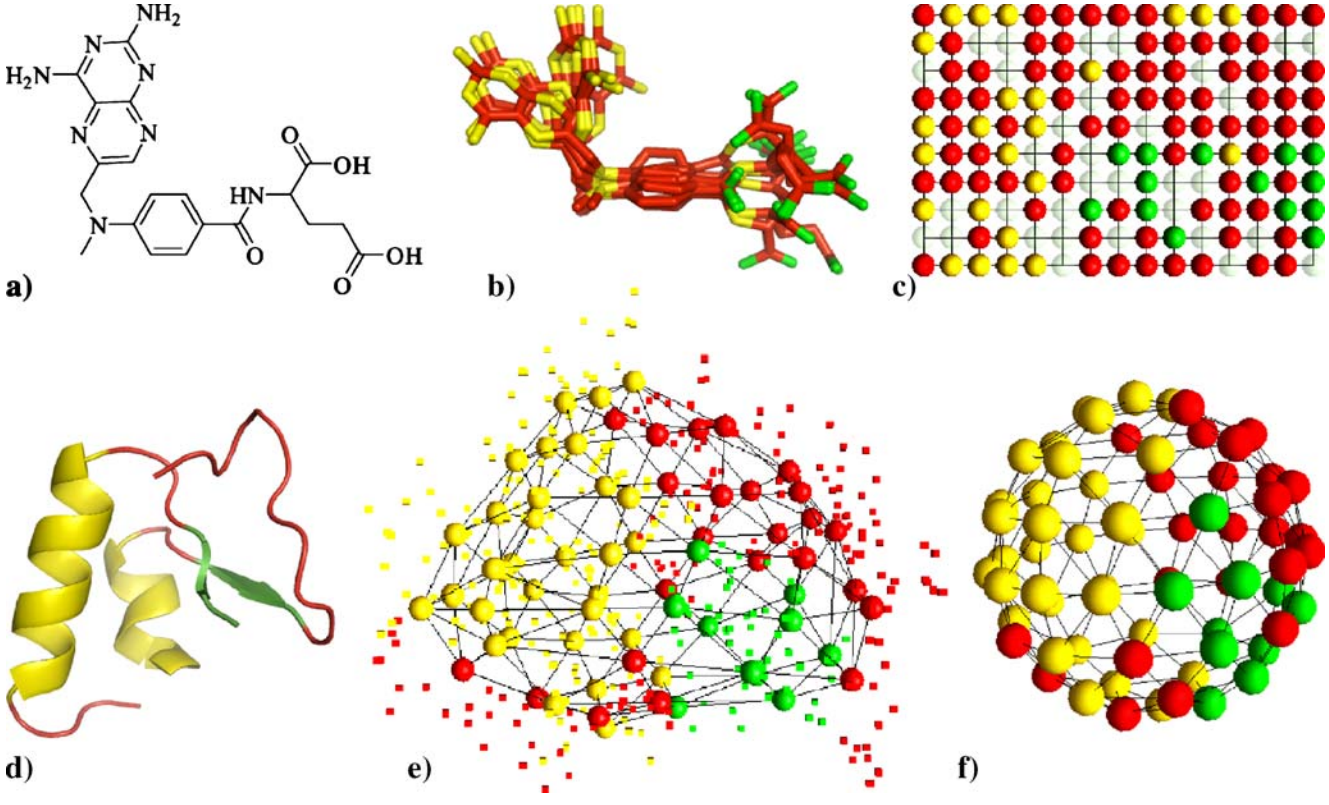

Fig. 1 Examples of SOM projections. **a**–**c** Low-energy conformations (**b**) of methotrexate (**a**) are projected onto a planar 2D SOM (**c**). Neurons are colored using an atom coloring scheme (red: carbon, yellow: nitrogen, green: oxygen). Gray spots indicate empty neurons. **d**–**f** A cartoon representation of crambin (PDB: 1cbn; visualized using PyMOL (DeLano Scientific, San Carlos, CA, USA, http://www.pymol.org); color scheme: red: coil; yellow: alpha-helix, green: beta-strand) is projected onto a spherical SOM (**f**). In (**e**) the SOM neurons are shown in 3D data space containing normalized atom coordinates of the protein

tessellation. In particular, while most neurons have six neighbors linked to them, there are six neurons that only have four neighbors. In spite of this irregularity, we chose this tessellation method over the icosahedron method because it allows for a more fine-grained tuning of the number of SOM neurons.

Training algorithm

The algorithm implemented in SOMMER is based on the work of Loos and Fritzke (Loos HS, Fritzke B (1998) DemoGNG v.1.5 Manual (http://www.neuroinformatik.ruhr-uni-bochum.de/ini/VDM/research/gsn/DemoGNG/tex/DemoGNG/DemoGNG.html). Since the grid-based distance metric used therein is not applicable to all available topologies, we generalized the distance between two neurons on the grid to a graph-based topological distance, such that $d_{topo}(n_1, n_2)$ is equal to the number of graph edges on the shortest path between the two neurons n_1 and n_2. With this slight modification, the algorithm is suitable for training an SOM with any topology, as long as some topological distance is defined.

The "winner neuron" $n_w(\xi)$ of a given data pattern ξ is defined as the neuron with minimal distance $d(\mathbf{n}, \xi)$, where $\mathbf{n}$ is the neuron's prototype vector and ξ the vector associated with the data pattern. Two distance functions are available in SOMMER: the Euclidean distance and the Manhattan (or city-block) metric.

In every training epoch t, the prototype vector $\mathbf{n}$ of each neuron is updated according to Eq. 2:

$$\Delta\mathbf{n} = \lambda(t) \cdot \nu(n_w, n, t) \cdot (\xi - \mathbf{n}) \tag{2}$$

with n_w the winner neuron, and the time-dependent learning rate $\lambda(t)$ (Eq. 3):

$$\lambda(t) = \lambda_i \left(\frac{\lambda_f}{\lambda_i}\right)^{\frac{t}{t_{max}}}. \tag{3}$$

The indices i and f refer to initial (t=0) and final (t=t_{max}) values. The neighborhood function ν determines how strongly a neuron n is adapted relative to the winner neuron n_w. *We use a Gaussian neighborhood function (Eq. 4):*

$$\nu(n_w, n, t) = \exp\left(\frac{-d_{topo}(n_w, n)^2}{2\sigma(t)^2}\right), \text{ with}$$
$$\sigma(t) = \sigma_i\left(\frac{\sigma_f}{\sigma_i}\right)^{\frac{t}{t_{max}}}. \tag{4}$$

The SOM training algorithm works as follows:

(1) Initialize the prototype vector $\mathbf{n}$ of each neuron (i.e., its coordinates in data space) to a random vector.
(2) Choose a data pattern ξ and determine the "winner neuron" $n_w(\xi)$ with minimal distance $d(\mathbf{n},\xi)$.
(3) Adapt each neuron n according to Eq. 2.
(4) Increase the training epoch t=t+1.
(5) If $t<t_{max}$ continue with step 2, else terminate.

Example

Two examples of SOM projections using SOMMER are shown in Fig. 1. Figure 1a–c shows how a planar map can be employed to visualize an ensemble of molecular conformations. The second example illustrates the use of a spherical SOM that captures the secondary-structure distribution of a small protein. There are many potential applications of 3D-SOMs, and SOMMER provides a convenient tool for this purpose.

SOMMER is freely available from the gecco!® website, http://www.gecco.org.chemie.uni-frankfurt.de/gecco.html.

Acknowledgements Norbert Dichter is thanked for technical assistance. This work was supported by the Beilstein-Institut zur Förderung der Chemischen Wissenschaften, Frankfurt.

References

1. Kohonen T (1982) Biol Cybern 43:59–69
2. Kohonen T (2001) Self-organizing maps, 3rd edn. Springer, Berlin Heidelberg New York
3. Arrigo P, Giuliano F, Scalia F, Rapallo A, Damiani G (1991) Comput Appl Biosci 7:353–357
4. Ferran EA, Ferrara P (1991) Biol Cybern 65:451–458
5. Schuchhardt J, Schneider G, Reichelt J, Schomburg D, Wrede P (1996) Protein Eng 9:833–842
6. Hanke J, Reich JG (1996) Comput Appl Biosci 12:447–454
7. Schneider G, Sjoling S, Wallin E, Wrede P, Glaser E, von Heijne G (1998) Proteins 30:49–60
8. Aires-de-Sousa J, Aires-de-Sousa L (2003) Bioinformatics 19:30–36
9. Schneider G, Fechner U (2004) Proteomics 4:1571–1580
10. Fankhauser N, Maser P (2005) Bioinformatics 21:1846–1852
11. Bensmail H, Golek J, Moody MM, Semmes JO, Haoudi A (2005) Bioinformatics 21:2210–2224
12. Schneider G, Wrede P (1998) Prog Biophys Mol Biol 70:175–222
13. Polanski J, Walczak B (2000) Comput Chem 24:615–625
14. Givehchi A, Dietrich A, Wrede P, Schneider G (2003) QSAR Comb Sci 22:549–559
15. Schneider G, Nettekoven M (2003) J Comb Chem 5:233–237
16. Teckentrup A, Briem H, Gasteiger J (2004) J Chem Inf Comput Sci 44:626–634
17. Xiao YD, Clauset A, Harris R, Bayram E, Santago P, Schmitt JD (2005) J Chem Inf Model 45:1749–1758
18. Gasteiger J, Li X, Uschold A (1994) J Mol Graph 12:90–97
19. Anzali S, Barnickel G, Krug M, Sadowski J, Wagener M, Gasteiger J, Polanski J (1996) J Comput Aided Mol Des 10:521–534

20. Hasegawa K, Matsuoka S, Arakawa M, Funatsu K (2002) Comput Chem 26:583–589
21. Roche O, Trube G, Zuegge J, Pflimlin P, Alanine A, Schneider G (2002) Chembiochem 3:455–459
22. Balakin KV, Ivanenkov YA, Savchuk NP, Ivashchenko AA, Ekins S (2005) Curr Drug Discov Technol 2:99–113
23. Stahl M, Taroni C, Schneider G (2000) Protein Eng 13:83–88
24. Del Carpio-Munoz CA, Ichiishi E, Yoshimori A, Yoshikawa T (2002) Proteins 48:696–732
25. Ritter H (1998) Self-organizing maps in non-Euclidian spaces. In: Oja E, Kaski S (eds) Kohonen maps. Elsevier, Amsterdam, pp 97–108
26. Sangole A, Knopf GK (2003) Comput Graph 27:963–976
27. Sangole A, Knopf GK (2003) Int J Smart Eng Sys Des 5:439–454
28. Liou C, Kuo Y (2005) Visual Comput 21:340–353
29. Wu Y, Takatsuka M (2004) The geodesic self-organizing map and its error analysis. In: Estivill-Castro V (ed) Proceedings of the 28th Australasian Conference on computer science, vol 38. Australian Computer Society, Darlinghurst, pp 343–351