# Combined Method for Effective Clustering based on Parallel SOM and Spectral Clustering

Lukáš Vojáček, Jan Martinovič, Kateřina Slaninová,
Pavla Dráždilová, and Jiří Dvorský

Department of Computer Science, FEI, VSB – Technical University of Ostrava,
17. listopadu 15, 708 33, Ostrava-Poruba, Czech Republic
lukas.vojacek@vsb.cz, jan.martinovic@vsb.cz, slaninova@opf.slu.cz,
pavla.drazdilova@vsb.cz, jiri.dvorsky@vsb.cz

**Abstract.** The paper is oriented to the problem of clustering for large datasets with high-dimensions. We propose a two-phase combined method with regard to high dimensions and exploiting the standard clustering algorithm. The first step of the method is based on the learning phase using artificial neural network, especially Self organizing map, which we find as a suitable method for the reduction of the problem complexity. Due to the fact, that the learning phase of artificial neural networks can be time-consuming operation (especially for large high-dimensional datasets), we decided to accelerate this phase using parallelization to improve the computational efficiency. The second phase of the proposed method is oriented to clustering. Because the visualization provided by Self organizing maps is depending on the map dimension, and is not as clear and comprehensible in the cases of clustering applications, we decided to use spectral clustering algorithm to obtain sufficient clusters. According to our results, the proposed combined method is sufficiently rapid and quite accurate.

## 1 Theoretical Background

*Artificial neural networks* (ANN) are the mathematical models inspired by the structure and functionality of the biological neural systems capable of parallel and distributed computation [11]. The ANN can be thought of the non-linear statistical data modeling tool to find and visualize complex relationships between the input data collections and the output map. Moreover, ANN can be used as an adaptive system that is able to change its structure through the learning phase in relation to external input or internal information in the network.

The model typically consists of interconnected groups of neurons, organized in the layers of the system. The basic system of ANN has three layers (the input neurons, the second layer of neurons and the output layer of neurons). All the layers are interconnected through synapses, which have assigned weights used for the calculations of network function $f(x)$. The network function is then defined as a composition of other functions appropriate to the neurons on each layer of the network.

V. Snášel, J. Pokorný, K. Richta (Eds.): Dateso 2011, pp. 120–131, ISBN 978-80-248-2391-1.

ANN is typical of its possibility of learning. Given a class of functions $F$, learning is a process of finding the optimal solution $f^* \in F$ for a specific task, using a set of observations. For the efficiency measurement is used a cost function $C : F \to \mathbb{R}$ such, that for the optimal solution $f^*$ is $C(f^*) = \min C(f), \forall f \in F$.

There are known two basic approaches to the learning phase of ANN: supervised learning, unsupervised learning and their extensions or combinations. Supervised learning is the approach, where the network function is inferred from the supervised training data collection. The set of training examples consists on pairs of the input vectors and the appropriate output values. The network function is then typically used for pattern recognition or classification (for discrete output) or for regression (for continuous output). As an example of commonly used supervised algorithms, we can mention multilayer Perceptron with Backpropagation method.

Unsupervised learning approach is used for solving the problems oriented to discovery and determination of the data structure. Among commonly used algorithms we can include Self organizing maps (SOM) and its extensions. The ANN with unsupervised learning are commonly used for the tasks like clustering, estimation of statistical distributions, filtering or compression problems.

### 1.1 Self Organizing Maps

*Self organizing map* (SOM), also called Kohonen map, is a type of artificial neural network invented by professor Teuvo Kohonen in 1982 [16]. The input space of the training samples is represented in a low dimensional (often two-dimensional) space, called map. The model is capable of projecting the high-dimensional space to the lower-dimensional space [19] and is efficient in the structure visualization due to its feature of the topological preservation using a neighborhood function. The obtained low-dimensional map is often used for pattern detection, clustering, or for characterization and analysis of the input space.

SOM technique has been applied in many spheres like speech recognition [21, 8], image classification [15, 2], document clustering [14, 9] etc. The detailed description of the SOM application is provided in [8].

The model of SOM consists of two layers of nodes. The input layer for receiving and transmitting the input information and the output layer called the map represented the output characteristics. The output layer is commonly organized as the two-dimensional map of nodes, but there are known extensions as, for example, hexagonal grid of output layer. The both layers are feed-forward connected. It is known, that the maps with the smaller grid of the output layer have the behavior similar to K-means clustering [1]. Thr larger output maps have the ability to describe the topological characteristics of the input data collection (often with using U-Matrix for the interpretation of the distance between the nodes).

The SOM input layer is given by input vectors $\boldsymbol{x} \in \mathbb{R}^n$. Each node of this layer is then connected with one of the output nodes $K$ by means of a weight of reference vector $\boldsymbol{w}_k \in \mathbb{R}^n, k = 1, \ldots, K$. During the learning phase, the weight vector $\boldsymbol{w}_k(t)$ is computed for the network at time $t$, where $t = 0, 1, \ldots$ is discrete

time index for each input vector $\boldsymbol{x}(t)$. The passage through the network at time $t$ is an epoch. The learning (training) phase is performed through competitive learning, where for each training example $\boldsymbol{x}(t)$ is computed similarity to all weight vectors $\boldsymbol{w}_k(t)$. The output neuron with the most similar vector is then called as the *best matching unit* (BMU). The weights of the winning neuron and the neurons in the closest neighborhood are then updated and adjusted for the appropriate input vector. The weight vector initialization is commonly assigned randomly, or by using other data mining methods. Concrete implementation of the SOM depends on the method used for the weight vectors' actualization during the training phase.

SOM networks are especially suitable for hidden knowledge presentation. Both the structure of data clusters and query result can be easily visualized. For the overall view of learned data, we use the so-called *Unified distance matrix* (U-matrix), which records the values in clusters and cluster boundaries. The values are assigned to the neuron which wins competition for them, and the distances between neighbouring neurons are recorded with grayness level. Darker colors usually mean greater distance. On the other hand, close data can be colored with similar colors, in this case the boundary between clusters is shown as a steep change in color hue.

### 1.2 Spectral Clustering

Spectral clustering algorithm uses eigenvalues and eigenvectors of a similarity matrix derived from the data set to find the clusters.

Given a set of data points $\{x_1, \dots x_n\} \in \mathbb{R}^l$ and similarity (cosine measure) $a_{ij} \geq 0$ between all pairs of the data points $x_i$ and $x_j$.

Let $G = (V, E)$ be an undirected graph with vertex set $V = \{v_1, \dots, v_n\}$. Each vertex $v_i$ in this graph represents the data point $x_i$. Two vertices are connected, if the similarity $a_{ij}$ between the corresponding data points $x_i$ and $x_j$ is positive, and the edge is weighted by $a_{ij}$. The weighted adjacency matrix of the graph is the matrix $A = (a_{ij})$ $i, j = 1, \dots, n$. If $a_{ij} = 0$ than $(v_i, v_j) \notin E(G)$. For undirected graph it governs that $A$ is symmetric. The degree of a vertex $v_i \in V$ is defined as $d_i = \sum_{j=1}^{n} a_{ij}$. The degree matrix $D$ is defined as the diagonal matrix with the degrees $d_1, \dots, d_n$ on the diagonal. The unnormalized graph Laplacian matrix is defined as $L = D - A$. In[6] Fiedler defines the second smallest eigenvalue $a(G)$ of the of Laplacian matrix $L(G)$ as *algebraic connectivity* of the graph $G$. In his honor, the corresponding eigenvector is called *Fiedler vector*. The *Spectral Partitioning Algorithm* which uses Fiedler vector is summarized in[22]. The other properties of the algebraic connectivity are in [7].

The survey of data clustering relevant to the clustering document collection is published in [12], while the analysis of spectral clustering is described in [13]. Kannan et al. developed a natural bicriteria measure for assessing the quality of the clustering. How to use the spectral algorithm is studied in [13] by Cheng et al. The practical implementation of the clustering algorithm is presented in [3]. In [5] Ding et al. proposed a new graph partition method based on the min-max

clustering principle: the similarity between two subgraphs (cut set) is minimized, while the similarity within each subgraph (summation of similarity between all pairs of nodes within a subgraph) is maximized. Shi and Malik [23] treated image segmentation as a graph partitioning problem and proposed a global criterion, the normalized cut, for segmenting the graph. They showed that an efficient computational technique based on a generalized eigenvalue problem can be used to optimize this criterion. Recursive algorithm is used in [4]. Dasgupta et al. analyzed the second eigenvector technique of spectral partitioning on the planted partition random graph model, by constructing a recursive algorithm.

## 2 SOM Partitioning

### 2.1 SOM Algorithms

There are known several variants of the SOM algorithm interpretations [17, 20]. Depending up to the implementation, we can use serial or parallel version of the algorithms.

**Serial SOM Algorithms** As conventional variant of the serial algorithm interpretations can be considered standard On-line SOM.

*On-line SOM Algorithm* is the conventional method, where the weight vectors $\boldsymbol{w}_k(t)$ are updated during the training phase recursively for each input vector $\boldsymbol{x}(t)$. The BMU $d_c$ is commonly selected by calculating the similarity using Euclidean distance:

$$d_k(t) = \|\boldsymbol{x}(t) - \boldsymbol{w}_k(t)\|^2, \tag{1}$$

$$d_c(t) \equiv \min_k d_k(t). \tag{2}$$

The weight vectors are then updated using a learning-rate factor $\sigma(t)$ and a neighborhood function $h_{ck}(t)$:

$$\boldsymbol{w}_k(t+1) = \boldsymbol{w}_k(t) + \sigma(t)h_{ck}(t)[\boldsymbol{x}(t) - \boldsymbol{w}_k(t)]. \tag{3}$$

The learning-rate factor $\sigma(t)$ is used for the correction of the weight vectors; during the learning phase is reduced. The concrete updated weight vectors $\boldsymbol{w}_k(t)$ are set by the neighbor function $h_{ck}(t)$, which determines the distance between nodes $c$ and $k$. The distance is typically decreasing during the learning phase, from an initial value (often comparable to the dimension/or the half of dimension of the lattice) to the value equal to one neuron (one node in the lattice). Commonly is used the standard Gaussian neighborhood function. For the serial online SOM algorithm were published several variants to improve its computational efficiency; as an example we can mention WEBSOM [18].

**Parallel SOM Algorithms** Till lately, most of the conventional algorithms were designed as sequential. The sequential algorithms were well suited to the past generation of computers, which basically performed the operations in the sequential fashion. With the development of the parallel computation, where the operations were performed simultaneously, there is growing the necessity to redesign the serial algorithms to their parallel implementations.

The parallelization of SOM learning algorithms can be implemented by the network partitioning. The *network partitioning* is the implementation, where the neural network is partitioned among the processors. Then, each input data sample is processed by its assigned processor or the parallel task. The network partitioning was implemented by several authors [10, 24].

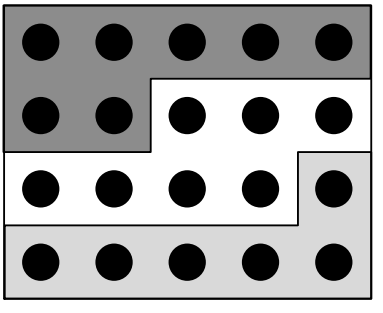


**Fig. 1.** SOM Map Division

The main principle of our parallel implementation is based on division of the neural network into the parts, where each part is assigned to one processor. This division is shown in the Fig. 1, where we have the map of $5 \times 4$ nodes. This map is divided into 3 parts which are associated with 3 processors. Not always there is the possibility to divide the map into the identical parts. In these cases, there is the neural network (map) divided using the principle, that the parts of the network differ in at the most one neuron.

The training phase is based on the serial version of the SOM algorithm. Each process finds its own BMU in its part of the map; this node is then compared with other BMU obtained by other processes. The information about the BMU of the whole network is then transmitted to all the processes, which in accordance with this information update weights of the appropriate nodes in their parts of the network.

### 2.2 Spectral Clustering

The experiment is oriented to the problem of effective clustering for the large datasets with the high-dimension. Due to this reason, we had to find suitable (sufficiently adequate) method for the reduction of the problem complexity. We have decided to used the SOM method. Consider the training set of $n$-dimensional objects $O = \{o_{ij}; o_i \in \mathbb{R}^n, i = 1, \dots m\}$, where $|O| = m$. SOM allows us to transfer the original problem with $m \times n$ dimension to 2-dimensional matrix $A$ of dimension $l \times l$ represented by the SOM map, where $l \ll \max(m, n)$. Our motivation was to facilitate tasks with objects of high dimension.

Because the visualization provided by SOM is depending on the map dimension, and is not as clear and comprehensible in the cases of clustering applications, we decided to use the appropriate algorithm for clustering. As the SOM map can be thought as a graph, and the node weights can be thought as similarity measures, we decided to use the spectral clustering method for dividing the

SOM map into the clusters with the close nodes. In other words, we transformed the SOM map to the similarity matrix of the individual nodes.

For each graph $G$, represented by the similarity matrix, the second smallest eigenvalue $\lambda_2$ of the Laplacian matrix designates algebraic connectivity $a(G)$ [6]. This value can be represented as a lower bound for edge and vertices graph connectivity [7]. We used this knowledge while computing eigenvector incident to this second smallest eigenvalue (algebraic connectivity).

By the recursive application of minimal cut to the graph we have obtained the sequence of the clusters. For each connected subgraph $G_i$ we have obtained its $a(G_i)$, algebraic connectivity. The graph division is finished, when $a(G_i)$ becomes descending as presented in Algorithm 1.

**Algorithm 1** Application of Spectral Clustering Method to SOM

1. Construction of the SOM map.
2. Consider graph $G_1 = (V, E)$, which is given by the similarity matrix $A(G_1)$ of dimension $l \times l$ from the SOM map.
3. Construct Laplacian matrix $L(G_1) = D(G_1) - A(G_1), d_{ii} = \sum_{i=1}^{l} a_{ij}$.
4. Compute $a(G_1)$, $u(G_1)$, where algebraic connectivity $a(G_1) = \lambda_2(G_1)$, $u(G_1)$ is appropriate eigenvector.
5. Divide graph $G_j$ to connected subgraphs $G'_{j+1} = \{v_i \in V_j$, where $u_i \leq 0\}$, $G''_{j+1} = \{v_j \in V_j$, where $u_j > 0\}$, but it is possible that this subgraph is not connected. Then find the all connected components which create the subgraphs $G^{(i)}_{j+1}$.
6. For each subgraph compute $a(G^{(i)}_{j+1})$ and appropriate $u(G^{i}_{j+1})$.
7. If $a(G_{j+1}) < \theta$ then do not divide else step 5, where $\theta \in \mathbb{R}$ is the threshold for algebraic connectivity.

## 3 Experiments

The experiments were divided into two phases according to the phases of proposed algorithm. The first phase of the experiments was oriented to the acceleration of the SOM algorithm. As mentioned above, we have tested the parallel implementation of the SOM algorithm training phase for various dimensions of the SOM map and the input vector.

The second phase of the experiments was related to the division of the SOM map using spectral clustering. Both phases are documented by obtained issues and the appropriate figures, see below.

### 3.1 SOM Acceleration

All the experiments were performed on Windows HPC server 2008 with 6 computing nodes, where each node has 8 processors with 12 GB of memory.

The first experiment was provided on the training set of 300 2-dimensional samples, while the dimension of the neural network (SOM map) was changed. The outputs are presented in the Table 1, where the records with asterisk (*) were provided only by one computing node. In these cases, there is not provided the network communication between processes and due to this fact is the computation faster.

**Table 1.** Computing Time Dependence on Map Dimension and Number of Processors

| Processors | Computing Time [hh:mm:ss] | | |
|---|---|---|---|
| | Map Dimension $100 \times 100$ | Map Dimension $300 \times 300$ | Map Dimension $500 \times 500$ |
| 1* | 0:01:20 | 0:12:57 | 0:35:57 |
| 8* | 0:00:16 | 0:01:56 | 0:21:43 |
| 16 | 0:00:14 | 0:01:06 | 0:03:18 |
| 24 | 0:00:13 | 0:00:50 | 0:02:05 |
| 32 | 0:00:15 | 0:00:48 | 0:01:37 |
| 40 | 0:00:15 | 0:00:38 | 0:01:21 |

The second experiment was provided on the map with selected dimension of $100 \times 100$ nodes, while the input vector dimension of the training data set was changed. There was used the training set of 150 records. The outputs are presented in the Table 2, where the records with asterisk (*) were provided only by one computing node. In this cases, there is not provided the network communication between processes; due to this fact is the computation faster.

**Table 2.** Computing Time Dependence on Input Vector Dimension and Number of Processors

| Processors | Computing Time [hh:mm:ss] | |
|---|---|---|
| | Input Vector Dimension 4 | Input Vector Dimension 8 |
| 1* | 0:06:29 | 0:12:08 |
| 8* | 0:01:01 | 0:01:48 |
| 16 | 0:00:36 | 0:01:00 |
| 24 | 0:00:27 | 0:00:44 |
| 32 | 0:00:25 | 0:00:38 |
| 40 | 0:00:23 | 0:00:34 |

As we can see from Tables 1 and 2, the acceleration of the SOM algorithm is appreciable. With growing number of processors is increasing the computation effectiveness, and the computational time is sufficiently reducing.

### 3.2 SOM Map Division

The next phase of the experiments was oriented to the testing of the proposed algorithms for the SOM map division using spectral clustering, concretely with the Fiedler vector. We have used three training data collections called TwoDiamonds, Lsun a Hepta from the Fundamental Clustering Problems Suite (FCPS)[1]. Short description of selected dataset used in our experiments is given in Table 3.

**Table 3.** Fundamental Clustering Problems Suite – selected datasets

| Name | Cases | #Vars | #Clusters | Main Clustering Problem |
|---|---|---|---|---|
| Hepta | 212 | 3 | 7 | different densities in clusters |
| Lsun | 400 | 2 | 3 | different variances in clusters |
| TwoDiamonds | 800 | 2 | 2 | touching clusters |

For comprehensible interpretation we have used U-matrix and its 3D visualization of the SOM map obtained after the dimension reduction of the input dataset . The visualization of the first input data set *TwoDiamonds*, the SOM map after the first phase of proposed algorithm and its division by spectral clustering are presented on the Fig. 2. The U-matrix for this dataset, presented on the Figs. 2(b) and 2(c), is accurately matching to the division provided by the Fiedler vector on the Fig. 2(d).

The same situation occurs for the second input data set *Lsun*, for visualization see Fig. 3(a). As we can see from the Figs. 3(b) and 3(c), the division provided by the SOM and Fiedler vector, Fig. 3(d), is also accurately matching as in the experiment with the first dataset.

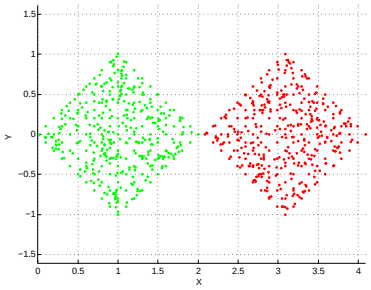
The visualization of the third input data set *Hepta*, the SOM map after the first phase of proposed algorithm and its division by spectral clustering are presented on the Fig. 4. From the Figs. 4(b) and 4(c) we can see, that the division provided by Fig. 4(d) is not as corresponding as in both previous experiments. The experiment with dataset Hepta demonstrates, that some edges are determined in spectral clustering differently then in U-Matrix, but both results are correct.

## 4 Conclusion

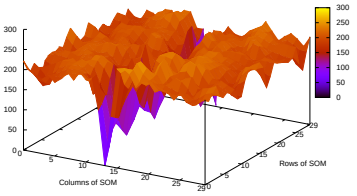
In this paper we presented the parallel implementation of the SOM neural network algorithm. Parallel implementation was tested on HPC cluster containing 6 nodes and 40 processor cores. The achieved speed-up was very good.

Moreover, the partitioning of the resulting SOM map using spectral clustering method was presented. The spectral clustering was applied on U-matrix. This method automatically detected the parts of the SOM that represent the

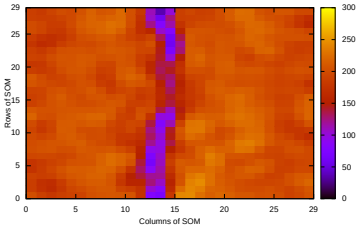
[1] `http://www.uni-marburg.de/fb12/datenbionik/data?language_sync=1`



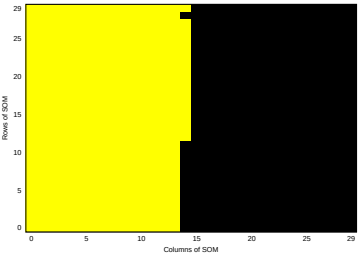

(a) Cluster Visualization




(b) U-Matrix 3D

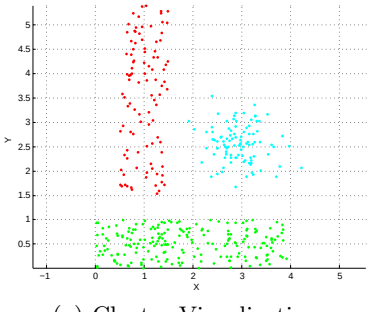



(c) U-Matrix

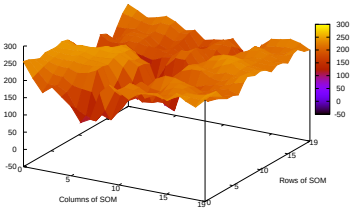



(d) SOM Division using Fiedler Vector

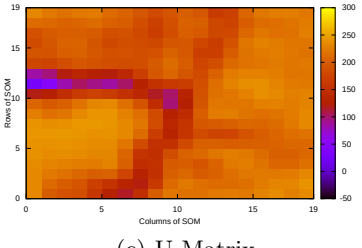
**Fig. 2.** Fundamental Clustering Problems Suite – Two Diamonds



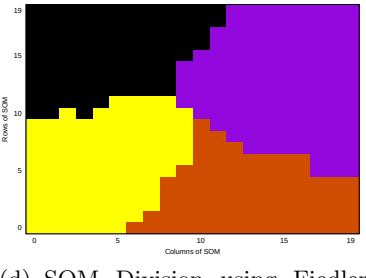

(a) Cluster Visualization




(b) U-Matrix 3D

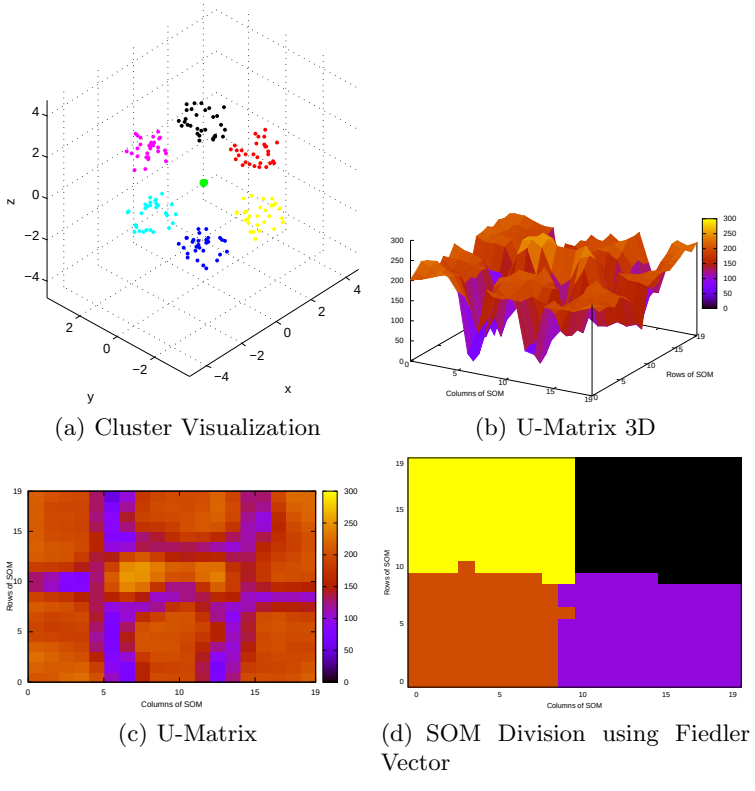



(c) U-Matrix




(d) SOM Division using Fiedler Vector

**Fig. 3.** Fundamental Clustering Problems Suite – Lsun




(a) Cluster Visualization

(b) U-Matrix 3D

(c) U-Matrix

(d) SOM Division using Fiedler Vector

**Fig. 4.** Fundamental Clustering Problems Suite – Hepta

clusters in original high-dimensional data. The detected clusters correspond to the clusters perceived by the human being.

In the future work we intend to focus on more precise specification of the threshold for the algebraic connectivity in the spectral clustering.

## Acknowledgment

This work is partially supported by Grant of Grant Agency of Czech Republic No. 205/09/1079, and SGS, VSB – Technical University of Ostrava, Czech Republic, under the grant No. SP2011/172.

## References

1. F. Bacao, V. Lobo, and M. Painho. Self-organizing maps as substitutes for k-means clustering. In *Computational Science - ICCS 2005, Pt. 3, Lecture Notes in Computer Science*, pages 209–217, 2005.
2. H. Bekel, G. Heidemann, and H. Ritter. Interactive image data labeling using self-organizing maps in an augmented reality scenario. *Neural Networks*, 18(5-6):566–574, June-July 2005.
3. D. Cheng, R. Kannan, S. Vempala, and G. Wang. On a recursive spectral algorithm for clustering from pairwise similarities. Technical report, MIT, 2003.
4. A. Dasgupta, J. Hopcroft, R. Kannan, and P. Mitra. Spectral clustering by recursive partitioning. In *ESA'06: Proceedings of the 14th conference on Annual European Symposium*, pages 256–267. Springer-Verlag, 2006.
5. C. H. Q. Ding, X. He, H. Zha, M. Gu, and H. D. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *ICDM '01: Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 107–114, Washington, DC, USA, 2001. IEEE Computer Society.
6. M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, pages 298 – 305, 1973.
7. M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*, pages 619–633, 1975.
8. B. Gas, M. Chetouani, J.-L. Zarader, and C. Charbuillet. Predictive kohonen map for speech features extraction. In W. Duch, J. Kacprzyk, E. Oja, and S. Zadrozny, editors, *Artificial Neural Networks: Formal Models and Their Applications - ICANN 2005*, volume 3697 of *Lecture Notes in Computer Science*, pages 793–798. Springer Berlin / Heidelberg, 2005.
9. A. Georgakis and H. Li. An ensemble of som networks for document organization and retrieval. In *Proceedings of AKRR'05, International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning*, pages 141–147, 2005.
10. W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: portable parallel programming with the message-passing inferace.* MIT Press, 1999.
11. S. Haykin. *Neural Networks. A Comprehensive Foundation.* Macmillan, New York, 1994.

12. D. Húsek, J. Pokorný, H. Řezánková, and V. Snášel. Data clustering: From documents to the web. In *Web Data Management Practices: Emerging Techniques and Technologies*, chapter Data Clustering: From Documents to the Web, pages 1–33. Idea Group Inc, 2006.
13. R. Kannan, S. Vempala, and A. Vetta. On clusterings: Good, bad and spectral. *J. ACM*, 51(3):497–515, May 2004.
14. K. Kishida. Techniques of document clustering: A review. *Library and Information Science*, 35(1):106–120, January 2005.
15. O. Kohonen, T. Jaaskelainen, M. Hauta-Kasari, J. Parkkinen, and K. Miyazawa. Organizing spectral image database using self-organizing maps. *Journal of Imaging Science and Technology*, 49(4):431–441, July-August 2005.
16. T. Kohonen. *Self-Organization and Associative Memory*, volume 8 of *Springer Series in Information Sciences*. Springer, Berlin, Heidelberg, 1984. 3rd ed. 1989.
17. T. Kohonen. Things you haven't heard about the self-organizing map. In *Proc. ICNN'93, International Conference on Neural Networks*, pages 1147–1156, Piscataway, NJ, 1993. IEEE, IEEE Service Center.
18. T. Kohonen. Exploration of very large databases by self-organizing maps. In *Proceedings of ICNN'97, International Conference on Neural Networks*, pages PL1–PL6. IEEE Service Center, Piscataway, NJ, 1997.
19. T. Kohonen. *Self Organizing Maps*. Springer-Verlag, 3rd edition, 2001.
20. R. D. Lawrence, G. S. Almasi, and H. E. Rushmeier. A scalable parallel algorithm for self-organizing maps with applications to sparse data mining problems. *Data Mining and Knowledge Discovery*, 3:171–95, 1999.
21. S. Meenakshisundaram, W. L. Woo, and S. S. Dlay. Generalization issues in multiclass classification - new framework using mixture of experts. *Wseas Transactions on Information-Science and Applications*, 4:1676–1681, Dec 2004.
22. A. Pothen, H. D. Simon, and K.-P. Liou. Partitioning sparse matrices with wigenvectors of graphs. *SIAM J. Matrix Anal. Appl.*, 11(3):430–452, July 1990.
23. J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:888–905, 1997.
24. C. H. Wu, R. E. Hodges, and C. J. Wang. Parallelizing the self-organizing feature map on multiprocessor systems. *Parallel Computing*, 17(6–7):821–832, September 1991.