Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Neural Networks 17 (2004) 365–377

Neural Networks

www.elsevier.com/locate/neunet

Marginal median SOM for document organization and retrieval

A. Georgakis[1], C. Kotropoulos*, A. Xafopoulos, I. Pitas

Artificial Intelligence and Information Analysis Laboratory, Department of Informatics, Aristotle University of Thessaloniki, Box 451, Thessaloniki GR-54124, Greece

Received 20 November 2001; accepted 25 August 2003

Abstract

The self-organizing map algorithm has been used successfully in document organization. We now propose using the same algorithm for document retrieval. Moreover, we test the performance of the self-organizing map by replacing the linear Least Mean Squares adaptation rule with the marginal median. We present two implementations of the latter variant of the self-organizing map by either quantizing the real valued feature vectors to integer valued ones or not. Experiments performed using both implementations demonstrate a superior performance against the self-organizing map based method in terms of the number of training iterations needed so that the mean square error (i.e. the average distortion) drops to the $e^{-1} = 36.788\%$ of its initial value. Furthermore, the performance of a document organization and retrieval system employing the self-organizing map architecture and its variant is assessed using the average recall–precision curves evaluated on two corpora; the first comprises of manually selected web pages over the Internet having touristic content and the second one is the Reuters-21578, Distribution 1.0.

Keywords: Self-organizing maps; Order statistics; Marginal median

1. Introduction

Due to their wide range of applications, *artificial neural networks* (ANN) have been an active research area for the past three decades (Haykin, 1999). A large variety of learning algorithms (i.e. error-correction, memory-based, Hebbian, Boltzmann machines, supervised or unsupervised) have been evolved and being employed in ANNs. A further categorization divides the network architectures into three distinct categories: *feedforward*, *feedbackward*, and *competitive* (Haykin, 1999).

The self-organizing maps (SOMs) or Kohonen's feature maps are feedforward, competitive ANN that employ a layer of input neurons and a single computational layer (Kohonen, 1990, 1997). The neurons on the computational layer are fully connected to the input layer and are arranged on a N-dimensional lattice. Low-dimensional grids, usually two dimensional (2D) or 3D, have prominent visualization properties, and therefore, are employed on the visualization of high-dimensional data. In this paper, we shall use the SOM algorithm to cluster contextually similar documents into classes. Therefore, we shall focus on the 2D lattice in order to visualize the resulting classes on the plane. For the 2D lattice, the computational layer can have either a hexagonal or orthogonal topology. In hexagonal lattices, each neuron has six equal-distant neighbors, whereas orthogonal lattices can be either four- or eight-connected. As for the competitive nature of the algorithm, this is expressed by the fact that only the neuron which is 'closer' to the input feature vector with respect to a given metric as well as its neighbors are updated every time a new feature is presented to the ANN.

The SOMs are capable of forming a nonlinear transformation or mapping from an arbitrary dimensional data manifold, the so-called *input space*, onto the low-dimensional lattice (Haykin, 1999; Kohonen, 1997). The algorithm takes into consideration the relations between the input feature vectors and computes a set of reference vectors in the output space that provide an efficient vector quantization of the input space. Moreover, the density of neurons, i.e. the number of neurons in a small volume of the input space

* Corresponding author. Tel.: +30-2310-998225; fax: +30-2310-998453.

E-mail addresses: costas@zeus.csd.auth.gr (C. Kotropoulos); apostolos.georgakis@tfe.umu.se (A. Georgakis).

[1] Present address: Digital Media Laboratory (DML), Department of Applied Physics and Electronics, Umeå University, Umeå SE-90187, Sweden.

0893-6080/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2003.08.008

matches the probability density function (pdf) of the feature vectors. Generally, the approximation error is measured by the Mean Square Error (MSE). In doing so, the algorithm employs a linear Least Mean Square adaptation rule for updating the reference vector of each neuron. When the training procedure is led to equilibrium it results to a partition of the domain of the vector-valued observations called *Voronoi tessellation* (Kohonen, 1997; Ritter & Schulten, 1988). The convergence properties of SOMs are studied in Erwin, Obermayer, and Schulten (1992) and Ritter and Schulten (1988).

A complete and thorough investigation regarding the available variants of the SOM algorithm can be found in Kangas, Kohonen, and Laaksonen (1990) and Kohonen (1997). One such frequently used variant is the batch-map. The batch-map estimates the sample mean of the feature vectors that are assigned to each reference vector and subsequently smooths the sample mean to yield an updated reference vector. A trade off is made between the speed and degradation of the clustering accuracy (Fort, Letremy, & Cottrell, 2002). The batch-map is faster than the on-line SOM algorithm. However, it produces unbalanced classes of inferior quality than those produced by on-line SOM algorithm. In the experiments reported in Section 5, the precision rate of the batch SOM algorithm is always less than that of the on-line SOM for all recall rates.

The ability of the SOM algorithm to produce spatially organized representations of the input space can be utilized in document organization, where organization refers to the representation and storage of the available data. In this paper, we exploit this algorithm also for document retrieval. Retrieval refers to the exploration of the organized document repository through specific user-defined queries (Yates & Neto, 1999).

Prior to the document indexing, due to the nature of the SOM algorithm the available textual data have to be transcribed into a numerical form. Among the three widely accepted encoding models that are used by the information retrieval (IR) community (Yates & Neto, 1999), namely the *boolean*, the *probabilistic*, and the *vector space* model, the latter model is the most appropriate for the SOM algorithm. In the vector space model, the documents and the queries used in the training and the retrieval phase are represented by high-dimensional vectors. Each vector component corresponds to a different *word type* (i.e. a distinct word appearance) in the document collection (also called *corpus*). Subsequently, the documents can be easily clustered into contextually related collections by using any distance metric, such as the Euclidean, the Mahalanobis, the city-block, etc. Such a clustering is based on the assumption that the contextual correlation between the documents continues to exist in their vectorial representation. The degree of similarity between a given query and the documents is measured using the same distance metric and the documents marked as being relevant to the query can be ranked in a decreasing order of similarity according to this distance metric (Yates & Neto, 1999).

An architecture based on the SOM algorithm that is capable of clustering documents according to their semantic similarities is the so-called *WEBSOM* architecture (Kohonen, 1998; Kohonen et al., 1999, 2000). The WEBSOM consists of two distinct layers where the SOM algorithm is applied. The first layer is used to cluster the words found in the available training documents into semantically related collections. The second layer, which is activated after the completion of the first layer, clusters the available documents into classes that with high probability contain relevant documents with respect to their semantic content (i.e. context). Due to that, the WEBSOM architecture is regarded as a prominent candidate for document organization and retrieval.

In this paper, we test the performance of the SOM algorithm by replacing the linear Least Mean Squares adaptation rule with the marginal median for document organization and retrieval. The proposed algorithm has similarities with the batch-map because both of them use the Voronoi sets, that is, the set of feature vectors that have been assigned to each neuron, in order to update the reference vector of the neuron. Its difference lies in the replacement of the averaging procedure employed in the batch-map by the marginal median operator in the proposed variant. However, the proposed algorithm remains an on-line algorithm.

The outline of the paper is as follows: Section 2 provides a brief description of the basic SOM algorithm, its mathematical foundations as well as a brief summary of the algorithm's native drawbacks. Section 3 describes the proposed variant with respect to the updating procedure of the reference vectors, which is based on marginal data ordering. It also contains a description of the two distinct implementations of the proposed algorithm. Section 4 is divided into three subsections: Section 4.1 covers the formation of the two corpora employed in our study and the preprocessing steps taken in order to remove any unwanted information from them. Section 4.2 describes the language model employed to encode the textual data into numerical vectors and Section 4.3 is devoted to word and document clustering. In Section 5, we assess the experimental results by using the MSE curves during the training phase of the proposed algorithm and the basic SOM method and the average recall–precision curves obtained by querying the information organization obtained in the training phase of both systems.

2. Self-organizing maps

Let us denote by $\mathscr{X}$ the set of vector-valued observations, $\mathscr{X} = \{\mathbf{x}_j \in \mathbb{R}^{N_w} | \mathbf{x}_j = (x_{1j}, x_{2j}, \ldots, x_{N_w j})^{\mathrm{T}}, \quad j = 1, 2, \ldots, N\}$, where N_w corresponds to the dimensionality of the vectors that encode the N available observations. Let also $\mathscr{W}$ denote the set of reference vectors of the neurons, that is,

$\mathscr{W} = \{\mathbf{w}_l(k) \in \mathbb{R}^{N_w}, l = 1, 2, \ldots, L\}$, where the parameter k denotes discrete time and L is the number of neurons on the lattice. Finally, let $\mathbf{w}_l(0)$ be located on a regular lattice that lies on the hyperplane which is determined by the two eigenvectors that correspond to the largest eigenvalues of the covariance matrix of $\mathbf{x}_j \in \mathscr{X}$ (*linear initialization*) (Kohonen, 1997).

There are two kinds of vector-valued observations that we are interested in: the word vectors and the document vectors. A detailed description of the formation of these vectors can be found in Section 4.2.

Due to its competitive nature, the SOM algorithm identifies the best-matching, winning reference vector $\mathbf{w}_s(k)$ (or winner for short), to a specific feature vector $\mathbf{x}_j$ with respect to a certain distance metric. The index s of the winning reference vector is given by:

$$s = \arg\min_{l=1}^{L} \|\mathbf{x}_j - \mathbf{w}_l(k)\|, \tag{1}$$

where $\|\cdot\|$ denotes the Euclidean distance.

The reference vector of the winner as well as the reference vectors of the neurons in its neighborhood are modified toward $\mathbf{x}_j$ using:

$$\mathbf{w}_i(k+1) = \begin{cases} \mathbf{w}_i(k) + a(k)[\mathbf{x}_j - \mathbf{w}_i(k)] & \forall i \in \mathscr{N}_s \\ \mathbf{w}_i(k) & \forall i \notin \mathscr{N}_s \end{cases}, \tag{2}$$

where $a(k)$ is the learning rate and $\mathscr{N}_s$ denotes the neighborhood of the winner. A neighborhood updating, especially in the early iterations, is performed in order to achieve a global ordering of the input space onto the lattice, which is crucial for the good resolution of the map (Kohonen, 1997). The term basic SOM will henceforth denote the on-line algorithm proposed by Kohonen (1997) without any modifications or speed-up techniques.

Eq. (2) can be rewritten as follows:

$$\mathbf{w}_i(k+1) = \mathbf{w}_i(k) + a(k)c_{ij}(k)[\mathbf{x}_j - \mathbf{w}_i(k)], \tag{3}$$

where $c_{ij}(k) = 1$ if the jth feature vector is assigned to the ith neuron during the kth iteration, otherwise $c_{ij}(k) = 0$.The reference vector of any neuron at the end of the $(k+1)$th iteration of the training phase is a linear combination of the input vectors assigned to it during all the previous iterations:

$$\begin{aligned} \mathbf{w}_i(k+1) = \mathbf{w}_i(0)\prod_{n=1}^{k+1}[1 - a(n)c_{ij}(n)]^N + \sum_{v=1}^{k}\prod_{n=v}^{k}[1 - a(n+1)c_{ij} \\ \times(n+1)]^N\left[a(N)\sum_{b=1}^{N}[1 - a(v)c_{ij}(v)]^{N-b}c_{ib}(v)\mathbf{x}_b\right] \\ + a(k+1)\sum_{b=1}^{N}[1 - a(k+1)c_{ij}(k+1)]^{N-b} \\ \times c_{ib}(k+1)\mathbf{x}_b. \end{aligned} \tag{4}$$

Eq. (4) is proven in Appendix A.

Let us denote by $f_i(\mathbf{x})$, $i = 1, 2, \ldots, L$, the pdfs of the various data classes. If sample data from these classes are mixed to form the sample set with a priori probabilities ε_i, $i = 1, 2, \ldots, L$, such that $\sum_{i=1}^{L} \varepsilon_i = 1$, the sample set distribution has the form

$$f(\mathbf{x}) = \sum_{i=1}^{L} \varepsilon_i f_i(\mathbf{x}). \tag{5}$$

For the sake of the discussion simplicity, let us assume a mixture of two 1D Gaussian pdfs, $f_i(\mathbf{x})$. An important goal is to decompose such a mixture (Eq. (5)) into two Gaussian-like distributions. Nearest mean reclassification algorithms, such as the K-means may have a serious shortcoming, particularly when a mixture distribution consists of several overlapping distributions (Fukunaga, 1990). An important goal is to decompose a mixture into several Gaussian-like distributions. However, the clustering procedures decompose the mixture by using a properly defined threshold. As a result, the distribution of class 1 includes the tail of the distribution of class 2 and does not include the tail of the distribution of class 1. Accordingly, the estimated mean values from the 'truncated' distributions could be significantly different from the true ones. The same applies for the SOM whose threshold is simply the midpoint between the stationary weight vectors given by the conditional means (Ritter & Schulten, 1988):

$$\bar{\mathbf{w}}_i = \frac{\int_{\mathbf{X}_i(\bar{\mathbf{W}})} \mathbf{x} f(\mathbf{x})\mathrm{d}\mathbf{x}}{\int_{\mathbf{X}_i(\bar{\mathbf{W}})} f(\mathbf{x})\mathrm{d}\mathbf{x}}, \tag{6}$$

$$i = 1, 2, \ldots, L, \quad \bar{\mathbf{W}} = (\bar{\mathbf{w}}_1^{\mathrm{T}}|\cdots|\bar{\mathbf{w}}_L^{\mathrm{T}})^{\mathrm{T}},$$

where $\mathbf{X}_i(\bar{\mathbf{W}})$ is the Voronoi neighborhood of the ith neuron. Obviously, the samples from the tail of the distribution of class 2 are outliers, when the reference vector for class 1 is computed. Despite the nonlinear weights $c_{ij}(k)$, SOM employs a linear estimation of location. Accordingly, its robustness properties are poor in the presence of outliers (Huber, 1981; Lehmann, 1983). To overcome these problems and to enhance the performance of the basic SOM method, a variant of the SOM algorithm is studied that employs *multivariate order statistics* (Barnett, 1976). The inherited robustness properties of the order statistics allow this variant to treat efficiently the presence of outliers in the data set, as has been demonstrated in (Pitas, Kotropoulos, Nikolaidis, Yang, & Gabbouj, 1996).

3. Marginal median SOM

Order statistics have played an important role in the statistical data analysis and especially in the robust analysis of data contaminated with outlying observations (Pitas & Venetasanopoulos, 1990). The lack of any obvious and unambiguous extension of ordering multivariate observations has led to several sub-ordering methods such as

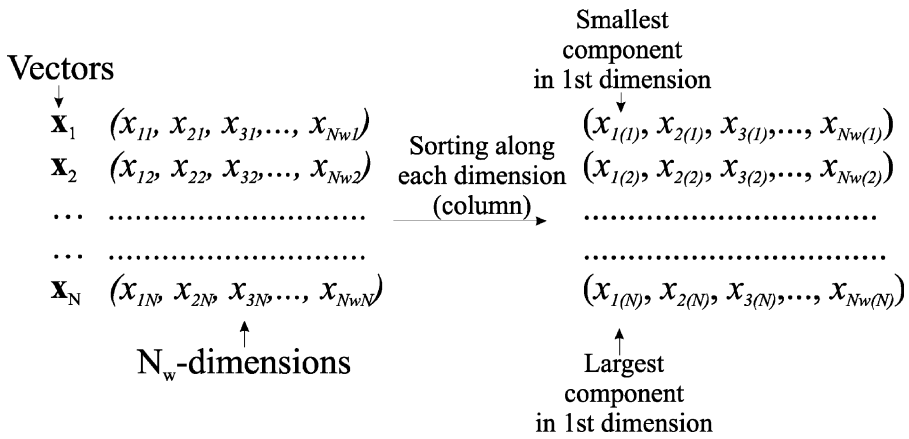

Fig. 1. The components of the feature vectors are column-wise sorted (each dimension independently). To the left, the vector components are not-ordered. To the right, the vector components are ordered along each of the N_w-dimensions.

marginal ordering, *reduced* (*aggregate*) *ordering*, *partial ordering* and *conditional* (*sequential*) *ordering*. A discussion on these principles can be found in Barnett (1976).

The SOM variant used in this paper relies on the concept of marginal ordering. The marginal ordering of N feature vectors, $\mathbf{x}_1, \ldots, \mathbf{x}_N$, where $\mathbf{x}_j = (x_{1j}, x_{2j}, \ldots, x_{N_w j}) \in \mathbb{R}^{N_w}$, is performed by ordering the vector components independently along each of the N_w-dimensions:

$$x_{q(1)} \leq x_{q(2)} \leq \cdots \leq x_{q(N)}, \qquad q = 1, 2, \ldots, N_w, \tag{7}$$

with q denoting the index of a component inside the feature vector. In Eq. (7) $x_{q(j)}$ is the so-called jth *order statistic*. The component-wise ordering is depicted in Fig. 1. Then, the marginal median, $\mathbf{x}_{\text{med}}$, of N feature vectors is defined by

$$\mathbf{x}_{\text{med}} = \text{marginal\_median}\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \triangleq$$
$$= \begin{cases} (x_{1(\nu+1)}, x_{2(\nu+1)}, \ldots, x_{N_w(\nu+1)})^{\mathrm{T}} & \text{for } N = 2\nu+1 \\ \left(\dfrac{x_{1(\nu)} + x_{1(\nu+1)}}{2}, \ldots, \dfrac{x_{N_w(\nu)} + x_{N_w(\nu+1)}}{2} \right)^{\mathrm{T}} & \text{for } N = 2\nu \end{cases}. \tag{8}$$

The concept of the marginal median is applied to the basic SOM algorithm in the following way. Let $\mathbf{X}_i(k-1)$ denote the ith Voronoi set, $i = 1, 2, \ldots, L$, until the $(k-1)$th iteration. That is,

$$\mathbf{X}_i(k-1) = \{\mathbf{x}_j \in \mathscr{X} \,|\, \|\mathbf{x}_j - \mathbf{w}_i(k-1)\| < \|\mathbf{x}_j - \mathbf{w}_l(k-1)\|, \; l = 1, 2, \ldots, i-1, i+1, \ldots, L\} \cup \mathbf{X}_i(k-2), \tag{9}$$

under the condition $\mathbf{X}_i(0) = \varnothing$.

At the kth iteration, the winning reference neuron, $\mathbf{w}_s(k)$, corresponding to a given feature vector $\mathbf{x}_j$ is identified by using Eq. (1). The winner is then updated by

$$\mathbf{w}_s(k+1) = \text{marginal\_median}\{\mathbf{x}_j \cup \mathbf{X}_s(k-1)\}, \tag{10}$$

where the marginal median operator is given by Eq. (8). Thus all the previously assigned feature vectors to the winner neuron as well as the current feature vector $\mathbf{x}_j$ are used in the computation of the marginal median. Accordingly, all past class assignment sets $\mathbf{X}_i(k)$, $i = 1, 2, \ldots, L$, are needed.

The neighboring neurons, $i \in \mathscr{N}_s(k)$, are updated using

$$\mathbf{w}_i(k+1) = \text{marginal\_median}\{a(k)\mathbf{x}_j \cup \mathbf{X}_i(k-1)\}, \tag{11}$$

in order to achieve global ordering. The parameter $a(k)$ in Eq. (11) admits a value in (0, 1) and has the following effect: at the beginning of the training phase the parameter is significantly larger than zero and allows to the feature vector $\mathbf{x}_j$ to participate in the updating of the neighboring neurons. In the lapse of time, $a(k)$ tends toward zero and $a(k)\mathbf{x}_j$ no longer affects the reference vector of the neighboring classes. Table 1 summarizes the proposed *Marginal Median SOM* (MMSOM) variant.

For relatively large data collections, a drawback of the MMSOM is the computational complexity with respect to the identification of the marginal median vector in Eq. (8) and the updating of both the winner neuron and its neighbors. To overcome this problem, two alternative shortcuts are proposed. In the first shortcut, the real-valued data are being quantized into 256 quantization levels. Subsequently, a modification of the *running median* algorithm is employed (Hung et al., 1979; Pitas et al., 1996). The algorithm uses the histogram of past feature vector assignment to each neuron for each data dimension. The histograms are being constantly updated as new feature vectors are assigned to each neuron. The main advantage of this approach is the computational savings at the cost of

Table 1
Overview of the marginal median SOM

Linear initialization of the reference vectors $\mathbf{w}_i(0)$, $i = 1, 2, \ldots, L$
Initialize the Voronoi set of each reference vector, that is, $\mathbf{X}_i(0) = \varnothing$
For each iteration, $k = 1, 2, \ldots$
  For each feature vector $\mathbf{x}_j$:
    Find the winning reference vector according to Eq. (1)
    Update the winning reference vector according to Eq. (9)
    Update the winning reference vector using Eq. (10)
    Update also the reference vectors of the neighboring neurons, $i \in \mathscr{N}_s(k)$, according to Eq. (11)

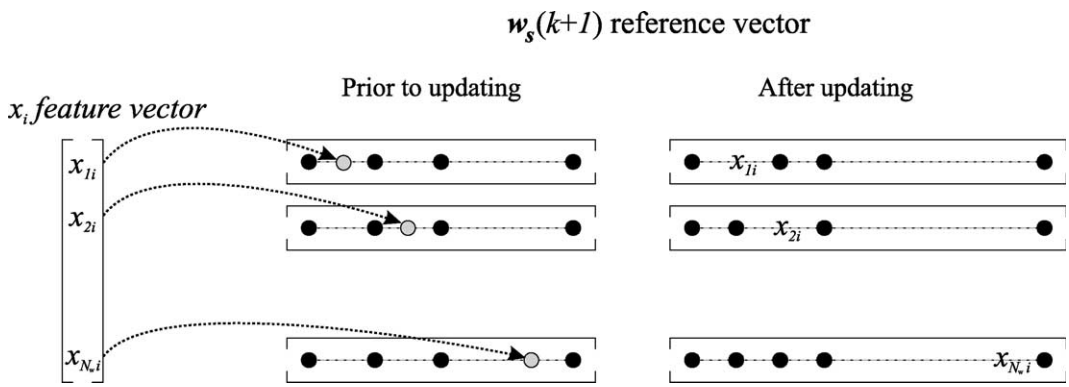

Fig. 2. For each component of an 'unseen' feature vector x_i the correct position is identified using binary search and the component is inserted to the appropriate position.

quantization errors. This variant shall be referred to as the *Marginal Median Quantized* SOM (MMQ-SOM).

The second shortcut avoids any quantization. Each neuron is equipped with a dynamically expanding matrix that stores the feature vectors assigned to it. In this matrix, the number of rows equals the dimensionality of the input patterns and the number of its columns equals the number of feature vectors assigned to the neuron since the beginning of the training phase. Each row (dimension) is sorted into ascending order. When a new feature vector is assigned to a particular class, for each vector component, the 'correct' position inside the row is located using binary search according to Eq. (7), and the component is inserted at this particular position. The sole drawback of this approach is the memory required to store all the available training 'history' for each neuron. The aforementioned shortcut will be termed as the *Marginal Median Without Quantization* SOM (MMWQ-SOM). Fig. 2 briefly depicts the just described procedure.

4. Marginal median SOM application to document retrieval

The performance evaluation of the proposed variant against the basic SOM method is described here for document retrieval. The training has been performed on two corpora, namely the *Hypergeo* corpus (described subsequently) and the *Reuters-21578* corpus (Lewis, 1997). The objective is to divide the corpora into contextually related document classes and then query these classes using sample query-documents, to find the closest document class. The major advantage of the SOM approach is that it can handle both keyword- as well as document-based queries since both of them can be represented by a vector that has to be assigned to a class formed during the training phase. In Section 4.1 we briefly describe the corpora and quote some statistics related to them. In Section 4.2 the vector space model encoding of the word stems into feature vectors is presented. These vectors are clustered using both the basic SOM and the proposed variant to construct classes of semantically related words. Finally, in Section 4.3 the resulted word classes are exploited in order to encode the documents with numerical vectors and both algorithms are used to cluster them into contextually related classes.

4.1. Corpus description and preprocessing steps

The Hypergeo corpus comprises 606 *HTML* files manually collected over the Internet. These files are web pages of touristic content mostly from Greece, Spain, Germany, and France. They were collected during the European Union funded project HYPERGEO. The selected files are annotated by dividing them into 18 categories related to tourism, such as accommodation, history, geography, etc., so that a ground truth is incorporated into the files.

The second corpus, is the Distribution 1.0 of the Reuters-21578 text categorization collection compiled by Lewis (1997). It consists of 21578 documents which appeared on the Reuters newswire in 1987. The documents are marked up using *SGML* tags and are manually annotated according to their content into 135 topic categories. Fig. 3 depicts

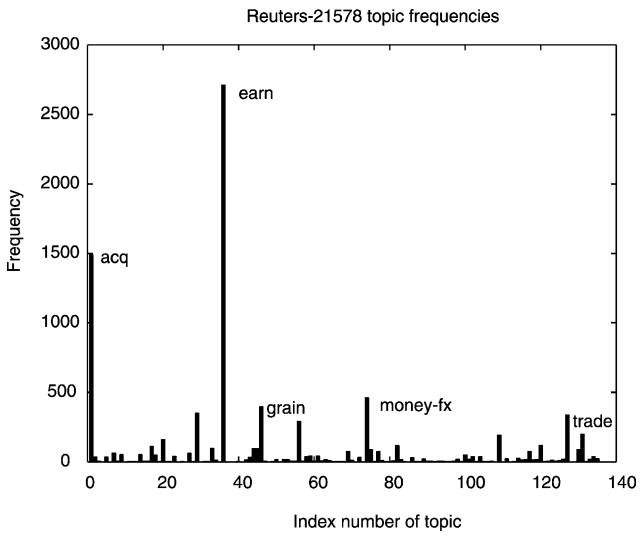

Fig. 3. The frequencies of the topics in the Reuters-21578.

the topic frequencies, with the topics being arranged into lexicographical order.

Due to the nature of the SOM algorithm, a series of actions are taken in order to encode the words into numerical vectors. During the first step, the HTML and SGML tags and entities are removed. Subsequently, plain text cleaning is performed. Text cleaning refers to the removal of URLs, email addresses, numbers, and punctuation marks. The sole punctuation mark left intact is the full stop which is preserved in order to provide a sentence delimiter. This is done because the context for a given word is confined by the limits of the sentence. Furthermore, the collocations (i.e. expressions consisting of two or more words) are meaningful only within the limits of a sentence (Manning & Schüte, 1999). Stopping is also performed so that some common English words such as articles, determiners, prepositions, pronouns, conjunctions, complementizers, abbreviations and some frequent non-English terms are removed.

Subsequently, *stemming* is performed. Stemming refers to the elimination of word suffixes, to shrink the vocabulary without significantly altering the context. It can be considered as an elementary clustering technique, with the word roots (stems) forming distinct classes. The underlying assumption for the successful usage of a stemming program, called *stemmer*, is that the morphological variants of words are semantically related (Frakes & Baeza-Yates, 1992). The commonly used Porter stemmer was applied to both corpora (Porter, 1980).

Finally, prior to encoding the word stems into vectors, the stems, whose frequency was below a certain threshold were eliminated. For both corpora the threshold was set to 20. Table 2 depicts their statistics. The third column of Table 2 contains the number of documents that were used after the completion of all the aforementioned preprocessing steps. It must be noted that the number of retained documents in the Reuters-21578 corpus is nearly 12% lower than its initial value. This is due to the fact that some documents did not contain textual information to start with or lost all their textual information due to the preprocessing and the thresholding steps. Furthermore, the resulting Reuters-21578 corpus was partitioned into two distinct sets, a training set and a test set, according to the recommended *Modified Apte* split of the collection (Lewis, 1997). The first set was used for document clustering during the training phase of the algorithms, whereas the second one was used to assess the quality of document clustering through retrieval experiments that employ its documents as query-documents during the test phase.

4.2. Feature vector construction

When encoding the textual data into numerical vectors one must take into account for every encoded word its preceding and the following words. This is the well known *n-gram* modeling, where the notion n denotes the number of preceding and succeeding words taken into consideration when encoding a specific word. When this model is used, the contextual statistics for every word stem in the corpus must be computed. For this purpose, the second version of the CMU-Cambridge Statistical Language Modeling Toolkit was used (Clarkson & Rosenfeld, 1997). In a first attempt, the following maximum likelihood estimates of conditional probabilities can be used to encode the jth word stem in the vocabulary:

$$x_{jl} = \frac{n_{jl}}{N_j}, \qquad l = 1, 2, \ldots, N, \tag{12}$$

where n_{jl} is the number of times the pair (jth word stem, lth word stem) occurred in the corpus, N_j is the number of times the jth word stem occurred in the corpus, and N is the number of word stems in the vocabulary. Let $\mathbf{e}_j$ denote the $(N \times 1)$ unit vector having one in the jth position and zero elsewhere. By using Eq. (12), the following word vectors, $\tilde{\mathbf{x}}_j$, can be computed:

$$\tilde{\mathbf{x}}_j = \frac{1}{N_j} \begin{bmatrix} \sum_{\substack{l=1 \\ l \neq j}}^{N} n_{lj}\mathbf{e}_l \\ \beta \mathbf{e}_j \\ \sum_{\substack{m=1 \\ m \neq j}}^{N} n_{jm}\mathbf{e}_m \end{bmatrix}. \tag{13}$$

The upper vector part in Eq. (13) encodes the 'average' context prior to the ith word (history), whereas the lower vector part encodes the 'average' context after the jth word. Furthermore, β is a small scaling factor ($\beta \simeq 0.2$).

Due to the high-dimensional nature of the textual data, the vectors derived from Eq. (13) have exceptionally high dimensionality ($3N - 2$ dimensions). This problem must be tackled by dimensionality reduction to N_w ($N_w \ll 3N - 2$), which can be achieved by the linear projection $\mathbf{x}_j = \mathbf{\Phi}\tilde{\mathbf{x}}_j$. Kaski et al. suggested a suboptimal approach to the previous problem using a random matrix $\mathbf{\Phi}$ that has the following properties (Kaski, 1998):

Table 2
Corpora statistics

Corpus	Number of original documents	Number of retained documents	Word tokens	Stem types	
				Before thresholding	After thresholding
Hypergeo	606	606	290,973	16,397	1524
Reuters-21578	21,578	19,043	2,642,893	28,670	4671

1. The components in each column are chosen to be independent, identically distributed Gaussian variables with zero mean and unit variance.
2. Each column is normalized to unit norm.

4.3. Clustering

After the preprocessing phase and the construction of the word feature vectors, $\mathbf{x}_j$, we perform training for both the basic SOM method and the two proposed implementations of the MMSOM variant. In each case, the feature vectors are presented iteratively an adequate number of times to the neural networks which perform clustering in an effort to build word classes containing semantically related words. This is based on empirical and theoretical observations that semantically related words have more or less the same preceding and succeeding words.

The above process yields the so-called *word categories map* (WCM) (Kohonen, 1998). The WCMs computed using MMWQ-SOM can be seen in Fig. 4 for the Hypergeo corpus and in Fig. 5 for the Reuters-21578 corpus. Each hexagon on these maps corresponds to one word class. The grey levels on the maps correspond to different word densities. Hexagons with grey levels near 255 (white color) imply that fewer word stems have been assigned to these neurons, whereas, grey levels near 0 (black color) imply larger densities. The word categories of some characteristic nodes can also be seen on the maps. For instance, classes containing words related to 'accommodation' and 'sightseeing' are highlighted in Fig. 4. In Fig. 5, the highlighted nodes correspond to classes related to 'finance', 'oil', and 'energy'.

Subsequently, for each document in the corpus, a histogram of word classes is computed to form the so-called *document vector* $\mathbf{a}_j$. The histogram is calculated as follows. For each word stem in a document, the WCM neuron is found where it was classified to. The histogram value is increased by one for this word class. An example is shown in Fig. 6.

After the computation of the document vectors the basic SOM method as well as its MMSOM variants are used to cluster them. The document vectors substitute the feature vectors in both algorithms, i.e. $\mathbf{x}_j = \mathbf{a}_j$.

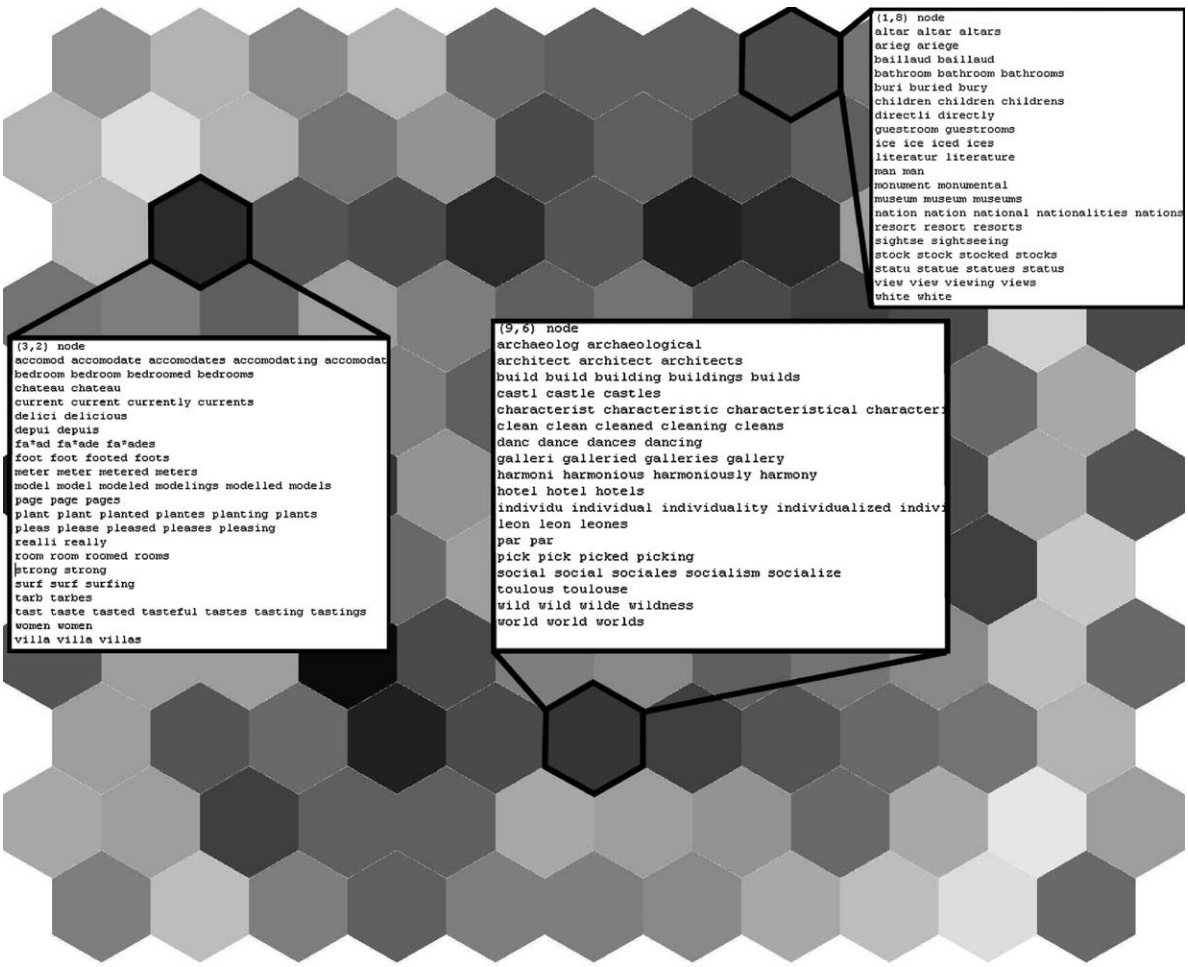

Fig. 4. Word categories map using the MMWQ-SOM for the Hypergeo corpus on a 11×11 neural network. The highlighted neurons correspond to word categories related to 'accommodation' (left) and 'sightseeing' (middle and right).

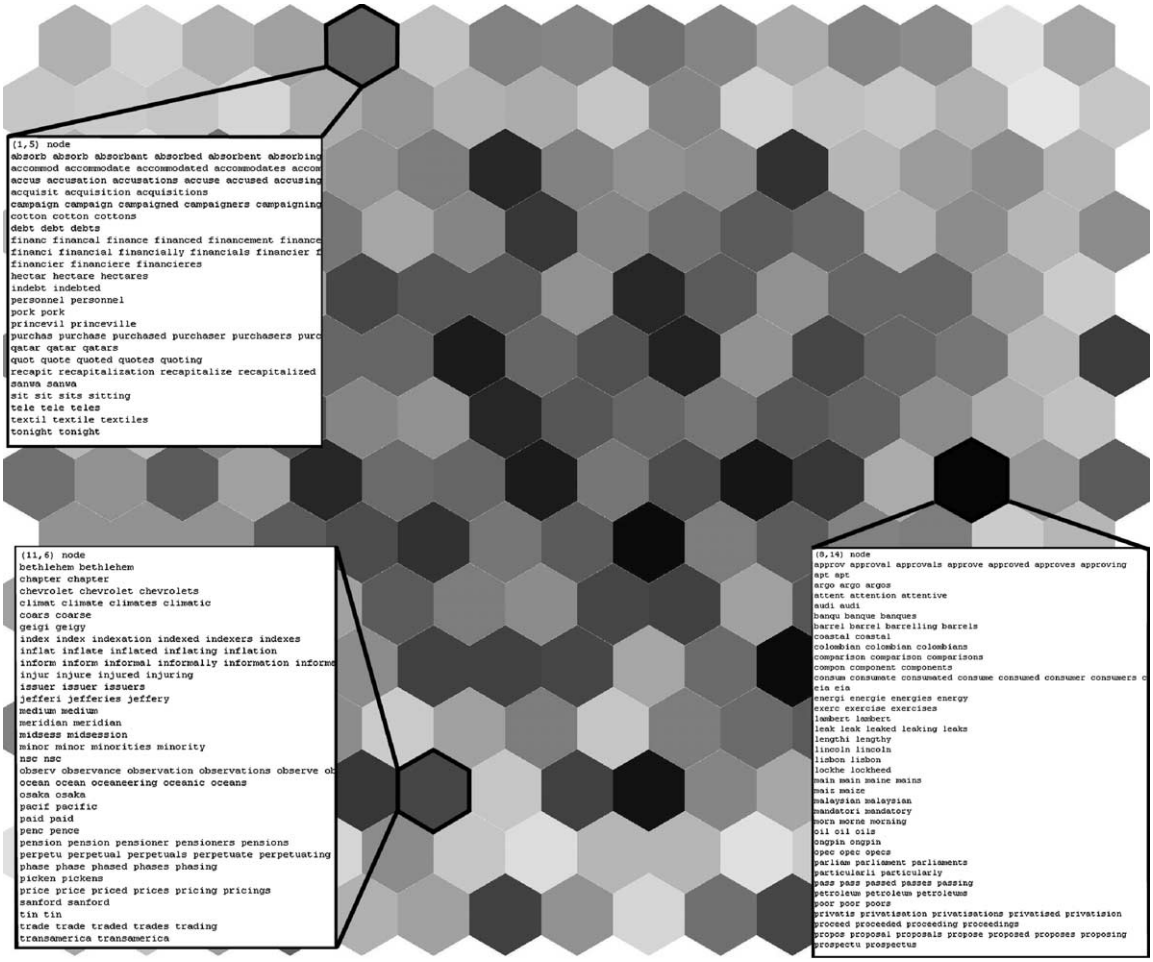

Fig. 5. Word categories map using the MMWQ-SOM for the Reuters-21578 corpus on a 15 × 15 neural network. The highlighted neurons correspond to word classes related to ‘finance’ (top left), ‘oil’ and ‘energy’ (bottom right).

It is expected that the constructed document classes contain contextually similar documents. The resulting map is called *document map* (DM) (Kohonen, 1998). The DM computed by the MMWQ-SOM for the Reuters-21578 corpus is depicted in Fig. 7 and the corresponding one for the Hypergeo corpus can be seen in Fig. 8. The highlighted nodes in the Reuters’ DM correspond to classes containing documents related to ‘debts’ and ‘economic revenues’. In the DM corresponding to the Hypergeo corpus, the highlighted neurons associated to clusters of web pages related to ‘sightseeing in Dresden’ and ‘mountains’.

The computed DM is the output of the training phase. An important advantage regarding such a system is the inherent ability for handling document-based queries. During the recall phase document-based queries are tested. That is, instead of using keywords as input to the retrieval system one can use full-text documents. The sample document used in a query undergoes all the preprocessing steps and then, with the help of the WCM computed during the training phase, the corresponding document vector $\mathbf{a}_j$ is computed. The document vector corresponds to the feature vector in Eq. (1). The neuron whose reference vector minimizes Eq. (1) represents with high probability the class which contains the most relevant documents to the query document in the corpus.

5. Experimental results

The performance of the MMSOM against the basic SOM method is measured using the MSE between the reference vectors and the document vectors assigned to each neuron in the training phase. Furthermore, the recall–precision performance is measured using query-documents from the test set during the recall phase is used as an indirect measure of the quality of document organization provided by both algorithms. Fig. 9 depicts the MSE curves during the formation of the WCM using the basic SOM architecture and the marginal median variant without quantization for the Hypergeo corpus. Similar MSE curves are plotted in Fig. 10 that correspond to the training phase of both

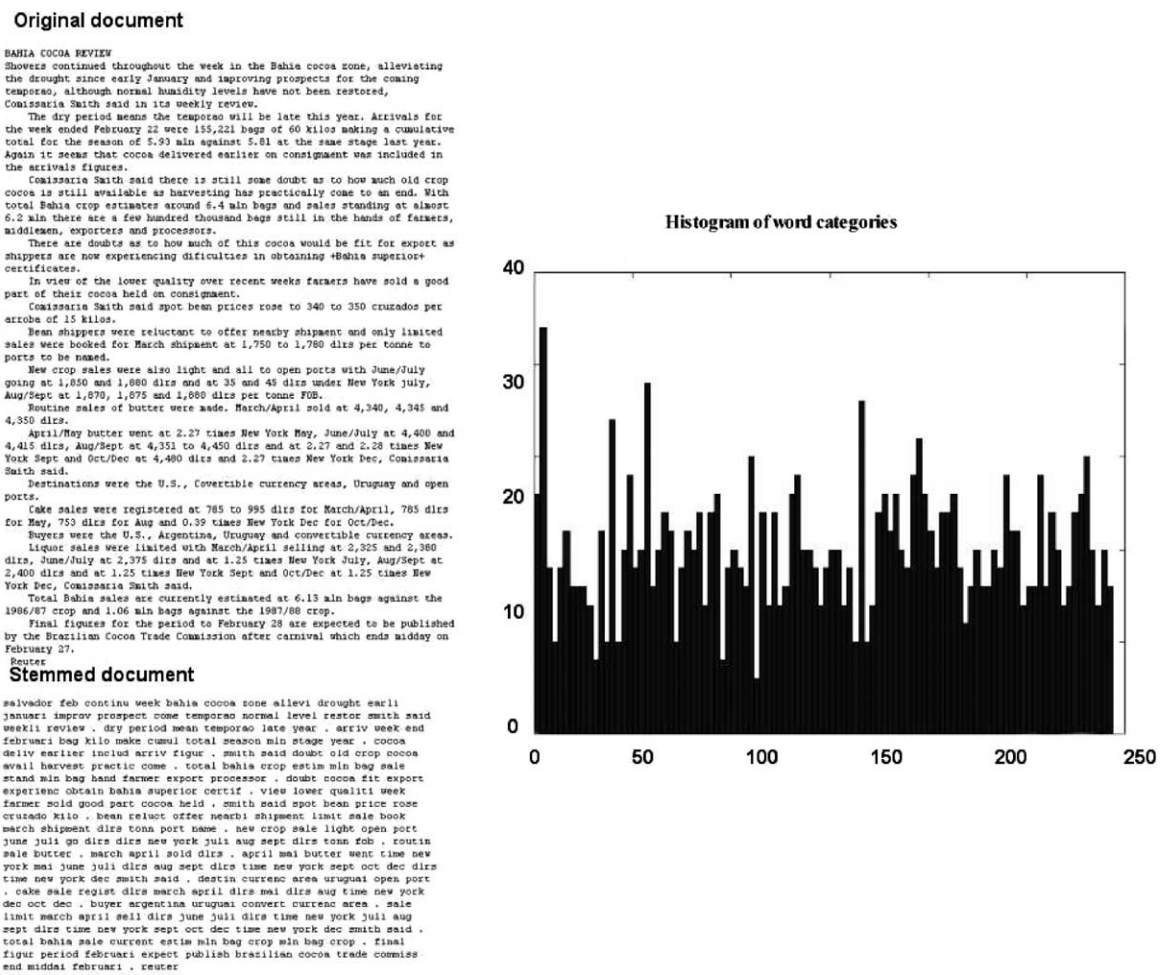

Fig. 6. The three distinct steps in the formation of the document vector $\mathbf{a}_j$. From the raw textual data (top left) to the stemmed document (bottom left) and the histogram of the word categories (middle right).

algorithm when the Reuters-21578 corpus is used. Both algorithms were initialized in the same way. It must be noted that even from the beginning of the training phase, the marginal median SOM outperforms the basic SOM algorithm. This can be explained by the presence of many outliers in the early iterations of the training procedure. The outlier rejection of the marginal median operator reduces quickly the initial MSE which is the same for both algorithms. During the formation of the WCM, the number of training iterations needed by the basic SOM so that the MSE drops to the e^{-1} of its initial value was nearly 15% higher than the MMWQ-SOM. Regarding the execution time for the completion of the training phase, the basic SOM completed the process nearly 22% faster than the proposed variant due to the computational cost of the marginal median operator.

Aiming at assessing the retrieval performance of the MMWQ-SOM against that of the basic SOM two retrieval systems were trained using the available corpora. For comparison purposes, we also trained a system using the batch SOM algorithm (Kohonen, 1997). Afterwards, the systems were queried using the same query-documents for each corpus. For each document-based query, the system retrieves those training documents that are represented by the best matching neuron of the DM. Subsequently, the training documents retrieved are ranked according to their Euclidean distance from the test document. Finally, the retrieved documents are classified as being either relevant or not to the query-document with respect to the annotation category they bear. Table 3 is the 2×2 contingency table which shows how the collection of retrieved documents is divided (Korfhage, 1997). In Table 3, n_1 denotes the total number of relevant documents in the training corpus, n_2 is the number of retrieved training documents, and r corresponds to the number of relevant documents that are retrieved.

To measure the effectiveness of a retrieval system two widely used ratios are employed: the *precision* and the *recall* (Korfhage, 1997). Precision is defined as the proportion of retrieved documents that are relevant:

$$P = \frac{r}{n_2}, \tag{14}$$

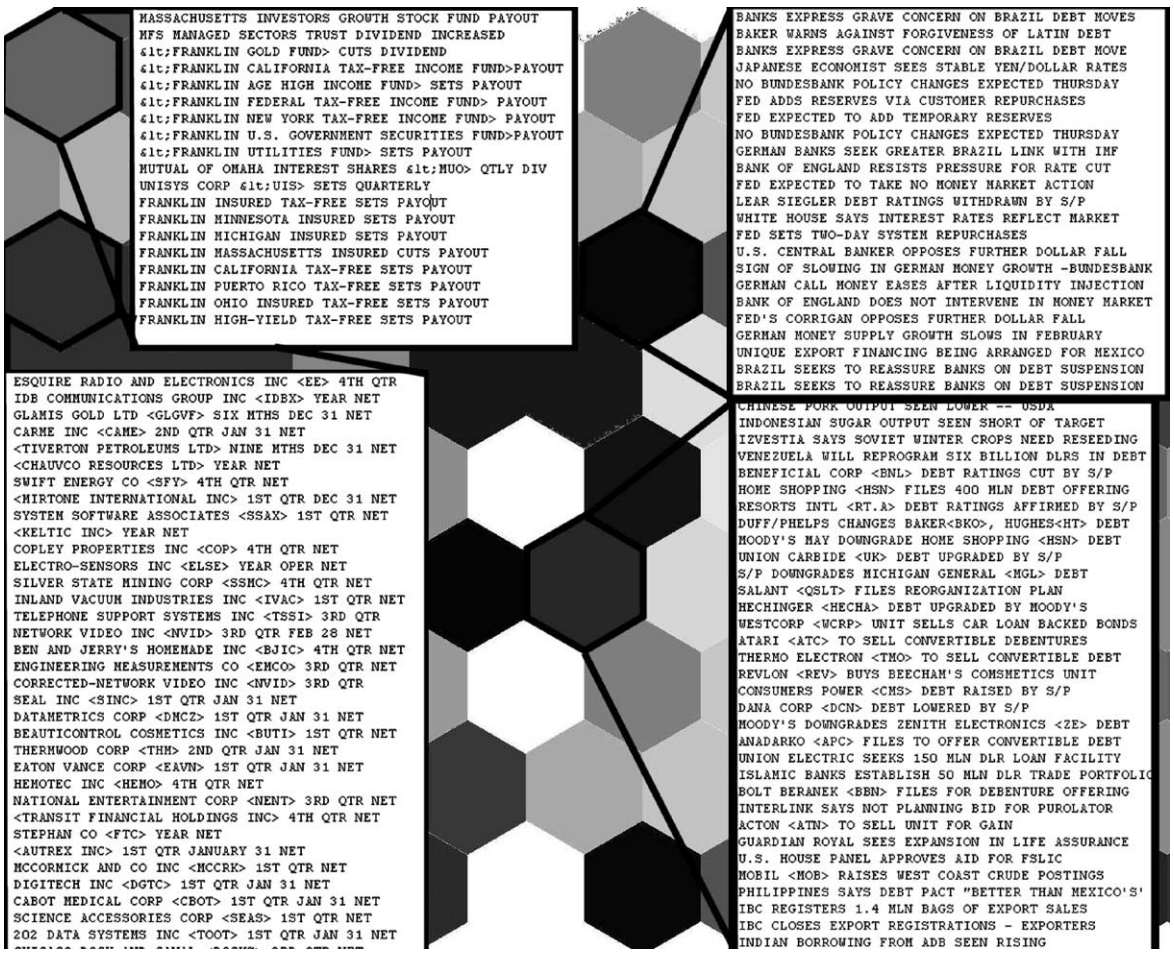

Fig. 7. The document map constructed for the Reuters-21578 corpus for a 9×9 neural network using the MMWQ-SOM. The document titles are listed for each document class.

Recall is the proportion of relevant documents that are retrieved:

$$R = \frac{r}{n_1}. \tag{15}$$

As the volume of retrieved documents increases the above ratios are expected to change. The sequence of $(recall, precision)$ pairs obtained yields the so-called *recall–precision* curve. Each query-document in the test set produces one recall–precision curve. An average over all the curves corresponding to query documents of the same topic obtained from the test set produces the *average* recall–precision curve (Korfhage, 1997). If the recall level does not equal one we proceed with the second best winner neuron and repeat the same procedure and so on. The comparison of the effectiveness between the retrieval system utilizes the above-mentioned curve. Fig. 11a and b depicts the average recall–precision curves for the basic SOM, the batch SOM, and the MMWQ-SOM architecture for 'Mergers and Acquisitions (ACQ)' and 'Earnings and Earnings Forecasts (EARN)' topics from Reuters corpus. It can be seen that the marginal median variant performs better than the basic and batch SOM for a wide range of recall volumes. More specifically, the performance of the marginal median is superior to the basic SOM as well as the batch SOM in small recall volumes ($R < 0.2$), which is extremely important given the fact that an average user is interested in high precisions ratios even from the beginning of the list of returned relevant documents.

Fig. 12 depicts the recall–precision curves for the Hypergeo corpus of the basic SOM, the batch SOM as well as the MMSOM without quantization variant (MMWQ-SOM). The MMWQ-SOM architecture is found again to be superior to the other two SOM architectures with respect to recall–precision curves.

Moreover, we have compared the average precision of the MMSOM to that of the SOM document map implementation reported for the CISI collection in (Lagus, 2002)

Fig. 8. The document map constructed for the Hypergeo corpus for a 7×7 neural network using the MMWQ-SOM. The document titles as well as their respective URL addresses are listed for each class.

under the same experimental set-up. A 1.4% higher average precision was achieved by the document map of the MMSOM compared to that of the SOM document map in Langus (2002) for 50 retrieved documents. The corresponding improvements in the average precision against Salton's vector space model and the latent semantic indexing were 1.6 and 3.2%, respectively, for 50 retrieved documents.

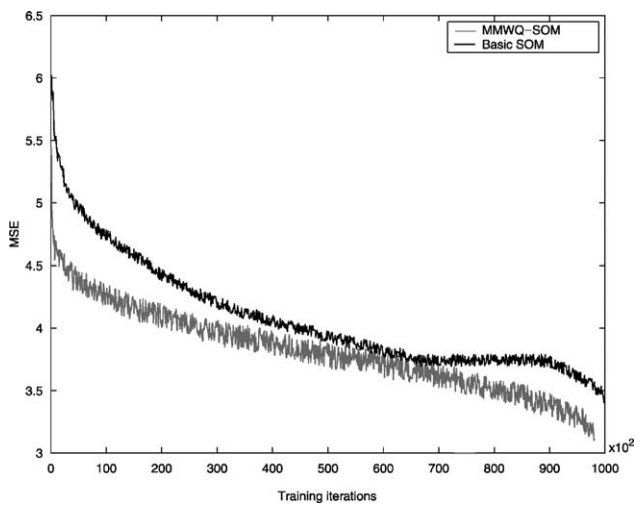

Fig. 9. The mean squared error curves for the basic SOM and the MMWQ-SOM variant in a 11×11 neural network using the Hypergeo corpus.

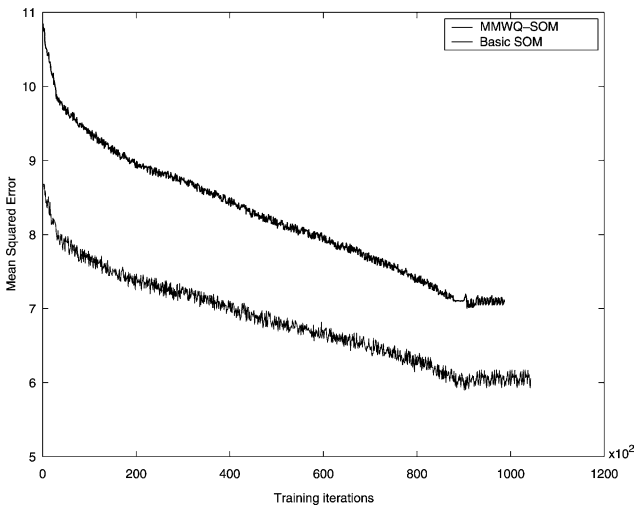

Fig. 10. The mean squared error curves for the basic SOM and the MMWQ-SOM variant using a 15×15 neural network for the Reuters-21578 corpus.

Table 3
Contingency table for evaluating retrieval

	Retrieved	Not retrieved	
Relevant	r	x	$n_1 = r + x$
Not relevant	y	z	
	$n_2 = r + y$		

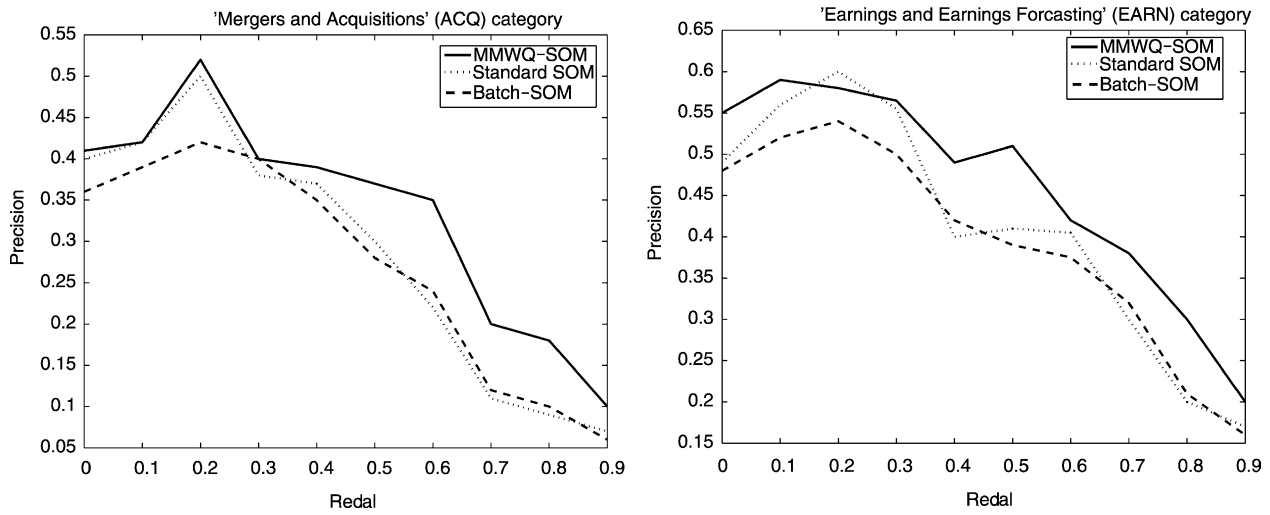

Fig. 11. (a) The average recall–precision curves for the basic SOM, the batch SOM and the MMWQ-SOM variant for the 'Mergers and Acquisitions (ACQ)' category of the Reuters-21578 corpus, respectively. (b) The average recall–precision curves for each one of the architectures for the 'Earnings and Earnings Forecasts (EARN)' category of the Reuters-21578 corpus.

6. Conclusions

The inherent drawbacks of the SOM algorithm with respect to the treatment of data outliers in the input space and the suboptimal estimation of the class means has given impetus to the development of a SOM variant that utilizes the marginal median and is capable to handle these drawbacks. Two implementations of the SOM variant that employ the multivariate median operator in order to update the reference vectors of the neurons have been discussed. A superior performance of the proposed variant with respect to the MSE curve related to the training phase of the algorithm, and the average recall–precision curve related to the retrieval effectiveness during the test phase has been demonstrated, when the basic SOM algorithm is replaced by the proposed MMSOM for document organization and retrieval.

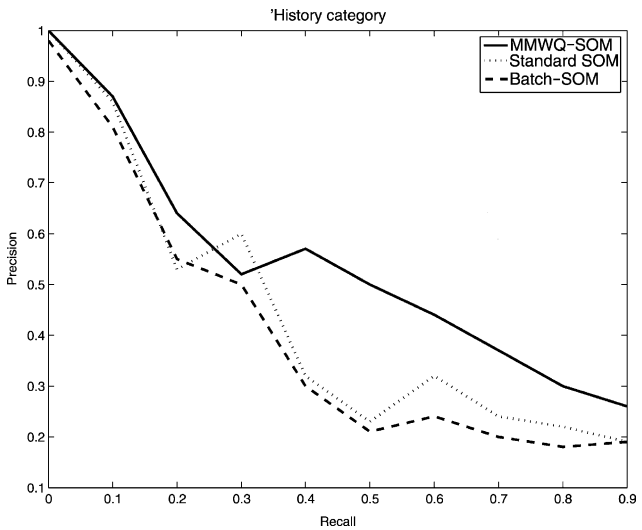

Fig. 12. The average recall–precision curves for each technique for the Hypergeo corpus. The sample test document was classified into the 'history' category.

Acknowledgements

The authors would like to thank their colleagues, G. Albanidis and N. Bassiou, Aristotle University of Thessaloniki, Greece, for their contribution in the formation of the Hypergeo corpus. This work was supported by the European Union IST Project 'HYPERGEO: Easy and friendly access to geographic information for mobile users' (IST-1999-11641).

Appendix A

In proving Eq. (4) some modifications are made in the definition and the notation of the reference vector. That is,

during the kth iteration of the algorithm, and for the jth feature vector, the reference vector $\mathbf{w}_i(k)$ is updated using the following equation:

$$\mathbf{w}_{ij}(k) = \mathbf{w}_{i,j-1}(k) + a(k)c_{ij}(k)[\mathbf{x}_j - \mathbf{w}_{i,j-1}(k)]$$
$$= \mathbf{w}_{i,j-1}(k)m(k) + a(k)c_{ij}(k)\mathbf{x}_j, \quad \text{(A.1)}$$

where $m(k) = [1 - a(k)c_{ij}(k)]$. The additional index in the definition of the reference vector is used to denote the last feature vector used to update the ith reference vector. For simplicity reasons we introduce the notation:

$$B_N(k) = a(k)\sum_{b=1}^{N} m^{N-b}(k)c_{ib}(k)\mathbf{x}_b. \quad \text{(A.2)}$$

Induction is used to prove Eq. (4). For $k = 1$ and

$$j = 1: \qquad \mathbf{w}_{i1}(1) = \mathbf{w}_{i0}(1)m(1) + a(1)c_{i1}(1)\mathbf{x}_1 \quad \text{(A.3)}$$

$j = 2$:

$$\mathbf{w}_{i2}(1) = \mathbf{w}_{i1}(1)m(1) + a(1)c_{i2}(1)\mathbf{x}_2$$
$$= \mathbf{w}_{i0}(1)m^2(1) + a(1)m(1)c_{i1}(1)\mathbf{x}_1 + a(1)c_{i2}(1)\mathbf{x}_2$$
$$= \mathbf{w}_{i0}(1)m^2(1) + B_2(1) \quad \text{(A.4)}$$

$$\vdots$$

$$j = N: \qquad \mathbf{w}_{iN}(1) = \mathbf{w}_{i0}(1)m^N(1) + B_N(1). \quad \text{(A.5)}$$

In the transition phase of kth iteration to the $(k+1)$th the following boundary condition is applied: $\mathbf{w}_{i0}(k+1) = \mathbf{w}_{iN}(k)$. Furthermore, $\mathbf{w}_{i0}(1) = \mathbf{w}_i(0)$. For $k+1 = 2$ and $j = 1$ we have

$$\mathbf{w}_{i1}(2) = \mathbf{w}_{i0}(2)m(2) + B_1(2) = \mathbf{w}_{iN}(1)m(2) + B_1(2)$$
$$= (\mathbf{w}_{i0}(1)m^N(1) + B_N(1))m(2) + B_1(2)$$
$$= \mathbf{w}_{i0}(1)m^N(1)m(2) + B_N(1)m(2) + B_1(2), \quad \text{(A.6)}$$

and finally

$$\mathbf{w}_{iN}(2) = \mathbf{w}_{i0}(1)m^N(1)m^N(2) + B_N(1)m^N(2) + B_N(2). \quad \text{(A.7)}$$

At the end of the $(k+1)$th iteration we get

$$\mathbf{w}_{iN}(k+1) = \mathbf{w}_{i0}(1)\prod_{n=1}^{k+1} m^N(n) + \sum_{v=1}^{k}\prod_{n=v}^{k} m^N(n+1)B_N(v)$$
$$+ B_N(k+1). \quad \text{(A.8)}$$

By substituting $m(k)$ and $B_N(\cdot)$ into Eq. (A.8) we obtain Eq. (4).

References

Barnett, V. (1976). The ordering of multivariate data. *Journal of the Royal Statistical Society A*, *139*(3), 318–354.

Clarkson, P., & Rosenfeld, R. (1997). *Statistical language modeling using the CMU-Cambridge toolkit. Proceedings of the Eurospeech'97*, pp. 2707–2710.

Erwin, E., Obermayer, K., & Schulten, K. (1992). Self-organizing maps: ordering, convergence properties and energy functions. *Biological Cybernetics*, *67*, 47–55.

Fort, J.-C., Letremy, P., & Cottrell, M. (2002). *Advantages and drawbacks of the Batch Kohonen algorithm. Proceedings of the 10th European Symposium on Artificial Neural Networks (ESANN).*

Frakes, W. B., & Baeza-Yates, R. (1992). Information retrieval: data structures and algorithms. Upper Saddle River: Prentice-Hall.

Fukunaga, K. (1990). Introduction to statistical pattern recognition (2nd ed.). San Diego: Academic Press.

Haykin, S. (1999). Neural networks: a comprehensive foundation. Upper Saddle River, NY: Prentice-Hall.

Huang, T. S., Yang, G. J., & Tang, G. Y. (1979). A fast two-dimensional median filtering algorithm. *IEEE Transactions on Accoustics, Speech and Signal Processing*, *27*(1), 13–18.

Huber, P. J. (1981). Robust statistics. New York: Wiley.

Kangas, J. A., Kohonen, T., & Laaksonen, J. T. (1990). March Variants of self-organizing maps. *IEEE Transactions on Neural Networks*, *1*(1), 93–99.

Kaski, S. (1998). *Dimensionality reduction by random mapping: Fast similarity computation for clustering (Vol. 1). Proceedings of IJCNN'98*, IEEE, pp. 413–418.

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, *78*, 1464–1480.

Kohonen, T. (1997). Self organizing maps. Berlin: Springer.

Kohonen, T. (1998). *Self-organization of very large document collections: State of the art (Vol. 1). Proceedings of ICANN*, pp. 65–74.

Kohonen, T., Kaski, S., Lagus, K., Salojärvi, J., Paatero, V., & Saarela, A. (2000). Organization of a massive document collection. *IEEE Transactions on Neural Networks*, *11*(3), 574–585.

Kohonen, T., Kaski, S., Lagus, K., Salojrvi, J., Honkela, J., Paatero, V., & Saarela, A. (1999). *Self organization of a massive text document collection. Kohonen Maps*, Amsterdam: Elsevier, pp. 171–182.

Korfhage, R. R. (1997). Information storage and retrieval. New York: Wiley.

Lagus, K. (2002). Text retrieval using self-organized document maps. *Neural Processing Letters*, *15*(1), 21–29.

Lehmann, E. L. (1983). Theory of point estimation. New York: Wiley.

Lewis, D. D (1997). *Reuters-21578 text categorization test collection, distribution 1.0*. http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html

Manning, D., & Schütze, H. (1999). Foundations of statistical natural language processing. Cambridge, MA: MIT Press.

Pitas, I., Kotropoulos, C., Nikolaidis, N., Yang, R., & Gabbouj, M. (1996). Order statistics learning vector quantizer. *IEEE Transactions on Image Processing*, *5*(6), 1048–1053.

Pitas, I., & Venetsanopoulos, A. N. (1990). Nonlinear digital filters: Principles and applications. Dordrecht: Kluwer.

Porter, M. (1980). An algorithm for suffix stripping. *Program*, *14*(3), 130–137.

Ritter, H., & Schulten, K. (1988). Convergence properties of Kohonen's topology conserving maps: fluctuation, stability, and dimension selection. *Biological Cybernetics*, *60*, 59–71.

Yates, R. B., & Neto, B. R. (1999). Modern information retrieval. New York: ACM Press.