Cross-entropy embedding of high-dimensional data using the neural gas model

Pablo A. Estévez[a,*], Cristián J. Figueroa[a], Kazumi Saito[b]

[a]*Department of Electrical Engineering, University of Chile, Casilla 412-3, Santiago, Chile*
[b]*NTT Communication Science Laboratories, 2-4 Hikaridai, Seika, Kyoto 619-0237, Japan*

Abstract

A cross-entropy approach to mapping high-dimensional data into a low-dimensional space embedding is presented. The method allows to project simultaneously the input data and the codebook vectors, obtained with the Neural Gas (NG) quantizer algorithm, into a low-dimensional output space. The aim of this approach is to preserve the relationship defined by the NG neighborhood function for each pair of input and codebook vectors. A cost function based on the cross-entropy between input and output probabilities is minimized by using a Newton–Raphson method. The new approach is compared with Sammon's non-linear mapping (NLM) and the hierarchical approach of combining a vector quantizer such as the self-organizing feature map (SOM) or NG with the NLM recall algorithm. In comparison with these techniques, our method delivers a clear visualization of both data points and codebooks, and it achieves a better mapping quality in terms of the topology preservation measure q_m.

1. Introduction

Multidimensional data projection and visualization have been subject to extensive research[1]. Many methods have been developed to embed objects, described by high-dimensional vectors, into a two- or three-dimensional output space. The resulting mappings provide insight in the structure of clusters and the distribution of data, and have found a variety of applications in data mining, multivariate data analysis, cluster analysis and pattern recognition. Classical projection methods include multidimensional scaling (MDS) (Togerson, 1958), and the related Sammon's non-linear mapping (NLM) (Sammon, 1969). The NLM tries to minimize the mean square difference between the pairwise distances of points in the original space and in the projection space, i.e. it builds a distance preserving mapping. However the computational complexity of NLM grows quadratically with M, the number of input vectors. Therefore this technique becomes impractical for large data sets.

Neural networks present another approach to data projection and visualization. Self-organizing neural networks are used for the quantization of a data manifold using a finite set of codebook vectors. The self-organizing feature map (SOM) associates every codebook vector of high dimensionality with nodes in a fixed output grid of low dimensionality (Kohonen, 1995). The SOM defines a neighborhood function in the fixed output grid, based on the Euclidean distance. Another self-organizing neural network is the Neural Gas (NG) (Martinetz & Schulten, 1991), which produces a more efficient vector quantization than SOM, because it does not have a fixed output structure that constraints the movement of the codebook vectors. In NG the neighborhood function is defined in the input space, based on the rank order of the codebook vectors with respect to the winner codebook (best matching unit, BMU) (Martinetz, Berkovich & Schulten, 1993).

König (2000) proposed an off-line visualization scheme that combines in cascade the SOM as a vector quantizer and the NLM as a projection method. In the first step, N codebook vectors are computed by SOM or any other clustering technique, and then projected onto the output space by NLM. As a result of this mapping, each codebook vector has associated a codebook position in the output space. If N is small, the mapping of the codebook vectors is

* Corresponding author. Tel.: +56 2 678 4211; fax: +56 2 672 0162.
E-mail addresses: pestevez@ing.uchile.cl (P.A. Estévez), cfiguero@ing.uchile.cl (C.J. Figueroa), saito@cslab.kecl.ntt.co.jp (K. Saito).

very fast. Secondly, the NLM recall (NLMR) algorithm is applied to map the entire data set. In NLMR the codebook positions determined in the previous step are fixed, and the data points presented in the recall operation are projected by computing only the distances with respect to these N fixed codebook positions. The interpoint distances of the recall data are not considered in the mapping. By using this hierarchical approach, the computational complexity of the mapping step is reduced to $O(M)$, for $N \ll M$. However, there is a trade-off between mapping quality and computation time. In fact, the more codebook vectors, the lower is the quantization error and the better is the mapping quality.

TOPSOM (Figueroa & Estévez, 2004) is an extension of the original SOM that replaces the traditional fixed grid by a dynamic and continuous output space. This visualization scheme allows to place simultaneously the codebook vectors in the input space and their respective codebook positions, in the low-dimensional output space. By changing the vector quantization method to NG, the TOPNG visualization method is obtained. The results of TOPSOM and TOPNG are comparable to those of the SOM/NLM in terms of the Sammon stress function, and a topology preservation measure.

Other mapping methods include Isomap (Tenenbaum, de Silva & Langford, 2000) and Local Linear Embedding (LLE) (Roweis & Saul, 2000). The former is a global approach while the latter is a local approach. Local approaches try to map nearby points in the high-dimensional space to nearby points in the low-dimensional embedding space. Global approaches attempt to preserve the global geometric structure of the data. A probabilistic approach to embedding objects into a low-dimensional space with neighborhood preservation is the Stochastic Neighbor Embedding (SNE) (Hinton & Roweis, 2003). The method minimizes the Kullback-Leibler divergence of Gaussian neighborhoods defined in the input and output spaces. SNE focuses mainly on visualizing a set of data points without considering a model structure, such as the relationship between codebook vectors and input data. The Parametric Embedding (PE) method (Iwata, Saito, Ueda, Stromsten, Griffiths & Tenenbaum, 2005) is a kind of generalization of SNE, which simultaneously embeds high-dimensional objects and their classes into a low-dimensional space. It tries to preserve the original classification structure based on posterior probabilities in the embedding space.

In this paper, a cross-entropy based data visualization method is presented. The aim of this method is to preserve the relationship between codebook vectors and input data, as defined by the neighborhood ranking function of the Neural Gas model. Our algorithm for data visualization builds on previous work in document (Iwata et al., 2005) and network (Yamada, Saito & Ueda, 2003) embedding, but it combines these two methods in novel ways to enable new capabilities. In addition, we have added two types of new features, a technique for scaling initial positions and a proposal of the upper bound function. The former enables the use of the results obtained by other method, while the latter allows stable minimization by the Newton–Raphson method. Furthermore, the quality of the mappings is evaluated using the topology preservation measure q_m (König, 2000), on a benchmark data set and three high-dimensional real-world data sets.

Preliminary results of this research were presented in (Estévez, Figueroa & Saito, 2005). The proposed method outperformed classical MDS in terms of the cross-entropy and the topology preservation measure q_m, for several data sets.

2. Visualization algorithm

This section provides an outline of our cross-entropy approach to simultaneously visualizing both input and codebook vectors in a K-dimensional Euclidean space. A visualization algorithm based on the Newton–Raphson method is described.

2.1. Problem setting

Let $\{\mathbf{x}_i : 1 \le i \le M\}$ and $\{\mathbf{w}_j : 1 \le j \le N\}$ be input and codebook vectors, respectively. In the Neural Gas model, the neighborhood function of the input vector $\mathbf{x}_i$ with respect to the codebook vector $\mathbf{w}_j$ is defined as follows:

$$h_\lambda(\mathbf{x}_i, \mathbf{w}_j) = \exp(-k(\mathbf{x}_i, \mathbf{w}_j)/\lambda). \quad (1)$$

The parameter λ controls the neighborhood function width, and $k(x_i, \mathbf{w}_j) \in \{0,1,\ldots,N-1\}$ ranks the codebook vector $\mathbf{w}_j$ by the distance from the input vector $\mathbf{x}_i$. Here $k=0$ is associated with the nearest codebook vector.

For a given set of the trained codebook vectors in the Neural Gas model, our problem is to compute a K dimensional visualization of both the M input vectors and the N codebook vectors so as to preserve the relationships defined by the neighborhood function for each pair of input and codebook vectors. With this aim, in what follows we propose a cross-entropy approach.

2.2. Cross-entropy approach

Consider each value of a neighborhood function as a probability that an input vector $\boldsymbol{x}_i$ belongs to a codebook vector $\boldsymbol{w}_j$ as follows:

$$p_{i,j} = h_\lambda(\mathbf{x}_i, \mathbf{w}_j), \quad (2)$$

where $0 \le p_{i,j} \le 1$. An independent binomial distribution is assumed for each pair of input and codebook vectors.

Let $\{\mathbf{y}_i : 1 \le i \le M\}$ and $\{\mathbf{z}_j : 1 \le j \le N\}$ be the positions of the corresponding M input vectors and N codebook vectors in a K-dimensional Euclidean space. The $\mathbf{y}_i$ and $\mathbf{z}_j$ vectors are called output vectors and codebook positions, respectively. As usual, we define the squared Euclidean

distance between $\mathbf{y}_i$ and $\mathbf{z}_j$ as follows:

$$d_{i,j} = ||\mathbf{y}_i - \mathbf{z}_j||^2 = \sum_{k=1}^{K}(y_{i,k} - z_{j,k})^2. \quad (3)$$

Here we introduce a monotonic decreasing function $\rho(u) \in [0,1]$ with respect to $u \geq 0$, where $\rho(0)=1$ and $\rho(\infty)=0$. Clearly the neighborhood function in the Neural Gas model is a monotonic decreasing function of this type. Since $\rho(d_{i,j})$ can also be regarded as a binomial probability between $\mathbf{y}_i$ and $\mathbf{z}_j$, we introduce a cross-entropy[2] (cost) function between $p_{i,j}$ and $\rho(d_{i,j})$ as follows:

$$E_{i,j} = -p_{i,j}\ln \rho(d_{i,j}) - (1 - p_{i,j})\ln(1 - \rho(d_{i,j})). \quad (4)$$

Since $E_{i,j}$ is minimized when $\rho(d_{i,j}) = p_{i,j}$, this minimization with respect to $\mathbf{y}_i$ and $\mathbf{z}_j$ basically coincides with our problem setting. In this paper, we employ $\rho(u) = \exp(-u/2)$ as the monotonic decreasing function, but note that our approach is not restricted to this one. Then the total cost function (objective function) can be defined as follows:

$$E = \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{N} p_{i,j}d_{i,j} - \sum_{i=1}^{M}\sum_{j=1}^{N}(1 - p_{i,j})\ln(1 - \rho(d_{i,j})). \quad (5)$$

Namely our approach is formalized as a function minimization problem defined in (5) with respect to $\{\mathbf{y}_i : 1 \leq i \leq M\}$ and $\{\mathbf{z}_j : 1 \leq j \leq N\}$.

2.3. Penalty function

In our preliminary experiments, the cost function defined in (5) sometimes produced relatively poor results when the number of codebooks becomes large. The problem was that some groups of codebooks were placed too closely, and thus it was not easy to see the relationships between input and codebook vectors.

In order to separate each codebook position slightly, we introduce the following penalty function.

$$\Omega = \sum_{j=1}^{N}\sum_{s\neq j}\Omega_{j,s}, \quad \Omega_{j,s} = -\ln(1 - \rho(c_{j,s})), \quad c_{j,s} = ||z_j - z_s||^2. \quad (6)$$

Clearly when $\|\mathbf{z}_j - \mathbf{z}_s\|$ is near zero, Ω becomes a very large penalty. Therefore, we can define the final objective function as follows:

$$J = \alpha\sum_{i=1}^{M}\sum_{j=1}^{N} E_{i,j} + \beta\sum_{j=1}^{N}\sum_{s\neq j}\Omega_{j,s}. \quad (7)$$

[2] A formal definition of the cross-entropy between two distributions is given in Bishop (1995), p. 244.

Here the normalizing constants, α and β, are introduced as a trade-off between both cost functions, E and Ω. In this paper, these values are set to $\alpha = 1/(MN)$ and $\beta = 1/(N(N-1))$.

2.4. Learning algorithm

As the basic structure of our learning algorithm, we adopt a coordinate strategy just like the EM (Expectation-Maximization) algorithm. Firstly, we move the output vectors, so as to minimize the objective function by freezing the codebook positions, then we move the codebook positions by freezing the output vectors. These two steps are repeated until convergence.

In the first minimization step, we need to calculate the derivative of the objective function with respect to $\mathbf{y}_i$ as follows:

$$J_{y_i} = \frac{\partial J}{\partial y_i} = \alpha\frac{\partial E}{\partial y_i} = \alpha\sum_{j=1}^{N}\frac{p_{i,j} - \rho(d_{i,j})}{1 - \rho(d_{i,j})}(y_i - z_j). \quad (8)$$

Since $y_{i'}(i' \neq i)$ disappears in (8), we can update $\mathbf{y}_i$ without considering the other output vectors. In the second minimization step, we need to calculate the following derivative,

$$J_{z_j} = \frac{\partial J}{\partial z_j} = \alpha\frac{\partial E}{\partial z_j} - 2\beta\sum_{s\neq j}\frac{\rho(c_{j,s})}{1 - \rho(c_{j,s})}(z_j - z_s), \quad (9)$$

$$\frac{\partial E}{\partial z_j} = \sum_{i=1}^{M}\frac{p_{i,j} - \rho(d_{i,j})}{1 - \rho(d_{i,j})}(z_j - y_i). \quad (10)$$

In the latter case, we update $\mathbf{z}_j$ by freezing the other codebook positions. Then our algorithm can be summarized as follows:

1. Initialize positions $\mathbf{y}_1,\ldots,\mathbf{y}_M$ and $\mathbf{z}_1,\ldots,\mathbf{z}_N$.
2. Calculate gradient vectors $J_{y_1},\ldots,J_{y_M}$.
3. Update output vectors $\mathbf{y}_1,\ldots,\mathbf{y}_M$.
4. Calculate gradient vectors $J_{z_1},\ldots J_{z_N}$.
5. Update codebook positions $\mathbf{z}_1,\ldots,\mathbf{z}_N$.
6. Stop if max $\{||J_{y_1}||,\ldots,||J_{y_M}||,||J_{z_1}||,\ldots,||J_{z_N}||\} < \varepsilon$..
7. Return to **2**.

In the following, we give more details about several steps of the above algorithm.

2.5. Initialization procedure

In Step **1** we might want to initialize the positions using conventional methods such as the MDS technique. However, the scale of distance might be quite different from those derived by our algorithm. In order to estimate an adequate scale of the initial positions, we introduce a scaling factor $\mu > 0$. Then the objective function with respect to μ

can be defined as follows:

$$F(\mu) = \alpha\tilde{F}(\mu) - \beta\sum_{j=1}^{N}\sum_{s\neq j}\ln(1-\rho(\mu c_{j,s})), \tag{11}$$

where

$$\tilde{F}(\mu) = \frac{\mu}{2}\sum_{j=1}^{N}\sum_{j=1}^{N}p_{i,j}d_{i,j} - \sum_{i=1}^{M}\sum_{j=1}^{N}(1-p_{i,j})\ln(1-\rho(\mu d_{i,j})). \tag{12}$$

The first-order derivative is as follows:

$$\frac{\partial F}{\partial \mu} = \alpha\frac{\partial\tilde{F}}{\partial\mu} - \frac{1}{2}\beta\sum_{j=1}^{N}\sum_{s\neq j}\frac{\rho(\mu c_{j,s})}{1-\rho(\mu c_{j,s})}c_{j,s}, \tag{13}$$

where

$$\frac{\partial\tilde{F}}{\partial\mu} = \frac{1}{2}\sum_{j=1}^{M}\sum_{j=1}^{N}\frac{p_{i,j}-\rho(\mu d_{i,j})}{1-\rho(\mu d_{i,j})}d_{i,j}, \tag{14}$$

and the second-order derivative is as follows:

$$\frac{\partial^2 F}{\partial\mu^2} = \alpha\frac{\partial^2\tilde{F}}{\partial\mu^2} + \frac{1}{4}\beta\sum_{j=1}^{N}\sum_{s\neq j}\frac{\rho(\mu c_{j,s})}{(1-\rho(\mu c_{j,s}))^2}c_{j,s}^2, \tag{15}$$

where

$$\frac{\partial^2\tilde{F}}{\partial\mu^2} = \frac{1}{4}\sum_{i=1}^{M}\sum_{j=1}^{N}\frac{(1-p_{i,j})\rho(\mu d_{i,j})}{(1-\rho(\mu d_{i,j}))^2}d_{i,j}^2. \tag{16}$$

Clearly the objective function $F(\mu)$ is convex because the second-order derivative is always non-negative as shown above. Therefore, we can obtain the optimal scaling μ^* by using some minimization algorithm like the Newton–Raphson method described below. In summary, let $\{\mathbf{y}_i: 1\leq i\leq M\}$ and $\{\mathbf{z}_j: 1\leq j\leq N\}$ be the positions obtained by some method like MDS, then the scaled initial positions are $\{\sqrt{\mu^*}y_i : 1\leq i\leq M\}$ and $\{\sqrt{\mu^*}z_j : 1\leq j\leq N\}$.

2.6. Update procedure

In Steps **3** and **5** we employ the Newton–Raphson method for updating the current positions as mentioned earlier. In this subsection, we focus on calculating the modification vector $\Delta\mathbf{y}_i$ by freezing the codebook positions. In this case we can optimize J by minimizing E with respect to $\mathbf{y}_i$. According to the Newton–Raphson method, we can obtain the modification vector, if the Hessian matrix of (5) with respect to $\mathbf{y}_i$ is positive definite,

$$\Delta y_i = -\left[\frac{\partial^2 E}{\partial y_i \partial y_i^T}\right]^{-1}\frac{\partial E}{\partial y_i}, \tag{17}$$

where $\mathbf{y}^T$ stands for the transposed vector of $\mathbf{y}$. The Hessian matrix can be expressed as

$$\frac{\partial^2 E}{\partial\mathbf{y}_i\partial\mathbf{y}_i^T} = \sum_{j=1}^{N}g_{i,j}\mathbf{I} + \sum_{j=1}^{N}h_{i,j}(\mathbf{y}_i-\mathbf{z}_j)(\mathbf{y}_i-\mathbf{z}_j)^T, \tag{18}$$

where $\mathbf{I}$ stands for the identity matrix in the K-dimensional space, and $g_{i,j}$ and $h_{i,j}$ are defined as follows:

$$g_{i,j} = \frac{p_{i,j}-\rho(d_{i,j})}{1-\rho(d_{i,j})}, \quad h_{i,j} = \frac{(1-p_{i,j})\rho(d_{i,j})}{(1-\rho(d_{i,j}))^2}. \tag{19}$$

Clearly $\sum_{j=1}^{N}g_{i,j}$ can be a negative value, and the Hessian matrix could become non-positive definite. To cope with this situation, we consider a relaxation problem. Firstly, we define the updated distance as follows:

$$d_{i,j}(\Delta\mathbf{y}_i) = ||\mathbf{y}_i + \Delta\mathbf{y}_i - \mathbf{z}_j||^2, \tag{20}$$

and the updated cost function as follows:

$$E_{i,j}(\Delta\mathbf{y}_i) = \frac{1}{2}p_{i,j}d_{i,j}(\Delta\mathbf{y}_i) - (1-p_{i,j})\ln(1-\rho(d_{i,j}(\Delta\mathbf{y}_i))). \tag{21}$$

Replacing the second term in the right-hand-side of (21) by

$$-(1-p_{i,j})\ln(1-\rho(d_{i,j}(\Delta\mathbf{y}_i) - ||\Delta\mathbf{y}_i||^2)), \tag{22}$$

we can consider the upper bound cost function $U_{i,j}(\Delta\mathbf{y}_i)$. It is easy to see that $U_{i,j}(\Delta\mathbf{y}_i)\geq E_{i,j}(\Delta\mathbf{y}_i)$ because $-\ln(1-\rho(u))$ monotonically increases as $u>0$ decreases. By noting that $E_{i,j}=E_{i,j}(\mathbf{0})=U_{i,j}(\mathbf{0})$, if we can find some $\Delta\mathbf{y}_i$ that satisfies $\sum_{j=1}^{N}U_{i,j}(0) > \sum_{j=1}^{N}U_{i,j}(\Delta\mathbf{y}_i)$, we can observe the following relationships:

$$\sum_{j=1}^{N}E_{i,j} > \sum_{j=1}^{N}U_{i,j}(\Delta\mathbf{y}_i) > \sum_{j=1}^{N}E_{i,j}(\Delta\mathbf{y}_i). \tag{23}$$

Minimizing the objective function constructed by the upper bound cost function guarantees a reduction of the original objective function. On the other hand, the gradient vector of the objective function derived from $U_{i,j}$ is as follows:

$$\frac{\partial U(0)}{\partial\Delta\mathbf{y}_i} = \sum_{j=1}^{N}g_{i,j}(\mathbf{y}_i-\mathbf{z}_j) = \frac{\partial E}{\partial\mathbf{y}_i}, \tag{24}$$

and the corresponding Hessian matrix is non-negative definite as shown below,

$$\frac{\partial^2 U(0)}{\partial\Delta\mathbf{y}_i\partial\Delta\mathbf{y}_i^T} = \sum_{j=1}^{N}p_{i,j}\mathbf{I} + \sum_{j=1}^{N}h_{i,j}(\mathbf{y}_i-\mathbf{z}_j)(\mathbf{y}_i-\mathbf{z}_j)^T. \tag{25}$$

As a consequence, if the Hessian matrix (18) is non-positive definite, we can use (25) and apply the Newton–Raphson method for updating the current positions.

2.7. Topology preservation measure

The topology preservation measure q_m used in (König, 2000) is considered as a performance measure. It is based on an assessment of rank order in the input and output spaces.

The n nearest codebook vectors $NN_{ji^w}(i \in [1,n])$ of each codebook vector j, ($j \in [1, N]$) and the n nearest codebook positions NN_{ji^z} of each codebook position j are computed. A credit scheme is constructed for each pair of points, $q_{m_{ji}}$:

$$q_{m_{ji}} = \begin{cases} 3, & \text{if } NN_{ji^w} = NN_{jl^z} \\ 2, & \text{if } NN_{ji^w} = NN_{jl^z}, l \in [1,n], i \neq l \\ 1, & \text{if } NN_{ji^w} = NN_{jt^z}, t \in [n,k], n < k \\ 0, & \text{else.} \end{cases} \tag{26}$$

The global q_{m} measure is defined as:

$$q_m = \frac{1}{3n \times N} \sum_{j=1}^{N} \sum_{i=1}^{n} q_{m_{ji.}} \tag{27}$$

Typical parameter settings are $n=4$ and $k=10$. The range of the q_{m} measure is between 0 and 1, where $q_{\mathrm{m}}=0$ indicates a poor neighborhood preservation, and $q_{\mathrm{m}}=1$ indicates a perfect neighborhood preservation. The input and output vectors can be also used for calculating the topology preservation measure q_{m}. The latter measure will be denoted as q_m^{xy} to distinguish it from the q_m^{wz} calculated with the codebook vectors and positions.

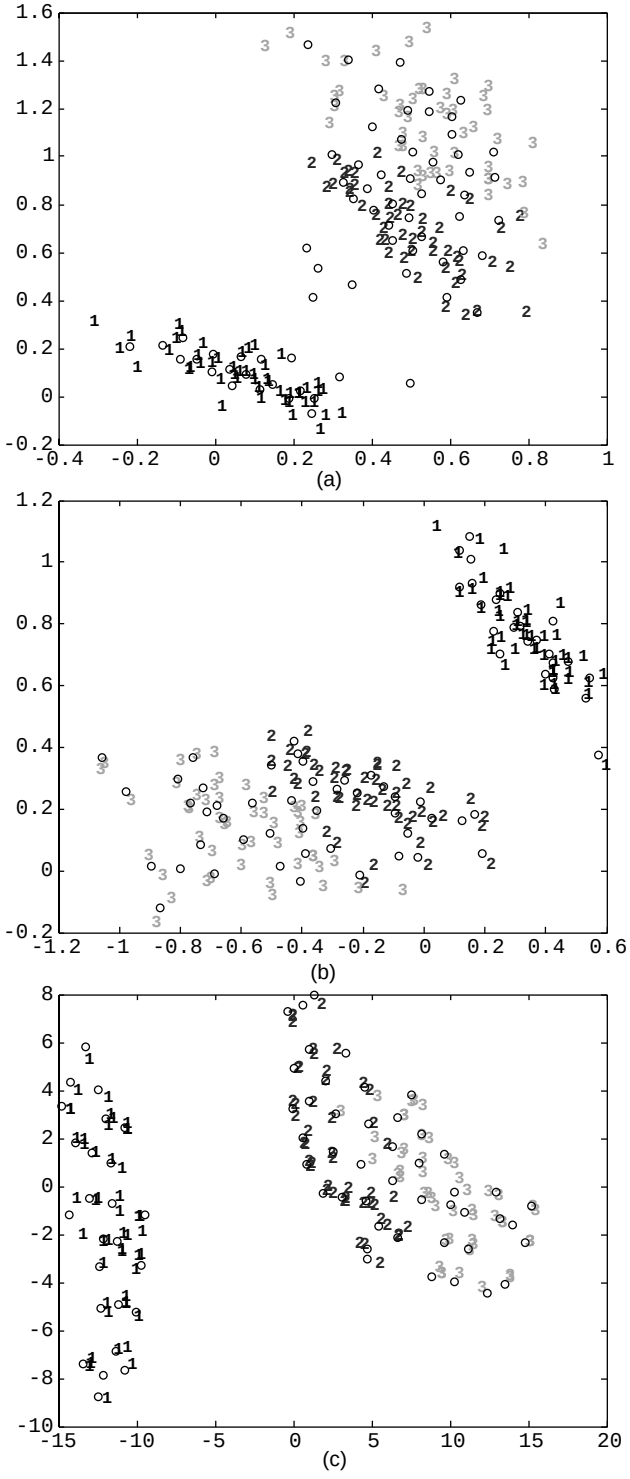

Fig. 1. Iris data set (150×4) projection results. Codebook positions (o) and output vectors (classes 1, 2, 3) are shown for (a) SOM-NLMR, (b) NG-NLMR, and (c) NG-CE.

3. Simulation results

For each data set, the results obtained for the proposed model, called NG-CE (Neural Gas Cross-Entropy) and the SOM-NLMR and NG-NLMR hierarchical strategies, are presented. The direct application of Sammon's NLM algorithm for projecting the entire data sets is considered as a reference.

3.1. Iris data

The Iris data set is a widely used benchmark in pattern recognition. It contains three classes of 50 samples each, where each class refers to a type of Iris plant (Iris setosa, Iris versicolor and Iris virginica). Fig. 1 shows the projection results obtained with the Iris data set and 70 codebook vectors for (a) SOM-NLMR, (b) NG-NLMR, and (c) NG-CE with $\lambda=1.5$. The best values of q_{m} are obtained for λ ranging from 1.0 to 3.0 (Estévez, Figueroa & Saito, 2005). In Table 1 the topology preserving measures for codebooks and data points are given for the three methods considered. In addition, the q_m^{xy} value obtained with the projection of the entire data set (150 examples) by NLM is provided as a reference. It can be observed that the NG-CE method obtained the best q_m^{xy} value. Table 2 shows the topology preservation measure q_{m} as a function of the number of codebook vectors. It can be observed that the q_m^{xy} value for

Table 1
Topology preservation measure (q_{m}) for the Iris dataset

Algorithm	q_m^{wz}	q_m^{xy}
SOM-NLMR	0.7712	0.6022
NG-NLMR	0.6988	0.6049
NG-CE	0.7417	0.6428
NLM (150)	–	0.6213

Table 2
Mapping quality versus codebook size for the Iris dataset

Codebook vectors	q_m^{wz}	q_m^{xy}
10	0.8083	0.4028
30	0.7917	0.5867
50	0.7350	0.6378
70	0.7417	0.6428
90	0.7324	0.6739
110	0.7287	0.6861
130	0.7250	0.6894
150	0.7300	0.7000

input-output vectors increases when the number of codebook vectors grows, while the q_m^{wz} value for codebook vectors and codebook positions diminishes. This shows clearly that there is a trade-off between the number of codebooks and the mapping quality. Fig. 2 shows the NG-CE visualization results, with $\lambda = 1.5$, for a varying number of codebook vectors, ranging from 10 to 150.

3.2. Fraud data

The Fraud data set contains information about telephone subscribers. It consists of 10721 samples of 26 features each (Estévez, Held & Perez, 2005). The samples are divided into

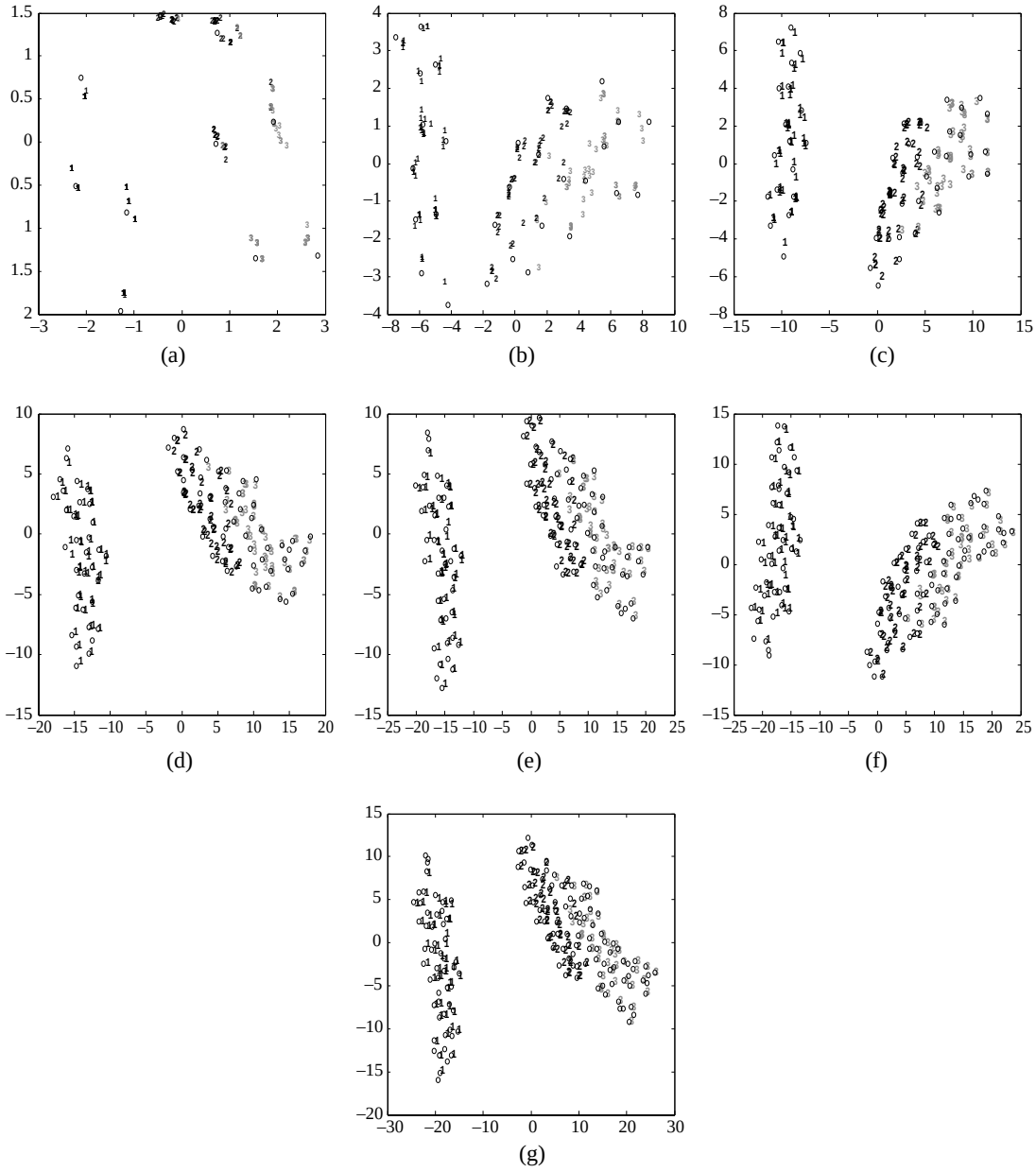

Fig. 2. Iris data set projection results for NG-CE with $\lambda = 1.5$, and a varying number of codebook vectors: (a) 10, (b) 30, (c) 50, (d) 90, (e) 110, (f) 130 and (g) 150.

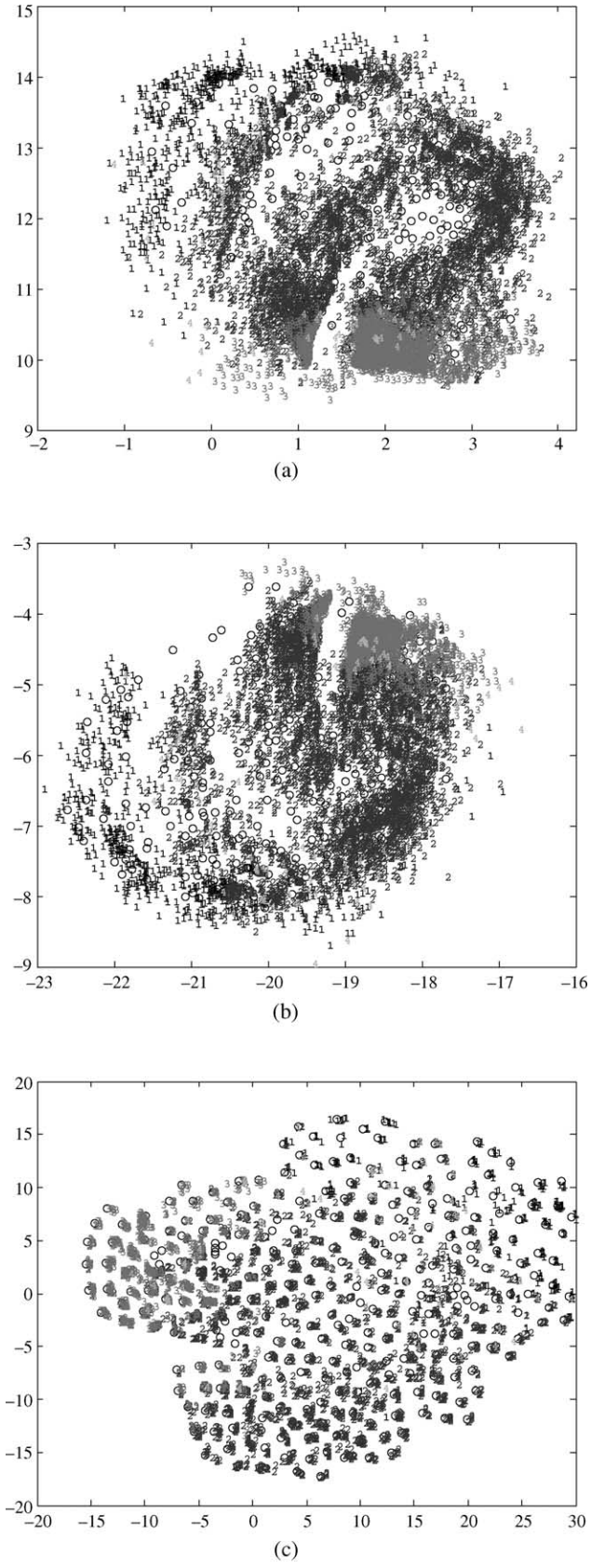

Fig. 3. Fraud data set (10721×26) projection results. Codebook positions (o) and output vectors (classes 1, 2, 3, 4) are shown for (a) SOM-NLMR, (b) NG-NLMR, and (c) NG-CE.

4 classes: 705 Fraud samples, 5095 Insolvent samples, 4824 Normal samples, and 97 Other samples. The vector quantization was performed using the NG algorithm with 300 codebook vectors.

Fig. 3 shows the projection results obtained with the Fraud data set and 300 codebook vectors for (a) SOM-NLMR, (b) NG-NLMR and (c) NG-CE with $\lambda = 0.75$. In Table 3 the quantifying measures are given for the models considered. It can be observed that the NG-CE method obtained the best q_m values.

3.3. Sleep data

This real-world data set consists of features extracted from infant polysomnograms used for sleep stage scoring, e.g. rapid eye movements (REM) in the EOG. The Sleep data set contains 6537 samples, and six features extracted from epochs (pages) of 30 s. (Estévez et al., 2002). This data set is divided into six classes that correspond to 919 Non-REM-I samples, 1479 Non-REM-II samples, 1658 Non-REM-III&IV samples, 890 REM samples, 1517 wakefulness samples and 74 indeterminate samples. The vector quantization was performed using the NG algorithm with 200 codebook vectors.

Fig. 4 shows the projection results obtained with the Sleep data set and 200 codebook vectors for (a) SOM-NLMR, (b) NG-NLMR, and (c) NG-CE with $\lambda = 2.0$. In Table 4 the quantifying measures are given for the models considered. Again, NG-CE obtained the best q_m values. Table 5 shows the topology preservation measure q_m as a function of the number of codebook vectors. It can be observed that the q_m^{xy} value for input-output vectors increases when the number of codebook vectors grows, while the q_m^{wz} value for codebook vectors and codebook positions diminishes. Again this shows clearly the trade-off between the number of codebooks and the mapping quality. Fig. 5 shows the NG-CE visualization results, with $\lambda = 2.0$, for a varying number of codebook vectors.

3.4. Wood data

This real-world data set contains 64-dimensional samples extracted from 16800 color images of wood boards (Estévez, Perez & Goles, 2003). This data set is divided into eleven classes that correspond to 2800 clearwood samples, 1400 birds eye samples, 1400 pocket samples, 1400 wane samples, 1400 split samples, 1400 blue stain samples, 1400

Table 3
Topology preservation measure (q_m) for the Fraud dataset

Algorithm	q_m^{wz}	q_m^{xy}
SOM-NLMR	0.4461	0.1948
NG-NLMR	0.3614	0.1823
NG-CE	0.4919	0.2727
NLM (10721)	–	0.2157

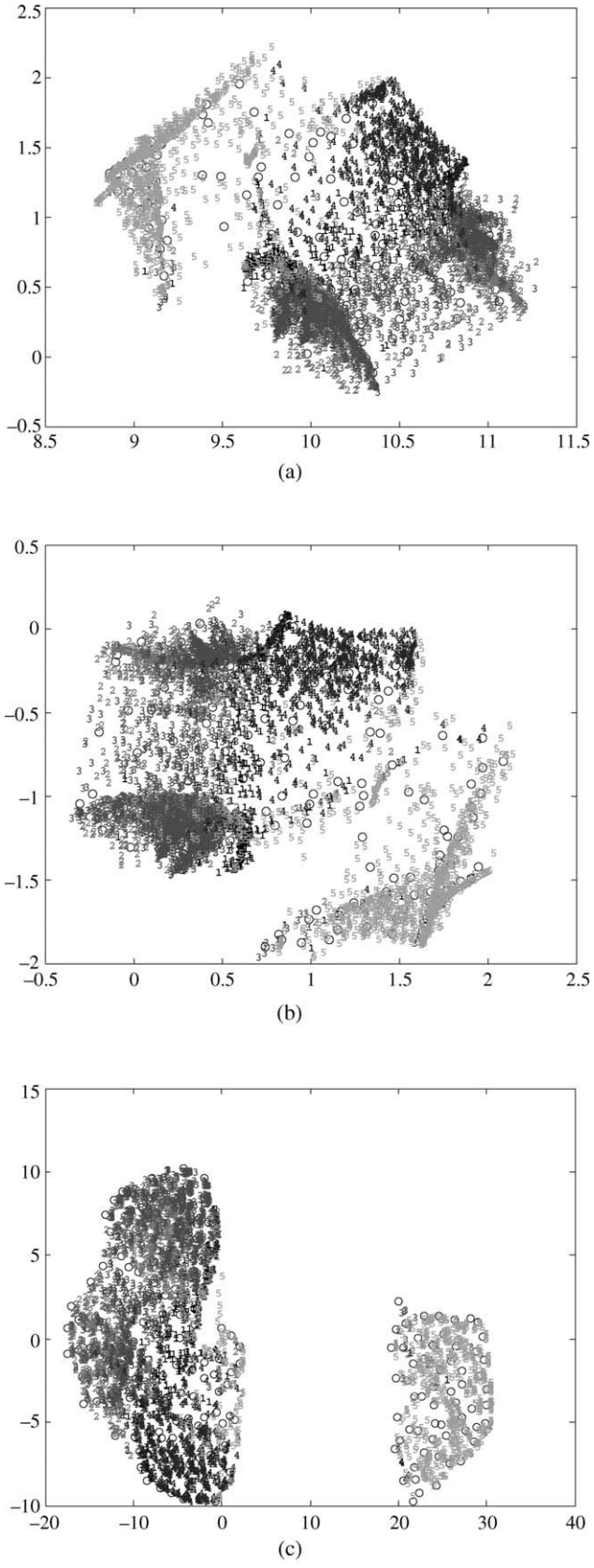

Fig. 4. Sleep data set (6537×6) projection results. Codebook positions (o) and output vectors (classes 1, 2,…,6) are shown for (a) SOM-NLMR, (b) NG-NLMR, and (c) NG-CE.

Table 4
Topology preservation measure (q_m) for the sleep dataset

Algorithm	q_m^{wz}	q_m^{xy}
SOM-NLMR	0.5745	0.3848
NG-NLMR	0.3954	0.3469
NG-CE	0.6558	0.4121
NLM (6537)	–	0.4123

stain samples, 1400 pith samples, 1400 dead knots, 1400 live knots and 1400 holes. For the vector quantization using the NG algorithm, 400 nodes were used. Fig. 6 shows the projection results obtained with the Wood data set for (a) SOM-NLMR, (b) NG-NLMR, and (c) NG-CE with $\lambda = 1.5$. In Table 6 the quantifying measures are given for the models considered. The best q_m values were obtained with NG-CE.

4. Discussion

The results show that the proposed NG-CE method outperforms SOM-NLMR and NG-NLMR for all the data sets considered, in terms of the q_m measure. The NG-CE results are better than full NLM for the Fraud and Wood data sets, and similar to NLM for the Iris and Sleep data sets. This implies that our method, which is not based on NLM as the other methods considered here, is able to preserve more faithfully the local neighborhoods.

In the NG-CE algorithm, the computational complexity of one-iteration is $O(M \times N \times K) + O(N^2 \times K)$ as shown in the objective function described in (21). In general, the number of input vectors M is much larger than the number of codebook vectors N, and the dimension of output space is usually set to $K = 2$ or 3. Thus, the main computational complexity is approximately $O(M)$. In contrast, other visualization methods such as Sammon mapping (Sammon, 1969) or SNE (Hinton & Roweis, 2003) have a computational complexity of $O(M^2)$. This is the reason why methods such as SOM-NLMR or NG-NLMR have been developed, in order to reduce the computational load. However, these methods are based on NLM, and get inferior results than full NLM in terms of topology preservation. The main advantage of NG-CE is its desirable $O(M)$

Table 5
Mapping quality versus codebook size for the sleep dataset

Codebook vectors	q_m^{wz}	q_m^{xy}
10	0.7500	0.0880
50	0.7233	0.3449
100	0.6692	0.3660
150	0.6756	0.3891
200	0.6558	0.4121
250	0.6346	0.4143
300	0.6083	0.4242
1000	0.5040	0.4774

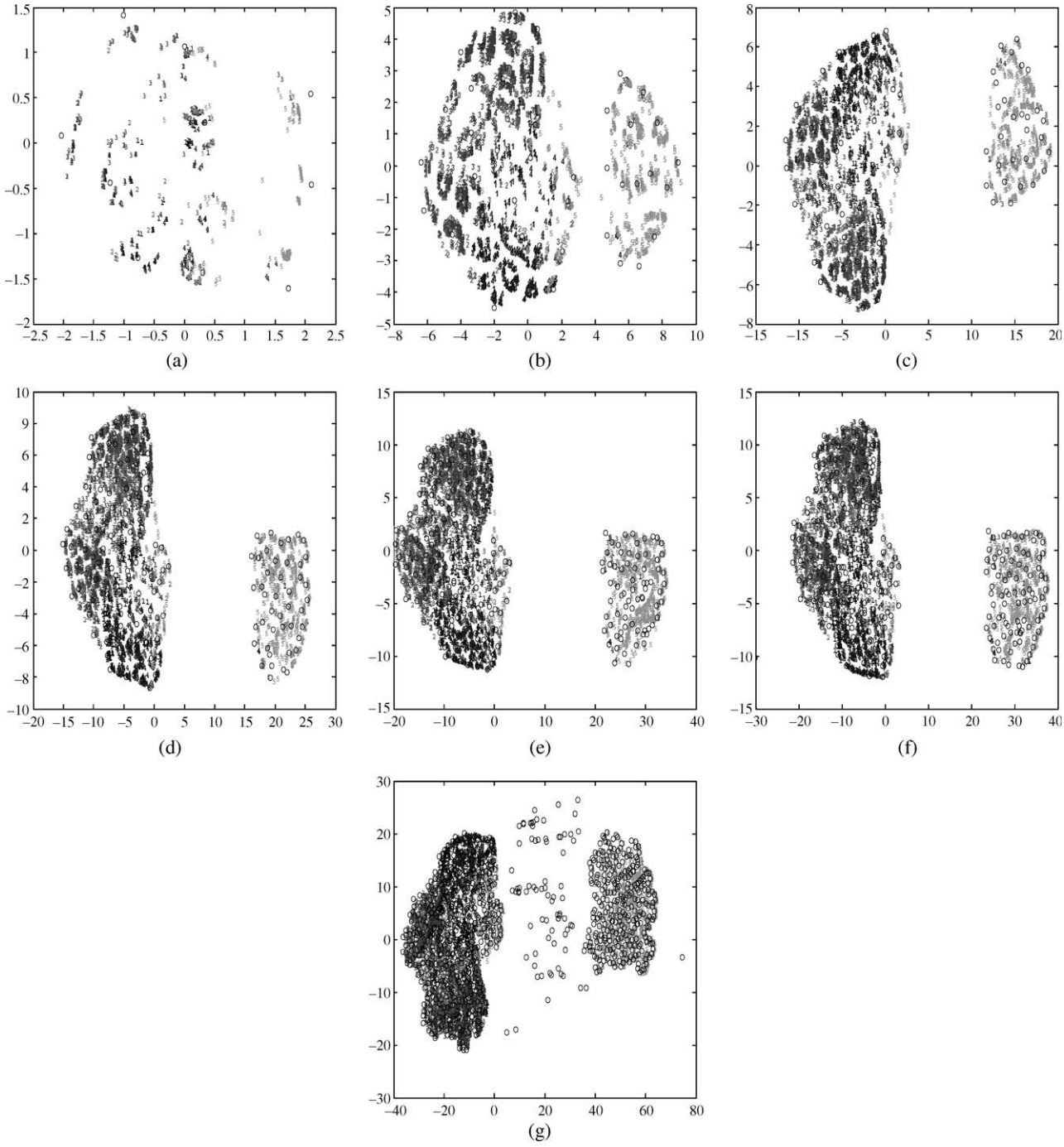

Fig. 5. Sleep data set projection results for NG-CE with $\lambda = 2.0$, and a varying number of codebook vectors: (a) 10, (b) 50, (c) 100, (d) 150, (e) 250, (f) 300 and (g) 1000.

computational complexity, and that provides better or similar results than full NLM in terms of topology preservation.

Our method is based on a probabilistic approach like SNE, but most previous work using this approach focused on visualizing a set of data points without a model structure. In contrast, NG-CE tries to preserve the relationship between codebook vectors and input vectors, as defined by the NG neighborhood ranking function. In future research it would be of interest to compare our approach with other local neighborhood preserving projection methods such as LLE (Roweis & Saul, 2000).

5. Conclusions

We have proposed a cross-entropy based method for embedding codebook vectors and input vectors

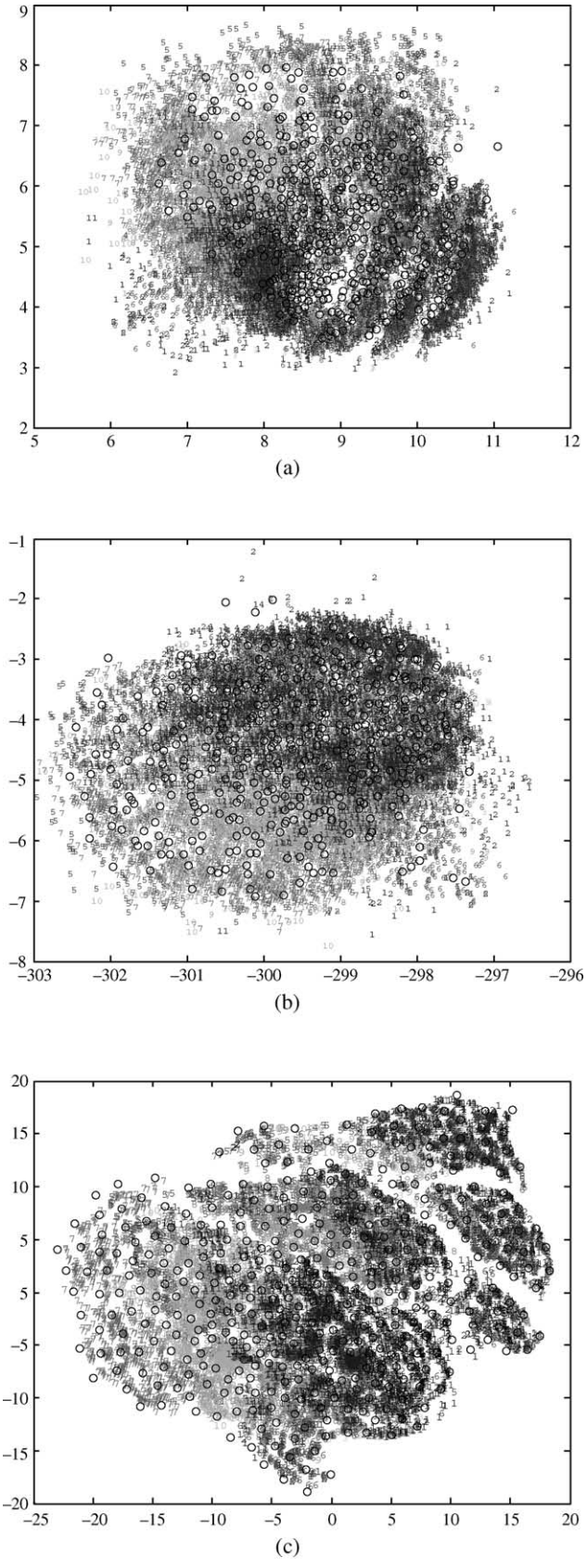

Fig. 6. Wood data set (16800×64) projection results. Codebook positions (o) and output vectors (classes 1, 2,…,11) are shown for (a) SOM-NLMR, (b) NG-NLMR, and (c) NG-CE.

Table 6
Topology preservation measure (q_m) for the wood dataset

Algorithm	q_m^{wz}	q_m^{xy}
SOM-NLMR	0.3107	0.0388
NG-NLMR	0.2054	0.0378
NG-CE	0.3804	0.1625
NLM (16800)	–	0.0399

simultaneously into a low-dimensional space. The results show that the proposed NG-CE method preserves more faithfully the local topology than SOM-NLMR and NG-NLMR, which are all methods of similar computational complexity, $O(M)$. In addition, NG-CE obtains better or similar results than full NLM in terms of the topology preservation measure q_m, but NLM has a $O(M^2)$ computational complexity. This makes NG-CE especially useful for large data sets. The Neural Gas neighborhood ranking function has been used as a case study, but any neighborhood function may be used. Effective visualizations were obtained for real world data sets of over 15000 samples. The proposed NG-CE method obtained better visualizations than SOM-NLMR and NG-NLMR, delivering a clear visualization of both data points and codebooks. This can be seen in the NG-CE projections of the Iris, Fraud, Sleep and Wood data sets.

Acknowledgements

This research was supported by Conicyt-Chile, under grant Fondecyt 1050751. This work was carried out under a joint research agreement between NTT and the University of Chile.

References

Bishop, C. M. (1995). *Neural networks for pattern recognition*. New York: Oxford University Press.

Estévez, P. A., Held, C. M., Holzmann, C. A., Perez, C. A., Pérez, J. P., Heiss, J., Garrido, M., & Peirano, P. (2002). Polysomnographic pattern recognition for automated classification of sleep-waking states in infants. *Medical and Biomedical Engineering and Computing*, *40*, 105–113.

Estévez, P. A., Perez, C. A., & Goles, E. (2003). Genetic input selection to a neural classifier for defect classification of radiata pine boards. *Forest Products Journal*, *53*(7/8), 87–94 (July/August).

Estévez, P. A., Figueroa, C. J., & Saito, K. (2005). Cross-entropy approach to data visualization based on the neural gas network. *Proceedings of the international joint conference on neural networks*, July 31–August 4, Montreal, Canada.

Estévez, P. A., Held, C. M., & Perez, C. A. (2005). Prevention of subscription fraud in telecommunications using fuzzy rules and neural networks. Available at http://www.die.uchile.cl/~pestevec/public.html.

Figueroa, C. J., Estévez, P. A. (2004). A new visualization scheme for self-organizing neural networks. *Proceedings of the international joint conference on neural networks*. (p. 757–762). Budapest, Hungary.

Hinton, G., & Roweis, R. (2003). Stochastic neighbor embedding. *Advances in Neural Information Processing Systems (NIPS'02)*, *15*, 833–840.

Iwata, T., Saito, K., Ueda, N., Stromsten, S., Griffiths, T. L., & Tenenbaum, J. B. (2005). Parametric embedding for class visualization. *Advances in neural information processing systems*, vol. 17. Cambridge, MA: MIT Press (pp. 513–520).

Kohonen, T. (1995). *Self-organizing maps*. Berlin, Germany: Springer-Verlag.

König, A. (2000). Interactive visualization and analysis of hierarchical neural projections for data mining. *IEEE Transactions on Neural Networks*, *11*(3), 615–624.

Martinetz, T. M., Berkovich, S. G., & Schulten, K. J. (1993). Neural gas network for vector quantization and its application to time-series prediction. *IEEE Transactions on Neural Networks*, *4*(4), 558–569.

Martinetz, T. M., & Schulten, K. J. (1991). A neural gas network learns topologies. *Artificial Neural Networks*, 397–402.

Roweis, S., & Saul, L. (2000). Nonlinear dimensionality reduction by local linear embedding. *Science*, *290*, 2323–2326.

Sammon, J. W. (1969). A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, *C-18*, 401–409.

Tenenbaum, J. B., de Silva, V., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, *290*, 2319–2323.

Togerson, W. S. (1958). *Theory and methods of scaling*. New York: Wiley.

Yamada, T., Saito, K., & Ueda, N. (2003). Cross-entropy directed embedding of network data. *Proceedings of the Twentieth International Conference on Machine Learning*, 832-839.