Nonlinear Projection Using Geodesic Distances and the Neural Gas Network

Pablo A. Estévez, Andrés M. Chong, Claudio M. Held, and Claudio A. Perez

Dept. Electrical Engineering, University of Chile,
Casilla 412-3, Santiago, Chile
pestevez@cec.uchile.cl
http://www.die.uchile.cl/~pestevez

Abstract. A nonlinear projection method that uses geodesic distances and the neural gas network is proposed. First, the neural gas algorithm is used to obtain codebook vectors, and a connectivity graph is concurrently created by using competitive Hebbian rule. A procedure is added to tear or break non-contractible cycles in the connectivity graph, in order to project efficiently 'circular' manifolds such as cylinder or torus. In the second step, the nonlinear projection is created by applying an adaptation rule for codebook positions in the projection space. The mapping quality obtained with the proposed method outperforms CDA and Isotop, in terms of the trustworthiness, continuity, and topology preservation measures.

1 Introduction

Self-organizing feature maps (SOMs) [8] have been widely used for vector quantization (VQ) and for data projection and visualization. The VQ techniques encode a manifold of data by using a finite set of reference or "codebook" vectors. An enhancement to the SOM is the curvilinear component analysis (CCA) [2], which builds a mapping that preserves well the interpoint distances. In the first step, CCA performs VQ of the data manifold in input space using SOM. In the second step, CCA makes a nonlinear projection of the quantizing vectors, which is similar to multidimensional scaling (MDS) [1], since it minimizes a cost function based on the interpoint distances. However the computational complexity of CCA is $O(N)$, while MDS is $O(N^2)$. Another difference is that in CCA the output is not a fixed grid but a continuous space that is able to take the shape of the data manifold. The codebook vectors are projected as codebook positions in output space, which are updated by a special adaptation rule. An enhanced version of CCA, called CDA (curvilinear distance analysis), makes use of geodesic or curvilinear distances instead of Euclidean distances in the input space [11, 13]. In CDA the geodesic distance is approximated by computing the sum of the Euclidean lengths of all links in the shortest path of a graph. CDA can outperform CCA in the projection of highly nonlinear databases like curved manifolds. However, only a few methods are able to project efficiently 'circular' manifolds such as a cylinder or torus. In [10] a procedure was proposed to tear or cut manifolds with essential loops to make easier their embedding in a low-dimensional space. The tearing procedure represents the manifold by a connectivity graph G. An elementary cycle is defined as a 'circular' path in a graph with no more than C edges. A cycle is said to be non-contractible if it can not be deformed to elementary cycles. The first step of the tearing procedure consists in removing all cycles in the graph G, by computing a shortest spanning tree (SPT) [3] or alternatively, a minimum spanning tree (MST) [17]. In the second step, the removed edges are put back one at a time, but only if they do not generate any non-contractible cycle. In this way, a maximum subgraph without non-contractible cycles is created.

Another neural method for nonlinear projection is Isotop [12], which focuses on neighborhood preservation instead of distance preservation. Isotop uses competitive learning for the VQ step. The second step consists in creating a graph structure by linking the k-closest prototypes. The third step builds the mapping from the input space onto the projection space.

The neural gas (NG) is another well-known self-organizing neural network [16]. The main difference with SOM is that NG does not define an output space. The SOM is able to obtain good VQ results only if the topology of the output grid matches the topology of the data manifold. As a consequence, NG can outperform SOM when quantizing topologically arbitrary structured manifolds. Instead of a neighborhood function in the output grid, NG utilizes a neighborhood ranking of the codebook vectors within the input space. In addition, the NG adaptation rule for codebook vectors obeys a stochastic gradient descent of a global cost function, while no cost function exists for the SOM adaptation rule [15]. The NG algorithm can be combined with the competitive Hebbian rule (CHR) for forming connections among neighboring units [14]. The procedure forms induced Delaunay triangulations of the distribution of codebook vectors.

The authors have proposed elsewhere [5, 6], a nonlinear projection method, called OVI-NG, that makes use of the VQ results obtained with the NG algorithm. The codebook positions are adjusted in a continuous output space by using an adaptation rule that minimizes a cost function that favors the local distance preservation. In [4], an extension of the previous nonlinear projection method was proposed called GNLP-NG, that uses geodesic distances instead of the Euclidean ones. This method projects the codebook vectors, and it uses the competitive Hebbian rule for building a connectivity graph linking these prototypes. The proposed algorithm outperformed CDA and Isotop, in terms of several topology preserving measures.

In this paper, we present an extension to the GNLP-NG nonlinear projection method, that includes tearing or cutting graphs with non-contractible cycles. The performance of the new algorithm is compared with CDA and Isotop, using two examples of ‘circular’ manifolds that are not easy to project without a tearing procedure.

2 The Geodesic Nonlinear Projection Algorithm

The proposed projection method is called GNLP-NG (Geodesic Nonlinear Projection Neural Gas).

2.1 Problem Setting

Let $\{x_i : 1 \leq i \leq M\}$ and $\{w_j : 1 \leq j \leq N\}$ be D-dimensional input and codebook vectors, respectively. For a given set of input vectors, our problem is to adjust first the codebook vectors in the input space and then their respective codebook positions z_j ($j = 1, \ldots, N$) in a A-dimensional continuous output space, with $A << D$.

To obtain a distance preserving mapping a cost function is defined, which depends on the difference between the interpoint distances in both the input and the output spaces. Let $D_{j,k}$ be the Euclidean distance defined in the output space, and let $\delta_{j,k}$ be the geodesic distance between codebook vectors w_j and w_k, measured in input space.

In the Neural Gas model, the neighborhood ranking function of the codebook vectors $\mathbf{w}_j$, for $j = 1, \cdots, N$, with respect to a given input vector x_i, is defined as follows:

$$h_\lambda(x_i, w_j) = e^{-\frac{r(x_i(t), w_j(t))}{\lambda(t)}}, \tag{1}$$

where $r_{ij} = r(x_i, w_j) \in \{0, 1, \ldots, N-1\}$ denotes the rank of the j^{th} codebook vector, and the parameter $\lambda(t)$ controls the width of the neighborhood function,

$$\lambda(t) = \lambda_0 \left(\frac{\lambda_f}{\lambda_0}\right)^{\left(\frac{t}{t_{max}}\right)}, \tag{2}$$

where t_{max} is the maximum number of adaptation steps.

Likewise, for projection purposes we introduce a ranking of the codebook vectors in input space using geodesic distances, with respect to a given input vector x_i. The term $\bar{r}_{ij} = \bar{r}(x_i, w_j) \in \{0, 1, \ldots, N-1\}$ denotes the rank of the j^{th} codebook vector using geodesic distances. Here $r = 0$ and $\bar{r} = 0$ are associated with the nearest codebook vector using Euclidean and geodesic distances in input space, respectively.

2.2 Codebook Position Adaptation Rule

The following global cost function is considered:

$$E = \frac{1}{2}\sum_{j=1}^{N}\sum_{k\neq j}(D_{j,k} - \delta_{j,k})^2 F(\bar{r}_{j,k}) = \frac{1}{2}\sum_{j=1}^{N}\sum_{k\neq j} E_{j,k}, \tag{3}$$

where the function F is defined as

$$F(f) = e^{-\left(\frac{f}{\sigma(t)}\right)^2} \tag{4}$$

and $\sigma(t)$ is the width of the neighborhood that decreases with the number of iterations in the same way as eq. (2). The function $F(f)$ is a bounded and monotonically decreasing function, in order to favor local topology preservation.

Eq. (3) is minimized with respect to z_j, by using a simplified version of gradient descent. The codebook position associated to the winner unit, $z_j^*(t)$, is fixed, and the $N-1$ remaining positions are moved towards the winner's position, disregarding any interaction among them. The updating rule for codebook positions is given by Eq. (12).

The complexity of the proposed projection algorithm is $O(NlogN)$, due to the determination of the ranking in input space. Since the computational complexity of NG is also $O(NlogN)$ [15], the combined algorithm of NG as VQ and our GNLP visualization strategy has the same complexity.

2.3 NG Learning Algorithm

The learning algorithm combines NG for VQ, with CHR for forming connections between units. The initial topology of the network is a set of N neurons. Each neuron j has associated a D-dimensional codebook vector, w_j, ($j = 1, \ldots, N$). Let $\mathcal{C}_{in}$ be a connection set, that includes the connections between units in input space. Each connection $j - k$ has an age ($\mathrm{a}_{(jk)}$) that is defined as the number of adaptation steps without being refreshed since its creation. A connection is removed if its age exceeds its lifetime $T(t)$.

1. Initialize the codebook vectors, w_j, randomly. Set the connection set to the empty set: $\mathcal{C}_{in} = \emptyset$.
2. Present an input vector, $x_i(t)$ to the network ($i = 1, \ldots, M$) at iteration t.
3. Find the best matching unit (BMU), j^* using:
$$j^* = \mathrm{argmin}_{j=1\ldots N} \|x_i(t) - w_j(t)\|, \tag{5}$$
and generate the ranking $r_{ij} = r(x_i(t), w_j(t)) \in \{0, 1, \ldots, N-1\}$ for each codebook vector $w_j(t)$ with respect to the input vector $x_i(t)$.
4. Update the codebook vectors:
$$w_j(t+1) = w_j(t) + \epsilon(t) h_\lambda(t)(x_i(t) - w_j(t)) \tag{6}$$
where
$$h_\lambda(t) = h_\lambda(x_i, w_j) \tag{7}$$
is the neighborhood ranking function defined in (1) and $\epsilon(t)$ is the learning rate, which depends on the number of adaptation steps t, in the same way as (2).
5. If a connection between the BMU and the second closest unit does not exist already, create a connection between units $j^* - j^{*2}$: $\mathcal{C}_{in} = \mathcal{C}_{in} \cup \{w_j^*, w_j^{*2}\}$, where the index of the second nearest unit is:
$$j^{*2} = \mathrm{argmin}_{j \neq j^*} \|x_i(t) - w_j(t)\|. \tag{8}$$
6. If the following condition is satisfied
$$\|w_j^{*k} - w_j^{*(k+1)}\| < \|w_j^* - w_j^{*(k+1)}\|, \tag{9}$$
for $k = 2, \ldots, K$, then create a connection between the kth-nearest unit, j^{*k}, and the (k+1)th-nearest unit, $j^{*(k+1)}$, if it does not exist already: $\mathcal{C}_{in} = \mathcal{C}_{in} \cup \{w_j^{*k}, w_j^{*(k+1)}\}$, else create a connection between units $j^* - j^{*(k+1)}$: $\mathcal{C}_{in} = \mathcal{C}_{in} \cup \{w_j^*, w_j^{*(k+1)}\}$.
Set the age of the connection $j^{*m} - j^{*n}$ to zero, for $m, n = 1 \ldots K$ ("refresh" the connection if it exists):
$$\mathrm{a}_{(j^{*m}, j^{*n})} = 0. \tag{10}$$
7. Increment the age of all edges emanating from w_j^{*k}, for $k = 1 \ldots K$:
$$\mathrm{a}_{(j^*, \ell)} = \mathrm{a}_{(j^*, \ell)} + 1, \forall \ell \in \mathcal{N}_{w_j^{*k}}, \tag{11}$$
where $\mathcal{N}_{w_j^{*k}}$ is the set of all direct topological neighbors of w_j^{*k}. Remove those connections with an age exceeding the lifetime T, where $T(t)$ has the same dependency on time t as (2).
8. If $t < t_{max}$ go back to step 2.

2.4 GNLP Algorithm

1. Apply Dijkstra's algorithm [3] to find the shortest path tree (SPT) of the neighborhood graph defined by connection matrix $\mathcal{C}_{in}$.
2. If necessary, tear or cut non-contractible cycles in the neighborhood graph, following the ad-hoc procedure described in 2.5.
3. Initialize the codebook positions, z_j, randomly.
4. Present an input vector, $x_i(t)$ to the network $(i = 1, \ldots, M)$ at iteration t.
5. Find the best matching unit (BMU), j^* using eq. (5)
6. Generate the ranking using geodesic distances in input space $\bar{r}_{j^*j} = \bar{r}(w_j(t), w_j^*(t)) \in \{0, 1, \ldots, N-1\}$ for each codebook vector $w_j(t)$ with respect to the BMU.
7. Update the codebook positions:

$$z_j(t+1) = z_j(t) + \alpha(t) F(\bar{r}_{j,j^*}) \frac{(D_{j,j^*} - \delta_{j,j^*})}{D_{j,j^*}} (z_{j^*}(t) - z_j(t)), \qquad (12)$$

   where $F(\bar{r}_{j,j^*})$ is a neighborhood function that depends on the ranking of units in input space according to the geodesic distance, and $\alpha(t)$ is the learning rate, which typically decreases with the number of iterations t, in the same way as eq. (2).
8. If $t < t_{max}$ go back to step 5.

2.5 Tearing Non-contractible Cycles in Graphs

The tearing procedure described below is based on Lee and Verleysen's method (2005). The only difference is that they use a simplified search procedure to compute the shortest distances in a graph, while we use Dijkstra's algorithm [3].

Let $G = (V, E)$ be the connectivity graph generated by the NG-CHR procedure described in 2.3, where V is the set of N nodes and E is the set of edges. Let $\mathcal{C}_{in}$ be the connectivity matrix associated to graph G. Let G_T be the shortest path tree (SPT). Let $\mathcal{C}_T$ be the connectivity matrix associated to graph G_T. Moreover, let G_C be a subgraph of G with no cycles including more than C edges, and $\mathcal{C}_C$ its corresponding connectivity matrix.

1. Let U be a $N \times N$ auxiliary matrix. Initialize U as the null matrix, and set $\mathcal{C}_C = \mathcal{C}_T$.
2. Sort the nodes of G_T, using a breadth-first traversal strategy, and store this order in an array p^*.
3. Let J be the set of nodes directly connected to the ith node in G. Given the i^{th} node $v_i \in p^*$, find all its neighbors, v_j, with $j = 1, \cdots, J$, that have a direct connection to it in the connectivity graph G. Store the nodes v_j in an array p^{**}.
4. For $j = 1, \cdots, J$, consider node $v_j \in p^{**}$. If $U(i,j) = 0$ calculate the shortest distance, d_{ij}, between nodes i and j using Dijkstra's algorithm in G. Check for cycles of length less than C. If $d_{ij} + 1 < C$, set $\mathcal{C}_C(i,j) = \mathcal{C}_C(j,i) = 1$ and $U(i,j) = U(j,i) = 1$. Go back to step 4 while $j \leq J$, else if $j > J$ go to step 3. Otherwise, if $U(i,j) = 1$ then the node has already been visited.
5. Stop if $i > N$, else go back to step 4 if $j \leq J$, or to step 3 if $j > J$.

2.6 Mapping Quality

A projection onto an output space is said to be trustworthy if the set of k nearest neighbors of a point in the map are also close by in the original space. Let $U_k(i)$ be the set of samples that are in the neighborhood of the ith point in the map but not in the original space. The measure of trustworthiness of the visualization, M_1, is defined as [7, 18]:

$$M_1(k) = 1 - A(k) \sum_{i=1}^{N} \sum_{z_j \in U_k(i)} (r(w_i, w_j) - k), \tag{13}$$

where $A(k) = 2/(Nk \times (2N - 3k - 1))$, and $r(w_i, w_j)$ is the ranking in input space.

A projection onto an output space is said to be continuous if the set of k closest neighbors of a point in the original space are also close by in the output space. Let $V_k(i)$ be the set of samples that are in the neighborhood of the ith point in the original space but not in the map. The measure of continuity, M_2, is defined as:

$$M_2(k) = 1 - A(k) \sum_{i=1}^{N} \sum_{w_j \in V_k(i)} (s(z_i, z_j) - k), \tag{14}$$

where $s(z_i, z_j)$ is the ranking in output space.

The topology preservation measure q_m [9] 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. The parameter n was set to $n = 0.4 \times k$, and k was varied from 5 to a maximum value K_q. The q_m measure takes values in $[0, 1]$, where $q_m = 1$ indicates a perfect neighborhood preservation. The global q_m measure is defined as:

$$q_m = \frac{1}{3nN} \sum_{j=1}^{N} \sum_{i=1}^{n} q_{m_{ji}}, \tag{15}$$

where

$$q_{m_{ji}} = \begin{cases} 3, \text{ if } NN_{ji^w} = NN_{ji^z} \\ 2, \text{ if } NN_{ji^w} = NN_{jl^z}, l \in [1, n], i \neq l \\ 1, \text{ if } NN_{ji^w} = NN_{jt^z}, t \in [n, k], n < k \\ 0, \text{ else.} \end{cases} \tag{16}$$

3 Simulation Results

In all simulations the parameters of the NG algorithm were set as follows: $\epsilon_0 = 0.5$, $\epsilon_f = 0.005$, $\lambda_0 = 30.0$, $\lambda_f = 0.01$, $T_0 = 20.0$, $T_f = 100.0$, $K = 2$. Likewise, the parameters of the GNLP algorithm were set as follows: $\alpha_0 = 0.3$ and $\alpha_f = 0.001$, $\sigma_0 = 0.7 \times N$, $\sigma_f = 0.1$, and $C = 6$, where N is the number of neurons.

Two artificial data sets were considered[1]: Cylinder and Torus. For each data set 300 codebook vectors were considered.The number of training epochs for NG was 20, the

[1] See http://www.dice.ucl.ac.be/˜lee

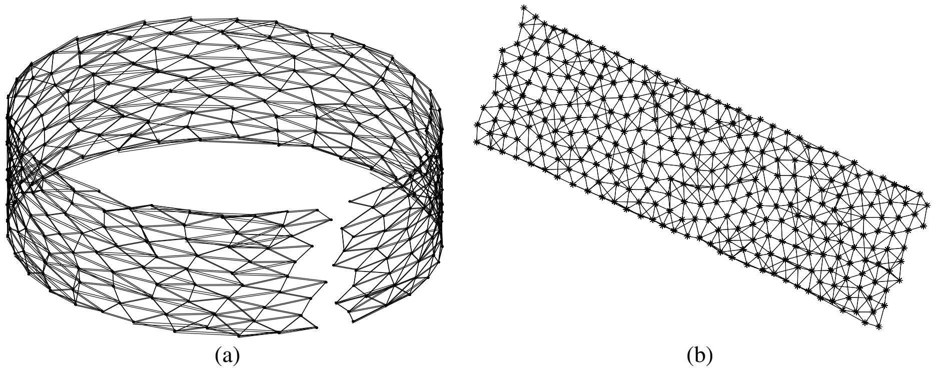

Fig. 1. (a) Cylinder quantized data set using 300 prototypes and a graph tearing procedure, and (b) Cylinder data projection with GNLP-NG

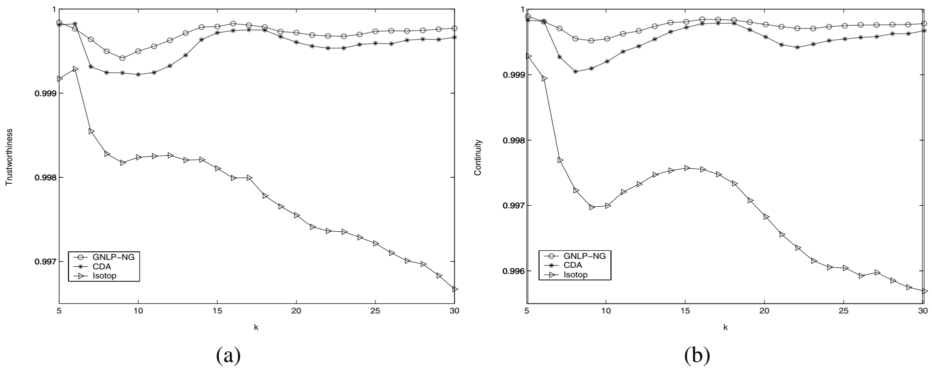

Fig. 2. (a) Trustworthiness measure and (b) continuity measure as a function of the number of neighbors k, for the Cylinder data set

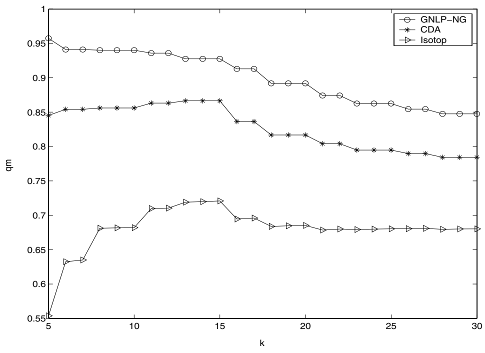

Fig. 3. Topology preservation measure q_m as a function of the number of neighbors, k, for the Cylinder data set

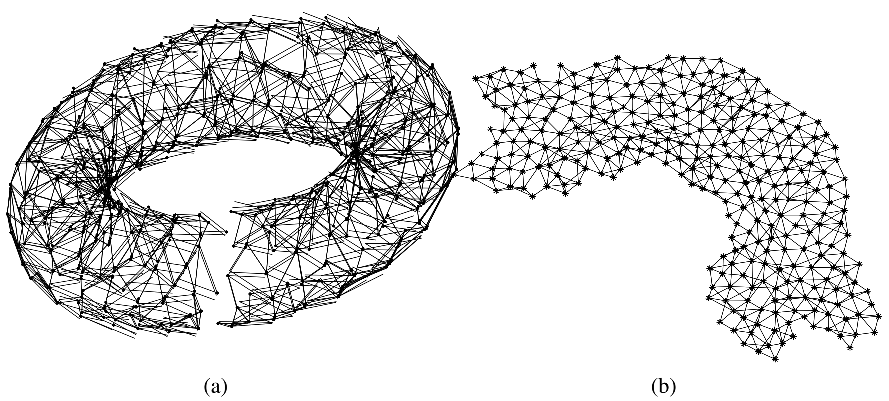

Fig. 4. (a) Torus quantized data using 300 prototypes and a graph tearing procedure, and (b)Torus data projection with GNLP-NG

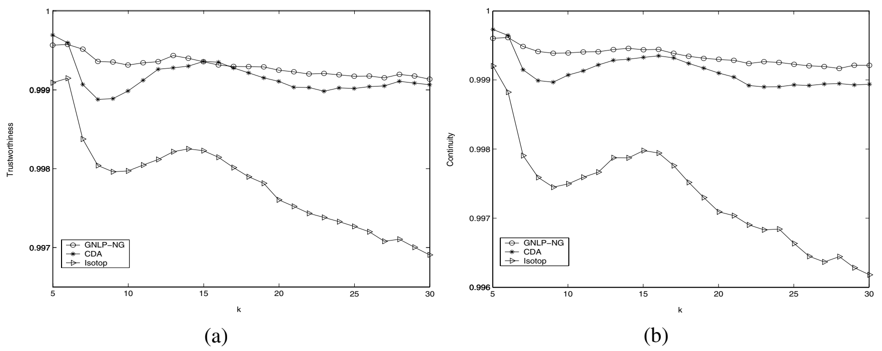

Fig. 5. (a) Trustworthiness measure and (b) continuity measure as a function of the number of neighbors k, for the Torus data set

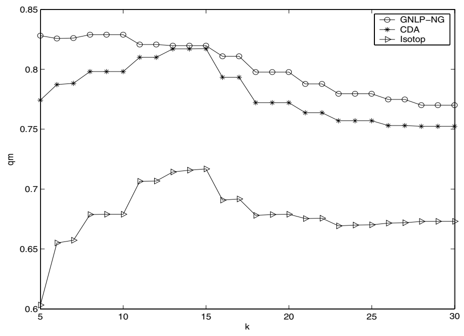

Fig. 6. Topology preservation measure q_m as a function of the number of neighbors, k, for the Torus data set

Table 1. CPU time, T, for the different steps of the GNLP-NG algorithm

	Cylinder	Torus
Algorithm	T [s]	T [s]
NG-CHR	347	188
SPT+Tearing	17	18
GNLP	472	270

same for GNLP. In the case of CDA, 30 epochs were used for training the competitive learning, 50 epochs for the nonlinear projection, and the parameter C was set to $C = 5$. The same setting was used for Isotop.

The Cylinder data set corresponds to a two-dimensional manifold embedded in a 3D space, from which 19600 samples were drawn. Due to the circularity of this manifold, its projection would get deformed or superimposed unless the corresponding connectivity graph is torn. The Torus data set corresponds to a 3D manifold, from which 10000 samples were drawn.

Fig. 1(a) shows the torn graph of the Cylinder quantized data set using NG and the tearing procedure. Fig. 1(b) illustrates the projection obtained with GNLP in output space. Fig. 2 shows (a) the trustworthiness measurement and (b) the continuity measurement, as a function of the number of nearest neighbors, k, for the Cylinder data set. It can be observed that the GNLP-NG method obtained the best trustworthiness and continuity measurements for all the k values explored (except one point), outperforming both CDA and Isotop. Fig. 3 shows the topology preservation measure q_m as a function of the number of neighbors. Here the GNLP-NG method outperformed CDA and Isotop for all k values.

Fig. 4(a) shows the torn graph of the Torus quantized data using NG and the tearing procedure. Fig. 4(b) illustrates the Torus data projection obtained with GNLP. Fig. 5 shows (a) the trustworthiness measurement and (b) the continuity measurement, as a function of the number of nearest neighbors, k, for the Torus data set. It can be observed that the best trustworthiness and continuity were obtained by GNLP-NG for $k > 6$, while CDA obtained a slightly better result for $k = 5$. Fig. 6 shows the topology preservation measure q_m as a function of the number of neighbors, for the Torus data set. Again GNLP-NG outperformed CDA and Isotop for all values of k explored.

Table 1 shows the CPU time in seconds for the different steps of the GNLP-NG algorithm, for both the Cylinder and Torus data sets. The algorithms were implemented in Matlab (except the Dijkstra's algorithm, which was implemented in C++), and executed in a Pentium IV 2.4GHz, 1GB RAM.

4 Conclusions

In the proposed GNLP-NG method the neighborhood graph is built online along with the vector quantization. This procedure allows obtaining a better graph representation than using competitive learning. A procedure for tearing or cutting graphs with non-contractible cycles was implemented, in order to make easier the nonlinear projection of manifolds with essential loops. The method is computationally efficient with

$O(N\log N)$ complexity. For two artificial data manifolds containing essential loops, the GNLP-NG method outperformed CDA and Isotop, in terms of the trustworthiness, continuity, and topology preservation measures.

Acknowledgement

This research was supported by Conicyt-Chile, under grant Fondecyt 1050751. We thank Dr. John Lee for providing us with a software implementation of CDA and Isotop.

References

1. Cox, T.F. and Cox, M.A.A.: *Multidimensional Scaling*, 2nd Edition, Boca Raton, Chapman & Hall/CRC, 2001.
2. Demartines, P., and Hérault, J.: Curvilinear component analysis: A self-organizing neural network for nonlinear mapping of data sets. *IEEE Trans. on Neural Networks*, **8** (1997) 148–154.
3. Dijkstra, F.W.: A note on two problems in connection with graphs. *Num. Math.*, **1** (1959) 269–271.
4. Estévez, P.A., and Chong, A.M.: Geodesic Nonlinear Projection using the Neural Gas Network. *Int. Joint Conference on Neural Networks*, Vancouver, Canada, 2006 (to appear).
5. Estévez, P.A., and Figueroa, C.J.: Online data visualization using the neural gas network. *Neural Networks*, (2006) (in press).
6. Estévez, P.A., and Figueroa, C.J.: Online nonlinear mapping using the neural gas network. *Proceedings of the Workshop on Self-Organizing Maps (WSOM'05)*, (2005) 299–306.
7. Kaski, S., Nikkilä, J., Oja, M., Venna, J., Törönen, P., and Castrén, E.: Trustworthiness and metrics in visualizing similarity of gene expression" *BMC Bioinformatics*, 4:48, (2003).
8. Kohonen, T.: *Self–Organizing Maps*, Berlin, Germany, Springer-Verlag, 1995.
9. König, A.: Interactive visualization and analysis of hierarchical neural projections for data mining. *IEEE Trans. on Neural Networks*, **11** (2000) 615–624.
10. Lee, J.A., and Verleysen, M.: Nonlinear dimensionality reduction of data manifolds with essential loops. *Neurocomputing*, **67** (2005) 29–53.
11. Lee, J.A., Lendasse, A., and Verleysen, M.: Nonlinear projection with curvilinear distances: Isomap versus curvilinear distance analysis, *Neurocomputing*, **57** (2004) 49–76.
12. Lee, J.A., and Verleysen, M.: Nonlinear projection with the Isotop method. *Proceedings of the International Conference on Artificial Neural Networks*, (2002) 933–938.
13. Lee, J.A., Lendasse, A., Donckers, N., and Verleysen, M.: A robust nonlinear projection method. *Proceedings of the European Symposium on Artificial Neural Networks (ESSAN'2000)*, (2000) 13–20.
14. Martinetz, T.M., and Schulten, K.J.: Topology representing networks. *Neural Networks*, **7** (1994) 507–522.
15. Martinetz, T.M., Berkovich, S.G., and Schulten, K.J.: 'Neural gas" network for vector quantization and its application to time-series prediction. *IEEE Trans. on Neural Networks*, **4** (1993) 558–569.
16. Martinetz, T.M., and Schulten, K.J.: A neural gas network learns topologies. *Artificial Neural Networks*, (1991) 397–402, Elsevier.
17. Prim, R.C.: Shortest connection networks and some generalizations. *Bell System Tech. J.*, **36** (1957) 1389–1401.
18. Venna, J., and Kaski, S.: Local multidimensional scaling with controlled tradeoff between trustworthiness and continuity. *Proceedings of the Workshop on Self-Organizing Maps (WSOM'05)*, (2005) 695–702.