The self-organizing map as a visual neighbor retrieval method

Kristian Nybo[1], Jarkko Venna[1,2], Samuel Kaski[1]
1) Adaptive Informatics Research Centre and Helsinki Institute for Information Technology, Laboratory of Computer and Information Science, Helsinki University of Technology
P.O. Box 5400, FI-02015 TKK, Finland
2) Numos Oy, Vasamatie 6 B, FI-02630 ESPOO, Finland
email: {kristian.nybo, jarkko.venna, samuel.kaski}@tkk.fi

Keywords: information visualization, self-organizing map, visual information retrieval

Abstract— We have recently introduced rigorous goodness criteria for information visualization by posing it as a visual neighbor retrieval problem, where the task is to find proximate high-dimensional data based only on a low-dimensional display. Standard information retrieval criteria such as precision and recall can then be used for information visualization, and we introduced an algorithm, Neighbor Retrieval Visualizer (NeRV), to optimize the total cost of retrieval errors. NeRV was shown to outperform alternative methods, but the comparisons did not include one of the methods widely used for information visualization, namely the Self-Organizing Map (SOM). In empirical experiments of this paper the SOM turns out to be comparable to the best methods in terms of smoothed precision, but not in terms of recall. On a related measure called trustworthiness, the SOM outperforms all others. Finally, we suggest that for information visualization tasks the free parameters of the SOM could be optimized with cross-validation to maximize its visual information retrieval performance. This would remove the need to choose the size of the SOM grid and the final radius by rules of thumb.

1 Introduction

Traditionally nonlinear dimensionality reduction methods have been used either as (i) preprocessing methods to reduce the number of input variables or represent the inputs in terms of more natural variables describing the embedded data manifold, or as (ii) a way of making the data set more understandable, by making the similarity relationships between data points explicit through visualizations. Visualizations are commonly needed in exploratory data analysis and in interfaces to high-dimensional data.

Early nonlinear projection methods introduced a representation for the data points and optimized the representations to minimize representation error. Most can be interpreted as multidimensional scaling (MDS) methods [2] that minimize some measure of preservation of pairwise distances between data points.

Manifold learning methods construct the projection by searching for data manifolds embedded in the original data space. Isomap [13] infers the manifold through local neighborhood relationships, and visualizes it by MDS; Locally Linear Embedding (LLE) [10] approximates the manifold locally by linear surfaces; Laplacian Eigenmap (LE) [1] and Hessian Eigenmap (HLLE) [4], are very similar but based on graph theory; Semidefinite Embedding (SDE) [18] aims at maximizing the variance in the feature space while preserving the distances between neighbors; Alignment of Local Models (ALM) [17] and other similar approaches first fit local models to the data and then search for a transformation that aligns them globally. Finally, there are more heuristically derived but surprisingly well-performing algorithms, such as the Curvilinear Components Analysis (CCA) [3].

The Self-Organizing Map (SOM) [8] has been used for both of the dimensionality reduction tasks, but it has not been compared with the most recent methods. In this paper we will carry out the comparison. We will additionally introduce a method for finding the best grid size and final radius for visualizing a given data set with a SOM.

2 Visual neighbor retrieval

Although visualization algorithms and applications have sometimes been motivated using an information retrieval task, the connection has been vague. We show a more explicit connection by showing how typical usage of a visualization of a data set can be interpreted as an information retrieval task.

Consider a data analyst exploring a high-dimensional data set. For example, the data could consist of a large set of indicators of welfare for all the countries in the world. The analyst selects a country, say Finland, and wishes to find out which other countries have a similar welfare profile. He or she uses the visualization to pick a few countries that appear most similar, after which the profiles of these countries can be investigated in more detail to determine how exactly they are related to Finland's profile. To that end, the visualization system should provide a visual interface that allows neighbors, similar objects, to be selected from the display. Additionally, we assume that the analyst has not decided on the country of interest before he or she sees the visualization, so the display should not be biased towards any particular set of objects.

Assuming that the user of our visualization will have needs similar to those of the data analyst described above, we define our visualization task as one of *visual neighbor retrieval*, where the goal of the user is to find a small set of nearest neighbors of a few interesting data points. The k nearest neighbors of a data point are the set of k data points that lie closest to it in the original data set. The purpose of a visualization algorithm is then to provide a single display that allows the user to find the neighbors of a data point as faithfully as possible, without prior knowledge of which data points the user is going to select.

Three methods have been specifically designed with visual neighbor retrieval in mind: Neighbor Retrieval Visualizer (NeRV) [16], fastNeRV [16] and LocalMDS [15]. There is also some evidence [7, 14] that the SOM would perform well in this task.

2.1 Neighbor Retrieval Visualizer

The Stochastic Neighbor Embedding (SNE) algorithm [5] was originally motivated as a method for placing a set of objects into a low-dimensional space in a configuration that preserves neighbor identities. Such a projection does not try to preserve pairwise distances as such, as MDS does, but instead the *probabilities* of points being neighbors.

A probability distribution is defined in the input space, based on the pairwise distances, to describe how likely it is that the point i is a neighbor of point j. The same is done in the low-dimensional output or projection space. The algorithm then optimizes the configuration of points in the output space so that the original distribution of neighborness is approximated as closely as possible in the output space.

More formally, the probability p_{ij} of the point i being a neighbor of point j in the input space is defined to be

$$p_{ij} = \frac{\exp\left(-\frac{d(\mathbf{x}_i,\mathbf{x}_j)^2}{\sigma_i^2}\right)}{\sum_{k\neq i}\exp\left(-\frac{d(\mathbf{x}_i,\mathbf{x}_k)^2}{\sigma_i^2}\right)}, \tag{1}$$

where $d(\mathbf{x}_i, \mathbf{x}_j)$ is the Euclidean distance between the data points $\mathbf{x}_i$ and $\mathbf{x}_j$. The width of the Gaussian, σ_i, is set either manually or by fixing the entropy of the distribution. Setting the entropy equal to $\log k$ sets the “effective number of neighbors” to k.

Similarly, the probability of the point i being a neighbor of point j in the output space is defined to be

$$q_{ij} = \frac{\exp\left(-\frac{\|\mathbf{y}_i-\mathbf{y}_j\|^2}{\sigma_i^2}\right)}{\sum_{k\neq i}\exp\left(-\frac{\|\mathbf{y}_i-\mathbf{y}_k\|^2}{\sigma_i^2}\right)}. \tag{2}$$

The SNE algorithm searches for the configuration of points $\mathbf{y}_i$ that minimizes the KL divergence D between the probability distributions in the input and output spaces, averaged over all points.

It was shown in [16] that optimizing the cost function of SNE is equivalent to optimizing a smoothed version of recall in a visual neighbor retrieval context, and that by reversing the direction of the KL-divergence in the cost function of SNE we get a method that optimizes smoothed precision instead of recall. In practice, however, it would be best to optimize a compromise between precision and recall. The compromise has a fundamental first-principles interpretation. Precision is proportional to the number of false positives and recall to the number of misses. If we know the relative cost of each type of error, we can compute the total cost as their weighted sum, and minimization of the total cost obviously is *the* natural objective.

If we assign a relative cost λ to misses and $(1-\lambda)$ to false positives, then the total cost function to be optimized is

$$\begin{aligned} E_{\text{NeRV}} &= \lambda E_i[D(p_i, q_i)] + (1-\lambda)E_i[D(q_i, p_i)] \\ &= \lambda \sum_i \sum_{j\neq i} p_{ij}\log\frac{p_{ij}}{q_{ij}} + (1-\lambda)\sum_i\sum_{j\neq i} q_{ij}\log\frac{q_{ij}}{p_{ij}}. \end{aligned} \tag{3}$$

For step functions and small δ this reduces to a total information retrieval cost, a compromise between precision and recall, and for Gaussian functions as in SNE it can be interpreted as a smoothed cost. We call the method that optimizes (3) *Neighbor Retrieval Visualizer (NeRV)* , since it interprets the visualization problem as a problem of retrieving neighbors based on the visualization display.

By setting the parameter $\lambda \in [0, 1]$ we choose to focus more on either the probabilities that are high in the input space (recall) or in the output space (precision). When $\lambda = 1$ the method becomes SNE and when $\lambda = 0$ it focuses purely on avoiding false positives.

We optimize the cost function using a conjugate gradient algorithm. A heuristic but very effective way of avoiding local minima is to initialize the optimization by starting with a large width of the Gaussian neighborhood, σ_i^2, and reducing it stepwise after each optimization step until the final value is reached. After this initialization, normal conjugate gradients are run with a fixed Gaussian for each data point.

2.2 fNeRV

The main drawback of the NeRV algorithm is its computational complexity. Each gradient step is of complexity $\mathcal{O}(n^3)$, where n is the number of data points. This makes NeRV practical only for relatively small data sets. One approach to reducing the computational cost is to use approximations of the cost function of NeRV [16].

Zhu and Rohwer [19] have developed an information-geometric extension of the Kullback-Leibler divergence that is valid for all positive measures instead of just normalized ones. The extended divergence is

$$D_{KLe}(p_i, q_i) = \sum_{j \neq i} q_{ij} - p_{ij} + p_{ij} \log \frac{p_{ij}}{q_{ij}}. \quad (4)$$

By replacing the Kullback-Leibler divergences in the NeRV cost function with the extended divergence we can use the exponential density values in p_{ij} and q_{ij} without the normalization. This reduces the complexity of the gradient step to $\mathcal{O}(n^2)$ which is comparable with other distance based methods. The method optimizing this approximate cost function is called *fast Neighbor Retrieval Visualizer (fNeRV)*.

2.3 LocalMDS

We have previously [15] introduced a method called *Local Multidimensional Scaling (LocalMDS)* that can be thought of as a heuristic but faster method than NeRV, aimed at achieving the same goals. It is a derivative of CCA with an indirect ability to control the tradeoff between precision and recall.

The CCA cost function penalizes errors in preserving distances of points that are neighbors in the *output space*. By disregarding errors in distances between points that are far from each other in the visualization, CCA allows discontinuities to be created in the mapping. These discontinuities allow better preservation of local distances on both sides of a discontinuity (a “cut” in the data manifold), which usually leads to high precision in the resulting visualization. On the other hand, the discontinuities reduce recall.

In LocalMDS a term is added to the cost function to discourage formation of discontinuities. This is achieved by additionally penalizing errors in distances between points that are close by in the *input space*. The tradeoff between these two terms governs the tradeoff between trustworthiness and continuity. The cost function of LocalMDS is

$$\begin{aligned} E = \\ \frac{1}{2}\sum_i \sum_{j \neq i} [(1-\lambda)(d(\mathbf{x}_i, \mathbf{x}_j) - d(\mathbf{y}_i, \mathbf{y}_j))^2 F(d(\mathbf{y}_i, \mathbf{y}_j), \sigma_i) \\ + \lambda(d(\mathbf{x}_i, \mathbf{x}_j) - d(\mathbf{y}_i, \mathbf{y}_j))^2 F(d(\mathbf{x}_i, \mathbf{x}_j), \sigma_i)] \end{aligned} \quad (5)$$

where the parameter $\lambda \in [0 \ldots 1]$ controls the tradeoff and, as in the CCA algorithm, during the optimization the radius of the area of influence around data point i, σ_i, is slowly reduced. The coefficient $F(d(\mathbf{x}_i, \mathbf{x}_j), \sigma_i)$ emphasizes local distances; here, F is simply a step function that equals 1 when $d(\mathbf{x}_i, \mathbf{x}_j) < \sigma_i$ and 0 otherwise. The final radius is set equal to the distance of the k:th nearest neighbor of the data point i in the original space. In practice a value around $k = 20$ has proven to be a good choice on the data sets tested.

When λ is set to zero the cost function is reduced to that of the basic CCA algorithm, with the exception that the neighborhood end radiuses are set differently for each data point. A good setting for λ is usually in the range $[0 \ldots 0.5]$. The cost function is optimized with the stochastic gradient descent introduced for CCA in [3].

2.4 The dredviz package

An implementation of the NeRV, fNeRV and local MDS algorithms, as well as the trustworthiness and continuity measures, is available at http://www.cis.hut.fi/projects/mi/software/dredviz. The implementation is open source software and written in ANSI/ISO C++, meaning that it should compile and run on any platform with an ANSI/ISO C++ compiler.

3 Comparison of the SOM with other information visualization methods

We compared the visualization performance of the SOM with that of other methods on three data sets. The first is a small, artificial, low-dimensional set; the two others are high-dimensional real-world sets. The SOM had a hexagonal grid of 12 by 16 units, resulting in approximately five data points for each grid unit.

3.1 Data sets

Thick S-curve. The first data set is a simple toy set of 1000 points sampled from a folded low-dimensional manifold, a two-dimensional S-shaped surface embedded in $\mathbf{R}^3$. The surface is generated by sampling points uniformly from a planar S-shaped curve made up of two unit circle halves and adding a uniformly distributed displacement orthogonal to the plane of the curve. We introduce some noise into the manifold by adding a spherical normally distributed displacement with a standard deviation of 0.6.

Mouse gene expression. The second data set is a collection of gene expression profiles from different mouse tissues [12]. Expression of over 13,000 mouse genes had been measured in 45 tissues. We used an extremely simple filtering method, similar to that originally used in [12], to select the genes for visualization. Of the mouse genes clearly expressed (average difference in Affymetrix chips, AD $>$ 200) in at least one of the 45 tissues (dimensions), a random sample of 1600 genes (points) was selected. After this the variance in each tissue was normalized to unity.

Gene expression compendium. The third data set is a large collection of human gene expression arrays [11, `http://dags.stanford.edu/cancer`]. Since the current implementations of all methods do not tolerate missing data we removed samples with missing values altogether. First we removed genes that were missing from more than 300 arrays. Then we removed the arrays for which values were still missing. This resulted in a data set containing 1278 points and 1339 dimensions.

3.2 Dimensionality reduction methods included

In addition to NeRV and fNeRV, we compared the SOM with the following dimensionality reduction methods: Principal Component Analysis (PCA) [6], metric Multidimensional Scaling (MDS) [2], Locally Linear Embedding (LLE) [10], Laplacian Eigenmap (LE) [1], Hessian Eigenmap (HLLE) [4], Isomap [13], Curvilinear Component Analysis (CCA) [3], Local MDS (LMDS) [15], and Curvilinear Distance Analysis (CDA) [9], which is a variant of CCA that uses graph distances to approximate the geodesic distances in the data. LLE, LE, HLLE and Isomap were computed using code from their developers; MDS, CCA, CDA, and LMDS use our own code.

3.3 Performance measures used

We used three pairs of performance measures to compare the SOM with the other methods. Given our formulation of information visualization in terms of visual neighbor retrieval, smoothed recall and smoothed precision are a natural first choice.

Smoothed recall and smoothed precision are sensitive to errors in local structure: rather than just checking whether the k nearest points are the same in both spaces, which would fulfill the requirements of simplest formulations of information retrieval, the measures also penalize violations in the internal structure of the neighborhood. In many tasks this sensitivity is desirable, but if the user of the visualization is only interested in retrieving a certain number of neighbors, and does not care about the exact structure of the retrieved neighborhood, the measures do not quite match the task. Thus we chose trustworthiness and continuity [7] to complement our first pair of measures: they are well established and completely indifferent to the relationships between the k nearest points as long as the k nearest points are the same in the input and output space.

Finally, because our motivation is based in information retrieval, we also plotted standard precision–recall curves.

3.4 Results

The right column of Figure 1 shows trustworthiness and continuity plotted for all methods and all three data sets. In terms of trustworthiness, the SOM beats every other method by a wide margin on both gene expression data sets, and is also among the top performers on the S-curve data. In the Figure trustworthiness and continuity are calculated with a neighborhood of $k = 20$, but we found that on the gene expression data sets, the SOM had the highest trustworthiness for any value of k between 1 and 50. The SOM also did very well in terms of smoothed precision, matching or beating NeRV on the gene expression data sets, even though NeRV optimizes the measure directly for $\lambda = 0$. When measured by smoothed recall, however, NeRV was clearly better on all three data sets.

The reason why the SOM can sometimes be even better in terms of precision than a method designed to optimize that measure is probably that, in the SOM displays, we search for the neighboring data points using the path distance computed along the SOM lattice. The edges are weighted by the distance between the corresponding model vectors, which corresponds roughly to visual distances on so-called U-matrix displays. The weighted path distances are not Euclidean any longer, whereas the comparison methods have to settle for Euclidean distances. If we use non-weighted (and hence two-dimensional) distances, the SOM is not as good.

The effect of varying the final radius on the trustworthiness–continuity trade-off was erratic: while trustworthiness tended to decrease as the final radius was increased, a corresponding systematic increase in continuity is visible only for the S-curve data set. Smoothed precision and smoothed recall responded slightly more regularly, as increasing the final radius resulted in a marked increase in smoothed recall on two of the three data sets. Clearly the final radius affects these trade-offs, but it does not seem as if we can, in general, expect to control them by adjusting that parameter alone.

4 Optimizing the SOM for information visualization

Choosing the size of the grid and the final radius is a traditional non-trivial problem faced in any application of the SOM. We saw in Section 3.4 that the final radius influences all our measures of goodness for information visualization, in some cases very strongly. Unfortunately we also noticed that there was no simple, general relationship between the measures and the final radius that would allow us to formulate a rule for choosing the parameter optimally for information visualization. We suspected that the reason why the measures responded so unpredictably to changes in the final radius was that they were also affected by the size of the SOM grid, and that the effects of the two parameters were not independent, which would mean that one parameter could not be chosen optimally without regard to the other.

The solution we propose involves n-fold cross-validation using trustworthiness and continuity for testing the maps. We divide the training data set S into n equal-sized parts $S_1, \ldots S_n$. For each combination of parameters that we wish to test, we then train n maps, where the training data

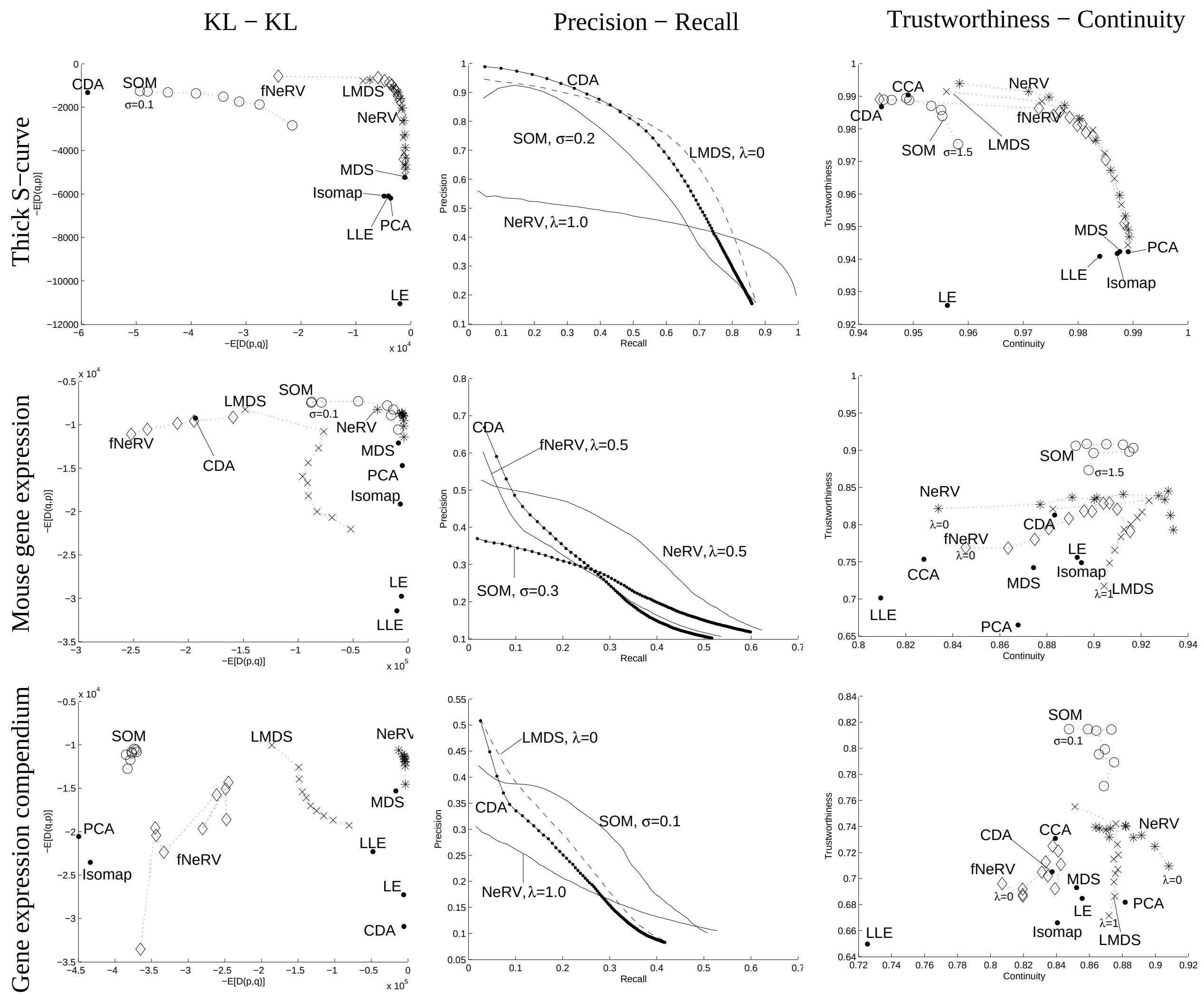

Figure 1: KL–KL curves (left), precision–recall curves (middle) and trustworthiness–continuity curves (right) for different values of λ (NeRV, fNerv, LocalMDS) and σ (SOM) on three data sets. The precision–recall curves were calculated with the 20 nearest neighbors in the input space as the set of relevant items and the number of retrieved items (neighbors) is varied from 1 to 100. The KL–KL curve and the trustworthiness–continuity curves were calculated using a neighborhood of 20 points. For the SOM, both KL–KL and trustworthiness–continuity were calculated using the path distance computed along the SOM lattice. On each plot, the best performance is in the top right corner. PCA: Principal Component Analysis; MDS: metric Multidimensional Scaling; LLE: Locally Linear Embedding; LE: Laplacian Eigenmap; CCA: Curvilinear Component Analysis; CDA: CCA with geodesic distances; HLLE: Hessian Eigenmap.

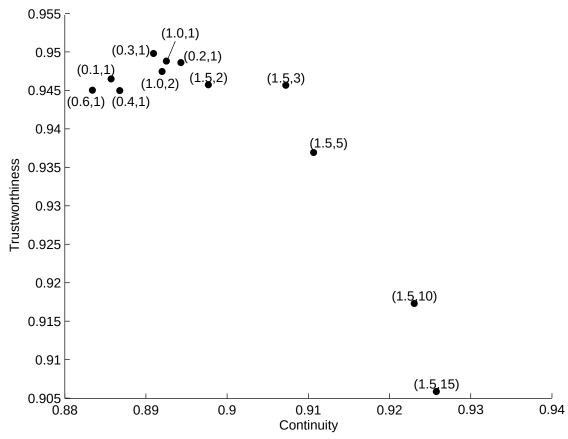

Figure 2: The results of the cross-validation procedure. Each point corresponds to one combination of final radius and grid size: the first number of the label is the final radius, and the second the number of data points per grid unit. For clarity, only the best performers, as well as the combinations that performed best in the actual test, are included.

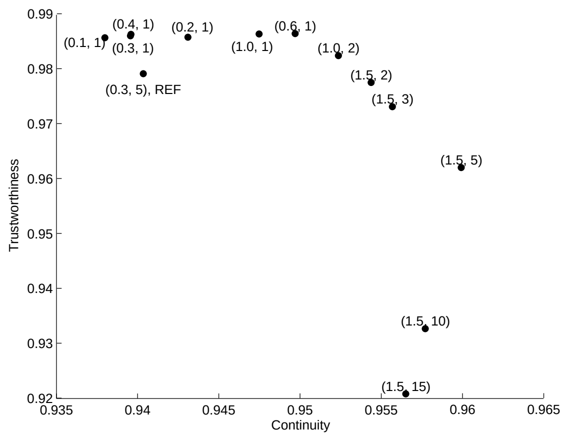

Figure 3: Continuity and trustworthiness ($k = 20$) for SOMs of various parameter combinations when applied to a data set different from (but generated from the same distribution as) the one they were trained on. Each point corresponds to one combination of final radius and grid size: the first number of the label is the final radius, and the second the number of data points per grid unit. For clarity, only the best performers, as well as the combinations that performed best in the cross-validation, are included. The most trustworthy parameter combination in the method comparison in Section 3, (0.3, 5), is shown for reference.

for the ith map consists of $S \backslash S_i$, that is, S with S_i removed. The removed part S_i is used as validation data: we find the best-matching units on the newly-trained map for the points in S_i, and then calculate the trustworthiness T_i and continuity C_i of that projection. We then use the averages of these measures, $T = (1/n) \sum_{i=1}^{n} T_i$ and $C = (1/n) \sum_{i=1}^{n} C_i$, to compare different parameter combinations.

We applied the cross validation-based method to the S-curve data set. This data set was chosen in part because it was by far the fastest to visualize due to its low dimensionality, but above all because we could easily generate a new data set with same distribution as the training set for testing purposes, which we could not have done with either of the real world data sets. We divided the data set into seven parts. The length to width ratio of the grid of each map trained was approximately 1.5. Using the results for the cross-validation, displayed in Figure 2, we would now choose the parameter combination that best matches our preferences. For example, if we wanted to maximize trustworthiness at any cost to continuity, we would use a final radius of 0.3 and a grid with as many units as there are points in the data set.

To test how good the parameter combinations recommended by our method actually were, we trained one map for every parameter combination with the complete S-data set, and then calculated trustworthiness and continuity for each map using a new set generated from the same distribution. The results are plotted in Figure 3.

When we compare Figures 2 and 3, we see that the parameter combinations suggested by our method are trustworthy: if we look at the best combinations suggested by our method, displayed in Figure 2, we see that all of these are among or near the best performing combinations in the actual test Figure 3. There are also no parameter combinations that turn out to be significantly better than the best combinations suggested by our method.

In particular, note that the cross-validation method would give us better parameter combinations than the ones we used in our method comparison in Section 3, where the grid size was fixed to five data points per code vector using a rule of thumb.

5 Discussion

We have compared the SOM with a variety of nonlinear dimensionality reduction methods in the task of visual neighbor retrieval. The SOM produced the most trustworthy projection by far for the two real world data sets, and was similar to or even slightly better than NeRV in terms of smoothed precision. Good performance as measured by these two measures is of primary importance in visual neighbor retrieval, because the user of the visualization must be able to trust the information provided by the visualization for it to be useful. The SOM also did well, if not quite as well as NeRV, in terms of continuity, but when measured by smoothed recall, its performance was clearly worse than that of NeRV.

We introduced a method based on n-fold cross-validation for choosing the grid size and final radius of a SOM to produce an optimal visualization of a given data set. The results of our experiment with the artificial S-curve data set were very encouraging. All the produced parameter combinations performed well with test data different from the one the SOMs were trained on (precision), and the method managed to find most of the best-performing parameter combinations (recall). What is more, our results showed a clear relationship between the parameters and the trustworthiness–continuity trade-off for this data set. A small final radius with a large grid produced the most trustworthy but least continuous projections, whereas a large final radius and small grid produced the most continuous but least trustworthy projections. Maps with small final radii and small grids produced the worst projections in terms of both measures.

Although the results are promising, further study is needed to determine how the cross-validation method for choosing parameters works for more general data sets. The data set we used was a low-dimensional folded manifold, whereas many real world data sets are very high-dimensional and do not have such clear-cut structure.

We would additionally like to point out that calculating trustworthiness and continuity directly for the mapping of the validation data, as we did in the cross-validation, may be misleading when the grid size of the map is large compared to the size of the validation data. If the validation data consists of relatively few points scattered all over the map, the resolution is very different from what it would be for the whole data set. One way to get around this potential problem would be to project both the training data and the validation data, and then calculate trustworthiness and continuity, but ignoring any errors that do not involve one of the points belonging to the validation data.

Acknowledgements

This work was supported by the Academy of Finland (decision number 207467), as well as the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views. All rights reserved because of other commitments.

References

[1] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 585–591, Cambridge, MA, 2002. MIT Press.

[2] Ingwer Borg and Patrick Groenen. *Modern Multidimensional Scaling*. Springer, New York, 1997.

[3] Pierre Demartines and Jeanny Hérault. Curvilinear component analysis: A self-organizing neural network for nonlinear mapping of data sets. *IEEE Transactions on Neural Networks*, 8(1):148–154, January 1997.

[4] David L. Donoho and Carrie Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *PNAS*, 100:5591–5596, 2003.

[5] Geoffrey Hinton and Sam Roweis. Stochastic neighbor embedding. In *Advances in Neural Information Processing Systems 15*, pages 833–840. MIT Press, 2002.

[6] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441,498–520, 1933.

[7] Samuel Kaski, Janne Nikkilä, Merja Oja, Jarkko Venna, Petri Törönen, and Eero Castrén. Trustworthiness and metrics in visualizing similarity of gene expression. *BMC Bioinformatics*, 4:48, 2003.

[8] Teuvo Kohonen. *Self-Organizing Maps*. Springer, Berlin, 3rd edition, 2001.

[9] John Aldo Lee, Amaury Lendasse, and Michel Verleysen. Nonlinear projection with curvilinear distances: Isomap versus curvilinear distance analysis. *Neurocomputing*, 57:49–76, 2004.

[10] Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

[11] Eran Segal, Nir Friedman, Daphne Koller, and Aviv Regev. A module map showing conditional activity of expression modules in cancer. *Nature Genetics*, 36:1090–1098, 2004.

[12] Andrew I. Su, Michael P. Cooke, Keith A. Ching, Yaron Hakak, John R. Walker, Tim Wiltshire, Anthony P. Orth, Raquel G. Vega, Lisa M. Sapinoso, Aziz Moqrich, Ardem Patapoutian, Garret M. Hampton, Peter G. Schultz, and John B. Hogenesch. Large-scale analysis of the human and mouse transcriptomes. *Proceedings of the National Academy of Sciences*, 99:4465–4470, 2002.

[13] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[14] Jarkko Venna and Samuel Kaski. Neighborhood preservation in nonlinear projection methods: An experimental study. In G. Dorffner, H. Bischof, and K. Hornik, editors, *Proceedings of ICANN 2001, International Conference on Artificial Neural Networks*, pages 485–491, Berlin, 2001. Springer.

[15] Jarkko Venna and Samuel Kaski. Local multidimensional scaling. *Neural Networks*, 19:889–899, 2006.

[16] Jarkko Venna and Samuel Kaski. Nonlinear dimensionality reduction as information retrieval. In Marina Meila and Xiaotong Shen, editors, *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics (AISTATS*07)*, pages 568–575, 2007.

[17] Jakob J Verbeek, Sam T Roweis, and Nikos Vlassis. Non-linear CCA and PCA by alignment of local models. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, Cambridge, MA, 2004. MIT Press.

[18] Kilian Q. Weinberger, Fei Sha, and Lawrence K. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *ICML '04: Proceedings of the 21st International Conference on Machine Learning*, pages 839–846, New York, NY, USA, 2004. ACM Press.

[19] Huaiyu Zhu and Richard Rohwer. Information geometric measurement of generalization. Technical Report NCRG/4350, Neural Computing Research Group, Aston University, 1995.