Journal of Machine Learning Research 11 (2010) 451-490 Submitted 4/09; Revised 12/09; Published 2/10

Information Retrieval Perspective to Nonlinear Dimensionality Reduction for Data Visualization

Jarkko Venna JARKKO.VENNA@NUMOS.FI
Jaakko Peltonen JAAKKO.PELTONEN@TKK.FI
Kristian Nybo KRISTIAN.NYBO@TKK.FI
Helena Aidos HELENA.AIDOS@TKK.FI
Samuel Kaski SAMUEL.KASKI@TKK.FI
Aalto University School of Science and Technology
Department of Information and Computer Science
P.O. Box 15400, FI-00076 Aalto, Finland

Editor: Yoshua Bengio

Abstract

Nonlinear dimensionality reduction methods are often used to visualize high-dimensional data, although the existing methods have been designed for other related tasks such as manifold learning. It has been difficult to assess the quality of visualizations since the task has not been well-defined. We give a rigorous definition for a specific visualization task, resulting in quantifiable goodness measures and new visualization methods. The task is *information retrieval* given the visualization: to find similar data based on the similarities shown on the display. The fundamental tradeoff between precision and recall of information retrieval can then be quantified in visualizations as well. The user needs to give the relative cost of missing similar points vs. retrieving dissimilar points, after which the total cost can be measured. We then introduce a new method NeRV (*neighbor retrieval visualizer*) which produces an optimal visualization by minimizing the cost. We further derive a variant for supervised visualization; class information is taken rigorously into account when computing the similarity relationships. We show empirically that the unsupervised version outperforms existing unsupervised dimensionality reduction methods in the visualization task, and the supervised version outperforms existing supervised methods.

Keywords: information retrieval, manifold learning, multidimensional scaling, nonlinear dimensionality reduction, visualization

1. Introduction

Visualization of high-dimensional data sets is one of the traditional applications of nonlinear dimensionality reduction methods. In high-dimensional data, such as experimental data where each dimension corresponds to a different measured variable, dependencies between different dimensions often restrict the data points to a manifold whose dimensionality is much lower than the dimensionality of the data space. Many methods are designed for *manifold learning*, that is, to find and unfold the lower-dimensional manifold. There has been a research boom in manifold learning since 2000, and there now exist many methods that are known to unfold at least certain kinds of manifolds successfully. Some of the successful methods include isomap (Tenenbaum et al., 2000), locally linear embedding (LLE; Roweis and Saul, 2000), Laplacian eigenmap (LE; Belkin and Niyogi, 2002a), and maximum variance unfolding (MVU; Weinberger and Saul, 2006).

It has turned out that the manifold learning methods are not necessarily good for information visualization. Several methods had severe difficulties when the output dimensionality was fixed to two for visualization purposes (Venna and Kaski, 2007a). This is natural since they have been designed to find a manifold, not to compress it into a lower dimensionality.

In this paper we discuss the specific visualization task of projecting the data to points on a two-dimensional display. Note that this task is different from manifold learning, in case the inherent dimensionality of the manifold is higher than two and the manifold cannot be represented perfectly in two dimensions. As the representation is necessarily imperfect, defining and using a *measure of goodness* of the representation is crucial. However, in spite of the large amount of research into methods for extracting manifolds, there has been very little discussion on what a good two-dimensional representation should be like and how the goodness should be measured. In a recent survey of 69 papers on dimensionality reduction from years 2000–2006 (Venna, 2007) it was found that 28 ($\approx 40\%$) of the papers only presented visualizations of toy or real data sets as a proof of quality. Most of the more quantitative approaches were based on one of two strategies. The first is to measure preservation of all pairwise distances or the order of all pairwise distances. Examples of this approach include the multidimensional scaling (MDS)-type cost functions like Sammon's cost and Stress, methods that relate the distances in the input space to the output space, and various correlation measures that assess the preservation of all pairwise distances. The other common quality assurance strategy is to classify the data in the low-dimensional space and report the classification performance.

The problem with using the above approaches to measure visualization performance is that their connection to visualization is unclear and indirect at best. Unless the purpose of the visualization is to help with a classification task, it is not obvious what the classification accuracy of a projection reveals about its goodness as a visualization. Preservation of pairwise distances, the other widely adopted principle, is a well-defined goal; it is a reasonable goal if the analyst wishes to use the visualization to assess distances between selected pairs of data points, but we argue that this is not the typical way how an analyst would use a visualization, at least in the early stages of analysis when no hypothesis about the data has yet been formed. Most approaches including ours are based on pairwise distances at heart, but we take into account the context of each pairwise distance, yielding a more natural way of evaluating visualization performance; the resulting method has a natural and rigorous interpretation which we discuss below and in the following sections.

In this paper we make rigorous the specific information visualization task of projecting a high-dimensional data set onto a two-dimensional plane for visualizing similarity relationships. This task has a very natural mapping into an information retrieval task as will be discussed in Section 2. The conceptualization as information retrieval explicitly reveals the necessary tradeoff between precision and recall, of making true similarities visible and avoiding false similarities. The tradeoff can be quantified exactly once costs have been assigned to each of the two error types, and once the total cost has been defined, it can be optimized as will be discussed in Section 3. We then show that the resulting method, called NeRV for *neighbor retrieval visualizer*, can be further extended to supervised visualization, and that both the unsupervised and supervised methods empirically outperform their alternatives. NeRV includes the previous method called stochastic neighbor embedding (SNE; Hinton and Roweis, 2002) as a special case where the tradeoff is set so that only recall is maximized; thus we give a new information retrieval interpretation to SNE.

This paper extends our earlier conference paper (Venna and Kaski, 2007b) which introduced the ideas in a preliminary form with preliminary experiments. The current paper gives the full justification and comprehensive experiments, and also introduces the supervised version of NeRV.

2. Visualization as Information Retrieval

In this section we define formally the specific visualization task; this is a novel formalization of visualization as an *information retrieval task*. We first give the definition for a simplified setup in Section 2.1, and then generalize it in Section 2.2.

2.1 Similarity Visualization with Binary Neighborhood Relationships

In the following we first define the specific visualization task and a cost function for it; we then show that the cost function is related to the traditional information retrieval measures *precision* and *recall*.

2.1.1 TASK DEFINITION: SIMILARITY VISUALIZATION

Let $\{\mathbf{x}_i\}_{i=1}^N$ be a set of input data samples, and let each sample i have an *input neighborhood* P_i, consisting of samples that are close to i. Typically, P_i might consist of all input samples (other than i itself) that fall within some radius of i, or alternatively P_i might consist of a fixed number of input samples most similar to i. In either case, let r_i be the size of the set P_i.

The *goal of similarity visualization* is to produce low-dimensional output coordinates $\{\mathbf{y}_i\}_{i=1}^N$ for the input data, usable in visual information retrieval. Given any sample i as a query, in visual information retrieval samples are retrieved based on the visualization; the retrieved result is a set Q_i of samples that are close to $\mathbf{y}_i$ in the visualization; we call Q_i the *output neighborhood*. The Q_i typically consists of all input samples j (other than i itself) whose visualization coordinates $\mathbf{y}_j$ are within some radius of $\mathbf{y}_i$ in the visualization, or alternatively Q_i might consist of a fixed number of input samples whose output coordinates are nearest to $\mathbf{y}_i$. In either case, let k_i be the number of points in the set Q_i. The number of points in Q_i may be different from the number of points in P_i; for example, if many points have been placed close to $\mathbf{y}_i$ in the visualization, then retrieving all points within a certain radius of $\mathbf{y}_i$ might yield too many retrieved points, compared to how many are neighbors in the input space. Figure 1 illustrates the setup.

The remaining question is what is a good visualization, that is, what is the cost function. Denote the number of samples that are in both Q_i and P_i by $N_{\mathrm{TP},i}$ (true positives), samples that are in Q_i but not in P_i by $N_{\mathrm{FP},i}$ (false positives), and samples that are in P_i but not Q_i by $N_{\mathrm{MISS},i}$ (misses). Assume the user has assigned a cost C_{FP} for each false positive and C_{MISS} for each miss. The total cost E_i for query i, summed over all data points, then is

$$E_i = N_{\mathrm{FP},i} C_{\mathrm{FP}} + N_{\mathrm{MISS},i} C_{\mathrm{MISS}} \,. \tag{1}$$

2.1.2 RELATIONSHIP TO PRECISION AND RECALL

The cost function of similarity visualization (1) bears a close relationship to the traditional measures of information retrieval, precision and recall. If we allow C_{MISS} to be a function of the total number of relevant points r, more specifically $C_{\mathrm{MISS}}(r_i) = C'_{\mathrm{MISS}}/r_i$, and take the cost per retrieved point by

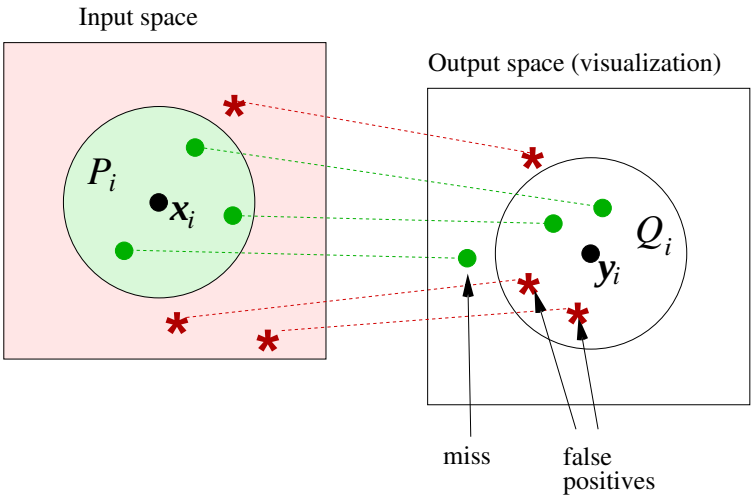

Figure 1: Diagram of the types of errors in visualization.

dividing by k_i, the total cost becomes

$$\begin{aligned} E(k_i, r_i) &= \frac{1}{k_i} E(r_i) = \frac{1}{k_i} \left(N_{\mathrm{FP},i} C_{\mathrm{FP}} + N_{\mathrm{MISS},i} C_{\mathrm{MISS}}(r_i) \right) \\ &= C_{\mathrm{FP}} \frac{N_{\mathrm{FP},i}}{k_i} + \frac{C'_{MISS}}{k_i} \frac{N_{\mathrm{MISS},i}}{r_i} \\ &= C_{\mathrm{FP}} (1 - \mathrm{precision}(i)) + \frac{C'_{MISS}}{k_i} (1 - \mathrm{recall}(i)) \, . \end{aligned}$$

The traditional definition of precision for a single query is

$$\mathrm{precision}(i) = \frac{N_{\mathrm{TP},i}}{k_i} = 1 - \frac{N_{\mathrm{FP},i}}{k_i} \, ,$$

and recall is

$$\mathrm{recall}(i) = \frac{N_{\mathrm{TP},i}}{r_i} = 1 - \frac{N_{\mathrm{MISS},i}}{r_i} \, .$$

Hence, fixing the costs C_{FP} and C_{MISS} and minimizing (1) corresponds to maximizing a specific weighted combination of precision and recall.

Finally, to assess performance of the full visualization the cost needs to be averaged over all samples (queries) which yields mean precision and recall of the visualization.

2.1.3 DISCUSSION

Given a high-dimensional data set, it is generally not possible to show all the similarity relationships within the data on a low-dimensional display; therefore, all linear or nonlinear dimensionality reduction methods need to make a tradeoff about which kinds of similarity relationships they aim to show on the display. Equation (1) fixes the tradeoff given the costs of the two kinds of errors. Figure 2 illustrates this tradeoff (computed with methods introduced in Section 3) with a toy example where a three-dimensional sphere surface is visualized in two dimensions. If we take some

query point in the visualization and retrieve a set of points close-by in the visualization, in display **A** such retrieval yields few false positives but many misses, whereas in display **B** the retrieval yields few misses but many false positives. The tradeoff can also be seen in the (mean) precision-recall curves for the two visualizations, where the number of retrieved points is varied to yield the curve. Visualization **A** reaches higher values of precision, but the precision drops much before high recall is reached. Visualization **B** has lower precision at the left end of the curve, but precision does not drop as much even when high recall is reached.

Note that in order to quantify the tradeoff, both precision and recall need to be used. This requires a rich enough retrieval model, in the sense that the number of retrieved points can be different from the number of relevant points, so that precision and recall get different values. It is well-known in information retrieval that if the numbers of relevant and retrieved items (here points) are equal, precision and recall become equal. The recent "local continuity" criterion (Equation 9 in Chen and Buja, 2009) is simply precision/recall under this constraint; we thus give a novel information retrieval interpretation of it as a side result. Such a criterion is useful but it gives only a limited view of the quality of visualizations, because it corresponds to a limited retrieval model and cannot fully quantify the precision-recall tradeoff. In this paper we will use fixed-radius neighborhoods (defined more precisely in Section 2.2) in the visualizations, which naturally yields differing numbers of retrieved and relevant points.

The simple visualization setup presented in this section is a novel formulation of visualization and useful as a clearly defined starting point. However, for practical use it has a shortcoming: the overly simple binary fixed-size neighborhoods do not take into account *grades of relevance*. The cost function does not penalize violating the original similarity ordering of neighbor samples; and the cost function penalizes all neighborhood violations with the same cost. Next we will introduce a more practical visualization setup.

2.2 Similarity Visualization with Continuous Neighborhood Relationships

We generalize the simple binary neighborhood case by defining probabilistic neighborhoods both in the (i) input and (ii) output spaces, and (iii) replacing the binary precision and recall measures with probabilistic ones. It will finally be shown that for binary neighborhoods, interpreted as a constant high probability of being a neighbor within the neighborhood set and a constant low probability elsewhere, the measures reduce to the standard precision and recall.

2.2.1 PROBABILISTIC MODEL OF RETRIEVAL

We start by defining the neighborhood in the output space, and do that by defining a probability distribution over the neighbor points. Such a distribution is interpretable as a model about how the user does the retrieval given the visualization display.

Given the location of the query point on the display, $\mathbf{y}_i$, suppose that the user selects one point at a time for inspection. Denote by $q_{j|i}$ the probability that the user chooses $\mathbf{y}_j$. If we can define such probabilities, they will define a *probabilistic model of retrieval* for the neighbors of $\mathbf{y}_i$.

The form of $q_{j|i}$ can be defined by a few axiomatic choices and a few arbitrary ones. Since the $q_{j|i}$ are a probability distribution over j for each i, they must be nonnegative and sum to one over j; therefore we can represent them as $q_{j|i} = \exp(-f_{i,j})/\sum_{k\neq i}\exp(-f_{i,k})$ where $f_{i,j} \in \mathbb{R}$. The $f_{i,j}$ should be an increasing function of distance (dissimilarity) between $\mathbf{y}_i$ and $\mathbf{y}_j$; we further assume that $f_{i,j}$ depends only on $\mathbf{y}_i$ and $\mathbf{y}_j$ and not on the other points $\mathbf{y}_k$. It remains to choose the form of

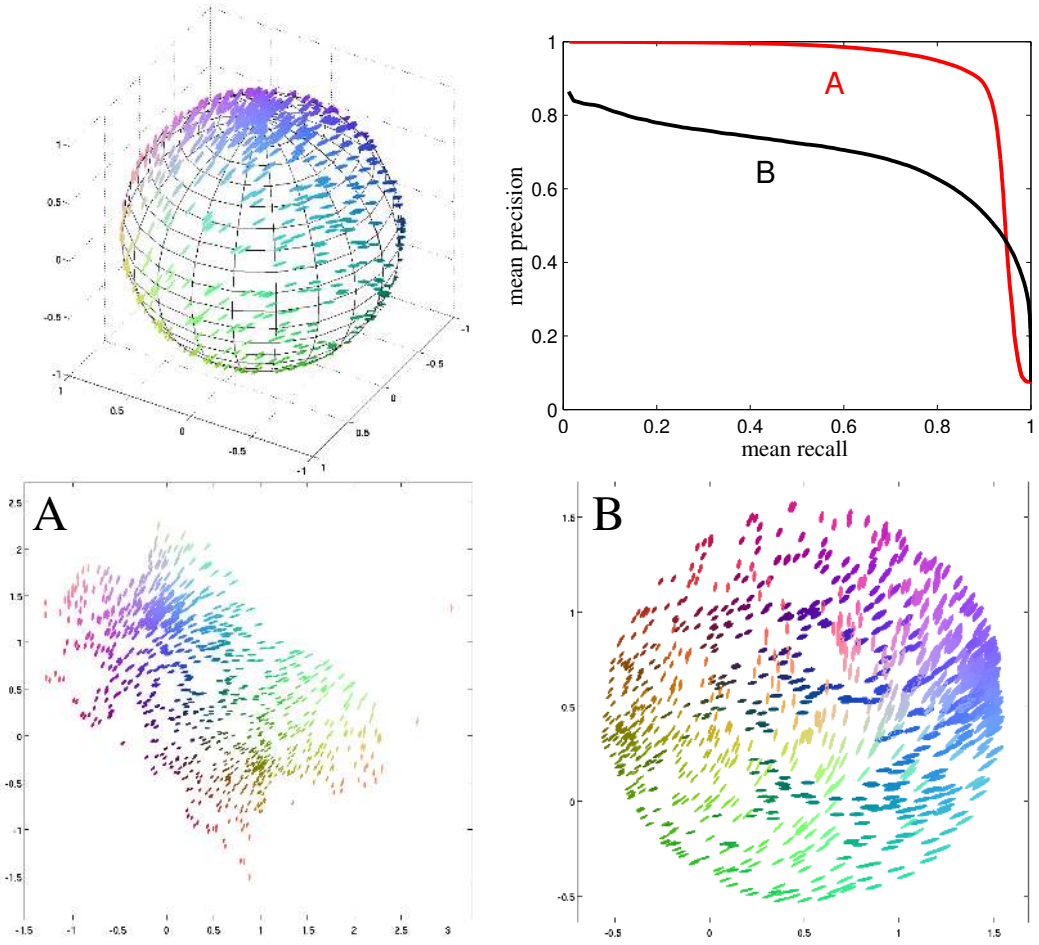

Figure 2: Demonstration of the tradeoff between false positives and misses. **Top left:** A three-dimensional data set sampled from the surface of a sphere; only the front hemisphere is shown for clarity. The glyph shapes (size, elongation, and angle) show the three-dimensional coordinates of each point; the colors in the online version show the same information. **Bottom:** Two embeddings of the data set. In the embedding **A**, the sphere has been cut open and folded out. This embedding eliminates *false positives*, but there are some *misses* because points on different sides of the tear end up far away from each other. In contrast, the embedding **B** minimizes the number of misses by simply squashing the sphere flat; this results in a large number of false positives because points on opposite sides of the sphere are mapped close to each other. **Top right:** mean precision-mean recall curves with input neighborhood size $r = 75$, as a function of the output neighborhood size k, for the two projections. The embedding **A** has better precision (yielding higher values at the left end of the curve) whereas the embedding **B** has better recall (yielding higher values at the right end of the curve).

$f_{i,j}$. In general there should not be any reason to favor any particular neighbor point, and hence the form should not depend on j. It could depend on i, however; we assume it has a simple quadratic form $f_{i,j} = ||\mathbf{y}_i - \mathbf{y}_j||^2/\sigma_i^2$ where $||\mathbf{y}_i - \mathbf{y}_j||$ is the Euclidean distance and the positive multiplier $1/\sigma_i^2$ allows the function to grow at an individual rate for each i. This yields the definition

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

2.2.2 PROBABILISTIC MODEL OF RELEVANCE

We extend the simple binary neighborhoods of input data samples to probabilistic neighborhoods as follows. Suppose that if the user was choosing the neighbors of a query point i in the original data space, she would choose point j with probability $p_{j|i}$. The $p_{j|i}$ define a *probabilistic model of relevance* for the original data, and are equivalent to a neighborhood around i: the higher the chance of choosing this neighbor, the larger its relevance to i.

We define the probability $p_{j|i}$ analogously to $q_{j|i}$, as

$$p_{j|i} = \frac{\exp\left(-\frac{d(\mathbf{x}_i, \mathbf{x}_j)^2}{\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(-\frac{d(\mathbf{x}_i, \mathbf{x}_k)^2}{\sigma_i^2}\right)} , \tag{3}$$

where $d(\cdot,\cdot)$ is a suitable difference measure in the original data, and $\mathbf{x}_i$ refers to the point in the original data that is represented by $\mathbf{y}_i$ in the visualization. Some data sets may provide the values of $d(\cdot,\cdot)$ directly; otherwise the analyst can choose a difference measure suitable for the data feature vectors. Later in this paper we will use both the simple Euclidean distance and a more complicated distance measure that incorporates additional information about the data.

Given known values of $d(\cdot,\cdot)$, the above definition of the neighborhood $p_{j|i}$ can be motivated by the same arguments as $q_{j|i}$. That is, the given form of $p_{j|i}$ is a good choice if no other information about the original neighborhoods is available. Other choices are possible too; in particular, if the data directly includes neighbor probabilities, they can simply be used as the $p_{j|i}$. Likewise, if more accurate models of user behavior are available, they can be plugged in place of $q_{j|i}$. The forms of $p_{j|i}$ and $q_{j|i}$ need not be the same.

For each point i, the scaling parameter σ_i controls how quickly the probabilities $p_{j|i}$ fall off with distance. These parameters could be fixed by prior knowledge, but without such knowledge it is reasonable to set the σ_i by specifying how much flexibility there should be about the choice of neighbors. That is, we set σ_i to a value that makes the entropy of the $p_{\cdot|i}$ distribution equal to $\log k$, where k is a rough upper limit for the number of relevant neighbors, set by the user. We use the same relative scale σ_i both in the input and output spaces (Equations 2 and 3).

2.2.3 COST FUNCTIONS

The remaining task is to measure how well the retrieval done in the output space, given the visualization, matches the true relevances defined in the input space. Both were above defined in terms of distributions, and a natural candidate for the measure is the *Kullback-Leibler divergence*, defined as

$$D(p_i, q_i) = \sum_{j \neq i} p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

where p_i and q_i are the neighbor distributions for a particular point i, in the input space and in the visualization respectively. For the particular probability distributions defined above the Kullback-Leibler divergence turns out to be intimately related to precision and recall. Specifically, for any query i, the Kullback-Leibler divergence $D(p_i, q_i)$ is a generalization of recall, and $D(q_i, p_i)$ is a generalization of precision; for simple "binary" neighborhood definitions, the Kullback-Leibler divergences and the precision-recall measures become equivalent. The proof is in Appendix A.

We call $D(q_i, p_i)$ *smoothed precision* and $D(p_i, q_i)$ *smoothed recall*. To evaluate a complete visualization rather than a single query, we define aggregate measures in the standard fashion: mean smoothed precision is defined as $\mathbb{E}_i[D(q_i, p_i)]$ and mean smoothed recall as $\mathbb{E}_i[D(p_i, q_i)]$, where $\mathbb{E}$ denotes expectation and the means are taken over queries (data points i).

Mean smoothed precision and recall are analogous to mean precision and recall in that we cannot in general reach the optimum of both simultaneously. We return to Figure 2 which illustrates the tradeoff for nonlinear projections of a three-dimensional sphere surface. The subfigure **A** was created by maximizing mean smoothed precision; the sphere has been cut open and folded out, which minimizes the number of false positives but also incurs some misses because some points located on opposite edges of the point cloud were originally close to each other on the sphere. The subfigure **B** was created by maximizing mean smoothed recall; the sphere is squashed flat, which minimizes the number of misses, as all the points that were close to each other in the original data are close to each other in the visualization. However, there are then a large number of false positives because opposite sides of the sphere have been mapped on top of each other, so that many points that appear close to each other in the visualization are actually originally far away from each other.

2.2.4 EASIER-TO-INTERPRET ALTERNATIVE GOODNESS MEASURES

Mean smoothed precision and recall are rigorous and well-motivated measures of visualization performance, but they have one practical shortcoming for human analysts: the errors have no upper bound, and the scale will tend to depend on the data set. The measures are very useful for comparing several visualizations of the same data, and will turn out to be useful as optimization criteria, but we would additionally like to have measures where the plain numbers are easily interpretable. We address this by introducing *mean rank-based smoothed precision and recall*: simply replace the distances in the definitions of $p_{j|i}$ and $q_{j|i}$ with ranks, so that the probability for the nearest neighbor uses a distance of 1, the probability for the second nearest neighbor a distance of 2, and so on. This imposes an upper bound on the error because the worst case scenario is that the ranks in the data set are reversed in the visualization. Dividing the errors by their upper bounds gives us measures that lie in the interval $[0, 1]$ regardless of the data and are thus much easier to interpret. The downside is that substituting ranks for distances makes the measures disregard much of the neighborhood structure in the data, so we suggest using mean rank-based smoothed precision and recall as easier-to-interpret, but less discriminating complements to, rather than replacements of, mean smoothed precision and recall.

3. Neighborhood Retrieval Visualizer (NeRV)

In Section 2 we defined similarity visualization as an information retrieval task. The quality of a visualization can be measured by the two loss functions, mean smoothed precision and recall. These measures generalize the straightforward precision and recall measures to non-binary neighborhoods. They have the further advantage of being continuous and differentiable functions of the

output visualization coordinates. It is then easy to use the measures as *optimization criteria* for a visualization method. We now introduce a visualization algorithm that optimizes visual information retrieval performance. We call the algorithm the *neighborhood retrieval visualizer* (NeRV).

As demonstrated in Figure 2, precision and recall cannot in general be minimized simultaneously, and the user has to choose which loss function (average smoothed precision or recall) is more important, by assigning a cost for misses and a cost for false positives. Once these costs have been assigned, the visualization task is simply to minimize the total cost. In practice the relative cost of false positives to misses is given as a parameter λ. The NeRV cost function then becomes

$$E_{\mathrm{NeRV}} = \lambda \mathbb{E}_i[D(p_i, q_i)] + (1-\lambda)\mathbb{E}_i[D(q_i, p_i)] \propto \lambda \sum_i \sum_{j \neq i} p_{j|i} \log \frac{p_{j|i}}{q_{j|i}} + (1-\lambda) \sum_i \sum_{j \neq i} q_{j|i} \log \frac{q_{j|i}}{p_{j|i}} \quad (4)$$

where, for example, setting λ to 0.1 indicates that the user considers an error in precision $(1-0.1)/0.1 = 9$ times as expensive as a similar error in recall.

To optimize the cost function (4) with respect to the output coordinates $\mathbf{y}_i$ of each data point, we use a standard conjugate gradient algorithm. The computational complexity of each iteration is $O(dn^2)$, where n is the number of data points and d the dimension of the projection. (In our earlier conference paper a coarse approximate algorithm was required for speed; this turned out to be unnecessary, and the $O(dn^2)$ complexity does not require any approximation.) Note that if a pairwise distance matrix in the input space is not directly provided as data, it can as usual be computed from input features; this is a one-time computation done at the start of the algorithm and takes $O(Dn^2)$ time, where D is the input dimensionality.

In general, NeRV optimizes a user-defined cost which forms a tradeoff between mean smoothed precision and mean smoothed recall. If we set $\lambda = 1$ in Equation (4), we obtain the cost function of stochastic neighbor embedding (SNE; see Hinton and Roweis, 2002). Hence we get as a side result a new interpretation of SNE as a method that maximizes mean smoothed recall.

3.0.5 PRACTICAL ADVICE ON OPTIMIZATION

After computing the distance matrix from the input data, we scale the input distances so that the average distance is equal to 1. We use a random projection onto the unit square as a starting point for the algorithm. Even this simple choice has turned out to give better results than alternatives; a more intelligent initialization, such as projecting the data using principal component analysis, can of course also be used.

To speed up convergence and avoid local minima, we apply a further initialization step: we run ten rounds of conjugate gradient (two conjugate gradient steps per round), and after each round decrease the neighborhood scaling parameters σ_i used in Equations (2) and (3). Initially, we set the σ_i to half the diameter of the input data. We decrease them linearly so that the final value makes the entropy of the $p_{j|i}$ distribution equal to an effective number of neighbors k, which is the choice recommended in Section 2.2. This initialization step has the same complexity $O(dn^2)$ per iteration as the rest of the algorithm. After this initialization phase we perform twenty standard conjugate gradient steps.

4. Using NeRV for Unsupervised Visualization

It is easy to apply NeRV for unsupervised dimensionality reduction. As in any unsupervised analysis, the analyst first chooses a suitable unsupervised similarity or distance measure for the input data; for vector-valued input data this can be the standard Euclidean distance (which we will use here), or it can be some other measure suggested by domain knowledge. Once the analyst has specified the relative importance of precision and recall by choosing a value for λ, the NeRV algorithm computes the embedding based on the distances it is given.

In this section we will make extensive experiments comparing the performance of NeRV with other dimensionality reduction methods on unsupervised visualization of several data sets, including both benchmark data sets and real-life bioinformatics data sets. In the following subsections, we describe the comparison methods and data sets, briefly discuss the experimental methodology, and present the results.

4.1 Comparison Methods for Unsupervised Visualization

For the task of unsupervised visualization we compare the performance of NeRV with the following unsupervised nonlinear dimensionality reduction methods: principal component analysis (PCA; Hotelling, 1933), metric multidimensional scaling (MDS; see Borg and Groenen, 1997), locally linear embedding (LLE; Roweis and Saul, 2000), Laplacian eigenmap (LE; Belkin and Niyogi, 2002a), Hessian-based locally linear embedding (HLLE; Donoho and Grimes, 2003), isomap (Tenenbaum et al., 2000), curvilinear component analysis (CCA; Demartines and Hérault, 1997), curvilinear distance analysis (CDA; Lee et al., 2004), maximum variance unfolding (MVU; Weinberger and Saul, 2006), landmark maximum variance unfolding (LMVU; Weinberger et al., 2005), and our previous method local MDS (LMDS; Venna and Kaski, 2006).

Principal component analysis (PCA; Hotelling, 1933) finds linear projections that maximally preserve the variance in the data. More technically, the projection directions can be found by solving for the eigenvalues and eigenvectors of the covariance matrix $\mathbf{C}_x$ of the input data points. The eigenvectors corresponding to the two or three largest eigenvalues are collected into a matrix $\mathbf{A}$, and the data points $\mathbf{x}_i$ can then be visualized by projecting them with $\mathbf{y}_i = \mathbf{A}\mathbf{x}_i$, where $\mathbf{y}_i$ is the obtained low-dimensional representation of $\mathbf{x}_i$. PCA is very closely related to linear multidimensional scaling (linear MDS, also called classical scaling; Torgerson, 1952; Gower, 1966), which tries to find low-dimensional coordinates preserving squared distances. It can be shown (Gower, 1966) that when the dimensionality of the sought solutions is the same and the distance measure is Euclidean, the projection of the original data to the PCA subspace equals the configuration of points found by linear MDS. This implies that PCA tries to preserve the squared distances between data points, and that linear MDS finds a solution that is a linear projection of the original data.

Traditional *multidimensional scaling* (MDS; see Borg and Groenen, 1997) exists in several different variants, but they all have a common goal: to find a configuration of output coordinates that preserves the pairwise distance matrix of the input data. For the comparison experiments we chose *metric MDS* which is the simplest nonlinear MDS method; its cost function (Kruskal, 1964), called the raw stress, is

$$E = \sum_{i,j} (d(\mathbf{x}_i, \mathbf{x}_j) - d(\mathbf{y}_i, \mathbf{y}_j))^2, \tag{5}$$

where $d(\mathbf{x}_i, \mathbf{x}_j)$ is the distance of points $\mathbf{x}_i$ and $\mathbf{x}_j$ in the input space and $d(\mathbf{y}_i, \mathbf{y}_j)$ is the distance of their corresponding representations (locations) $\mathbf{y}_i$ and $\mathbf{y}_j$ in the output space. This cost function is minimized with respect to the representations $\mathbf{y}_i$.

Isomap (Tenenbaum et al., 2000) is an interesting variant of MDS, which again finds a configuration of output coordinates matching a given distance matrix. The difference is that Isomap does not compute pairwise input-space distances as simple Euclidean distances but as *geodesic distances* along the manifold of the data (technically, along a graph formed by connecting all k-nearest neighbors). Given these geodesic distances the output coordinates are found by standard linear MDS. When output coordinates are found for such input distances, the manifold structure in the original data becomes unfolded; it has been shown (Bernstein et al., 2000) that this algorithm is asymptotically able to recover certain types of manifolds. We used the isomap implementation available at `http://isomap.stanford.edu` in the experiments.

Curvilinear component analysis (CCA; Demartines and Hérault, 1997) is a variant of MDS that tries to preserve only distances between points that are near each other in the visualization. This is achieved by weighting each term in the MDS cost function (5) by a coefficient that depends on the corresponding pairwise distance in the visualization. In the implementation we use, the coefficient is simply a step function that equals 1 if the distance is below a predetermined threshold and 0 if it is larger.

Curvilinear distance analysis (CDA; Lee et al., 2000, 2004) is an extension of CCA. The idea is to replace the Euclidean distances in the original space with geodesic distances in the same manner as in the isomap algorithm. Otherwise the algorithm stays the same.

Local MDS (LMDS; Venna and Kaski, 2006) is our earlier method, an extension of CCA that focuses on local proximities with a tunable cost function tradeoff. It can be seen as a first step in the development of the ideas of NeRV.

The *locally linear embedding* (LLE; Roweis and Saul, 2000) algorithm is based on the assumption that the data manifold is smooth enough and is sampled densely enough, such that each data point lies close to a locally linear subspace on the manifold. LLE makes a locally linear approximation of the whole data manifold: LLE first estimates a local coordinate system for each data point, by calculating linear coefficients that reconstruct the data point as well as possible from its k nearest neighbors. To unfold the manifold, LLE finds low-dimensional coordinates that preserve the previously estimated local coordinate systems as well as possible. Technically, LLE first minimizes the reconstruction error $E(\mathbf{W}) = \sum_i \|\mathbf{x}_i - \sum_j W_{i,j}\mathbf{x}_j\|^2$ with respect to the coefficients $W_{i,j}$, under the constraints that $W_{i,j} = 0$ if i and j are not neighbors, and $\sum_j W_{i,j} = 1$. Given the weights, the low-dimensional configuration of points is next found by minimizing $E(\mathbf{Y}) = \sum_i \|\mathbf{y}_i - \sum_j W_{i,j}\mathbf{y}_j\|^2$ with respect to the low-dimensional representation $\mathbf{y}_i$ of each data point.

The *Laplacian eigenmap* (LE; see Belkin and Niyogi, 2002a) uses a graph embedding approach. An undirected k-nearest-neighbor graph is formed, where each data point is a vertex. Points i and j are connected by an edge with weight $W_{i,j} = 1$ if j is among the k nearest neighbors of i, otherwise the edge weight is set to zero; this simple weighting method has been found to work well in practice (Belkin and Niyogi, 2002b). To find a low-dimensional embedding of the graph, the algorithm tries to put points that are connected in the graph as close to each other as possible and does not care what happens to the other points. Technically, it minimizes $\frac{1}{2}\sum_{i,j} \|\mathbf{y}_i - \mathbf{y}_j\|^2 W_{i,j} = \mathbf{y}^T\mathbf{L}\mathbf{y}$ with respect to the low-dimensional point locations $\mathbf{y}_i$, where $\mathbf{L} = \mathbf{D} - \mathbf{W}$ is the graph Laplacian and $\mathbf{D}$ is a diagonal matrix with elements $\mathrm{D}_{ii} = \sum_j W_{i,j}$. However, this cost function has an undesirable trivial solution: putting all points in the same position would minimize the cost. This can be avoided by adding suit-

able constraints. In practice the low-dimensional configuration is found by solving the generalized eigenvalue problem $\mathbf{Ly} = \lambda\mathbf{Dy}$ (Belkin and Niyogi, 2002a). The smallest eigenvalue corresponds to the trivial solution, but the eigenvectors corresponding to the next smallest eigenvalues give the Laplacian eigenmap solution.

The Laplacian eigenmap algorithm reduces to solving a generalized eigenvalue problem because the cost function that is minimized is a quadratic form involving the Laplacian matrix $\mathbf{L}$. The *Hessian-based locally linear embedding* (HLLE; Donoho and Grimes, 2003) algorithm is similar, but the Laplacian $\mathbf{L}$ is replaced by the Hessian $\mathbf{H}$.

The *maximum variance unfolding* algorithm (MVU; Weinberger and Saul, 2006) expresses dimensionality reduction as a semidefinite programming problem. One way of unfolding a folded flag is to pull its four corners apart, but not so hard as to tear the flag. MVU applies this idea to projecting a manifold: the projection maximizes variance (pulling apart) while preserving distances between neighbors (no tears). The constraint of local distance preservation can be expressed in terms of the Gram matrix $\mathbf{K}$ of the mapping. Maximizing the variance of the mapping is equivalent to maximizing the trace of $\mathbf{K}$ under a set of constraints, which, it turns out, can be done using semidefinite programming.

A notable disadvantage of MVU is the time required to solve a semidefinite program for $n \times n$ matrices when the number of data points n is large. *Landmark MVU* (LMVU; Weinberger et al., 2005) addresses this issue by significantly reducing the size of the semidefinite programming problem. Like LLE, LMVU assumes that the data manifold is sufficiently smooth and densely sampled that it is locally approximately linear. Instead of embedding all the data points directly as MVU does, LMVU randomly chooses $m \ll n$ inputs as so-called landmarks. Because of the local linearity assumption, the other data points can be approximately reconstructed from the landmarks using a linear transformation. It follows that the Gram matrix $\mathbf{K}$ can be approximated using the $m \times m$ submatrix of inner products between landmarks. Hence we only need to optimize over $m \times m$ matrices, a much smaller semidefinite program. Other recent approaches for speeding up MVU include matrix factorization based on a graph Laplacian (Weinberger et al., 2007).

In addition to the above comparison methods, other recent work on dimensionality reduction includes *minimum volume embedding* (MVE; Shaw and Jebara, 2007), which is similar to MVU, but where MVU maximizes the whole trace of the Gram matrix (the sum of all eigenvalues), MVE maximizes the sum of the first few eigenvalues and minimizes the sum of the rest, in order to preserve the largest amount of eigenspectrum energy in the few dimensions that remain after dimensionality reduction. In practice, a variational upper bound of the resulting criterion is optimized.

Very recently, a number of unsupervised methods have been compared by van der Maaten et al. (2009) in terms of classification accuracy and our old criteria trustworthiness-continuity.

4.2 Data Sets for Unsupervised Visualization

We used two synthetic benchmark data sets and four real-life data sets for our experiments.

The *plain s-curve* data set is an artificial set sampled from an S-shaped two-dimensional surface embedded in three-dimensional space. An almost perfect two-dimensional representation should be possible for a non-linear dimensionality reduction method, so this data set works as a sanity check.

The *noisy s-curve* data set is otherwise identical to the plain s-curve data set, but significant spherical normally distributed noise has been added to each data point. The result is a cloud of points where the original S-shape is difficult to discern by visual inspection.

The *faces* data set consists of ten different face images of 40 different people, for a total of 400 images. For a given subject, the images vary in terms of lighting and facial expressions. The size of each image is 64×64 pixels, with 256 grey levels per pixel. The data set is available for download at `http://www.cs.toronto.edu/~roweis/data.html`.

The *mouse gene expression* data set is a collection of gene expression profiles from different mouse tissues (Su et al., 2002). 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 by Su et al. (2002), 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.

The *gene expression compendium* data set is a large collection of human gene expression arrays (`http://dags.stanford.edu/cancer`; Segal et al., 2004). 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.

The *sea-water temperature time series* data set (Liitiäinen and Lendasse, 2007) is a time series of weekly temperature measurements of sea water over several years. Each data point is a time window of 52 weeks, which is shifted one week forward for the next data point. Altogether there are 823 data points and 52 dimensions.

4.3 Methodology for the Unsupervised Experiments

The performance of NeRV was compared with 11 unsupervised dimensionality reduction methods described in Section 4.1, namely principal component analysis (PCA), metric multidimensional scaling (here simply denoted MDS), locally linear embedding (LLE), Laplacian eigenmap (LE), Hessian-based locally linear embedding (HLLE), isomap, curvilinear component analysis (CCA), curvilinear distance analysis (CDA), maximum variance unfolding (MVU), landmark maximum variance unfolding (LMVU), and local MDS (LMDS). LLE, LE, HLLE, MVU, LMVU and isomap were computed with code from their developers; MDS, CCA and CDA used our code.

4.3.1 GOODNESS MEASURES

We used four pairs of performance measures to compare the methods. The first pair is *mean smoothed precision-mean smoothed recall*, that is, our new measures of visualization quality. The scale of input neighborhoods was fixed to 20 relevant neighbors (see Section 2.2).

Although we feel, as explained in Section 2, that smoothed precision and smoothed recall are more sophisticated measures of visualization performance than precision and recall, we have also plotted standard *mean precision-mean recall* curves. The curves were plotted by fixing the 20 nearest neighbors of a point in the original data as the set of relevant items, and then varying the number of neighbors retrieved from the visualization between 1 and 100, plotting mean precision and recall for each number.

Our third pair of measures are the rank-based variants of our new measures, *mean rank-based smoothed precision-mean rank-based smoothed recall*. Recall that we introduced the rank-based

variants as easier-to-interpret, but less discriminating, alternatives to mean smoothed precision and mean smoothed recall. The scale of input neighborhoods was again fixed to 20 relevant neighbors.

Our fourth pair of measures is *trustworthiness-continuity* (Kaski et al., 2003). The intuitive motivation behind these measures was the same trade-off between precision and recall as in this paper, but the measures were defined in a more ad hoc way. At the time we did not have the clear connection to information retrieval which makes NeRV particularly attractive, and we did not optimize the measures. Trustworthiness and continuity can, however, now be used as partly independent measures of visualization quality. To compute the trustworthiness and continuity, we used neighborhoods of each point containing the 20 nearest neighbors.

As a fifth measure, when data classes are available, we use *classification error* given the display, with a standard k-nearest neighbor classifier where we set $k = 5$.

4.3.2 CHOICE OF PARAMETERS

Whenever we needed to choose a parameter for any method, we used the same criterion, namely the F-measure computed from the new rank-based measures. That is, we chose the parameter yielding the largest value of $2(P \cdot R)/(P+R)$ where P and R are the mean rank-based smoothed precision and recall.

Many of the methods have a parameter k denoting the number of nearest neighbors for constructing a neighborhood graph; for each method and each data set we tested values of k ranging from 4 to 20, and chose the value that produced the best F-measure. (For MVU and LMVU we used a smaller parameter range to save computational time. For MVU k ranged from 4 to 6; for LMVU k ranged from 3 to 9.) The exceptions are local MDS (LMDS), one of our own earlier methods, and NeRV, for which we simply set k to 20 without optimizing it.

Methods that may have local optima were run five times with different random initializations and the best run (again, in terms of the F-measure) was selected.

4.4 Results of Unsupervised Visualization

We will next show visualizations for a few sets, and measure quantitatively the results of several. We begin by showing an example of a NeRV visualization for the plain S-curve data set in Figure 3. Later in this section we will show a NeRV visualization of a synthetic face data set (Figure 8), and in Section 4.6 of the *faces* data set of real face images (Figure 11). The quantitative results are spread across four figures (Figures 4–7), each of which contains results for one pair of measures and all six data sets.

We first show the curves of *mean smoothed precision-mean smoothed recall*, that is, the loss functions associated with our formalization of visualization as information retrieval. The results are shown in Figure 4. NeRV and local MDS (LMDS) form curves parameterized by λ, which ranges from 0 to 1.0 for NeRV and from 0 to 0.9 for LMDS. NeRV was clearly the best-performing method on all six data sets, which is of course to be expected since NeRV directly optimizes a linear combination of these measures. LMDS has a relatively good mean smoothed precision, but does not perform as well in terms of mean smoothed recall. Simple metric MDS also stands out as a consistently reasonably good method.

Because we formulated visualization as an information retrieval task, it is natural to also try existing measures of information retrieval performance, that is, mean precision and mean recall, even though they do not take into account grades of relevance as discussed in Section 2.1. Standard

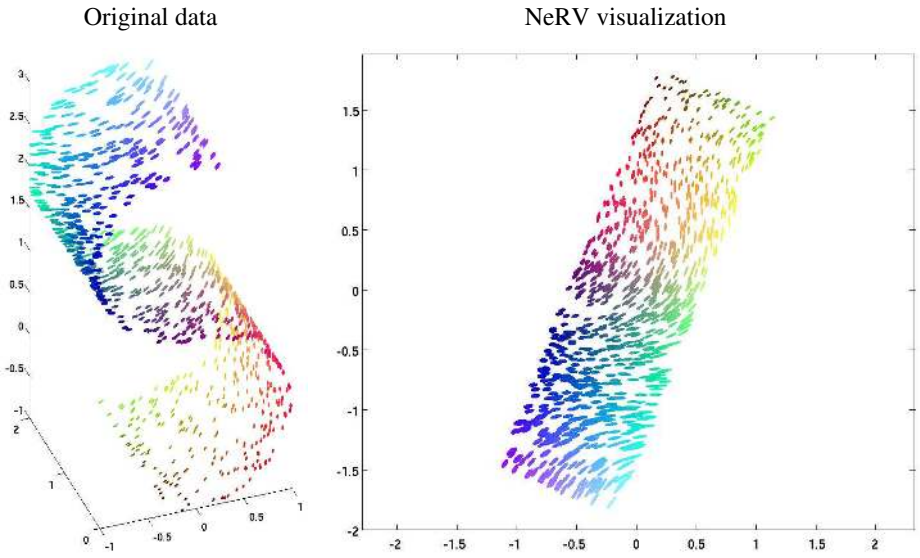

Figure 3: Left: Plain S-curve data set. The glyph shapes (size, elongation, and angle) show the three-dimensional coordinates of each point; the colors in the online version show the same information. Right: NeRV visualization (here $\lambda = 0.8$).

mean precision-mean recall curves are shown in Figure 5; for NeRV and LMDS, we show the curve for a single λ value picked by the F-measure as described in Section 4.3. Even with these coarse measures, NeRV shows excellent performance: NeRV is best on four data sets in terms of the area under the curve, CDA and CCA are each best on one data set.

Next, we plot our easier-to-interpret but less discriminating alternative measures of visualization performance. The curves of *mean rank-based smoothed precision-mean rank-based smoothed recall* are shown in Figure 6. These measures lie between 0 and 1, and may hence be easier to compare between data sets. With these measures, NeRV again performs best on all data sets; LMDS also performs well, especially on the seawater temperature data.

Finally, we plot the curves of *trustworthiness-continuity*, shown in Figure 7. The results are fairly similar to the new rank-based measures: once again NeRV performs best on all data sets and LMDS also performs well, especially on the seawater temperature data.

4.4.1 EXPERIMENT WITH A KNOWN UNDERLYING MANIFOLD

To further test how well the methods are able to recover the neighborhood structure inherent in the data we studied a synthetic face data set where a known underlying manifold defines the relevant items (neighbors) of each point. The SculptFaces data contains 698 synthetic images of a face (sized 64×64 pixels each). The pose and direction of lighting have been changed in a systematic way to create a manifold in the image space (`http://web.mit.edu/cocosci/isomap/datasets.html`; Tenenbaum et al., 2000). We used the raw pixel data as input features.

The pose and lighting parameters used to generate the images are available. These parameters define the manifold of the faces embedded in the very high-dimensional image space. For any face

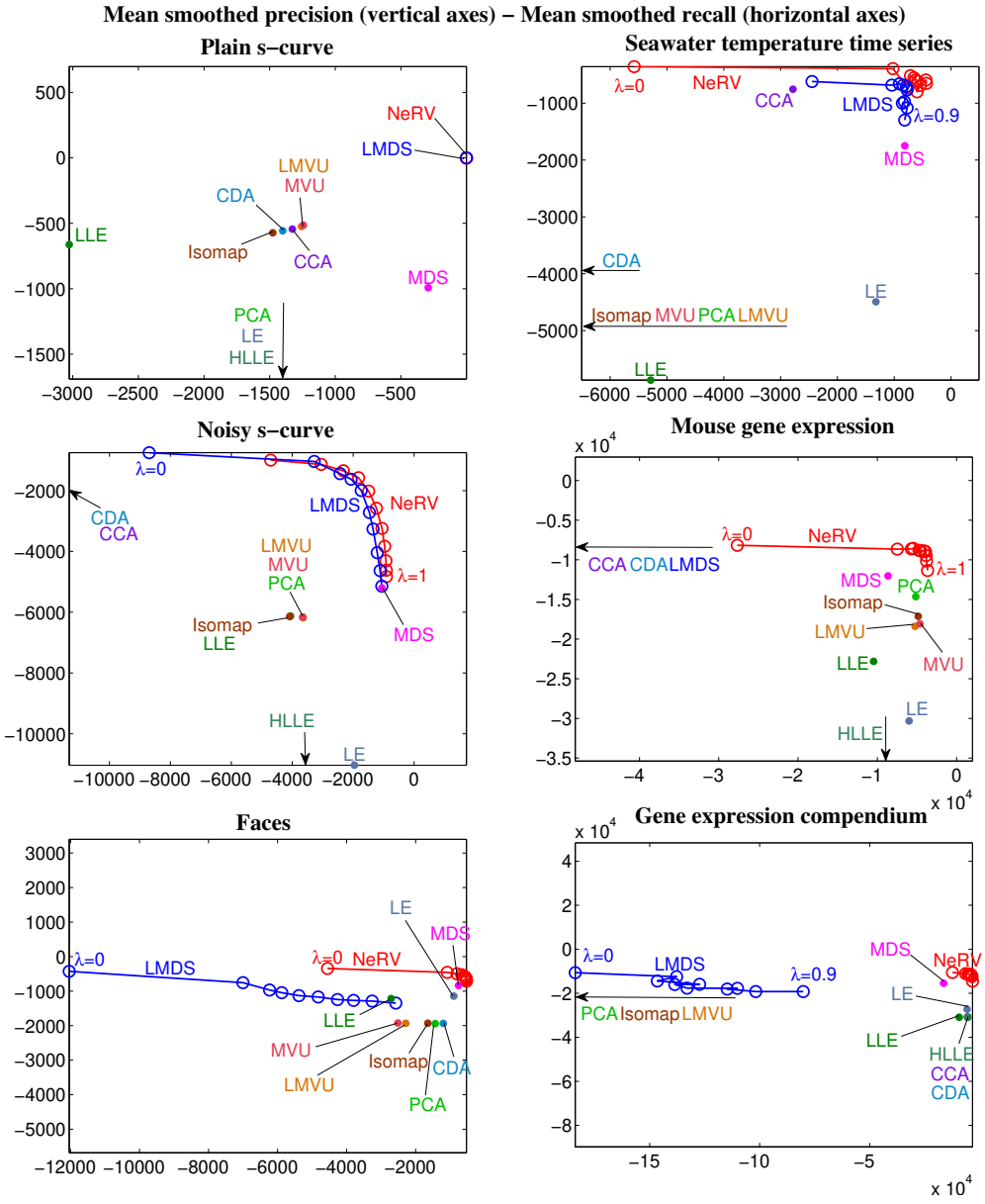

Figure 4: *Mean smoothed precision-mean smoothed recall* plotted for all six data sets. For clarity, only a few of the best-performing methods are shown for each data set. We have actually plotted $-1\cdot$(mean smoothed precision) and $-1\cdot$(mean smoothed recall) to maintain visual consistency with the plots for other measures: in each plot, the best performing methods appear in the top right corner.

image, the relevant other faces are the ones that are neighbors with respect to the pose and lighting parameters; we defined the ground truth neighborhoods using Euclidean distances in the pose and lighting space, and we fixed the scale of the ground truth neighborhoods to 20 relevant neighbors (see Section 2.2).

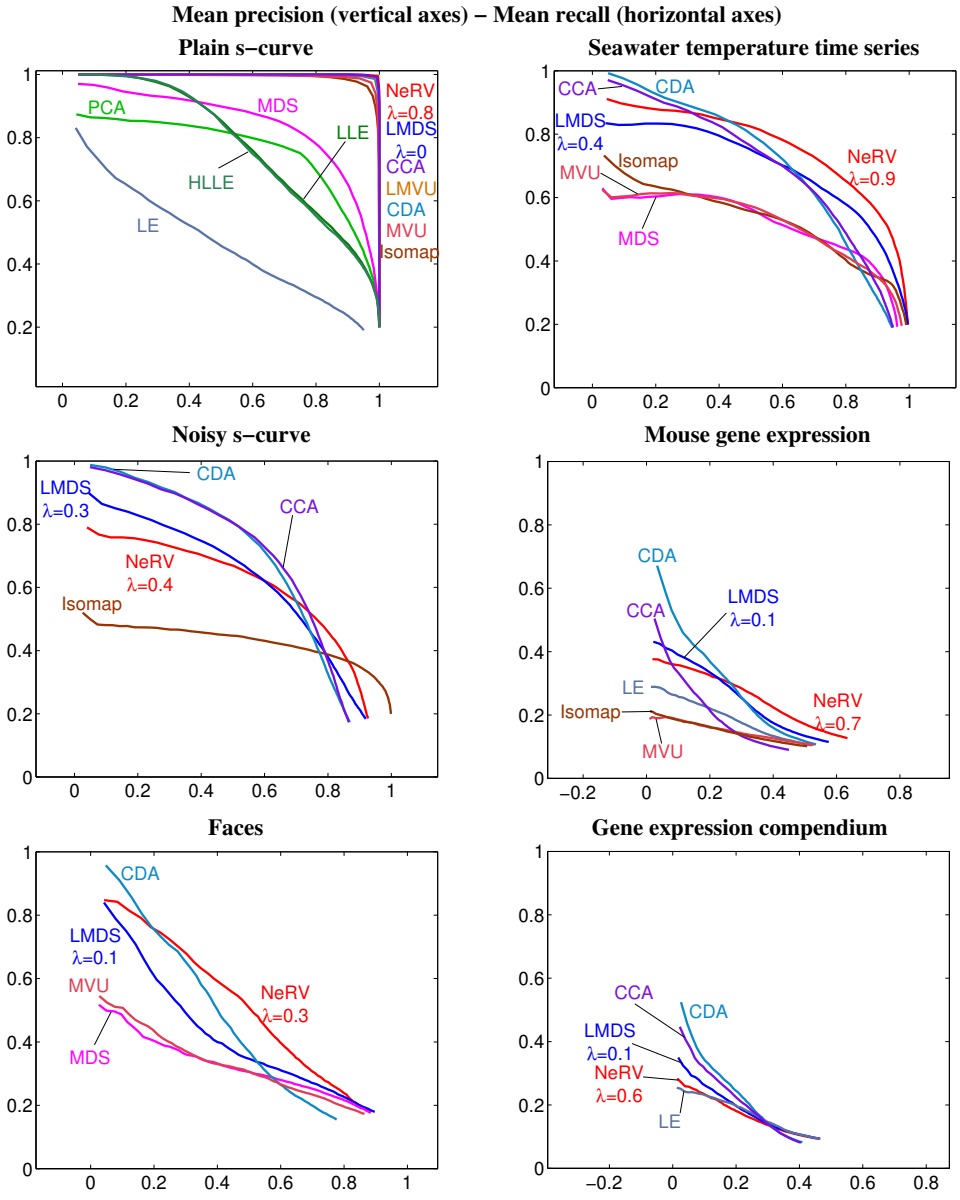

Figure 5: *Mean precision-mean recall* curves plotted for all six data sets. For clarity, only the best methods (with largest area under curve) are shown for each data set. In each plot, the best performance is in the top right corner. For NeRV and LMDS, a single λ value picked with the F-measure is shown.

We ran all methods for this data as in all experiments in this section, and then calculated four performance curves (*mean smoothed precision-mean smoothed recall*, *mean precision-mean recall*, *mean rank-based smoothed precision-mean rank-based smoothed recall*, and *trustworthiness-continuity*) using the neighborhoods in pose and lighting space as the ground truth. The results are shown in Figure 8. In spite of the very high dimensionality of the input space and the reduction of

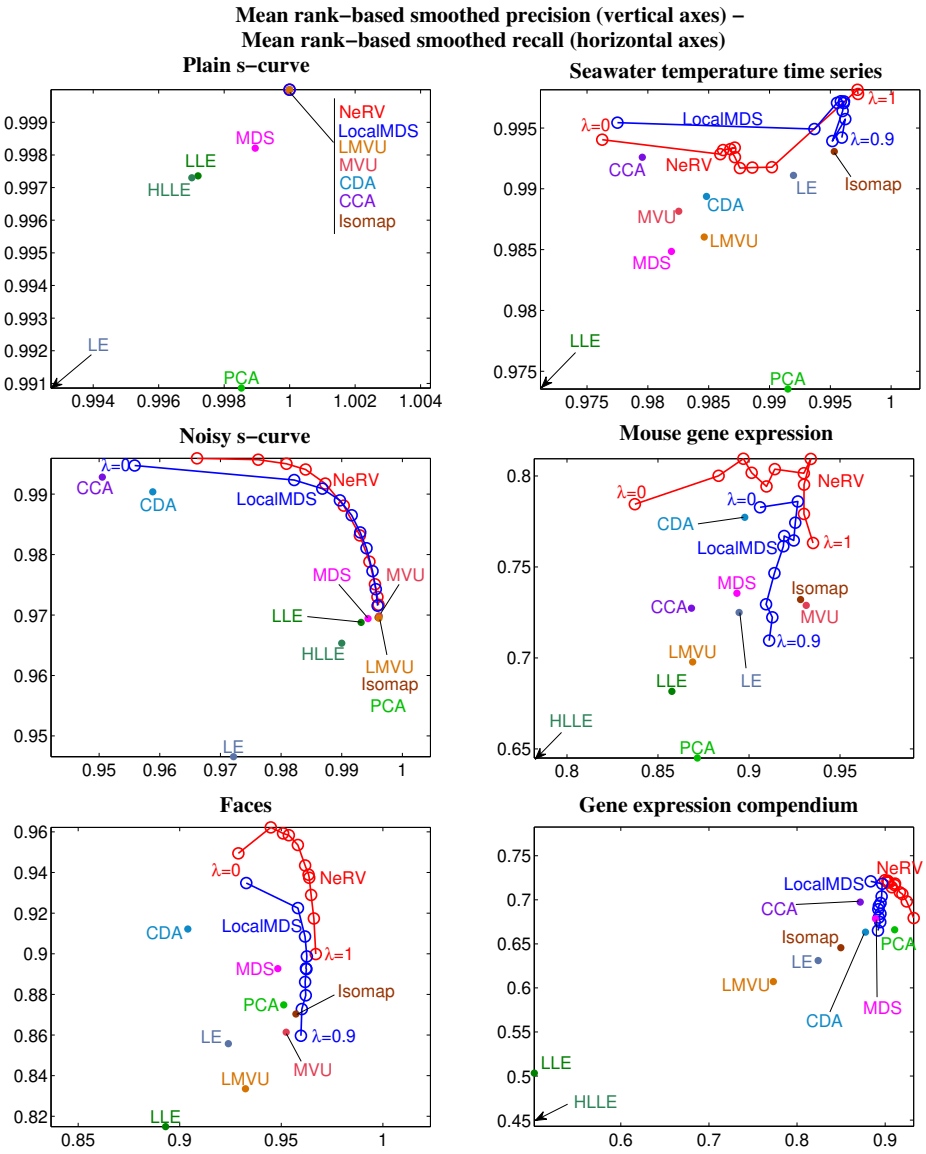

Figure 6: *Mean rank-based smoothed precision-mean rank-based smoothed recall* plotted for all six data sets. For clarity, only a few of the best performing methods are shown for each data set. We have actually plotted 1−(mean rank-based smoothed precision) and 1−(mean rank-based smoothed recall) to maintain visual consistency with the plots for other measures: in each plot, the best performance is in the top right corner.

the manifold dimension from three to two, NeRV was able to recover the structure well. NeRV is the best according to both of our proposed measures of visualization performance, mean smoothed precision and recall; MDS and local MDS also perform well. In terms of the simple mean precision and mean recall NeRV is the second best with CDA being slightly better. In terms of the rank-based measures, NeRV is the best in terms of precision; LE and MDS attain the best mean rank-based

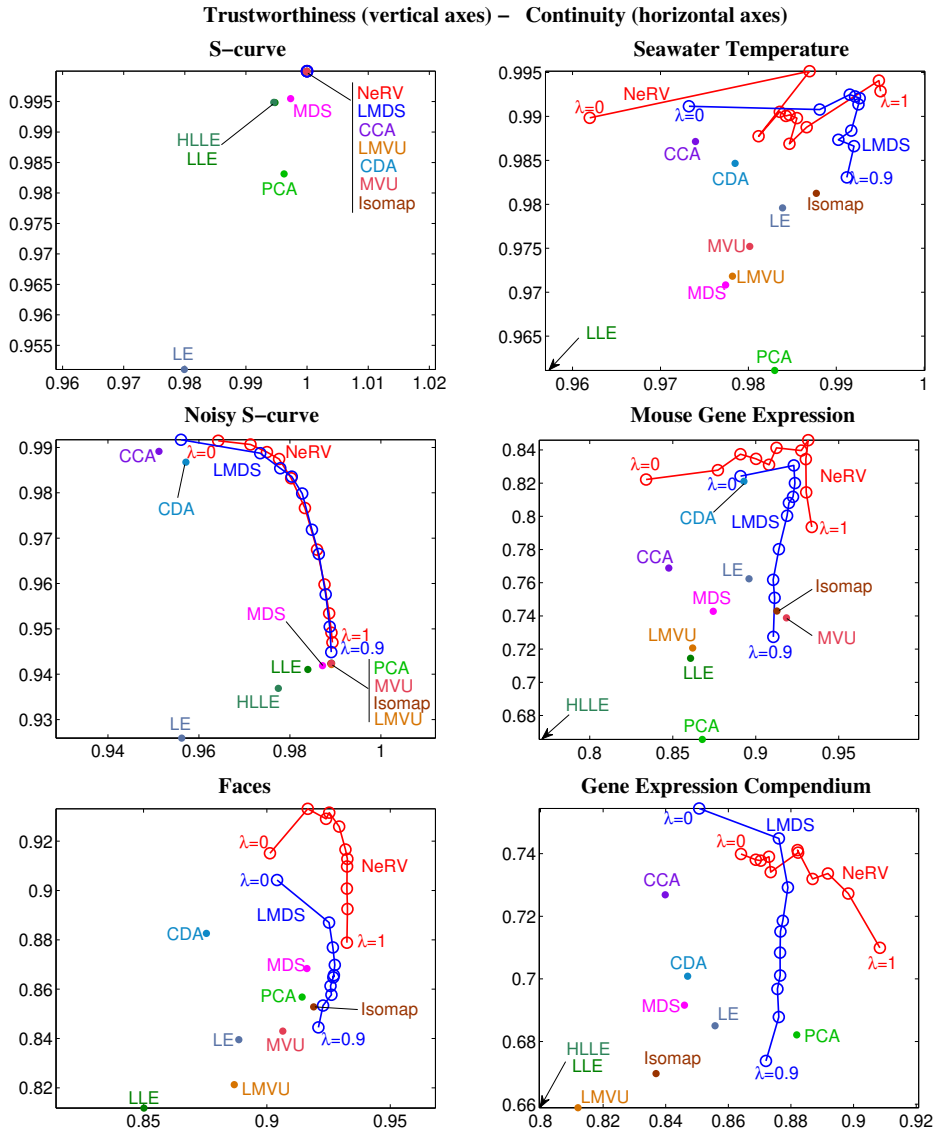

Figure 7: *Trustworthiness-continuity* plotted for all six data sets. For clarity, only a few of the best performing methods are shown for each data set. In each plot, the best performance is in the top right corner.

smoothed recall; and local MDS and CDA also perform well. When performance was measured with trustworthiness and continuity, NeRV was the best in terms of trustworthiness while MVU and Isomap attained the highest continuity.

Overall, NeRV was the best in these unsupervised visualization tasks, although it was not the best in all, and in some tasks it had tough competition.

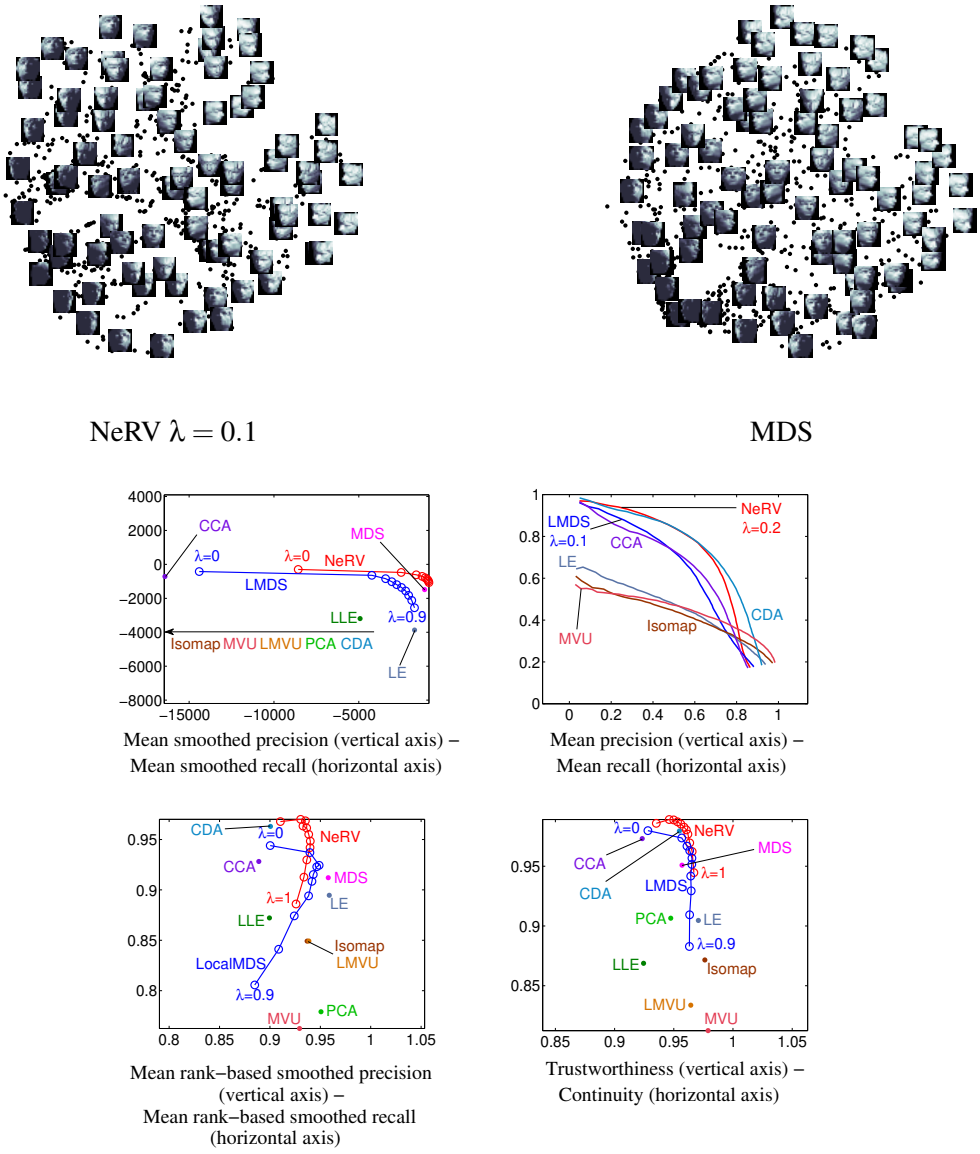

Figure 8: **Top:** Sample projections of the SculptFaces data set (NeRV vs. the best alternative). **Bottom:** How well were the ground truth neighbors in pose-lighting space retrieved from the image data, evaluated by four pairs of measures. The measures were computed the same way as before, as described in Section 4.3, but here taking the known pose and lighting information as the input data. Only the best performing methods are shown for clarity.

4.5 Comparison by Unsupervised Classification

For the data sets where a classification of samples is available, we additionally follow a traditional way to evaluate visualizations: we measure how well samples can be classified based on the visualization.

Data set	Dimensions	Classes
Letter	16	26
Phoneme	20	13
Landsat	36	6
TIMIT	12	41

Table 1: The data sets used in the unsupervised classification experiments.

Here all methods are unsupervised, that is, class labels of samples are not used in computing the visualization. The parameters of methods are again chosen as described in Section 4.3.2. Methods are evaluated by k-nearest neighbor classification accuracy (with $k = 5$), that is, each sample in the visualization is classified by majority vote of its k nearest neighbors in the visualization, and the classification is compared to the ground truth label.

We use four benchmark data sets, all of which include class labels, to compare the performances of the methods. The data sets are summarized in Table 1. For all data sets we used a randomly chosen subset of 1500 samples in the experiments, to save computation time.

The *letter recognition* data set (denoted Letter) is from the UCI Machine Learning Repository (Blake and Merz, 1998); it is a 16-dimensional data set with 26 classes, which are 4×4 images of the 26 capital letters of the alphabet. These letters are based on 20 different fonts which have been distorted to produce the final images.

The *phoneme* data set (denoted Phoneme) is taken from LVQ-PAK (Kohonen et al., 1996) and consists of phoneme samples represented by a 20-dimensional vector of features plus a class label indicating which phoneme is actually represented. There are a total of 13 classes.

The *landsat satellite* data set (denoted Landsat) is from UCI Machine Learning Repository (Blake and Merz, 1998). Each data point is a 36-dimensional vector, corresponding to a 3×3 satellite image measured in four spectral bands; the class label of the point indicates the terrain type in the image (6 possibilitities, for example red soil).

The *TIMIT* data set is taken from the DARPA TIMIT speech database (TIMIT). It is similar to the phoneme data from LVQ-PAK but the feature vectors are 12-dimensional and there are 41 classes in total.

The resulting classification error rates are shown in Table 2. NeRV is best on two out of four data sets and second best on a third set (there our old method LocalMDS is best). CDA is best on one.

4.6 NeRV, Joint Probabilities, and t-Distributions

Very recently, based on stochastic neighbor embedding (SNE), van der Maaten and Hinton (2008) have proposed a modified method called t-SNE, which has performed well in unsupervised experiments. The t-SNE makes two changes compared to SNE; in this section we describe the changes and show that the same changes can be made to NeRV, yielding a variant that we call t-NeRV. We then provide a new information retrieval interpretation for t-NeRV and t-SNE.

We start by analyzing the differences between t-SNE and the original stochastic neighbor embedding. The original SNE minimizes the sum of Kullback-Leibler divergences

$$\sum_i \sum_{j \neq i} p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

	Letter	Phoneme	Landsat	TIMIT
Eigenmap	0.914	0.121	0.168	0.674
LLE	n/a	0.118	0.212	0.722
Isomap	0.847	0.134	0.156	0.721
MVU	0.763	0.155	0.153	0.699
LMVU	0.819	0.208	0.151	0.787
MDS	0.823	0.189	0.151	0.705
CDA	**0.336**	0.118	0.141	0.643
CCA	0.422	0.098	0.143	0.633
NeRV	0.532	**0.079**	0.139	**0.626**
LocalMDS	0.499	0.118	**0.128**	0.637

Table 2: Error rates of k-nearest neighbor classification based on the visualization, for unsupervised visualization methods. The best results for each data set are in bold; n/a denotes that LLE did not yield a result for the Letter data. NeRV attains the lowest error rate for two data sets and second lowest error rate for one data set.

where $p_{j|i}$ and $q_{j|i}$ are defined by Equations (3) and (2). We showed in Section 2.2 that this cost function has an information retrieval interpretation: it corresponds to mean smoothed recall of retrieving neighbors of query points. The t-SNE method makes two changes which we discuss below.

4.6.1 COST FUNCTION BASED ON JOINT PROBABILITIES

The first change in t-SNE is to the cost function: t-SNE minimizes a "symmetric version" of the cost function, defined as

$$\sum_i \sum_{j \neq i} p_{i,j} \log \frac{p_{i,j}}{q_{i,j}}$$

where the $p_{i,j}$ and $q_{i,j}$ are now joint probabilities over both i and j, so that $\sum_{i,j} p_{i,j} = 1$ and similarly for $q_{i,j}$. The term "symmetric" comes from the fact that the joint probabilities are defined in a specific way for which $p_{i,j} = p_{j,i}$ and $q_{i,j} = q_{j,i}$; note that this need not be the case for all definitions of the joint probabilities.

4.6.2 DEFINITIONS OF THE JOINT PROBABILITIES

The second change in t-SNE is that the joint probabilities are defined in a manner which does not yield quite the same conditional probabilities as in Equations (3) and (2). The joint probabilities are defined as

$$p_{i,j} = \frac{1}{2n}(p_{i|j} + p_{j|i}) \tag{6}$$

where $p_{i|j}$ and $p_{j|i}$ are computed by Equation (3) and n is the total number of data points in the data set, and

$$q_{i,j} = \frac{(1 + ||\mathbf{y}_i - \mathbf{y}_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||\mathbf{y}_k - \mathbf{y}_l||^2)^{-1}} \,. \tag{7}$$

The former equation is intended to ensure that, in the input space, even outlier points will have some other points as neighbors. The latter equation means that, in the visualization, the joint probability

falls off according to a (normalized) t-distribution with one degree of freedom, which is intended to help with a *crowding problem*: because the volume of a small-dimensional neighborhood grows slower than the volume of a high-dimensional one, the neighborhood ends up stretched in the visualization so that moderately distant point pairs are placed too far apart. This tends to cause clumping of the data in the center of the visualization. Since the t-distribution has heavier tails than a Gaussian, using such a distribution for the $q_{i,j}$ makes the visualization less affected by the placement of the moderately distant point pairs, and hence better able to focus on other features of the data.

4.6.3 NEW METHOD: T-NERV

We can easily apply the above-described changes to the cost function in NeRV; we call the resulting variant t-NeRV. We define the cost function as

$$E_{\text{t-NeRV}} = \lambda \sum_i \sum_{j \neq i} p_{i,j} \log \frac{p_{i,j}}{q_{i,j}} + (1-\lambda) \sum_i \sum_{j \neq i} q_{i,j} \log \frac{q_{i,j}}{p_{i,j}} = \lambda D(p,q) + (1-\lambda) D(q,p) \qquad (8)$$

where p and q are the joint distributions over i and j defined by the $p_{i,j}$ and the $q_{i,j}$, and the individual joint probabilities are given by Equations (6) and (7).

It can be shown that this changed cost function again has a natural information retrieval interpretation: it corresponds to the tradeoff between *smoothed precision* and *smoothed recall* of a *two-step information retrieval task*, where an analyst looks at a visualization and (step 1) picks a query point and then (step 2) picks a neighbor for the query point. The probability of picking a query point i depends on how many other points are close to it (that is, it depends on $\sum_j q_{i,j}$), and the probability of picking a neighbor depends as usual on the relative closenesses of the neighbors to the query. Both choices are done based on the visualization, and the choices are compared by smoothed precision and smoothed recall to the relevant pairs of queries and neighbors that are defined based on the input space. The parameter λ again controls the tradeoff between precision and recall.

The connection between the $D(p,q)$ and the recall of the two-step retrieval task can be shown by a similar proof as in Appendix A, the main difference being that conditional distributions $p_{j|i}$ and $q_{j|i}$ are replaced by joint distributions $p_{i,j}$ and $q_{i,j}$, and the sums then go over both i and j. The connection between $D(q,p)$ and precision can be shown analogously.

As a special case, setting $\lambda = 1$ in the above cost function, that is, optimizing only smoothed recall of the two-step retrieval task, yields the cost function of t-SNE. We therefore provide a novel information retrieval interpretation of t-SNE as a method that maximizes recall of query points and their neighbors.

The main conceptual difference between NeRV and t-NeRV is that in t-NeRV the probability of picking a query point in the visualization and in the input space depends on the densities in the visualization and input space respectively; in NeRV all potential query points are treated equally. Which treatment of query points should be used depends on the task of the analyst. Additionally, NeRV and t-NeRV have differences in the technical forms of the probabilities, that is, whether t-distributions or Gaussians are used etc.

The t-NeRV method can be optimized with respect to visualization coordinates $\mathbf{y}_i$ of points, by conjugate gradient optimization as in NeRV; the computational complexity is also the same.

4.6.4 COMPARISON

We briefly compare t-NeRV and NeRV on the Faces data set. The setup is the same as in the previous comparison experiments (Figures 4–7). For t-NeRV we use the effective number of neighbors $k = 40$

to compute the joint probabilities $p_{i,j}$; this corresponds to the perplexity value used by the authors of t-SNE (van der Maaten and Hinton, 2008).

Figure 9 shows the results for the four unsupervised evaluation criteria. According to the mean smoothed precision and mean smoothed recall measures, t-NeRV does worse in terms of recall. The rank-based measures indicate a similar result; however, there t-NeRV does fairly well in terms of mean rank-based smoothed precision. The trustworthiness-continuity curves are similar to the rank-based measures. The curves of mean precision versus mean recall show that t-NeRV does achieve better precision for small values of recall (i.e., for small retrieved neighborhoods), while NeRV does slightly better for larger retrieved neighborhoods. These measures correspond to the information retrieval interpretation of NeRV which is slightly different from that of t-NeRV, as discussed above. Figure 9 **E** shows mean smoothed precision/recall in the t-NeRV sense, where t-NeRV naturally performs relatively better.

Lastly, we computed k-nearest neighbor classification error rate (using $k = 5$) with respect to the identity of the persons in the images. NeRV (with $\lambda = 0.3$) attained an error rate of 0.394 and t-NeRV (with $\lambda = 0.8$) an error rate of 0.226. Here t-NeRV is better; this may be because it avoids the problem of crowding samples near the center of the visualization.

Figures 10-12 show example visualizations of the faces data set. First we show a well-performing comparison method (CDA; Figure 10); it has arranged the faces well in terms of keeping images of the same person in a single area; however, the areas of each person are diffuse and close to other persons, hence there is no strong separation between persons on the display. NeRV, here optimized to maximize precision, makes clearly tighter clusters of each person (Figure 11), which yields better retrieval of neighbor face images. However, NeRV has here placed several persons close to each other in the center of the visualization. The t-NeRV visualization, again optimized to maximize precision (Figure 12) has lessened this behavior, placing the clusters of faces more evenly.

Overall, t-NeRV is a useful alternative formulation of NeRV, and may be useful for data sets especially where crowding near the center of the visualization is an issue.

5. Using NeRV for Supervised Visualization

In this section we show how to use NeRV for supervised visualization. The key idea is simple: NeRV can be computed based on any input-space distances $d(\mathbf{x}_i, \mathbf{x}_j)$, not only the standard Euclidean distances. All that is required for supervised visualization is to compute the input-space distances in a supervised manner. The distances are then plugged into the NeRV algorithm and the visualization proceeds as usual. Note that doing the visualization modularly in two steps is an advantage, since it will be later possible to easily change the algorithm used in either step if desired.

Conveniently, rigorous methods exist for learning a supervised metric from labeled data samples. Learning of supervised metrics has recently been extensively studied for classification purposes and for some semi-supervised tasks, with both simple linear approaches and complicated nonlinear ones; see, for instance, works by Xing et al. (2003), Chang and Yeung (2004), Globerson and Roweis (2006) and Weinberger et al. (2006). Any such metric can in principle be used to compute distances for NeRV. Here we use an early one, which is flexible and can be directly plugged in the NeRV, namely the *learning metric* (Kaski et al., 2001; Kaski and Sinkkonen, 2004; Peltonen et al., 2004) which was originally proposed for data exploration tasks.

We will call NeRV computed with the supervised distances "supervised NeRV" (SNeRV). The information retrieval interpretation of NeRV carries over to SNeRV. Under certain parameter set-

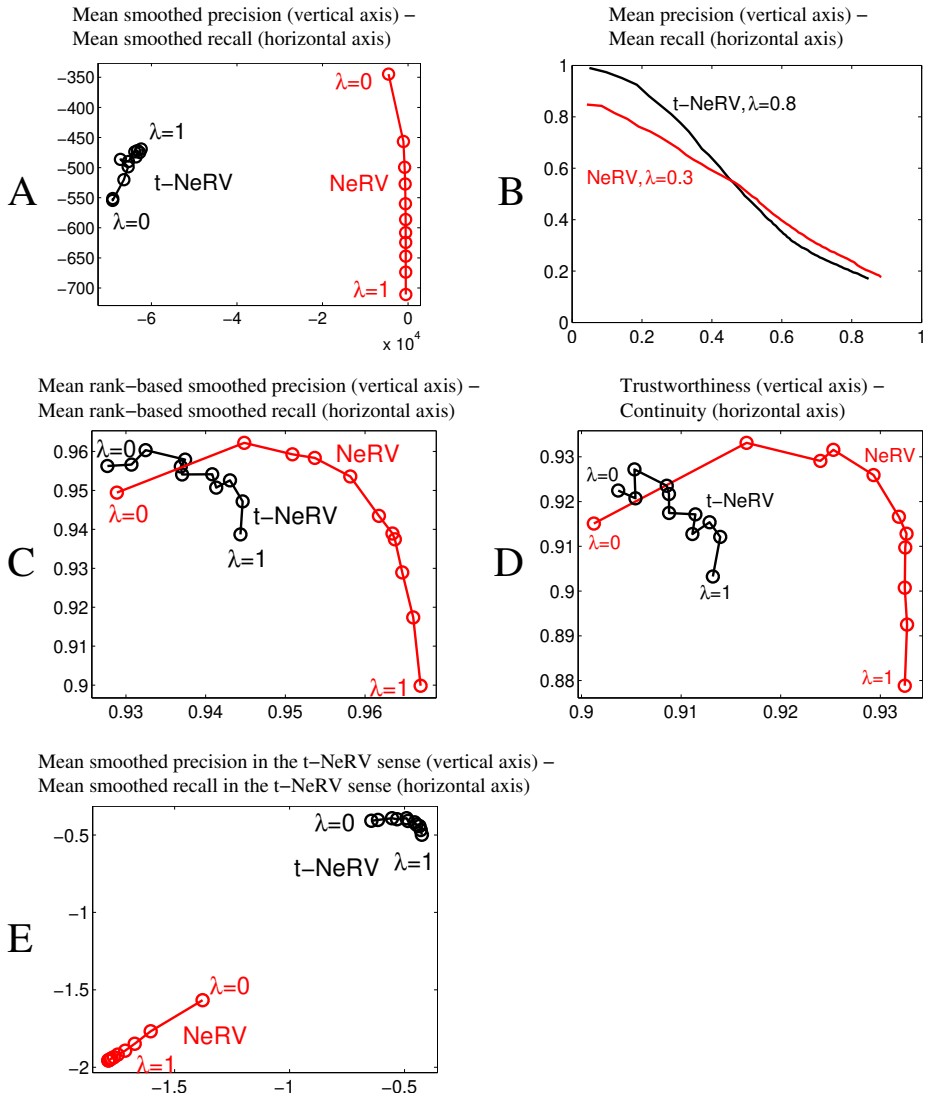

Figure 9: Comparison of NeRV and t-NeRV on the Faces data set according to the four goodness measures described in Section 4.3 (**A**-**D**), and for mean smoothed precision/recall corresponding to the information retrieval interpretation of t-NeRV (**E**; first and second terms of Eqn. 8).

tings SNeRV can be seen as a new, supervised version of stochastic neighbor embedding, but more generally it manages a flexible tradeoff between precision and recall of the information retrieval just like the unsupervised NeRV does.

SNeRV has the useful property that it can directly compute embeddings for unlabeled training points as well as labeled ones. By contrast, some supervised nonlinear dimensionality reduction methods (Geng et al., 2005; Liu et al., 2005; Song et al., 2008) only give embeddings for labeled points; for unlabeled points, the mapping is approximated for instance by interpolation or by training

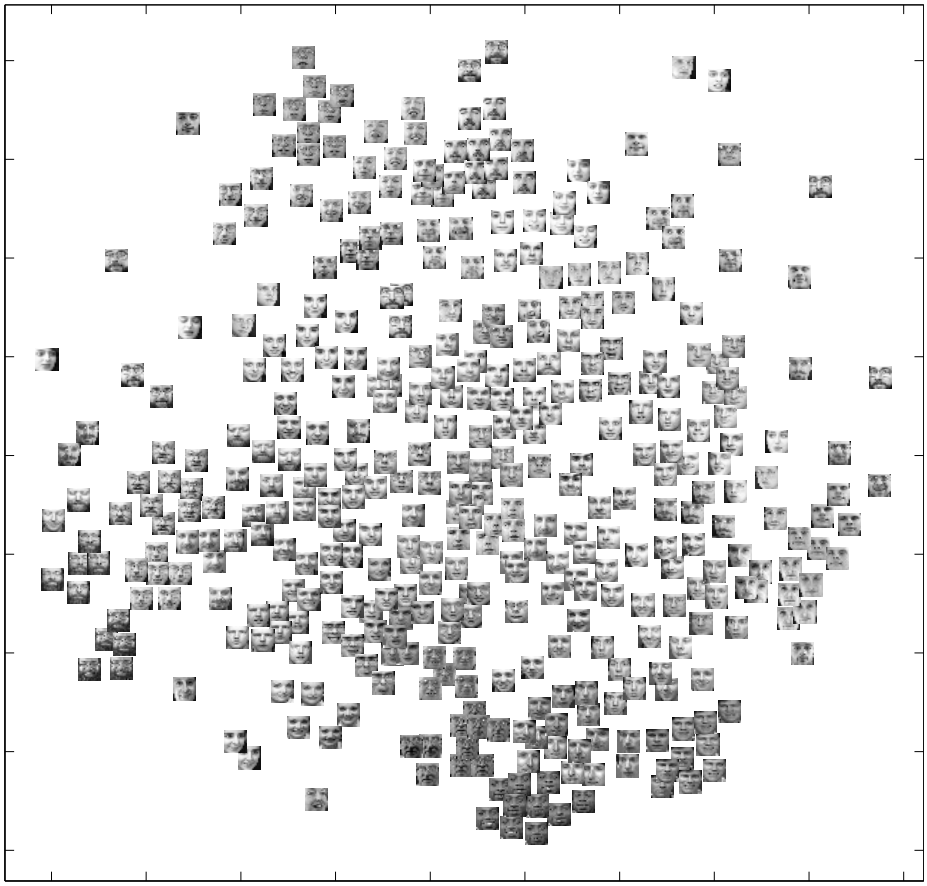

Figure 10: Example visualization of the Faces data set with CDA.

a neural network. Such approximation is not needed for SNeRV. (On the other hand, a trained neural network can embed not only unlabeled training points, but also previously unseen new points; if such generalization is desired, the same kinds of approximate mappings can be learned for SNeRV.)

In the next subsections we present the details of the distance computation, and then describe experimental comparisons showing that SNeRV outperforms several existing supervised methods.

5.1 Supervised Distances for NeRV

The input-space distances for SNeRV are computed using *learning metrics* (Kaski et al., 2001; Kaski and Sinkkonen, 2004; Peltonen et al., 2004). It is a formalism particularly suited for so-called "supervised unsupervised learning" where the final goal is still to make discoveries as in unsupervised learning, but the metric helps to focus the analysis by emphasizing useful features and, moreover, does that locally, differently for different samples. Learning metrics have previously been applied to clustering and visualization.

In brief, the learning metric is a Riemannian topology-preserving metric that measures distances in terms of changes in the class distribution. The class distribution is estimated through

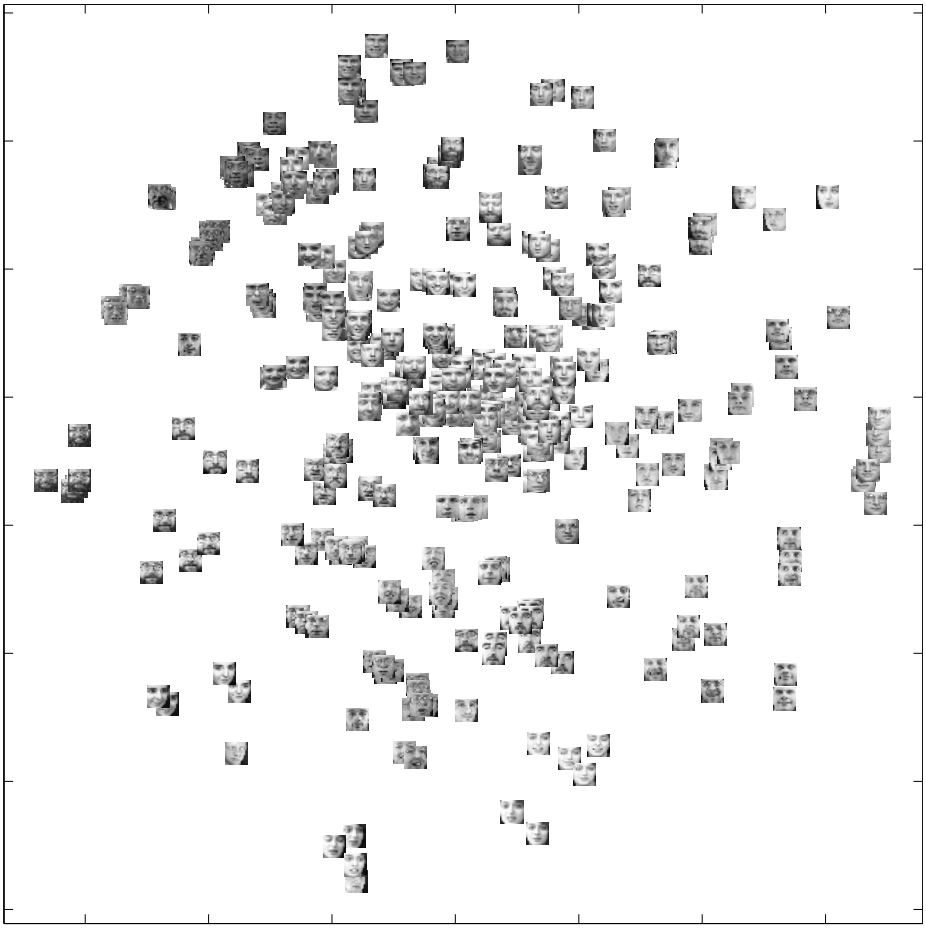

Figure 11: Example visualization of the Faces data set with NeRV, here maximizing precision (tradeoff parameter $\lambda = 0$).

conditional density estimation from labeled samples. Topology preservation helps in generalizing to new points, since class information cannot override the input space topology. In this metric, we can compute input-space distances between any two data points, and hence visualize the points with NeRV, whether they have known labels or not.

5.1.1 DEFINITION

The learning metric is a so-called Riemannian metric. Such a metric is defined in a local manner; between two (infinitesimally) close-by points it has a simple form, and this simple form is extended through path integrals to global distances.

In the learning metric, the squared distance between two close-by points $\mathbf{x}_1$ and $\mathbf{x}_2$ is given by the quadratic form

$$d_L(\mathbf{x}_1, \mathbf{x}_2)^2 = (\mathbf{x}_1 - \mathbf{x}_2)^T \mathbf{J}(\mathbf{x}_1)(\mathbf{x}_1 - \mathbf{x}_2). \tag{9}$$

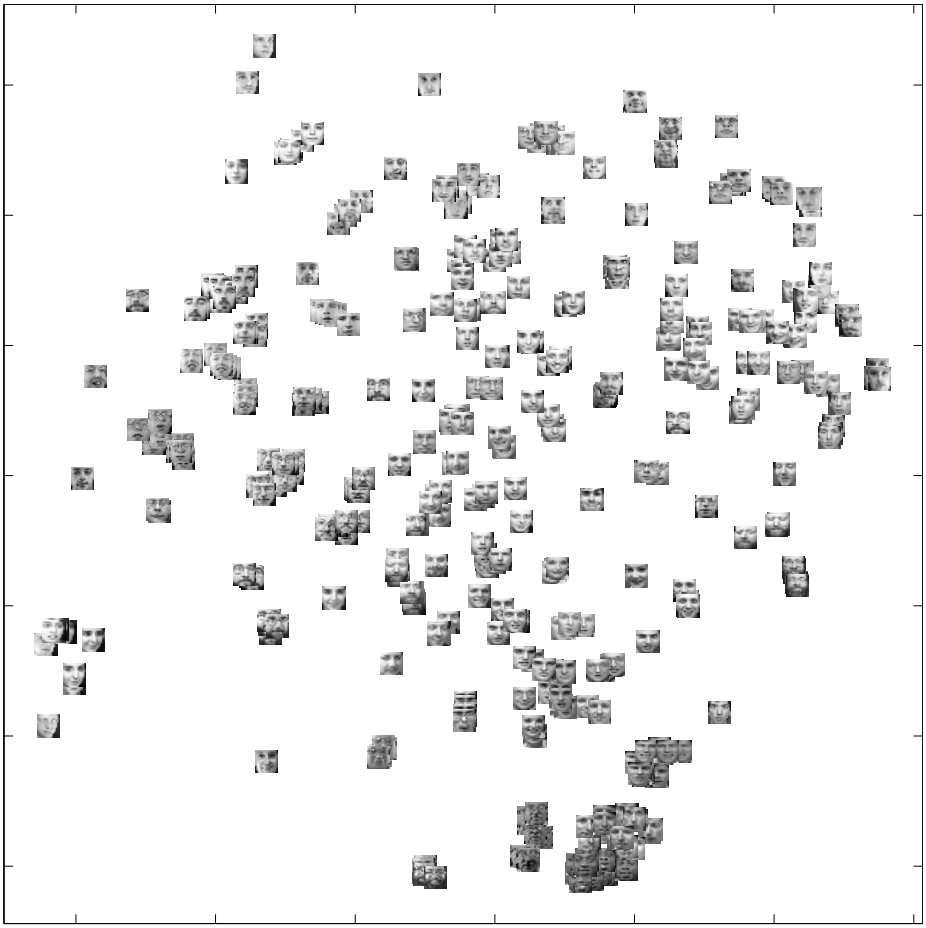

Figure 12: Example visualization of the Faces data set with t-NeRV, here maximizing precision (tradeoff parameter $\lambda = 0$).

Here $\mathbf{J}(\mathbf{x})$ is the Fisher information matrix which describes the local dependency of the conditional class distribution on the input features, that is,

$$\mathbf{J}(\mathbf{x}) = \sum_c p(c|\mathbf{x}) \left(\frac{\partial}{\partial \mathbf{x}} \log p(c|\mathbf{x}) \right) \left(\frac{\partial}{\partial \mathbf{x}} \log p(c|\mathbf{x}) \right)^T .$$

Here the c are the classes and the $p(c|\mathbf{x})$ are the conditional class probabilities at point $\mathbf{x}$. The idea is that the local distances grow the most along directions where the conditional class distribution $p(c|\mathbf{x})$ changes the most. It can be shown that the quadratic form (9) is, for close-by points, equivalent to the Kullback-Leibler divergence $D(p(c|\mathbf{x}_1), p(c|\mathbf{x}_2))$.

The general distance $d_L(\mathbf{x}_1, \mathbf{x}_2)$ between two far-away points $\mathbf{x}_1$ and $\mathbf{x}_2$ is defined in the standard fashion of Riemannian metrics: the distance is the *minimal path integral* over local distances, where the minimum is taken over all possible paths connecting $\mathbf{x}_1$ and $\mathbf{x}_2$. Notice that in a Riemannian metric, the straight path may not yield the minimum distance.

Learning metrics defined in the above manner satisfy the three criteria required of any metric: the distances d_L are nonnegative, symmetric, and satisfy the triangle inequality. Because the learning metric distances are defined as minimal path integrals they preserve the topology of the input space; roughly speaking, if the distance between two points is small, then there must be a path between them where distances are small along the entire path.

5.1.2 PRACTICAL COMPUTATION

In order to compute local distances using the Fisher information matrices $\mathbf{J}(\mathbf{x})$, we need an estimate for the conditional probability distributions $p(c|\mathbf{x})$. We learn the distributions by optimizing a discriminative mixture of labeled Gaussian densities for the data (Peltonen et al., 2004). The conditional density estimate is of the form

$$\hat{p}(c|\mathbf{x}) = \frac{\sum_{k=1}^{K} \beta_{ck} \exp(-||\mathbf{x}-\mathbf{m}_k||^2/2\sigma^2)}{\sum_{k=1}^{K} \exp(-||\mathbf{x}-\mathbf{m}_k||^2/2\sigma^2)} \tag{10}$$

where the number of Gaussians K, the centroids $\mathbf{m}_k$, the class probabilities β_{ck} and the Gaussian width σ (standard deviation) are parameters of the estimate; we require that the β_{ck} are nonnegative and that $\sum_c \beta_{ck} = 1$ for all k. The $\mathbf{m}_k$ and β_{ck} are optimized by a conjugate gradient algorithm to maximize the conditional class likelihood, and K and σ are chosen by internal cross-validation (see Section 5.3).

Given the Fisher matrices, we next need to compute the global distances between all point pairs. In most cases the minimal path integrals in the global distance definition cannot be computed analytically, and we use a graph-based approximation. We first form a fully connected graph between all known data points, where the path between each pair of points is approximated by a straight line. For these straight paths, the path integral can be computed by piecewise approximation (see Peltonen et al., 2004, for details; we use $T = 10$ pieces in all experiments). We could then use graph search (Floyd's algorithm) to find the shortest paths in the graph and use the shortest path distances as the learning metric distances. This graph approximation would take $O(n^3)$ time where n is the number of data points; note that this would not be excessive since a similar graph computation is needed in methods like isomap. However, in our experiments the straight line paths yielded about equally good results, so we simply use them, which takes only $O(n^2)$ time. Therefore SNeRV as a whole took only $O(n^2)$ time just like NeRV.

5.2 Comparison Methods for Supervised Visualization

For each data set to be visualized, the choice of supervised vs. unsupervised visualization is up to the analyst; in general, supervised embedding will preserve differences between classes better but at the expense of within-class details. In the experiments of this section we concentrate on comparing performances of supervised methods; we will compare SNeRV to three recent supervised nonlinear embedding methods.

Multiple relational embedding (MRE; Memisevic and Hinton, 2005) was proposed as an extension of stochastic neighbor embedding (Hinton and Roweis, 2002). MRE minimizes a sum of mismatches, measured by Kullback-Leibler divergence, between neighborhoods in the embedding and several different input neighborhoods: typically one of the input neighborhoods is derived from the input-space coordinates and the others are derived from auxiliary information such as labels.

MRE is able to use unlabeled data; for unlabeled points, divergences that involve neighborhoods based on labels are simply left out of the cost function.

Colored maximum variance unfolding (Song et al., 2008) is an extension of the unsupervised maximum variance unfolding. It maximizes the dependency between the embedding coordinates and the labels according to the Hilbert-Schmidt independence criterion, which is based on a cross-covariance operator. This leads to constrained optimization of the output kernel. Because of these details the method is also called maximum unfolding via Hilbert-Schmidt independence criterion (MUHSIC); we use this abbreviation.

Supervised isomap (S-Isomap; Geng et al., 2005) is an extension of the unsupervised isomap. The only difference to unsupervised isomap is a new definition of the input-space distances: roughly speaking, distances between points in different classes will grow faster than distances between same-class points. The actual embedding is done in the same way as in unsupervised isomap (described in Section 4.1). Other supervised extensions of isomap have been introduced by Li and Guo (2006) and Gu and Xu (2007).

Parametric embedding (PE; Iwata et al., 2007) represents the embedded data with a Gaussian mixture model with all Gaussians having the same covariances in the embedding space, and attempts to preserve the topology of the original data by minimizing a sum of Kullback-Leibler divergences.

Neighbourhood component analysis (NCA; Goldberger et al., 2005; see also Kaski and Peltonen, 2003, Peltonen and Kaski, 2005) is a linear and non-parametric dimensionality reduction method which learns a Mahalanobis distance measure such that, in the transformed space, k-nearest neighbor classification achieves the maximum accuracy.

5.3 Methodology for the Supervised Experiments

We used the four benchmark data sets having class information (Letter, Phoneme, Landsat, and TIMIT described in Section 4.5) to compare supervised NeRV and the five supervised visualization methods described in Section 5.2, namely multiple relational embedding (MRE), colored maximum variance unfolding (MUHSIC), supervised isomap (S-Isomap), parametric embedding (PE), and neighbourhood component analysis (NCA). We used a standard 10-fold cross-validation setup: in each fold we reserve one of the subsets for testing and use the rest of the data for training. For each data set, we use SNeRV and the comparison methods to find 2-dimensional visualizations.

In principle we could evaluate the results as in Section 4.3 for the unsupervised experiments, that is, by mean smoothed precision and recall; the only difference would be to use the supervised learning metric for the evaluation. However, unlike SNeRV, the other methods have not been formulated using the same supervised metrics. To make an unbiased comparison of the methods, we resort to a simple indirect evaluation: we evaluate the performance of the four methods by class prediction accuracy of the resulting visualizations. Although it is an indirect measure, the accuracy is a reasonable choice for unbiased comparison and has been used in several supervised dimensionality reduction papers. In more detail, we provide test point locations during training but not their labels; after the methods have computed their visualization results, we classify the test points by running a k-nearest neighbor classifier ($k = 5$) on the embedded data, and evaluate the classification error rates of the methods.

We use a standard internal 10-fold validation strategy to choose all parameters which are not optimized by their respective algorithms: each training set is subdivided into 10 folds where 9/10 of data is used for learning and 1/10 for validation; we learn visualizations with the different parameter

values; the values that yielded the best classification accuracy for the embedded validation points are then chosen and used to compute the final visualization for the whole training data.

We ran two versions of SNeRV using $\lambda = 0.1$ and $\lambda = 0.3$. The scaling parameters σ_i were set by fixing the entropy of the input neighborhoods as described in Section 2.2. Here we specified the rough upper limit for the number of relevant neighbors as $0.5 \cdot n/K$ where n is the number of data points and K is the number of mixture components used to estimate the metric; this choice roughly means that for well-separated mixture components, each data point will on average consider half of the data points from the same mixture component as relevant neighbors. A simplified validation sufficed for the number K and width σ of Gaussians: we did not need to run the embedding step but picked the values that gave best conditional class likelihood for validation points in the input space. For S-Isomap we chose its parameter α and its number of nearest neighbors using the validation sets, and trained a generalized radial basis function network to project new points, as suggested by Geng et al. (2005). For MUHSIC, the parameters are the regularization parameter ν, number of nearest neighbors, and number of eigenvectors in the graph Laplacian, and we used linear interpolation to project new points as suggested by the MUHSIC authors. For MRE the only free parameter is its neighborhood smoothness parameter σ_{MRE}. For the PE one needs to provide a conditional density estimate: we used the same one that SNeRV uses (see Equation 10) to obtain a comparison as unbiased as possible. Neighbourhood component analysis is a non-parametric method, therefore we did not need to choose any parameters for it.

5.4 Results of Supervised Visualization

Figure 13 shows the average error rate over the 10 folds as well as the standard deviation. The best two methods are SNeRV and PE, which give good results in all data sets. On two of the data sets (Letter and TIMIT) SNeRV is clearly the best; on the other two data sets (Phoneme and Landsat) SNeRV is about as good as the best of the remaining methods (S-Isomap and parametric embedding, respectively). MRE is clearly worse than the other methods, whereas MUHSIC and NCA results depend on the data set: on Letter they are the second and third worst methods after MRE, while in the other data sets they are not far from the best methods.

The value of the tradeoff parameter λ did not affect the performance of SNeRV much; both $\lambda = 0.1$ and $\lambda = 0.3$ produced good projections.

To evaluate whether the best method on each data set is statistically significantly better than the next best one, we performed a paired t-test of the performances across the 10 cross-validation folds (Table 3). The two best methods compared are always SNeRV and parametric embedding, except for the Phoneme data set for which the two best methods are SNeRV and S-Isomap. For the Letter and the TIMIT data sets SNeRV is significantly better than the next best method, while for the other two data sets the difference is not significant. In summary, all the significant differences are in favor of SNeRV.

Figure 14 presents sample visualizations of the letter recognition data set; projection results are shown for one of the 10 cross-validation folds, including both training and test points. Although there is some overlap, in general SNeRV shows distinct clusters of classes—for example, the letter "M" is a well separated cluster in the top of the figure.

Parametric embedding also manages to separate some letters, such as "A" and "I", but there is a severe overlap of classes in the center of the figure. In S-Isomap we see that there are a few very well separated clusters of classes, like the letters "W" and "N", but there is a large area with

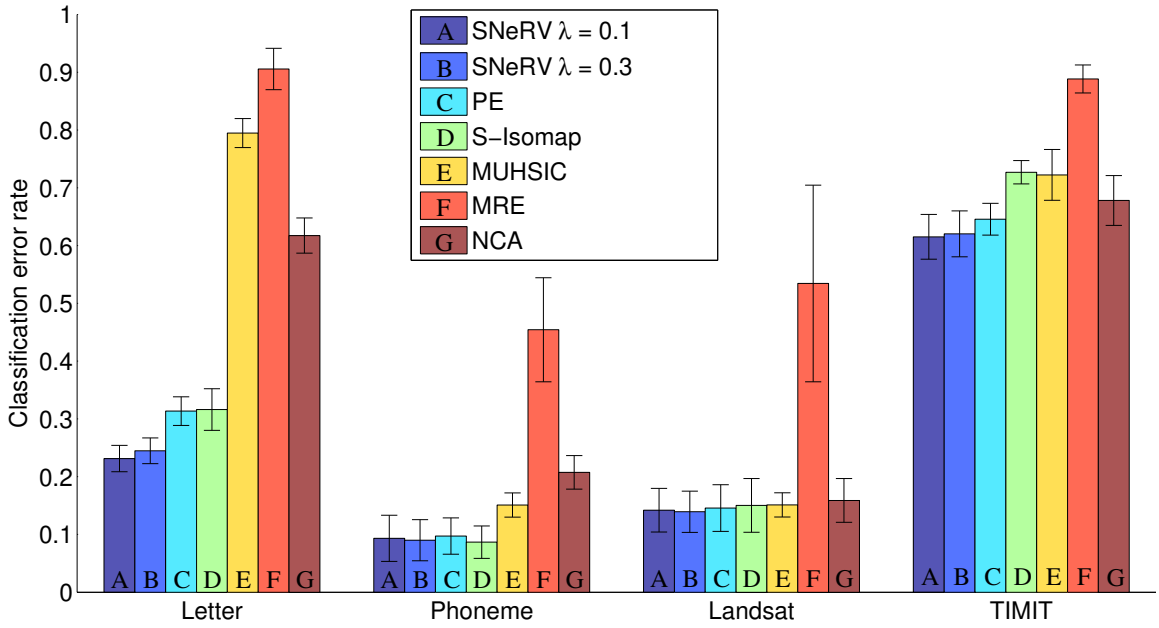

Figure 13: Performance of the supervised nonlinear embedding methods in each benchmark data set. The results are average classification error rates over 10 cross-validation folds (smaller is better), and the standard deviations are shown with error bars.

Data set	Best method	Second best	p-value
Letter	**SNeRV ($\lambda = 0.1$)**	PE	$1.8 \cdot 10^{-6}$
Phoneme	S-Isomap	SNeRV ($\lambda = 0.3$)	0.54
Landsat	SNeRV ($\lambda = 0.3$)	PE	0.28
TIMIT	**SNeRV ($\lambda = 0.1$)**	PE	$3.4 \cdot 10^{-3}$

Table 3: Statistical significance of the difference between the two best methods. The p-values are from a paired t-test of the 10-fold cross-validation results; statistically significant winners have been boldfaced.

overlapping classes near the center of the right edge of the figure. This overlap is worse than in SNeRV but still roughly comparable; by contrast, MUHSIC, MRE and NCA performed poorly on this data set, leaving most classes severely overlapped.

6. Conclusions and Discussion

By formulating the task of nonlinear projection for information visualization as an information retrieval task, we have derived a rigorously motivated pair of measures for visualization performance, *mean smoothed precision* and *mean smoothed recall*. We showed that these new measures are extensions of two traditional information retrieval measures: mean smoothed precision can be interpreted as a more sophisticated extension of mean precision, the proportion of false positives in the neigh-

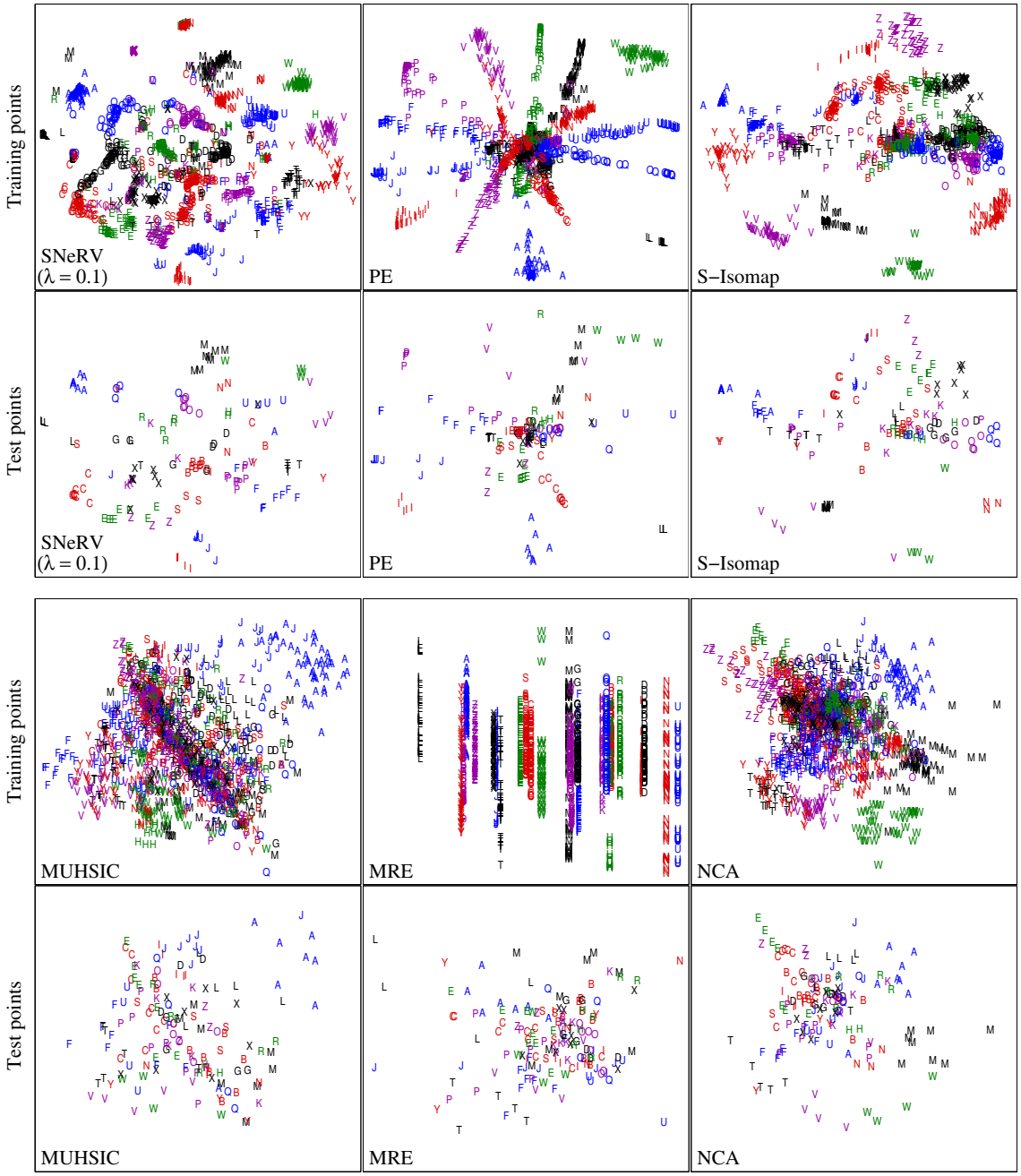

Figure 14: Visualizations of the letter recognition data set by all supervised methods.

borhood retrieved from the visualization. Analogously, mean smoothed recall is an extension of mean recall, the proportion of misses incurred by the retrieved neighborhood.

We introduced an algorithm called *neighbor retrieval visualizer* (NeRV) that optimizes the total cost, interpretable as a tradeoff between mean smoothed precision and mean smoothed recall. The tradeoff is governed by a parameter λ set by the user according to the desired cost of a miss relative

to the desired cost of a false positive. The earlier method stochastic neighbor embedding is obtained as a special case when $\lambda = 1$, optimizing mean smoothed recall.

We showed that NeRV can be used for both unsupervised and supervised visualization. For unsupervised visualization, we simply use fixed input distances; for supervised visualization we learn a supervised distance metric for the input space and plug the resulting input distances to the NeRV algorithm. In the latter case the key idea is to use supervision (labeled data) in a way that does not override the input feature space; we use a topology-preserving class-discriminative metric called the *learning metric* for the input space.

In unsupervised visualization, NeRV outperformed alternatives for most of the six data sets we tried, for four different pairs of measures, and was overall the best method. NeRV also performed well in a comparison by unsupervised classification. Many of the best manifold extraction methods perform surprisingly poorly, most likely because they have not been designed to reduce the dimensionality below the intrinsic dimensionality of the data manifold. In visualization, however, we generally have no choice but to reduce the dimensionality of the data to two or three, even if its intrinsic dimensionality is higher. NeRV is designed to find a mapping that is, in a well-defined sense, optimal for a certain type of visualization regardless of the intrinsic dimensionality of the data.

In supervised visualization, the supervised version of NeRV performed as well as or better than the best alternative method Parametric embedding; this shows that the plug-in learning metrics work well in incorporating supervision.

6.1 Discussion

NeRV models relevance using probability distributions, which makes sense if the total "amount" of relevance for any query is normalized to a fixed sum. Such normalization is desirable for any relevance measure, because for any query (point of interest) the relevance of a retrieved neighbor point should depend on its proximity relative to the proximities of other points, rather than on its absolute distance from the query point. (Our previous method, local MDS, can be thought of as an attempt to approximate NeRV without the normalization.)

The Kullback-Leibler divergences in NeRV are natural choices for measuring the difference between two probability distributions, but in principle other divergence measures could be used as well. The notions of neighbor retrieval and a probabilistic relevance model are the crucial parts of NeRV, not the specific divergence measure.

Our notion of plug-in supervised metrics could in principle be used with other methods too; other unsupervised embedding algorithms that work based on a distance matrix can also be turned into supervised versions, by plugging in learning metric distances into the distance matrix. We performed an initial experiment with Sammon's mapping (Peltonen et al., 2004); a similar idea for isomap appeared later (Weng et al., 2005). However, we believe that NeRV is an especially attractive choice for the embedding step since it has the information retrieval interpretation and it performed well empirically.

An implementation of the NeRV and local MDS algorithms as well as the *mean smoothed precision-mean smoothed recall* measures is available at `http://www.cis.hut.fi/projects/mi/software/dredviz`

Acknowledgments

The authors belong to the Adaptive Informatics Research Centre, a national CoE of the Academy of Finland, and to Helsinki Institute for Information Technology HIIT. Jarkko Venna is currently at Numos Ltd. JP was supported by the Academy of Finland, decision number 123983, and HA by the Portuguese Foundation for Science and Technology, scholarship number SFRH/BD/39642/2007. This work was also supported in part by the PASCAL2 Network of Excellence, ICT 216886.

Appendix A. Proof of the Connection between the Probabilistic Cost Functions and Precision and Recall

In Section 2.2 we introduced Kullback-Leibler divergences as cost functions for visual neighbor retrieval, based on probability distributions q_i and p_i which generalize the relevance model implicit in precision and recall. We will next show that in the simple case of "binary neighborhoods" the cost functions reduce to precision and recall. By "binary neighborhoods" we mean that, in both the input space and the visualization, (i) the point of interest has some number of relevant neighbors and all the other points are completely irrelevant, and (ii) the points that are relevant are all equally relevant.

In the probabilistic model the binary neighborhoods can be interpreted as follows. Let i be the point of interest, and let P_i be the set of relevant neighbors for point i in the input space. P_i can be the set of all points (other than i itself) falling inside some fixed radius from point i in the input space, or it can be the set containing some fixed number of points nearest to i in the input space. In either case, let r_i be the size of P_i.

We define that the relevant neighbors of the point of interest i have an equal non-zero probability of being chosen, and all the other points have a near-zero probability of being chosen. In other words, we define

$$p^*_{j|i} = \begin{cases} a_i \equiv \frac{1-\delta}{r_i}, & \text{if point } j \text{ is in } P_i \\ b_i \equiv \frac{\delta}{N-r_i-1}, & \text{otherwise.} \end{cases}$$

Here N is the total number of data points, and $0 < \delta \ll 0.5$ gives the irrelevant points a very small probability.

Similarly, let Q_i be the set of neighbors for point i in the visualization. Again, Q_i can be the set of all points (other than i itself) falling inside some fixed radius from point i in the visualization, or it can be the set containing some fixed number of points nearest to i in the visualization. In either case, let k_i be the size of Q_i. Note that the sizes of Q_i and P_i can be different, that is, k_i can be different from r_i.

We define the probability of choosing a neighbor from the visualization as

$$q^*_{j|i} = \begin{cases} c_i \equiv \frac{1-\delta}{k_i}, & \text{if point } j \text{ is in } Q_i \\ d_i \equiv \frac{\delta}{N-k_i-1}, & \text{otherwise.} \end{cases}$$

Consider the Kullback-Leibler divergence $D(p^*_i, q^*_i)$ for any fixed i. We now show that minimizing this divergence is equivalent to maximizing recall where point i is the query. The divergence is a sum over elements $p^*_{j|i} \log \frac{p^*_{j|i}}{q^*_{j|i}}$, thus the sum can be divided into four parts depending on which

value $p^*_{j|i}$ takes (two possibilities) and which value $q^*_{j|i}$ takes (two possibilities). We get

$$\begin{aligned}
D(p^*_i, q^*_i) = &\sum_{j\neq i, p^*_{j|i}=a_i, q^*_{j|i}=c_i} \left(a_i \log \frac{a_i}{c_i}\right) + \sum_{j\neq i, p^*_{j|i}=a_i, q^*_{j|i}=d_i} \left(a_i \log \frac{a_i}{d_i}\right) \\
&+ \sum_{j\neq i, p^*_{j|i}=b_i, q^*_{j|i}=c_i} \left(b_i \log \frac{b_i}{c_i}\right) + \sum_{j\neq i, p^*_{j|i}=b_i, q^*_{j|i}=d_i} \left(b_i \log \frac{b_i}{d_i}\right) \\
&= \left(a_i \log \frac{a_i}{c_i}\right) N_{\mathrm{TP},i} + \left(a_i \log \frac{a_i}{d_i}\right) N_{\mathrm{MISS},i} + \left(b_i \log \frac{b_i}{c_i}\right) N_{\mathrm{FP},i} + \left(b_i \log \frac{b_i}{d_i}\right) N_{\mathrm{TN},i}
\end{aligned}$$

where on the last line the terms inside parentheses are simply constant coefficients. Here $N_{\mathrm{TP},i}$ is the number of true positives for this query, that is, points for which the probability is high in both the data and the visualization. The number of misses, that is, the number of points that have a low probability in the visualization although the probability in the data is high, is $N_{\mathrm{MISS},i}$. The number of false positives (high probability in the visualization, low in the data) is $N_{\mathrm{FP},i}$. Finally the number of true negatives (low probability in both the visualization and the data) is $N_{\mathrm{TN},i}$.

It is straightforward to check that if δ is very small, then the coefficients for the misses and false positives dominate the divergence. This yields

$$\begin{aligned}
D(p^*_i, q^*_i) &\approx N_{\mathrm{MISS},i} \left(a_i \log \frac{a_i}{d_i}\right) + N_{\mathrm{FP},i} \left(b_i \log \frac{b_i}{c_i}\right) \\
&= N_{\mathrm{MISS},i} \frac{1-\delta}{r_i} \left(\log \frac{(N-k_i-1)}{\delta} + \log \frac{(1-\delta)}{r_i}\right) \\
&+ N_{\mathrm{FP},i} \frac{\delta}{N-r_i-1} \left(\log \frac{\delta}{N-r_i-1} - \log \frac{(1-\delta)}{k_i}\right) \\
&= N_{\mathrm{MISS},i} \frac{1-\delta}{r_i} \left(\log \frac{(N-k_i-1)}{r_i} + \log \frac{(1-\delta)}{\delta}\right) \\
&\qquad + N_{\mathrm{FP},i} \frac{\delta}{N-r_i-1} \left(\log \frac{k_i}{N-r_i-1} - \log \frac{(1-\delta)}{\delta}\right) . \quad (11)
\end{aligned}$$

Because the terms $\log[(1-\delta)/\delta]$ dominate the other logarithmic terms, (11) further simplifies to

$$\begin{aligned}
D(p^*_i, q^*_i) &\approx \left(N_{\mathrm{MISS},i} \frac{1-\delta}{r_i} - N_{\mathrm{FP},i} \frac{\delta}{N-r_i-1}\right) \log \frac{(1-\delta)}{\delta} \\
&\approx N_{\mathrm{MISS},i} \frac{1-\delta}{r_i} \log \frac{(1-\delta)}{\delta} = \frac{N_{\mathrm{MISS},i}}{r_i} C
\end{aligned}$$

where C is a constant that only depends on δ and not on i. Hence if we minimized this cost function, we would be maximizing the recall of the query, which is defined as

$$\mathrm{recall}(i) = \frac{N_{\mathrm{TP},i}}{r_i} = 1 - \frac{N_{\mathrm{MISS},i}}{r_i} .$$

We can analogously show that for any fixed i, minimizing $D(q^*_i, p^*_i)$ is equivalent to maximizing precision of the corresponding query.

Because $D(q^*_i, p^*_i)$ and $D(p^*_i, q^*_i)$ are equivalent to precision and recall, and p_i and q_i can be seen as more sophisticated generalizations of p^*_i and q^*_i, we interpret $D(q_i, p_i)$ and $D(p_i, q_i)$ as more sophisticated generalizations of precision and recall.

References

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 585–591. MIT Press, Cambridge, MA, 2002a.

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. Technical Report TR-2002-01, Department of Computer Science,The University of Chicago, 2002b.

Mira Bernstein, Vin de Silva, John C. Langford, and Joshua B. Tenenbaum. Graph approximations to geodesics on embedded manifolds. Technical report, Department of Psychology, Stanford University, 2000.

Catherine L. Blake and C. J. Merz. UCI repository of machine learning databases. `http://www.ics.uci.edu/∼mlearn/MLRepository.html`, 1998.

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

Hong Chang and Dit-Yan Yeung. Locally linear metric adaptation for semi-supervised clustering. In *Proceedings of the Twenty-first International Conference on Machine Learning 2004*, pages 153–160. Omnipress, Madison, WI, 2004.

Lisha Chen and Andreas Buja. Local multidimensional scaling for nonlinear dimension reduction, graph drawing and proximity analysis. *Journal of the American Statistical Association*, 104(485): 209–219, 2009.

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, 1997.

David L. Donoho and Carrie Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Sciences*, 100:5591–5596, 2003.

Xin Geng, De-Chuan Zhan, and Zhi-Hua Zhou. Supervised nonlinear dimensionality reduction for visualization and classification. *IEEE Transactions on Systems, Man, and Cybernetics–Part B: Cybernetics*, 35:1098–1107, 2005.

Amir Globerson and Sam Roweis. Metric learning by collapsing classes. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing 18*, pages 451–458. MIT Press, Cambridge, MA, 2006.

Jacob Goldberger, Sam Roweis, Geoffrey Hinton, and Ruslan Salakhutdinov. Neighbourhood components analysis. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 513–520. MIT Press, Cambridge, MA, 2005.

John C. Gower. Some distance properties of latent root and vector methods used in multivariate analysis. *Biometrika*, 53:325–338, 1966.

Rui-jun Gu and Wen-bo Xu. Weighted kernel Isomap for data visualization and pattern classification. In Y. Wang, Y.-m. Cheung, and H. Liu, editors, *Computational Intelligence and Security (CIS 2006)*, pages 1050–1057. Springer-Verlag, Berlin Heidelberg, 2007.

Geoffrey Hinton and Sam T. Roweis. Stochastic neighbor embedding. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural information processings systems 14*, pages 833–840. MIT Press, Cambridge, MA, 2002.

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

Tomoharu Iwata, Kazumi Saito, Naonori Ueda, Sean Stromsten, Thomas L. Griffiths, and Joshua B. Tenenbaum. Parametric embedding for class visualization. *Neural Computation*, 19:2536–2556, 2007.

Samuel Kaski and Jaakko Peltonen. Informative discriminant analysis. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, pages 329–336. AAAI Press, Menlo Park, CA, 2003.

Samuel Kaski and Janne Sinkkonen. Principle of learning metrics for exploratory data analysis. *Journal of VLSI Signal Processing, special issue on Machine Learning for Signal Processing*, 37: 177–188, 2004.

Samuel Kaski, Janne Sinkkonen, and Jaakko Peltonen. Bankruptcy analysis with self-organizing maps in learning metrics. *IEEE Transactions on Neural Networks*, 12:936–947, 2001.

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.

Teuvo Kohonen, Jussi Hynninen, Jari Kangas, Jorma Laaksonen, and Kari Torkkola. LVQ_PAK: The learning vector quantization program package. Technical Report A30, Helsinki University of Technology, Laboratory of Computer and Information Science, FIN-02150 Espoo, Finland, 1996.

Joseph B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrica*, 29(1):1–27, 1964.

John Aldo Lee, Amaury Lendasse, Nicolas Donckers, and Michel Verleysen. A robust nonlinear projection method. In M. Verleysen, editor, *ESANN'2000, Eighth European Symposium on Artificial Neural Networks*, pages 13–20. D-Facto Publications, Bruges, Belgium, 2000.

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

Chun-Guang Li and Jun Guo. Supervised Isomap with explicit mapping. In J.-S. Pan, P. Shi, and Y. Zhao, editors, *Proceedings of the First International Conference on Innovative Computing, Information and Control (ICICIC'06)*, volume 3, pages 345–348. IEEE, 2006.

E. Liitiäinen and A. Lendasse. Variable scaling for time series prediction: Application to the ESTSP'07 and the NN3 forecasting competitions. In *IJCNN 2007, International Joint Conference on Neural Networks*, pages 2812–2816. IEEE, Piscataway, NJ, 2007.

Ning Liu, Fengshan Bai, Jun Yan, Benyu Zhang, Zheng Chen, and Wei-Ying Ma. Supervised semi-definite embedding for email data cleaning and visualization. In Y. Zhang, K. Tanaka, J. Xu Yu, S. Wang, and M. Li, editors, *Web Technologies Research and Development–APWeb 2005*, pages 972–982. Springer-Verlag, Berlin Heidelberg, 2005.

Roland Memisevic and Geoffrey Hinton. Multiple relational embedding. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 913–920. MIT Press, Cambridge, MA, 2005.

Jaakko Peltonen and Samuel Kaski. Discriminative components of data. *IEEE Transactions on Neural Networks*, 16(1):68–83, 2005.

Jaakko Peltonen, Arto Klami, and Samuel Kaski. Improved learning of Riemannian metrics for exploratory analysis. *Neural Networks*, 17:1087–1100, 2004.

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

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.

Blake Shaw and Tony Jebara. Minimum volume embedding. In M. Meila and X. Shen, editors, *Proceedings of AISTATS*07, the 11th International Conference on Artificial Intelligence and Statistics (JMLR Workshop and Conference Proceedings Volume 2)*, pages 460-467, 2007.

Le Song, Alex Smola, Kersten Borgwardt, and Arthur Gretton. Colored maximum variance unfolding. In J. C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1385–1392. MIT Press, Cambridge, MA, 2008.

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.

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

TIMIT. CD-ROM prototype version of the DARPA TIMIT acoustic-phonetic speech database, 1998.

Warren S. Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17:401–419, 1952.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

Laurens van der Maaten, Eric Postma, and Jaap van der Herik. Dimensionality reduction: A comparative review. Technical report 2009–005, Tilburg centre for Creative Computing, Tilburg University, Tilburg, The Netherlands, 2009.

Jarkko Venna. *Dimensionality Reduction for Visual Exploration of Similarity Structures*. PhD thesis, Helsinki University of Technology, Espoo, Finland, 2007.

Jarkko Venna and Samuel Kaski. Local multidimensional scaling. *Neural Networks*, 19:889–99, 2006.

Jarkko Venna and Samuel Kaski. Comparison of visualization methods for an atlas of gene expression data sets. *Information Visualization*, 6:139–154, 2007a.

Jarkko Venna and Samuel Kaski. Nonlinear dimensionality reduction as information retrieval. In M. Meila and X. Shen, editors, *Proceedings of AISTATS*07, the 11th International Conference on Artificial Intelligence and Statistics (JMLR Workshop and Conference Proceedings Volume 2)*, pages 572–579, 2007b.

Kilian Weinberger, Benjamin Packer, and Lawrence Saul. Nonlinear dimensionality reduction by semidefinite programming and kernel matrix factorization. In R. G. Cowell and Z. Ghahramani, editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (AISTATS 2005)*, pages 381–388. Society for Artificial Intelligence and Statistics, 2005. (Available electronically at `http://www.gatsby.ucl.ac.uk/aistats/`).

Kilian Weinberger, John Blitzer, and Lawrence Saul. Distance metric learning for large margin nearest neighbor classification. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1473–1480. MIT Press, Cambridge, MA, 2006.

Kilian Q. Weinberger and Lawrence K. Saul. Unsupervised learning of image manifolds by semidefinite programming. *International Journal of Computer Vision*, 70:77–90, 2006.

Kilian Q. Weinberger, Fei Sha, Qihui Zhu, and Lawrence K. Saul. Graph Laplacian regularization for large-scale semidefinite programming. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 1489–1496. MIT Press, Cambridge, MA, 2007.

Shifeng Weng, Changshui Zhang, and Zhonglin Lin. Exploring the structure of supervised data by Discriminant Isometric Mapping. *Pattern Recognition*, 38:599–601, 2005.

Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, and Stuart Russell. Distance metric learning, with application to clustering with side information. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*. MIT Press, Cambridge, MA, 2003.