SUPERVISED NONLINEAR DIMENSIONALITY REDUCTION BY NEIGHBOR RETRIEVAL

Jaakko Peltonen, Helena Aidos, Samuel Kaski

Helsinki University of Technology, Department of Information and Computer Science
P.O. Box 5400, FI-02015 TKK, Finland
{jaakko.peltonen, helena.aidos, samuel.kaski}@tkk.fi

ABSTRACT

Many recent works have combined two machine learning topics, learning of supervised distance metrics and manifold embedding methods, into supervised nonlinear dimensionality reduction methods. We show that a combination of an early metric learning method and a recent unsupervised dimensionality reduction method empirically outperforms previous methods. In our method, the Riemannian distance metric measures local change of class distributions, and the dimensionality reduction method makes a rigorous tradeoff between precision and recall in retrieving similar data points based on the reduced-dimensional display. The resulting supervised visualizations are good for finding (sets of) similar data samples that have similar class distributions.

***Index Terms*—** dimensionality reduction, information retrieval, metric learning, supervised manifold embedding

1. INTRODUCTION

Dimensionality reduction for visualizing high-dimensional data is central in exploratory data analysis. It is also used for preprocessing in pattern recognition and data analysis. We consider setups where there are known class labels available for some of the data points; the labels can come from any *auxiliary information* of the data, such as category labels or ontologies. Given the labeled data, *supervised* dimensionality reduction is used to reveal between-class relationships. The labels supervise the dimensionality reduction by telling which kind of variation in data is worth analyzing, as opposed to noise or irrelevant trends. The idea is that variation in the input features is worth preserving in the dimensionality reduction if it corresponds to changes in the class labels.

A typical approach for supervised nonlinear embedding is to modify input-space distances to take into account class labels of individual data points (see e.g. [2]); a similar idea appears in [7]. Such discriminative distances cannot be rigorously computed without knowing the class labels of the points, which can yield poor generalization to new, unlabeled points. Alternatively, some methods add a penalty term to the cost function favoring embeddings with small within-class distances [1]; such methods must manage a tradeoff between class discrimination and trustworthiness of the visualizations. Another alternative is to make kernel versions of supervised linear dimensionality methods (e.g. [8]); complicated kernels can make the mappings hard to interpret, though.

We introduce a method, called Supervised Neighbor Retrieval Visualizer (SNeRV), that supervises the input-space distance metric and then reduces the dimensionality. Our method has two simple and unique properties. First, we derive a local, topology-preserving Riemannian metric through conditional density estimation [6]. Second, we apply the metric to a recent visualization method, the Neighbor Retrieval Visualizer (NeRV; [10]), with a unique justification: it optimizes *information retrieval performance* of the visualization.

SNeRV is a supervised version of NeRV, which formalized visualization as a form of information retrieval: similar samples in the input space are retrieved based on similarity in the output space (the visualization), and a good visualization method must manage a tradeoff between precision and recall of such retrieval. SNeRV inherits this interpretation. Under certain parameter settings SNeRV can be seen as a new, supervised version of Stochastic Neighbor Embedding (SNE; [3]), but more generally it manages a flexible tradeoff between precision and recall of the information retrieval. Moreover, unlike some supervised methods [2, 4, 9], SNeRV can directly embed unlabeled training points as well as labeled ones.

2. THE METHOD

We now present the Supervised Neighbor Retrieval Visualizer (SNeRV). It computes input-space distances in a supervised metric learned from the data, and then plugs them into the embedding algorithm of unsupervised NeRV. The advantage of this approach is that either step can be changed if desired.

2.1. Computing the Input-Space Distances

SNeRV computes input-space distances using *learning metrics* (see [6]). The learning metric is a Riemannian metric:

The authors belong to Helsinki Institute for Information Technology HIIT and the Adaptive Informatics Research Centre. This work was supported by the Academy of Finland, decision number 123983, by the Portuguese Foundation for Science and Technology, scholarship number SFRH/BD/39642/2007, and in part by the PASCAL2 Network of Excellence.

it has a simple definition between close-by points, which is extended through integrals to yield global distances.

In the learning metric, squared distance between two close-by points $\mathbf{x}_1$ and $\mathbf{x}_2$ is defined as $d_L(\mathbf{x}_1, \mathbf{x}_2)^2 = (\mathbf{x}_1 - \mathbf{x}_2)^T \mathbf{J}(\mathbf{x}_1)(\mathbf{x}_1 - \mathbf{x}_2)$. Here $\mathbf{J}(\mathbf{x})$ is a Fisher information matrix which represents local dependency of the conditional class distribution on the input features, that is, $\mathbf{J}(\mathbf{x}) = \sum_c p(c|\mathbf{x}) \left(\frac{\partial}{\partial \mathbf{x}} \log p(c|\mathbf{x})\right) \left(\frac{\partial}{\partial \mathbf{x}} \log p(c|\mathbf{x})\right)^T$ where the c are classes and the $p(c|\mathbf{x})$ are conditional class probabilities at point $\mathbf{x}$. The local distances grow the fastest along directions where class probabilities $p(c|\mathbf{x})$ change the most. It can be shown that for close-by points $d_L(\mathbf{x}_1, \mathbf{x}_2)^2$ is equivalent to the Kullback-Leibler divergence $D_{KL}(p(c|\mathbf{x}_1)||p(c|\mathbf{x}_2))$.

The 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 $\mathbf{x}_2$. In a Riemannian metric, the straight path may not yield the minimum distance: intuitively, it can be more efficient to walk around sticky mud than across it.

Learning metric distances d_L are rigorous: they are nonnegative and symmetric, and satisfy the triangle inequality. The distances (minimal path integrals) preserve the topology of the input space: if the distance between two points is small, there must be a path between them where distances are small.

Practical computation. To compute local distances through the Fisher information matrices $\mathbf{J}(\mathbf{x})$, we estimate the conditional probabilities $p(c|\mathbf{x})$ by optimizing a discriminative mixture of labeled Gaussian densities for the data [6]: this yields the estimate $\hat{p}(c|\mathbf{x}) = \frac{\sum_{k=1}^K \beta_{ck} \exp(-||\mathbf{x}-\mathbf{m}_k||^2/2\sigma^2)}{\sum_{k=1}^K \exp(-||\mathbf{x}-\mathbf{m}_k||^2/2\sigma^2)}$ 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 3).

We next compute global distances as minimal path integrals of local distances, using a graph based approximation [6]. We connect all points in our graph: a connection between two points denotes a straight path and the distance along it is computed by piecewise approximation (see [6]; we use $T = 10$ pieces). We could then find shortest paths between all points by graph search (Floyd's algorithm), and use the shortest path distances as the learning metric distances. With N points, graph search would take $O(N^3)$ time; note that similar graph computation is needed in methods like Isomap. However, in experiments the straight paths yielded about equally good results so we use them, which only takes $O(N^2)$ time.

2.2. Computing the Nonlinear Embedding

The nonlinear embedding in SNeRV is done by the same algorithm as in unsupervised NeRV; that algorithm outperformed many unsupervised methods in [10] and it has an information retrieval interpretation as we will discuss. The algorithm tries to make the neighborhood of each point in the visualization be similar to the corresponding neighborhood in the original input space. This is formalized in the cost function of SNeRV:

$$E_{SNeRV} = \lambda \sum_{i,j \neq i} p_{ij} \log \frac{p_{ij}}{q_{ij}} + (1-\lambda) \sum_{i,j \neq i} q_{ij} \log \frac{q_{ij}}{p_{ij}} \ . \quad (1)$$

The cost function (1) is a sum of two kinds of Kullback-Leibler divergences, which measure the difference between input-space neighborhoods and output-space neighborhoods.

Here the p_{ij} define the input-space neighborhoods; they are normalized probabilities for points j being neighbors of point i based on their input-space locations. That is, the probabilities are computed based on learning metric distances in the input space: $p_{ij} = \frac{\exp(-d_L(\mathbf{x}_i,\mathbf{x}_j)^2/\sigma_i^2)}{\sum_{k \neq i} \exp(-d_L(\mathbf{x}_i,\mathbf{x}_k)^2/\sigma_i^2)}$. Similarly, the q_{ij} define the output-space neighborhoods; they are normalized probabilities for points j being neighbors of point i which are computed based on Euclidean distances in the output space (the visualization): $q_{ij} = \frac{\exp(-||\mathbf{y}_i-\mathbf{y}_j||^2/\sigma_i^2)}{\sum_{k \neq i} \exp(-||\mathbf{y}_i-\mathbf{y}_k||^2/\sigma_i^2)}$. The smoothness parameter σ_i is chosen separately for each point by fixing the entropy of the p_{ij} distribution to be the same for each point, which corresponds to choosing a fixed "effective number of neighbors"; see [10] for details. In the experiments we chose the "effective number of 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.

SNeRV minimizes E_{SNeRV} with respect to the output coordinates $\mathbf{y}_i$ of each point, by a conjugate gradient algorithm; see [10] for details. The time complexity of that algorithm is $O(N^2)$ per gradient step (Venna et al., in preparation).

In E_{SNeRV}, the parameter λ controls the tradeoff between two kinds of Kullback-Leibler divergences, which has an *information retrieval interpretation*: it is shown in [10] that unsupervised NeRV optimizes performance in an information retrieval task where input-space neighbors of a point are retrieved based on its neighbors in the visualization. When $\lambda = 0$ NeRV maximizes (smoothed) precision of the retrieval task and when $\lambda = 1$ it maximizes (smoothed) recall; more generally it optimizes a tradeoff between precision and recall. Our supervised method SNeRV has the same interpretation, where precision and recall are now computed with respect to the supervised input-space metric. In experiments we use two intermediate values for the tradeoff: $\lambda = 0.1$ and $\lambda = 0.3$.

When $\lambda = 1$, the unsupervised NeRV is equivalent to Stochastic Neighbor Embedding (SNE; [3]); thus SNeRV with $\lambda = 1$ can be seen as a supervised version of SNE.

3. EXPERIMENTS

We compare SNeRV to three recent supervised nonlinear embedding methods: *Multiple Relational Embedding* (MRE; [5]) was proposed as an extension of Stochastic Neighbor

Embedding [3]. MRE minimizes a sum of mismatches between neighborhoods in the embedding and several input neighborhoods: typically one input neighborhood is derived from input-space coordinates and others from auxiliary information like labels. *Colored Maximum Variance Unfolding* (MUHSIC; [9]) is a supervised extension of Maximum Variance Unfolding. *Supervised Isomap* (S-Isomap; [2]) is one of several supervised extensions of Isomap; it uses a new definition of input-space distances where they grow faster between different-class points than between same-class points.

We use the methods to find 2-dimensional visualizations for four benchmark data sets. The *Letter recognition* and *Landsat satellite* data sets (denoted Letter and Landsat) are from the UCI Machine Learning Repository; Letter ($D = 16$ dimensions, $C = 26$ classes) contains images of different capital letters and Landsat ($D = 36$, $C = 6$) contains satellite images of different terrain types. The *Phoneme* data set ($D = 20$, $C = 13$) is from the LVQ-PAK program package and contains samples of different phonemes. The *TIMIT* data set ($D = 12$, $C = 41$) is from a CD-ROM prototype version of the DARPA TIMIT speech database; it is similar to the Phoneme data. For all data sets we used a randomly chosen subset of 1500 samples, to save computation time.

We compare supervised embedding methods; unsupervised ones often cannot find class-discriminative embeddings. Fig. 1 shows example embeddings by the supervised methods and by the unsupervised NeRV (which outperformed several unsupervised methods in [10]). The NeRV embedding has much class overlap near the center and is clearly less informative about classes than SNeRV. See the references for other comparisons of supervised and unsupervised embedding.

Methodology. We use 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.

We evaluate the performance of the four methods by class prediction accuracy of the resulting embeddings. We provide test point locations during training but not their labels; after the methods have made their embeddings, we classify the test points by running a k-NN 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 in the methods 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 embeddings with the different parameter values; the values that yielded best classification accuracy for the embedded validation points are then chosen and used to compute the final embedding for the whole training data.

We ran two versions of SNeRV using $\lambda = 0.1$ and $\lambda = 0.3$. 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 validated its parameter α and its number of nearest neighbors, and trained a generalized radial basis function network to project new points, as suggested by the authors of [2]. For MUHSIC we validated its 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 we validated its neighborhood smoothness parameter σ_{MRE}.

Table 1. Statistical significance of the difference between the two best methods. The p-values are from a paired t-test of 10-fold cross-validation results; when the result is statistically significant the winner is shown in bold. Numbers after SNeRV denote values of the λ parameter.

Data set	Best method	Second best	p-value
Letter	**SNeRV 0.1**	S-Isomap	$1.5 \cdot 10^{-4}$
Phoneme	S-Isomap	SNeRV 0.3	0.54
Landsat	SNeRV 0.3	S-Isomap	0.53
TIMIT	**SNeRV 0.1**	MUHSIC	$2.5 \cdot 10^{-6}$

Results. The results are shown in the top left subfigure of Fig. 1. We present the average error rate over the 10 folds and the standard deviation. The best two methods are SNeRV and S-Isomap. On Letter and TIMIT data SNeRV is clearly best; on Phoneme and Landsat SNeRV and S-Isomap are about equally good. MRE is clearly worse than others, whereas MUHSIC results depend on the data. For SNeRV, both values $\lambda = 0.1$ and $\lambda = 0.3$ of the tradeoff parameter yielded good embeddings. To test whether the best method on each data set is statistically significantly better than the next best one, we did a paired t-test across the 10 cross-validation folds. For SNeRV we use the version (λ value) which gave the better error rate in top left of Fig. 1. The results are shown in Table 1: on Letter and TIMIT data, SNeRV is significantly better than the other method; for the other two data the difference is not significant. All significant differences are in favor of SNeRV.

Fig. 1 shows sample embeddings of the Letter data. SNeRV shows distinct clusters of many classes. In S-Isomap a few classes like "W" and "N" are well separated but there is much overlap at center right. MUHSIC yielded severe class overlap on this data. In MRE, classes are well separated for training points, but test points are scattered loosely around training points yielding bad classification accuracy. Embedding by unsupervised NeRV is shown as a baseline; as discussed before, it has clearly worse overlap than SNeRV.

4. CONCLUSIONS AND DISCUSSION

We introduced the Supervised Neighbor Retrieval Visualizer (SNeRV), which uses class labels of points to learn a supervised nonlinear embedding of the data set. The key idea is that we learn a topology-preserving, class-discriminative *learning metric*, and distances in this metric are plugged into the em-

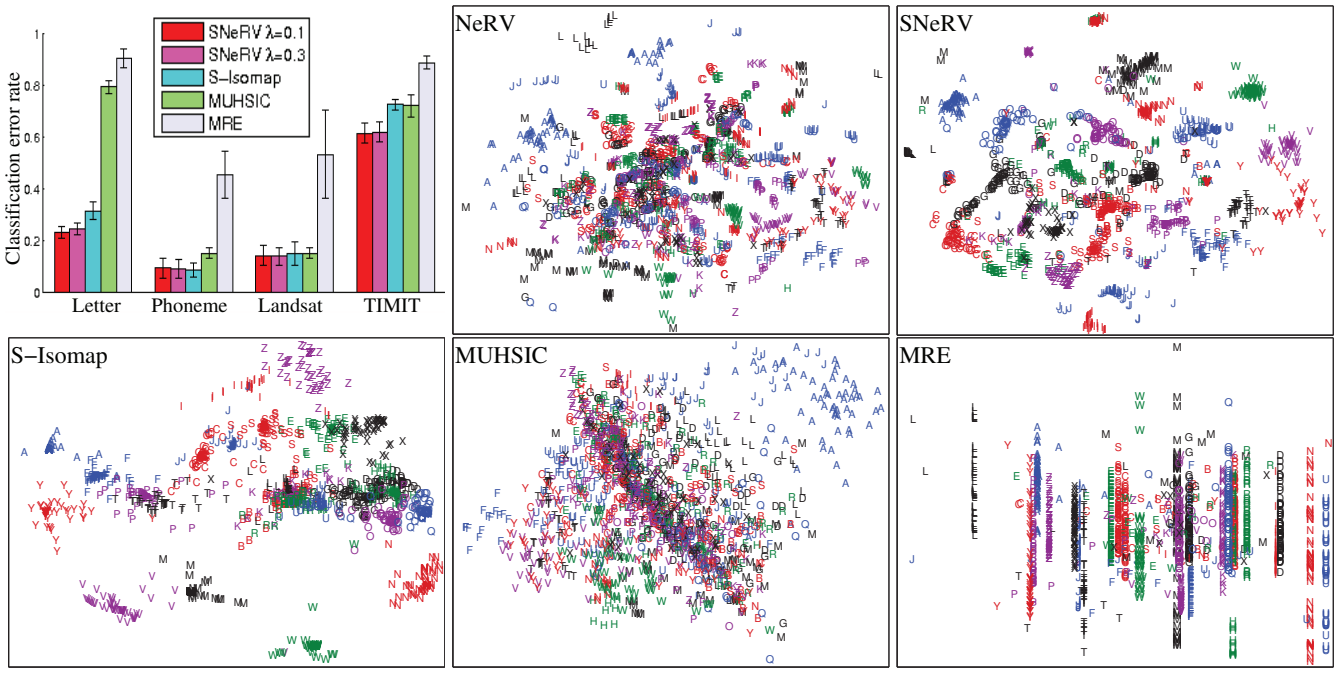

Fig. 1. Top left: performances of the supervised nonlinear embedding methods in each data set. Results are average classification error rates over 10 cross-validation folds (smaller is better); standard deviations are shown with error bars. The other subfigures show example embeddings of the Letter recognition data set by all supervised methods; an embedding by the unsupervised NeRV is shown as a baseline. Overall, SNeRV yields the best embeddings.

bedding algorithm from [10]. In the benchmark experiments SNeRV performed as well as or better than the best alternative method S-Isomap. Moreover, SNeRV allows immediate embedding for unlabeled points as well.

Learning metric distances could be plugged into other embedding algorithms that use a distance matrix; we made an experiment with Sammon's mapping in [6]; a similar idea for Isomap appeared later in [11]. NeRV is a good choice for the embedding step, though; it has an information retrieval interpretation and led to good performance.

5. REFERENCES

[1] J. A. Costa and A. O. Hero III, "Classification constrained dimensionality reduction," in *Proc. ICASSP'05*. IEEE, 2005, vol. 5, pp. 1077–1080.

[2] X. Geng, D.-C. Zhan, and Z.-H. Zhou, "Supervised nonlinear dimensionality reduction for visualization and classification," *IEEE Trans. Syst. Man Cybern. Part B: Cybern.*, vol. 35, pp. 1098–1107, 2005.

[3] G. Hinton and S. T. Roweis, "Stochastic neighbor embedding," in *Proc. NIPS 2002*. MIT Press, 2003, pp. 833–840.

[4] N. Liu, F. Bai, J. Yan, B. Zhang, Z. Chen, and W.-Y. Ma, "Supervised semi-definite embedding for email data cleaning and visualization," in *Proc. APWeb 2005*, pp. 972–982. Springer, 2005.

[5] R. Memisevic and G. Hinton, "Multiple relational embedding," in *Proc. NIPS 2004*. MIT Press, 2005, pp. 913–920.

[6] J. Peltonen, A. Klami, and S. Kaski, "Improved learning of Riemannian metrics for exploratory analysis," *Neural Networks*, vol. 17, pp. 1087–1100, 2004.

[7] D. de Ridder, M. Loog, and M. J. T. Reinders, "Local Fisher embedding," in *Proc. ICPR'04*. IEEE, 2004, vol. 2, pp. 295–298.

[8] V. Roth and V. Steinhage, "Nonlinear discriminant analysis using kernel functions," in *Proc. NIPS 1999*. MIT Press, 2000, pp. 568–574.

[9] L. Song, A. Smola, K. Borgwardt, and A. Gretton, "Colored maximum variance unfolding," in *Proc. NIPS 2007*. MIT Press, 2008, pp. 1385–1392.

[10] J. Venna and S. Kaski, "Nonlinear dimensionality reduction as information retrieval," in *Proc. AISTATS*07, San Juan, Puerto Rico, March 21-24*, 2007.

[11] S. Weng, C. Zhang, and Z. Lin, "Exploring the structure of supervised data by Discriminant Isometric Mapping," *Pattern Recognit.*, vol. 38, pp. 599–601, 2005.