Learning Topology of Curves with Application to Clustering

Hossein Mobahi [†‡] and **Shankar R. Rao** [‡] and **Yi Ma** [‡]
† Department of Computer Science
‡ Coordinated Science Laboratory
University of Illinois at Urbana Champaign, Urbana, IL 61801

Abstract

We propose a method for learning the intrinsic topology of a point set sampled from a curve embedded in a high-dimensional ambient space. Our approach does not rely on distances in the ambient space, and thus can recover the topology of sparsely sampled curves, a situation where extant manifold learning methods are expected to fail. We formulate a loss function based on the smoothness of a curve, and derive a greedy procedure for minimizing this loss function. We compare the efficacy of our approach with representative manifold learning and hierarchical clustering methods on both real and synthetic data.

Introduction

A fundamental task in machine learning is to learn the underlying structure of unorganized data. A common assumption is that the data lies on a low-dimensional *manifold* embedded in the high-dimensional ambient space. The structure of the manifold can differ significantly from the structure of the ambient space, so that data samples that are "close" in the ambient space are not necessarily "close" on the underlying manifold. Thus a key step in manifold learning is to infer the *intrinsic topology* of the embedded manifold from the data. That is, given a set of unstructured data samples believed to lie on a low-dimensional manifold, one seeks to infer a neighborhood relationship among samples that is independent of distances in the ambient space. Such a task is inherently ill-posed, as there are many possible ways such connectivities can be realized.

A simple idea is to build the intrinsic topology upon local neighborhood in the Euclidean space. For example, every point may be connected to its neighbor points determined by K-Nearest-Neighbors (KNN) or ϵ-ball. This idea is motivated from the definition of a smooth manifold; tiny regions on a manifold are topologically equivalent to patches of a Euclidean space.

Due to its simplicity and efficiency, a majority of manifold learning methods rely on KNN or ϵ-balls for learning the local topology, including Isomap (Tenenbaum et al., 2000), Local Linear Embedding (LLE) (Roweis & Saul, 2000), Laplacian Eigenmaps (Belkin & Niyogi, 2003), Semidefinite Embedding (Weinberger & Saul, 2005). Similarly, Principal Curves method (Hastie & Stuetzle, 1989) relies a neighborhood span for local averaging.

These methods tend to work well if the manifold is densely sampled. However, in many circumstances, obtaining large dense sample sets may be expensive or impossible. To cope with the lack of samples, a natural option is to involve *all* the samples collectively by fitting a global model to the data. To prevent a global model from overfitting the data, one needs a notion of model simplicity. In our context, the notion of simplicity will be the *smoothness* of the manifold.

A general framework for achieving this goal is to consider a parameterization of the manifold and then seek a solution whose parameterization is itself a smooth function. Along this path, (Smola et al., 2001) explores kernel methods for obtaining parametric representation of manifolds. For kernel-based methods, the definition of a smooth manifold varies depending on what kernel is chosen. In addition, it is not clear how smooth parameters in kernel-based representations are related to the actual smoothness of the manifold.

There has been some recent effort that integrate notions of global smoothness into manifold learning such as (Dollár et al., 2007) and (Bengio et al., 2006). However these methods still use a KNN or ϵ-balls as an initialization. Finding the intrinsic topology of a point set that maximizes smoothness *without* resorting to these local cues is a hard problem in general. In this work we take a first step toward this goal, by studying single 1-dimensional manifolds, aka curves. Learning curves, e.g. obtaining principal curves (Hastie & Stuetzle, 1989), and its applications to machine learning and pattern recognition have been addressed in the literature (Chang & Ghosh, 1998). This work seeks to learn the intrinsic topology without using *any* local information from a *sparse* set of samples. We formulate the problem as an optimization task and propose a simple and suitable greedy search for solving it. Our formulation is guaranteed to converge to a solution that is locally optimal in the solution space in a finite number of steps. As we will show, our method only requires a single parameter (the choice of ℓ^q-norm), and empirically, we have found that $q = 0.1$ works well for all of our experiments.

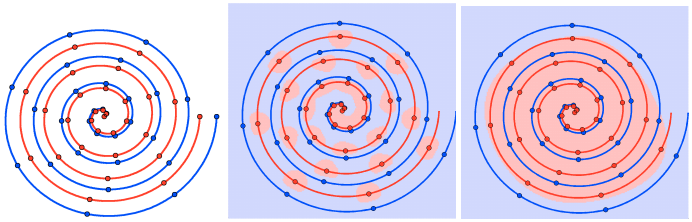

Figure 1: Left: Two Spiral Problem. Middle: SVM + RBF Kernel. Right: SVM + Polynomial Kernel

We evaluate the performance of our method against Isomap[1], as a representative manifold learning from a local neighborhood as well as typical hierarchical clustering algorithms. We use both synthetic and real data in our tests.

From Neighbors to Smoothness

In this section we provide an intuitive motivation for our approach using a simple example.

In the late 80s, Alexis Wieland posed the *two spirals problem* as a challenge to the artificial neural networks community. The goal was to learn models for two concentric and intertwined spirals using training samples (originally 97 per spiral), so that unlabeled test samples from either spiral would be correctly classified. The density of the samples increased moving inward along the spirals. Due to the non-linearity and uneven distribution of the samples, it was a very difficult problem for error backpropagation and relative approaches (Touretzky & Pomerleau, 1989). For many years, this problem was a benchmark for pattern recognition.

Many modern classification methods achieve outstanding performance on this problem. Specifically, *support vector machines* (SVM) using the *radial basis function* (RBF) kernel discovers the perfect decision boundary as illustrated in (Cristianini, 2001). However, this performance significantly deteriorates when the sample size is reduced. For example, consider the instance of the problem shown in Figure 1 with only 30 samples per spiral. The SVM results shown in Figure 1 are the best that it could achieve by a grid search for the parameters with the best generalization performance [2].

This observation is not very surprising as these kernels are well suited for *spherical-shaped distributions* while here each spiral is a thin curve. Thus, manifold learning seems to be a more promising avenue for learning such structure of more sparsely sampled datasets.

We test the efficiency of Isomap to model each spiral using the reduced dataset. The set of points for each spiral were input to Isomap separately. The intrinsic dimensionality of the manifold was set to one and then the projection of the points onto this 1-dimensional space were ordered. This order was used to connected the points in the original space. The results for neighborhoods $k = 1$ and $k = 2$ are shown in Figure 2. As these results demonstrate, the local nature of Isomap prevents it from inferring the global structure of the manifold from a more sparse set of samples [3].

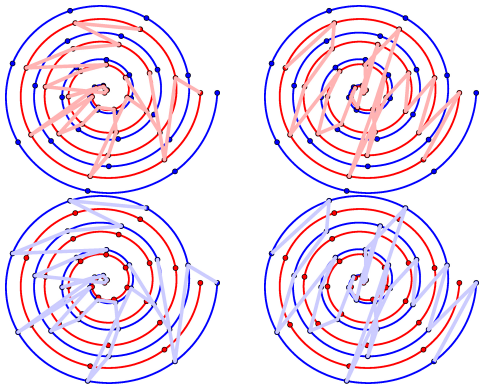

Figure 2: Red and Blue points ordered by Isomap. Left: k=1 , Right: k=2.

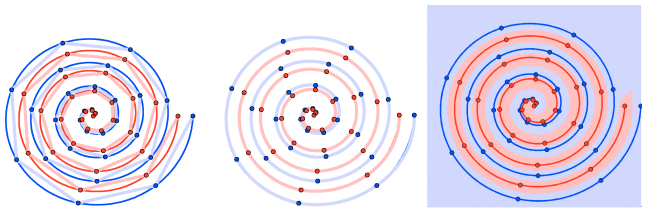

Figure 3: Left: Our method applied to 2 Spiral data. Middle: Result smoothed by cubic spline. Right: Decision region by nearest manifold

Now we apply the idea that we propose in this paper to this problem. The result is shown in Figure 3. The method will be explained in details in the next section. Briefly it tries to find an arrangement of points such that sharp angles are avoided (compare the angles in this figure with those of Isomap in figure 2). Similar to Isomap case, we fed red and blue points separately and obtained the corresponding curves. Once the order of the points along the curve is known, one can resort to spline methods for approximating the actual curve. Here we applied a cubic spline and it perfectly coincides with the underlying curves. Finally, the decision regions are shown by assigning each point on the plane to the nearest (splined) curve.

Methodology

In this section we discuss our approach in detail. We first formulate the problem as minimization of a loss function, propose a greedy approach to the optimization, and then summarize the method as an algorithm.

[1] Isomap assumes that the manifold is isometric to a convex subset of Euclidean space. However, because we only consider singly connected curves in this paper, this assumption is reasonable.

[2] For RBF kernel $\sigma = 1.2255$ and $C = 12.4637$ and for the polynomial kernel, degree is 4 and $C = 180.2722$.

[3] We also applied Isomap for neighborhoods $k = 3...29$, but did not observe any improvement.

Problem Formulation

Suppose we are given a collection of points $S = \cup_i \boldsymbol{x}_i \ , \ i = 1, ...n$ in a vector space $\mathcal{X}$ equipped with an inner product, so that angles between any pair of vectors is defined. The goal is to obtain a permutation of the points $P^* = (\tilde{\boldsymbol{x}_1}, \tilde{\boldsymbol{x}_2}, ..., \tilde{\boldsymbol{x}_n})$ such that the (piecewise linear) curve obtained by connecting the points in the order induced by P^* exhibits no sharp angle. More precisely, $P^* = \underset{P \in \mathcal{P}}{\operatorname{argmin}} \, L(P)$, where $\mathcal{P}$ is the space of all permutations of S and L is an appropriate loss function. We detail one such loss function below.

We first form a vector $\boldsymbol{\Theta}$, where each entry θ_i is a function of the angle formed at the junction of consecutive line segments. Since each angle relies on three consecutive points, there are $n-2$ components in $\boldsymbol{\Theta}$, obtained by traversing the path $(\tilde{\boldsymbol{x}_1}, \tilde{\boldsymbol{x}_2}, ..., \tilde{\boldsymbol{x}_n})$. For three consecutive points $\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_{i+1}$ and, $\tilde{\boldsymbol{x}}_{i+2}$, the trajectory along the path is most smooth when $\angle \tilde{\boldsymbol{x}}_i \tilde{\boldsymbol{x}}_{i+1} \tilde{\boldsymbol{x}}_{i+2}$ is most *obtuse* and least smooth when $\angle \tilde{\boldsymbol{x}}_i \tilde{\boldsymbol{x}}_{i+1} \tilde{\boldsymbol{x}}_{i+2}$ is most *acute*. One such function that possesses these desired properties is:

$$\begin{aligned} \theta_i &= 1 + \cos \angle \tilde{\boldsymbol{x}}_i \tilde{\boldsymbol{x}}_{i+1} \tilde{\boldsymbol{x}}_{i+2} \\ &= 1 + \frac{(\tilde{\boldsymbol{x}}_{i+2} - \tilde{\boldsymbol{x}}_{i+1}) \bullet (\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{x}}_{i+1})}{\|\tilde{\boldsymbol{x}}_{i+2} - \tilde{\boldsymbol{x}}_{i+1}\| \, \|\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{x}}_{i+1}\|} \end{aligned} \quad (1)$$

We compute the loss function as the ℓ^q-norm of $\boldsymbol{\Theta}$:

$$L(P) = \|\boldsymbol{\Theta}\|_q^q \quad (2)$$

$$= \sum_{i=1}^{n-2} \left(1 + \frac{(\tilde{\boldsymbol{x}}_{i+2} - \tilde{\boldsymbol{x}}_{i+1}) \bullet (\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{x}}_{i+1})}{\|\tilde{\boldsymbol{x}}_{i+2} - \tilde{\boldsymbol{x}}_{i+1}\| \|\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{x}}_{i+1}\|} \right)^q \quad (3)$$

Different choices of q affect the performance of the algorithm. We have found that small q, i.e. sparsity-inducing, tend to give the best results.

Greedy Optimization

Finding a global minimum of (3) over all permutations of S is a combinatorial optimization. We adopt a greedy local search scheme that attains a local minimum. The idea is to start from a random solution and iteratively improve it. In each iteration, the current solution is perturbed in its *neighborhood* in the solution space to produce a number of candidate solutions. The candidate which reduces the loss function the most replaces the current solution. The optimization terminates when there is no candidate able to reduce the loss.

One may look at $\mathcal{P}$ from a graph theoretic perspective where any $P \in \mathcal{P}$ is a Hamiltonian path in an undirected graph with n vertices. Here n vertices correspond to each of the n points in S. Remember that a Hamiltonian path visits each vertex exactly once.

This view leads us to a repository of methods developed for computing optimal path in a graph. For example, a simple method called "2-opt", obtains a neighbor solution by eliminating two edges and reconnecting the two resulting paths in a different way to create a new path. There is only one way to reconnect the paths that yield a valid Hamiltonian path. Therefore, the number of neighbor solutions is equal to the number of choosing two edges out of n edges. The 2-opt technique was proposed for solving the Traveling Salesman Problem (TSP) and has probably been the most widely used heuristic for this problem (Bentley, 1992). We choose 2-opt neighborhood for our search due to its simplicity and efficiency against other graph based techniques.

Algorithm 1 Smooth Curve Estimation via 2-opt

Input: Initial solution P_0
Initialize $P = P_0$
Initialize $\mathcal{L} = L(P)$
repeat
  $\mathcal{L}^{'} = \infty$
  for $i = 1$ **to** $|P|$ **do**
    for $j = i + 1$ **to** $|P|$ **do**
      $\tilde{P} = P$
      Reverse subsequence $p_i ... p_j$ in $\tilde{P}$
      if $L(\tilde{P}) < \mathcal{L}^{'}$ **then**
        $\mathcal{L}^{'} = L(\tilde{P})$
        $P^{'} = \tilde{P}$
      end if
    end for
  end for
  $P^* = P$; $\mathcal{L}^* = \mathcal{L}$
  $P = P^{'}$; $\mathcal{L} = \mathcal{L}^{'}$
until $\mathcal{L} \geq \mathcal{L}^{'}$
Output: Optimal permutation P^*

Algorithm

The procedure of finding a smooth ordering of a point set S is shown in Algorithm 1. It starts from a given initial solution P_0, possibly random, and iteratively creates 2-opt neighbors and improves upon. When using random initialization, adopting multiple start points is recommended. One can then pick the solution that attains the lowest cost.

The algorithm is guaranteed to converge because it reduces the loss function at each step and because the loss function is bounded from below by zero. In addition, algorithm terminates in finite steps due to the finiteness of the permutation space. The complexity of each iteration is $O(|S|^2)$, assuming the loss function is precomputed in a table.

Experiments

We test the performance of our method in a practically useful task for data clustering. We assume that the data of each cluster lie on a smooth curve $\mathcal{C}_i \ , \ i = 1, ...c$, aka a 1-dimensional manifold (c is the number of clusters). Since we do not know in advance which point belongs to which cluster, we fit a single smooth curve $\mathcal{C}$ to the whole data. Our hope is that this single curve emerges from gluing together the actual clusters, i.e. $\forall i \ \ \mathcal{C} \cap \mathcal{C}_i = \mathcal{C}_i$. This way all of the points from the same $\mathcal{C}_i$ are clustered along the curve $\mathcal{C}$.

Once such $\mathcal{C}$ is identified, it should be cut at the transition points to recover the clusters $\mathcal{C}_i$. One way to identify the transition points (a curve segment connecting two clusters) is assuming that the transition among the pieces $\mathcal{C}_i$ on $\mathcal{C}$ happens sharply. This way one can traverse along $\mathcal{C}$ and associate any sharp point as a cutoff point. These points are detected as the c most acute angles along the curve. Note that when there is a sharp turn due to a transition, two adjacent angles might be affected. Therefore, when extracting the c most acute angles, we constrain them to ignore the less acute of two adjacent angles. Now each of these c sharp angles identifies a transition point, from one of two consecutive clusters on the curve.

In the following, we investigate this idea in two different tasks. One task involves synthetic images of three animals undergone drastic illumination changes. The goal is to cluster the images based on the type of the animal. For the second task, we use objects in the Columbia Object Image Library (COIL-100) dataset, that consists of color images of 100 objects (Nene et al., 1996). This set provides different poses of various objects. The goal is to cluster all instances of the same object into the same group regardless of their pose.

We compare our results against Isomap, as well as typical clustering algorithms. For clustering, we adopt a hierarchical framework instantiated with a large variety of possibilities. We consider five "point distance metrics", namely, Euclidean, standardized Euclidean, cityblock, one minus cosine of the angle, and one minus correlation. We also consider six linkages (distance between two clusters), namely nearest distance (single), furthest distance (complete), average distance, centroid distance, median distance and ward distances. Specifically we try all 30 combinations of clustering methods from these ingredients.

We utilize the *kappa index* to compare the quality of a clustering C with respect to the ground truth G. The kappa index measures the proportional difference between positive acceptance and the expected random agreement between partitions. It constructs the $c \times c$ contingency matrix of C, G where c is the number of clusters. It first permutes rows (or columns) such that the trace of the matrix is maximal. Denoting the contingency matrix by A and $|C| = |G| = n$, the kappa index is computed as follows:

$$\kappa(C, G) = \frac{n\ Tr(A) - \sum_{i=1}^{c} \sum_{j=1}^{c} a_{ij} a_{ji}}{n^2 - \sum_{i=1}^{c} \sum_{j=1}^{c} a_{ij} a_{ji}} \qquad (4)$$

Illumination experiment

We use 3D models of three animals: a bunny, a horse, and a dragon. The bunny and dragon were obtained from Stanford 3D scanning repository (Turk & Levoy, 1994) and the horse model is courtesy of Cyberware Inc [4]. The animals where placed in a synthetic room with a light source on top. Some representative images are shown in Figure (4). The environment and lighting were handled by "Pane", a publicly available ray tracer (Banks & Beason, 2007). We took screen shots while moving the light source from left to right.

[4] www.cyberware.com

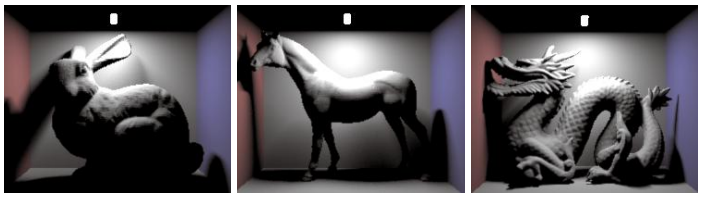

Figure 4: Synthesized animals and lighting

This way, the images of each animal would lie along a 1-d manifold in the image space. In total, five images were generated per animal, corresponding to the left, right, middle and two in between situations, as shown in Figure (7).

The low sample size makes learning the topology of the manifolds very challenging for local methods such as KNN. From a distance, the images with the same lighting conditions look more similar to each other albeit containing different animals. Thus, local methods would choose similar lighting conditions as their neighbors along the manifold. Each input image is 246×200 pixels of RGB channels. No special processing was involved; pixel values in each image were stacked to form a 147600 dimensional feature vector.

Among all 30 combinations of typical clustering methods, city block distance plus single linkage obtained the best $\kappa = 0.4$. The resulted clustering is shown in figure (5). We then applied our data on Isomap, setting the intrinsic dimensionality parameter to one. For no choice of k (number of neighbors) Isomap could detect the 3 connected components. For $k = 1$ it would detect 5 curves and for all other choices 1 curve. We investigated if the 1-component solutions were at least in a desired order. Therefore, for $k = 2, ..., 14$ that produce one connected component, we extracted three clusters from Isomap's solution. This was achieved by embedding the solution in 1-dimensional space (real line) and using their position on this line for ordering them. This way, the best result gained $\kappa = 0.3$ for choice of $k = 2$. The next best choice was $k = 4$ which obtained $\kappa = 0.2$. All other choices of $k > 1$ resulted in poor performance $\kappa = 0.1$. Figure 6 displays ordering for the best case ($k = 2$). The figure is augmented by the angles of each three successive points. A lot of acute angles are apparent along the curve.

The proposed method could *easily* recover perfect clustering. Despite the greedy nature of the algorithm, running it from a few random initial configurations and then picking the one with the lowest cost could always yield a perfect solution. This can be supported by investigating the chance of not seeing a perfect solution in n different runs. This probability drops exponentially fast in n. We modeled the failure/success process by a binomial distribution as the result of each run is independent from the other. The initial solution was randomly chosen from uniform distribution in permutation space. We obtained an estimation of 0.158 for the probability of failure, by 1000 runs of the algorithm. Thus the probability of not seeing a perfect solution in n runs is approximately 0.158^n.

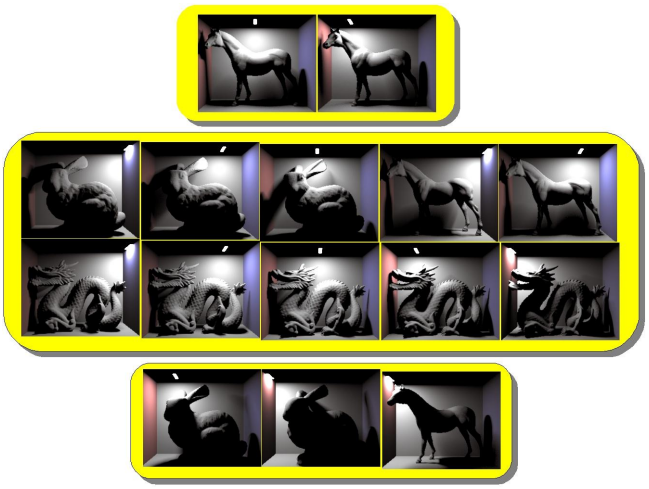

Figure 5: Hierarchical clustering by cityblock distance and single linkage

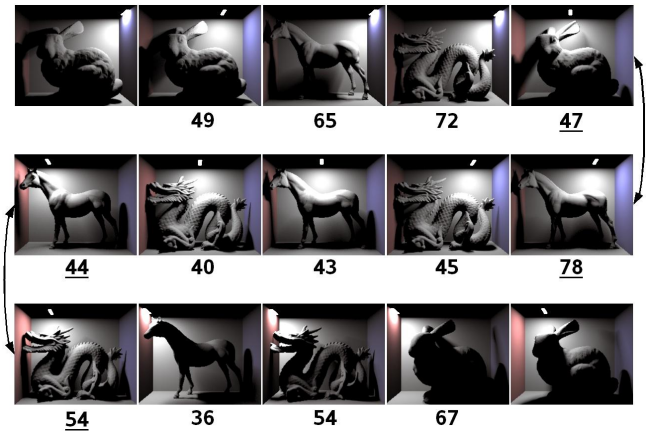

Figure 6: One dimensional embedding by Isomap (k=2)

Figure (7) displays a solution of the proposed method augmented with angles in between. Comparing to figure (6), it has much fewer sharp angles. In fact, acute angles only happen at the transition points. This indicates that one can identify the transition points merely by searching for acute angles without need of knowing any additional information.

Pose experiment

Dataset The goal in this experiment is to cluster real objects undergone pose transformation. This deals with real data from COIL-100. For this dataset, objects were placed on a turntable and against a black background. Images of the objects were taken at pose intervals of 5 degrees corresponding to 72 poses per object. The images were normalized in size. Since there is only one degree of freedom responsible for change in the appearance of an object, the resulting manifold is a curve in the image space.

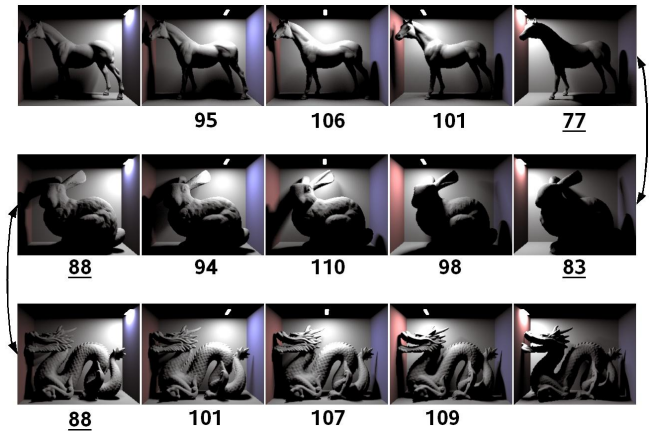

Figure 7: Recovered curve using the proposed method

We focus on two particular types of objects in COIL, namely boxes and cars. Each type has 10 different instances, as shown in figure (8). We selected these two types because it has been reported in the literature that they contain the most difficult objects for recognition purpose relative to the rest of objects in COIL. See (Pontil & Verri, 1998) or (Roth et al., 2002) for details about difficult objects.

We measure how well the methods can separate similar objects in the same category. For computational tractability[5], we define subtasks, where each subtask is combination of two objects in the same category. There are 45 subtasks for each category where each gains its own clustering score. We aggregate the performance of subtasks in different ways to obtain a performance index of the whole category.

To investigate the effect of sample size on the performance, we subsample available poses for each object. Specifically, we consider 5 cases where 36, 24, 14, 9 and 6 views per object are available. These are respectively obtained by starting from the first pose (zero angle) of each object and skipping every 2, 3, 5, 8, and 12 images respectively. The number of available views is denoted by m. In our experiments $m \in \{6, 9, 14, 24, 36\}$.

Preprocessing Each image is reduced to 30×30 and then convolved with a Gaussian kernel t times [6]. This Gaussian smoothing is a means of both reducing the noise and smoothing the underlying manifold. In the following experiments, we try $t \in \mathcal{T} = \{0, 5, 10, 15, 20, 25, 30, 35, 40\}$. Each of the RGB channels are convolved separately. Finally, each image is converted to a vector of dimension 270 by stacking up its RGB values at any pixel.

Parameters Each of these methods has its own tuning parameters. Hierarchical clustering depends on the combination of choices for linkage and distance functions, and

[5] Our current implementation is on Matlab and not optimized for speed.

[6] Multiple applications of a Gaussian kernel is equivalent to applying a single wider Gaussian.

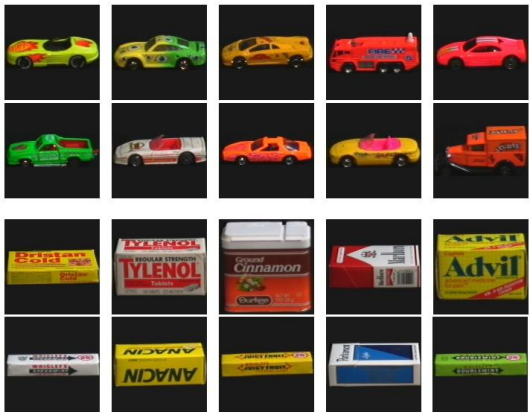

Figure 8: Two hardest categories used in our experiments.

Isomap depends on choice of neighborhood parameter k. Our method has only one parameter (the choice of ℓ^q-norm), which we empirically obtained $q = 0.1$ doing a good job across all the tasks we experimented. In addition, we consider t_h, t_i and t_p (the number of applications of the Gaussian convolution) as additional parameters of the hierarchical clustering, Isomap and our method.

Given a category, denote the kappa index of subtask i, j for given parameters θ, m, t by $\kappa_{\theta,m,t,i,j}$. Here i and j determine which pair of the 10 objects are considered in the subtask. θ denotes the parameters; linkage and distance when applied to hierarchical clustering, k for Isomap, and empty for our method. With this notation, we aggregate the performance of subtasks in two different ways:

$$\mathcal{J}_{m,t} = \max_{\theta} \frac{1}{\binom{n}{2}} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \kappa_{\theta,m,t,i,j} \tag{5}$$

$$\mathcal{K}_{m,t} = \frac{1}{\binom{n}{2}} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \max_{\theta} \kappa_{\theta,m,t,i,j} \tag{6}$$

Here n is the total number of objects in the category, i.e. 10. $\mathcal{J}$ uses a fixed best parameter vector over the entire subtasks while $\mathcal{K}$ picks the best parameter vector for each subtask independently. Of course, $\mathcal{J}$ is a more practical measure, but we also report results based on $\mathcal{K}$ as an extreme case. In order to simplify visualization of performance, we further reduce the degrees of freedom of the performance index by considering the best performance over all choices of t in the following sense:

$$\mathcal{J}_m = \max_{t \in \mathcal{T}} \mathcal{J}_{m,t} \qquad \mathcal{K}_m = \max_{t \in \mathcal{T}} \mathcal{K}_{m,t} \tag{7}$$

Results We assume the only given knowledge is the number of clusters within each subtask, which we denote by c (2 for our specific experiment). In this scenario, the hierarchical clustering stops when it reaches c clusters. For Isomap, a result is considered correct when the number of connected components matches with c. If it does, then the elements of each component are grouped to a cluster and the κ is computed accordingly. Otherwise, κ is set to zero. This should not hurt the performance measure of Isomap too much as long as there is at least one choice of k for which Isomap obtains the right c. This is because we only choose maximal κ in equation 5 among all choices of the parameter k.

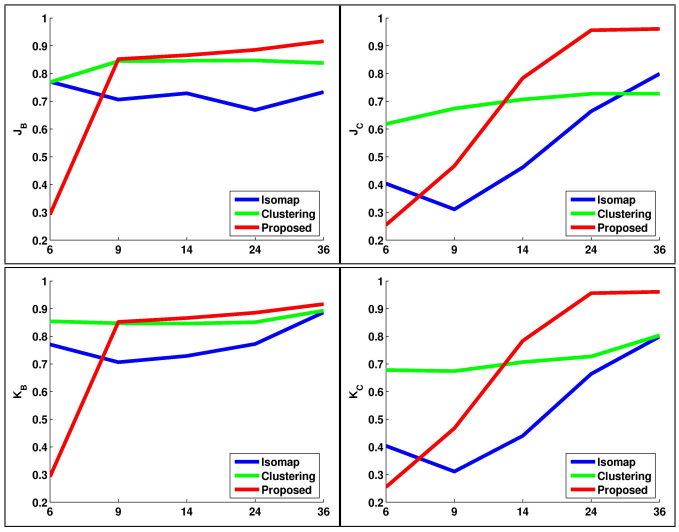

Figure 9: Performance vs. # of available views. Top: $\mathcal{J}$ Index. Bottom: $\mathcal{K}$ Index. Left: Results for Boxes. Right: Results for Cars.

The results are shown in Figure 9. The vertical axis indicates the performance index ($\mathcal{J}(m)$ and $\mathcal{K}(m)$) for boxes and cars and the horizontal axis corresponds to the number of views m provided to the algorithms per object. Note that curve associated with our proposed method is the same across the plots for each category as it does not depend on any tuning parameter θ.

The proposed method outperforms the others when the number of views reaches 14. This number seems to be the minimal set of points that can reveal the global topology of the pose curve for these data. In addition, it beats Isomap when the number of views reaches 9. Note that the performance of all three methods improves by increasing the number of views. However, in the regime where the data is moderately sparse (between 14 and 36 views, as our experiment shows), using global smoothness is demonstrably useful. The clustering results for two of the cars with 18 views per object is shown in Figures (10) (11) and (12) respectively for hierarchical clustering, Isomap and our method.

For the most computationally expensive scenario, where the number of samples for each object is 36, and therefore 72 samples in total for the object pair, our algorithm takes 0.042 seconds to find a solution. This is based on Matlab implementation on a 3.00Ghz machine. One should multiply this time by the number of random starts. We used 100 random starts for each subtask, and among them picked the one which attains the lowest cost.

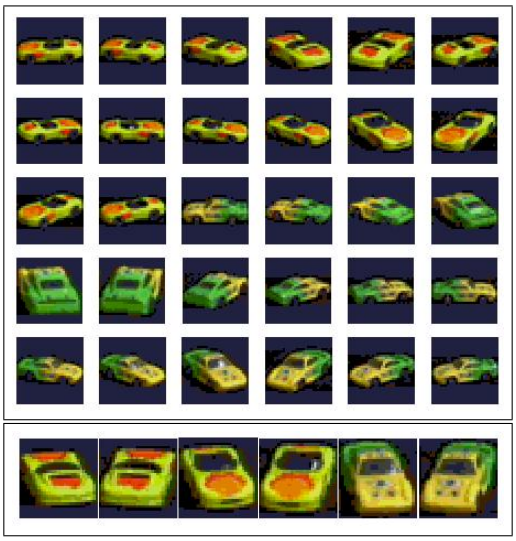

Figure 10: Two best groups found by hierarchical clustering

Figure 11: Two connected components obtained by Isomap (with red and green borders)

Conclusion

We formulated the problem of learning the intrinsic topology of a smooth curve without relying on any notion of geometric locality. We claim the notion helps when the sample set is sparse and not very noisy. We proposed a simple greedy algorithm for searching for a solution from a random initial solution. The proposed method is guaranteed to converge in finite steps. In addition, it has only one parameter, the choice of ℓ^q-norm. We experimentally observed that setting this norm to a small number ($q = 0.1$ in our case) produces good results across the tasks we experimented. This itself suggests a connection to sparse solutions.

We evaluated our method in the context of clustering. We aimed at recovering objects invariant to illumination and pose. Our algorithm greatly outperforms in the synthetic illumination case. It also outperforms for the real data of pose when the size of the sample set is large enough to identify the structure of the manifold.

This paper brings a new way of formulating manifold learning without relying on local geometry. One obvious avenue for future research is to determine whether this formulation can be extended to handle manifolds of arbitrary dimension (e.g., using the polar sine formulation of (Chen & Lerman, 2009)).

Acknowledgment

This work was supported by NSF IIS 07-03756, and ONR N00014-09-1-0230 grants.

References

Banks, D. C., & Beason, K. (2007). Fast global illumination for visualizing isosurfaces with a 3d illumination grid. *Computing in Science & Engineering*, *9*, 48–54.

Belkin, M., & Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, *15*, 1373–1396.

Bengio, Y., Monperrus, M., & Larochelle, H. (2006). Nonlocal estimation of manifold structure. *Neural Comput.*, *18*, 2509–2528.

Bentley, J. J. (1992). Fast algorithms for geometric traveling salesman problems. *ORSA Journal on Computing*, *4*, 387–411.

Chang, K., & Ghosh, J. (1998). Principal curve classifier: a nonlinear approach to pattern classification. *IEEE International Joint Conference on Neural Networks* (pp. 695–700).

Chen, G., & Lerman, G. (2009). Spectral curvature clustering (scc). *IJCV*, *81*, 317–330.

Cristianini, N. (2001). Icml tutorial about support vector machines:.

Dollár, P., Rabaud, V., & Belongie, S. (2007). Non-isometric manifold learning: Analysis and an algorithm. *ICML*.

Hastie, T., & Stuetzle, W. (1989). Principal curves. *Journal of the American Statistical Association*, *84*, 502–516.

Nene, S. A., Nayar, S. K., & Murase, H. (1996). *Columbia object image library (coil-100)* (Technical Report CUCS-006-96). Dept. of Computer Science, Columbia University.

Pontil, M., & Verri, A. (1998). Support vector machines for 3d object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*, 637–646.

Roth, D., Yang, M.-H., & Ahuja, N. (2002). Learning to recognize 3d objects. *Neural Computation*, *14*, 1071–1104.

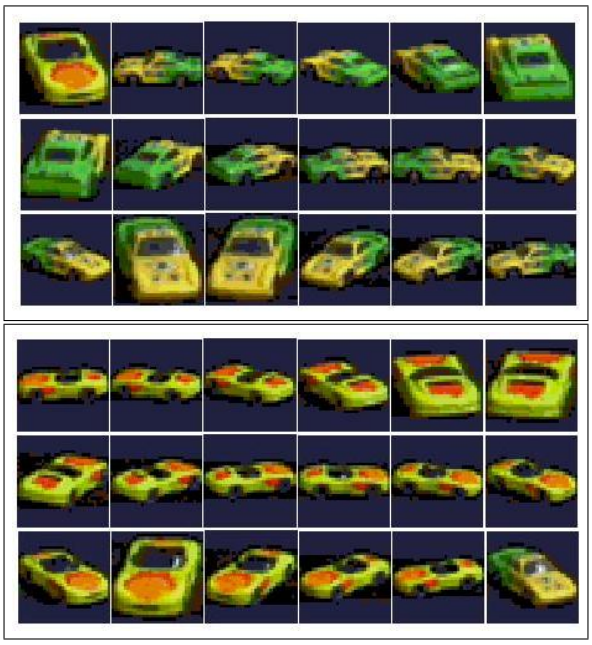

Figure 12: Two clusters discovered by the our method.

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

Smola, A. J., Mika, S., Schölkopf, B., & Williamson, R. C. (2001). Regularized principal manifolds. *J. Mach. Learn. Res.*, *1*, 179–209.

Tenenbaum, J., de Silva, V., & Langford, J. C. (2000). A Global Geometric Framework for Nonlinear Dimensionality Reduction. *SCIENCE*, *290*, 2319–2323.

Touretzky, D. S., & Pomerleau, D. A. (1989). What's hidden in the hidden layers? *BYTE*, *14*, 227–233.

Turk, G., & Levoy, M. (1994). Zippered polygon meshes from range images. *SIGGRAPH 94 Conference Proceedings* (pp. 311–318).

Weinberger, K. Q., & Saul, L. K. (2005). Unsupervised learning of image manifolds by semidefinite programming. *International Journal of Computer Vision*, *70*, 77–90.