Discovering Shared Structure in Manifold Learning

Yoshua Bengio and Martin Monperrus
Dept. IRO, Université de Montréal
P.O. Box 6128, Downtown Branch, Montreal, H3C 3J7, Qc, Canada
{bengioy,monperrm}@iro.umontreal.ca
Technical Report 1250, July 2nd 2004,
Département d'Informatique et Recherche Opérationnelle

July 6, 2004

Abstract

We claim and present arguments to the effect that a large class of manifold learning algorithms that are essentially local will suffer from at least four generic problems associated with (1) noise in the data, (2) curvature of the manifold, (3) dimensionality of the manifold, and (4) the presence of many manifolds with little data per manifold. This analysis suggests non-local manifold learning algorithms which attempt to discover shared structure in the tangent planes at different positions. A criterion for such an algorithm is proposed and experiments estimating a tangent plane prediction function are presented. The function has parameters that are shared across space rather than estimated based on the local neighborhood, as in current non-parametric manifold learning algorithms. The results show clearly the advantages of this approach with respect to local manifold learning algorithms.

1 Introduction

A central objective of statistical machine learning is to discover structure in the joint distribution between random variables, so as to be able to make predictions about new combinations of values of these variables. An extremely simplified way to describe such structure is to characterize the regions of high density versus the regions of low density. For example, clustering algorithms attempt to discover regions of high density that are centered around "cluster centers" or prototypes. Manifold learning algorithms generalize clustering by allowing the regions to have more general shapes. In particular, for very high-dimensional but structure-rich data (such as speech, language, or images), it makes sense to expect most directions of variations around a given observation to be unlikely, i.e. locally, the regions of high density are high-dimensional pancakes.

There has been in recent years a lot of work on unsupervised learning based on characterizing a possibly non-linear manifold near which the data would lie, such as

Locally Linear Embedding (LLE) (Roweis and Saul, 2000), Isomap (Tenenbaum, de Silva and Langford, 2000), kernel Principal Components Analysis (PCA) (Schölkopf, Smola and Müller, 1998), Laplacian Eigenmaps (Belkin and Niyogi, 2003), and Manifold Charting (Brand, 2003). These are all essentially non-parametric methods which represent the manifold on the basis of local neighborhood relations, very often constructed using the nearest neighbors graph (the graph with one vertex per observed example, and arcs between near neighbors). The above methods characterize the manifold through an embedding which associates each training example (an input object) with a low-dimensional coordinate vector (the coordinates on the manifold). Other closely related methods characterize the manifold as well as "noise" around it. Most of these methods consider the density as a mixture of flattened Gaussians, e.g. mixtures of factor analyzers (Ghahramani and Hinton, 1996), Manifold Parzen windows (Vincent and Bengio, 2003), and other local PCA models such as mixtures of probabilistic PCA (Tipping and Bishop, 1999). This is not an exhaustive list, and recent work also combines modeling through a mixture density and dimensionality reduction (Teh and Roweis, 2003; Brand, 2003).

In this paper we claim that there is a fundamental weakness with all of these methods (and the other similar non-parametric density estimation methods), due to the **locality of learning**: we show that the local tangent plane of the manifold at a point x is defined based mostly on the near neighbors of x according to some possibly data-dependent kernel K_D. A consequence of that weakness is that it is difficult to generalize to new combinations of values x that are "far" from the training examples x_i, where being "far" is a notion that should be understood in the context of several factors: the quantity of training data near x, the amount of noise around the manifold (the examples do not lie exactly on the manifold), and the dimensionality of the manifold. For example, if the manifold curves quickly around x, neighbors need to be closer for a locally linear approximation to be meaningful, which means that more data are needed in such high-curvature regions. Intrinsic dimensionality of the manifold compounds that problem because the amount of data thus needed will grow exponentially with this dimensionality. Thus saying that y is "far" from x means that *y is far from the tangent plane at x*.

This problem is fundamentally linked to the generalization principles used in a learning algorithm in order to be able to say something about a new example. In this paper we propose as a requirement for "high-dimensional" learning algorithms that they be non-local, i.e. that what is learned about the data in one region of input space could be used to help discover or predict structure in other far-away regions. We claim that this is a necessary condition to be able to generalize when the dimensionality, the noise and/or the curvature of the manifolds that characterize the density are high.

One way to address that problem is to estimate the tangent planes of the manifolds as a function of x, with parameters that can be estimated not only from the data around x but from the whole dataset. Note that there can be more than one manifold (e.g. in vision, one may imagine a different manifold for each "class" of object), but the structure of these manifolds may be related, something that many previous manifold learning methods did not take advantage of. Here we present experiments using multi-layer neural networks to represent those tangent planes, on a variety of tasks illustrating the weaknesses of the local manifold learning algorithms enumerated above. The main

advantage of non-local approaches such as the one introduced here is that it has at least **the potential to capture shared structure in many regions of input space.** On the other hand, a theoretical disadvantage of the proposed approach with respect to the local learning algorithms is that the optimization problem involved in learning is more difficult and generally non-convex. Fortunately, the experiments suggest that simple stochastic gradient descent may be sufficient to learn such shared structure.

2 Local Manifold Learning

By "local manifold learning", we mean a method that derives information about the local structure of the manifold (i.e. implicitly its tangent directions) at x based mostly on the training examples "around" x, where "proximity" may be the Euclidean distance or defined indirectly through a kernel.

Let us consider in turn a few of the most common local manifold learning methods to verify that this definition applies. In some cases it is pretty obvious, in others it is less.

2.1 Spectral Embedding Algorithms

As shown in (Bengio et al., 2004), there is a large group of manifold learning methods (as well as the spectral clustering methods) that share several characteristics. These include LLE (Roweis and Saul, 2000), Isomap (Tenenbaum, de Silva and Langford, 2000), kernel Principal Components Analysis (PCA) (Schölkopf, Smola and Müller, 1998) and Laplacian Eigenmaps (Belkin and Niyogi, 2003). They first build a data-dependent Gram matrix M with $n \times n$ entries $K_D(x_i, x_j)$, where $D = \{x_1, \ldots, x_n\}$ is the training set and K_D is a **data-dependent kernel**, and compute the eigenvector-eigenvalue pairs $\{(v_k, \lambda_k)\}$ of M. The embedding of the training set is obtained directly from the principal eigenvectors v_k of M (the i-th element of v_k gives the k-th coordinate of x_i's embedding, possibly scaled by $\sqrt{\frac{\lambda_k}{n}}$, i.e. $e_k(x_i) = v_{ik}$) and the embedding for a new example can be estimated using the Nyström formula (Bengio et al., 2004):

$$e_k(x) = \frac{1}{\lambda_k} \sum_{i=1}^{n} v_{ki} K_D(x, x_i)$$

for the k-th coordinate of x, where λ_k is the k-th eigenvalue of M (the optional scaling by $\sqrt{\frac{\lambda_k}{n}}$ would also apply). Here we will talk about "neighbors" of x to be those x_i for which $K_D(x, x_i)$ is significantly different from zero, when $K_D(x, x_i)$ decreases quickly as x_i is taken farther away from x. The above equation says that the embedding for a new example x is a local interpolation of the manifold coordinates of its neighbors x_i, with interpolating weights given by $\frac{K_D(x,x_i)}{\lambda_k}$. To see more clearly how the tangent plane may depend only on the neighbors of x, consider the relation between the tangent plane and the embedding function:

the tangent plane at x is simply the subspace spanned by the vectors $\frac{\partial e_k(x)}{\partial x}$.

In the case of very "local" kernels like that of LLE, spectral clustering with Gaussian kernel, Laplacian Eigenmaps or kernel PCA with Gaussian kernel, that derivative only depends significantly on the near neighbors of x:

$$\frac{\partial e_k(x)}{\partial x} = \frac{1}{\lambda_k} \sum_{i=1}^{n} v_{ki} \frac{\partial K_D(x, x_i)}{\partial x}.$$

For example, for a Gaussian kernel the derivative quickly becomes 0 as $||x - x_i||$ increases.

The case of Isomap is less obvious but we show below that it is also local. Let $\mathcal{D}(a, b)$ denote the graph geodesic distance going only through a, b and points from the training set. As shown in (Bengio et al., 2004), the corresponding data-dependent kernel can be defined as $K_D(x, x_i) = -\frac{1}{2}(\mathcal{D}(x, x_i)^2 - \frac{1}{n}\sum_j \mathcal{D}(x, x_j)^2 - \bar{\mathcal{D}}_i + \bar{\mathcal{D}})$ where $\bar{\mathcal{D}}_i = \frac{1}{n}\sum_j \mathcal{D}(x_i, x_j)^2$ and $\bar{\mathcal{D}} = \frac{1}{n}\sum_j \bar{\mathcal{D}}_j$. Let $\mathcal{N}(x, x_i)$ denote the index j of the training set example x_j that is a neighbor of x that minimizes $||x - x_j|| + \mathcal{D}(x_j, x_i)$. Then

$$\frac{\partial e_k(x)}{\partial x} = \frac{1}{\lambda_k} \sum_i v_{ki} \left(\frac{1}{n} \sum_j \mathcal{D}(x, x_j) \frac{(x - x_{\mathcal{N}(x,x_j)})}{||x - x_{\mathcal{N}(x,x_j)}||} - \mathcal{D}(x, x_i) \frac{(x - x_{\mathcal{N}(x,x_i)})}{||x - x_{\mathcal{N}(x,x_i)}||} \right) \quad (1)$$

which is a linear combination of vectors $(x - x_k)$, where x_k is a neighbor of x. *This clearly shows that the tangent plane at x associated with Isomap is included in the subspace spanned by the vectors $(x - x_k)$ where x_k is a neighbor of x.*

The fact that the estimated tangent plane at x must essentially lie in the subspace spanned by the near neighbors of x means that such methods cannot say anything meaningful when those neighbors are too far to give information about the true local tangent plane. Unfortunately, as we have argued, when the manifold is high dimensional, noisy, and curved, the neighbors of x are likely to often be too far to correctly estimate the true local plane.

2.2 Mixture of Pancakes Density Models

There are also a variety of local manifold learning algorithms which can be classified as "mixtures of pancakes" (Ghahramani and Hinton, 1996; Tipping and Bishop, 1999; Vincent and Bengio, 2003; Teh and Roweis, 2003; Brand, 2003). These are generally mixtures of Gaussians with a particular covariance structure. When the covariance matrix is approximated using its principal eigenvectors, this leads to "local PCA" types of methods. For these methods the local tangent directions directly correspond to the principal eigenvectors of the local covariance matrices. Again it is clear with these methods that the learning is local since it is mostly the examples around the Gaussian center that determine its covariance structure. The problem is not so much due to the form of the density as a mixture of Gaussians. The problem is that the local parameters (e.g. local principal directions) are estimated mostly based on local data. There is usually a non-local interaction between the different Gaussians, but its role is mainly of global coordination, e.g. where to set the Gaussian centers to allocate them properly

where there is data, and optionally how to orient the principal directions so as to obtain a globally coherent coordinate system for embedding the data.

Note that some of these methods (Teh and Roweis, 2003; Brand, 2003) provide both an embedding and a density model: the embedding is derived from the density model, represented by a mixture of pancake-like Gaussians. The central question that we will study below is how to estimate these Gaussians, using more than just local neighborhood information.

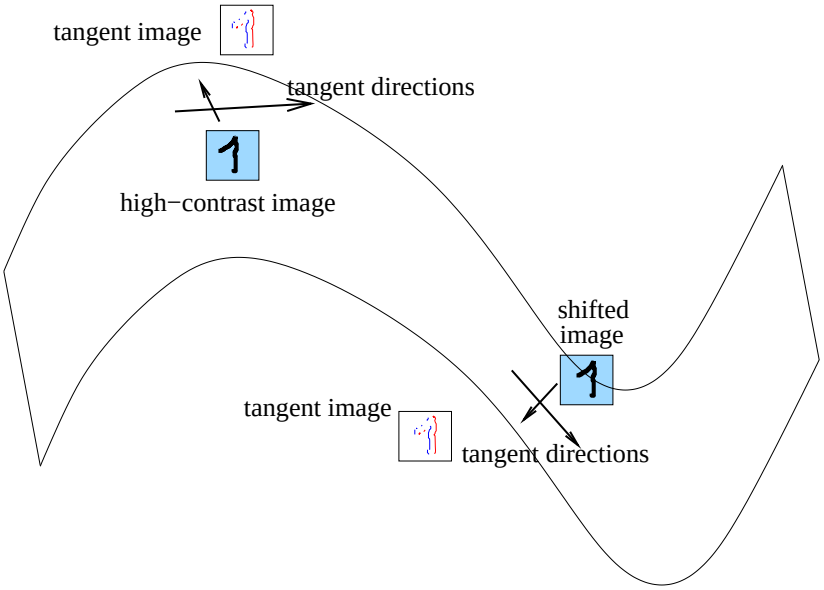

Figure 1: *The manifold of translations of a high-contrast image has very high curvature. The tangent plane for an image translated by only one pixel looks similar but changes abruptly since the edges are only one-pixel wide and are also shifted by one pixel. Hence the two tangent planes are almost orthogonal.*

2.3 Where Local Manifold Learning Would Fail

It is easy to imagine at least four failure causes for local manifold learning methods, and combining them will create even greater problems:

1. **Noise around the manifold**: data are not exactly lying on the manifold, i.e. the pancake is thick. In the case of PCA with d principal components and decreasing eigenvalues $\lambda_1, \lambda_2, \ldots$, this happens if $\frac{\lambda_d}{\lambda_{d+1}}$ is not large enough or if $\frac{\sum_{i=1}^{d} \lambda_i}{\sum_{i=d+1}^{n} \lambda_i}$ is not large enough. If that ratio is small then it takes more data to properly estimate the principal components. In the case of non-linear manifolds, the presence of noise means that more data around each pancake region will be needed to properly estimate the tangent directions of the manifold in that region. With

such noisy data, the tangent plane estimated from a local-PCA like method will often point away from the manifold, because a neighbor x_k of x may be forming a vector $x - x_k$ which forms a large angle with the tangent plane.

2. **High curvature of the manifold**. The above local manifold learning methods essentially approximate the manifold by the union of many locally linear patches. For this to work, there must be at least d close enough examples in each patch (in the sense of being close to the tangent plane at the center of the patch) for this to work. With noise, more data will be needed. With a high curvature manifold, more – smaller – patches will be needed, and the number of required patches will grow exponentially with the dimensionality of the manifold. To emphasize that this is a serious problem, consider the manifold of translations of a high-contrast image, in Figure 1. The tangent direction corresponds to the change in image due a small translation, i.e. it is non-zero only at edges in the image. After a one-pixel translation, the edges have moved by one pixel, and may not overlap much with the edges of the original image if it had high contrast. This is indeed a very high curvature manifold.

3. **High intrinsic dimension of the manifold**. We have already seen that high manifold dimensionality d is hurtful because $O(d)$ examples are required in each patch and $O(r^d)$ patches (for some r depending on curvature and noise) are necessary to span the manifold. In the translation example, if the image resolution is increased then many more training images will be needed to capture the curvature around the translation manifold with locally linear patches. Yet the physical phenomenon responsible for translation is expressed by a simple equation, which does not get more complicated with increasing resolution.

4. **Presence of many manifolds with little data per manifold**. In many real-world contexts there is not only one global manifold but a large number of manifolds which however share something about their structure. A simple example is the manifold of transformations (view-point, position, lighting,...) of 3D objects in 2D images. There is one manifold per object instance (corresponding to the successive application of small amounts of all of these transformations). The manifolds associated with different object instances may be connected to each other (i.e. when there is a continuum of plausible object images going from one to the other). However in general image data, there will be a large number of different object classes, each corresponding to a manifold disconnected from the other. If there are only a few examples for each such class then it is almost impossible to learn the manifold structures using only local manifold learning. However, if the manifold structures are generated by a common underlying phenomenon (as in changes due to view-point, position, lighting, etc...) then a non-local manifold learning method could potentially learn all of these manifolds and even generalize to manifolds for which a single instance is observed.

2.4 Relation to Non-Parametric Semi-Supervised Learning

We claim that the problems outline above also plague non-parametric semi-supervised learning algorithms, such as (Szummer and Jaakkola, 2002; Chapelle, Weston and Scholkopf, 2003; Belkin and Niyogi, 2003; Zhu, Ghahramani and Lafferty, 2003; Zhu, Lafferty and Ghahramani, 2003; Zhou et al., 2004). These algorithms basically rely on a local kernel and the nearest neighbor graph to "propagate" label information from labeled examples to unlabeled examples. If the classes are well separated in comparison to that kernel (i.e., generally with respect to Euclidean distance), than these methods can be very helpful. However, it is again easy to come up with simple examples where this approach would fail, because of noise around the manifold, dimensionality of the manifold, curvature of the manifold, and not enough data to characterize each manifold as a patchwork of local linear pancakes.

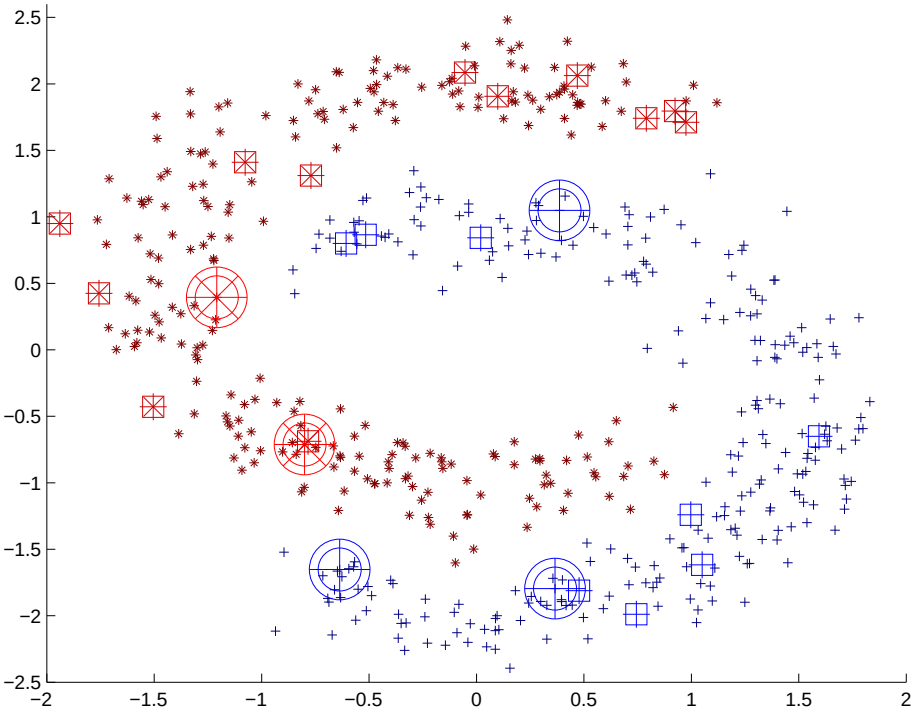

Figure 2: *Non-parametric semi-supervised learning works best on low dimensional data where there is enough data locally to estimate the manifolds associated with each class. The labeled examples (with a circle) are enough correctly capture the class of the unlabeled examples (red points on the left vs blue points on the right).*

The basic idea behind these algorithms is that we can do better than supervised learning because we can "follow" the manifold near which the data lie, as shown in figure 2. However, this assumes that we have characterized such manifold properly,

and we have well explained how this can failed if the manifold is estimated relying essentially on the neighbors of each unlabeled example.

If the estimation of the manifolds tangent plane is poor, than "label propagation" is likely to bring the wrong class information to large chunks of unlabeled examples.

3 Non-Local Manifold Tangent Learning

In order to better deal with the challenges described in section 2.3, we propose to compute a **smooth** of local learning algorithms: not only the local neighbors should be used to estimate the local tangent plane, i.e. shared structure (if there is one) should be exploited.

To discover such shared structure we propose here to characterize the manifolds in the data distribution through a matrix-valued function $F(x)$ that predicts at $x \in \mathbf{R}^n$ a basis for the tangent plane of the manifold near x, hence $F(x) \in \mathbf{R}^{d \times n}$ for a d-dimensional manifold.

We are going to consider a simple supervised learning setting to train this function. As with Isomap, we consider that the vectors $(x - x_i)$ with x_i a neighbor of x span a noisy estimate of the manifold tangent space. We propose to use them to define a "target" for training $F(x)$. In our experiments we simply collected the k nearest neighbors of each example x, but better selection criteria could be devised.[1] Points on the predicted tangent subspace can be written $F'(x)w$ with $w \in \mathbf{R}^d$ the combining weight vector. This is illustrated in figure 3.

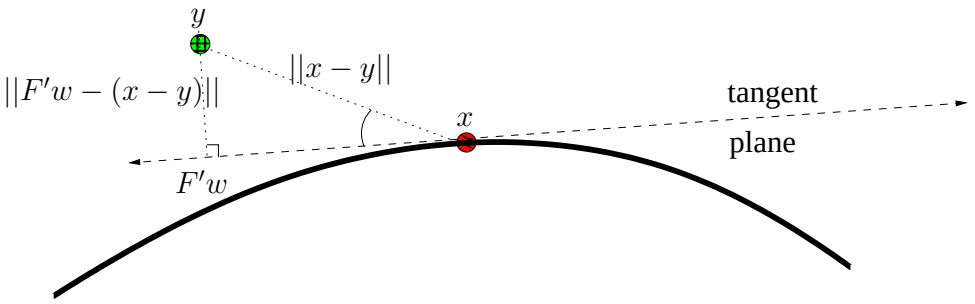

Figure 3: *The manifold tangent plane around x, with a neighbor y. The estimated tangent plane is the set of linear combinations $F'w = \sum_i w_i F_i$. We choose w that make $F'w$ the projection of y on the estimated tangent plane.*

Several criteria are possible to match the neighbors differences with the subspace defined by $F(x)$. One that yields to a simple analytic solution is simply to minimize the distance between the $x - x_j$ vectors and their projection on the subspace defined by $F(x)$. The weight vector $w_{tj} \in \mathbf{R}^d$ that matches neighbor x_j of example x_t is thus an extra free parameter that has to be optimized. Fortunately the solution to this

[1] possibly also using an ϵ-ball around x, or more sophisticated criteria in which we allow further away neighbors in the directions of low curvature and high variability.

optimization is obtained easily and analytically. The overall training criterion involves a double optimization over function F and projection weights w_{tj} of what we call the **relative projection error**:

$$\min_{F,\{w_{tj}\}} \sum_t \sum_{j \in \mathcal{N}(x_t)} \frac{||F'(x_t)w_{tj} - (x_t - x_j)||^2}{||x_t - x_j||^2} \tag{2}$$

where $\mathcal{N}(x)$ denotes the selected set of near neighbors of x. The normalization by $||x_t - x_j||^2$ is to avoid giving more weight to the neighbors that are further away. Recall that the subspace goes through the origin, so examples that are further away but at the same angle with respect to the subspace will have greater projection error. Taking the above ratio amounts to minimizing the square of the sinus of this angle. To perform the above minimization, we can do coordinate descent (which guarantees convergence to a minimum), i.e. alternate changes in F and changes in w's which at each step go down the total criterion. Since the minimization over the w's can be done separately for each example x_t and neighbor x_j, it is equivalent to minimize

$$\frac{||F'(x_t)w_{tj} - (x_t - x_j)||^2}{||x_t - x_j||^2} \tag{3}$$

over vector w_{tj} for each such pair (done analytically) and compute the gradient of the above over F (or its parameters) to move F slightly (we used stochastic gradient on the parameters of F). The solution for w_{tj} is obtained by solving the linear system

$$F(x_t)F'(x_t)w_{tj} = F(x_t)\frac{(x_t - x_j)}{||x_t - x_j||^2}. \tag{4}$$

In our implementation this is done robustly through a singular value decomposition $F'(x_t) = USV'$ and $w_{tj} = B(x_t - x_j)$ where B can be precomputed for all the neighbors of x_t: $B = (\sum_{k=1}^{d} 1_{S_k > \epsilon} V_{.k} V'_{.k}/S_k^2)F(x_t)$. The gradient of the criterion with respect to the i-th row of $F(x_t)$, holding w_{tj} fixed, is simply

$$2\sum_j \frac{w_{tji}}{||x_t - x_j||}(F'(x_t)w - (x_t - x_j)) \tag{5}$$

where w_{tji} is the i-th element of w_{tj}. In practice, it is not necessary to store more than one w_{tj} vector at a time. In the experiments, $F(\cdot)$ is parameterized as a an ordinary one hidden layer neural network with n inputs and $d \times n$ outputs. It is trained by stochastic gradient descent, one example x_t at a time.

Although the above algorithm provides a characterization of the manifold, it does not directly provide an embedding nor a density function. However, once the tangent plane function is trained, there are ways to use it to obtain all of the above. The simplest method is to apply existing algorithms that provide both an embedding and a density function based on a Gaussian mixture with pancake-like covariances, once the local principal components (i.e. the local tangent planes) have been estimated. For example one could use (Teh and Roweis, 2003) or (Brand, 2003), with each Gaussian being centered at a data point or at a representative point, and the covariance matrix can be constructed from $F'(x)diag(\sigma^2(x))F(x)$, where $\sigma_i^2(x)$ should estimate $Var(w_i)$ around x.

3.1 Previous Work on Non-Local Manifold Learning

The non-local manifold learning algorithm presented here (find $F(\cdot)$ which minimizes the criterion in eq. 2) is similar to the one proposed in (Rao and Ruderman, 1999) to estimate the generator matrix of a Lie group. That group defines a one-dimensional manifold generated by following the orbit $x(t) = e^{Gt}x(0)$, where G is an $n \times n$ matrix and t is a scalar manifold coordinate. Note that Lie groups are appropriate to approximately model some transformations on images such as translation, but not others (e.g. rotation, lighting change). A multi-dimensional manifold can be obtained by replacing Gt above by a linear combination of multiple generating matrices. In (Rao and Ruderman, 1999) the matrix exponential is approximated to first order by $(I + Gt)$, and the authors estimate G for a simple signal undergoing translations, using as a criterion the minimization of $\sum_{x,\tilde{x}} \min_t ||(I + Gt)x - \tilde{x}||^2$, where $\tilde{x}$ is a neighbor of x in the data. Note that the Lie group has a tangent plane that is a linear function of x, i.e. $F_1(x) = Gx$. By minimizing the above across many pairs of examples, a good estimate of G for the artificial data was recovered by (Rao and Ruderman, 1999). The proposal here extends this approach to multiple dimensions and non-linear relations between x and the tangent planes. Note also the earlier work on Tangent Distance (Simard, LeCun and Denker, 1993), in which the tangent planes are not learned but used to build a nearest neighbor classifier that is based on the distance between the tangent subspaces around two examples to be compared.

The main advantage of the approach proposed here over local manifold learning is that the parameters of the tangent plane predictor can be estimated using data from very different regions of space, thus in principle allowing to be less sensitive to all four of the problems described in 2.3, thanks to sharing of information across these different regions. Of course, higher dimensionality (of the manifold and of the raw data) requires more parameters because $F : \mathbf{R}^n \rightarrow \mathbf{R}^{d \times n}$. The proposed algorithm could in particular be improved with respect to the problem of the curvature, because we are still going to try to estimate the local tangent using the linear relation between x and its neighboring examples. However the estimation of $F(x)$ across all the data should smooth out some of the noise and hopefully some of the local effect of curvature. To fully take curvature into account, one possibility is to try to follow the manifold to go from x to its neighbors (e.g. using a Newton optimization approach as proposed in (Simard, LeCun and Denker, 1993)). Another is to parameterize the curvature locally (e.g. as in Lie group manifold learning (Rao and Ruderman, 1999), but approximating the matrix exponential with more than its first order Taylor expansion). However, "matching" x's neighbors to x becomes more difficult, whereas with the simple algorithm proposed here this matching can be done analytically.

4 Experimental Results

The objective of the experiments is to validate the proposed algorithm: does it estimate well the true tangent planes? does it learn better than a local manifold learning algorithm? Note that all the tasks tested involve tangent planes that are not a linear function of x, i.e. can't be represented by a Lie group based manifold of the form $F(x) = Gx$.

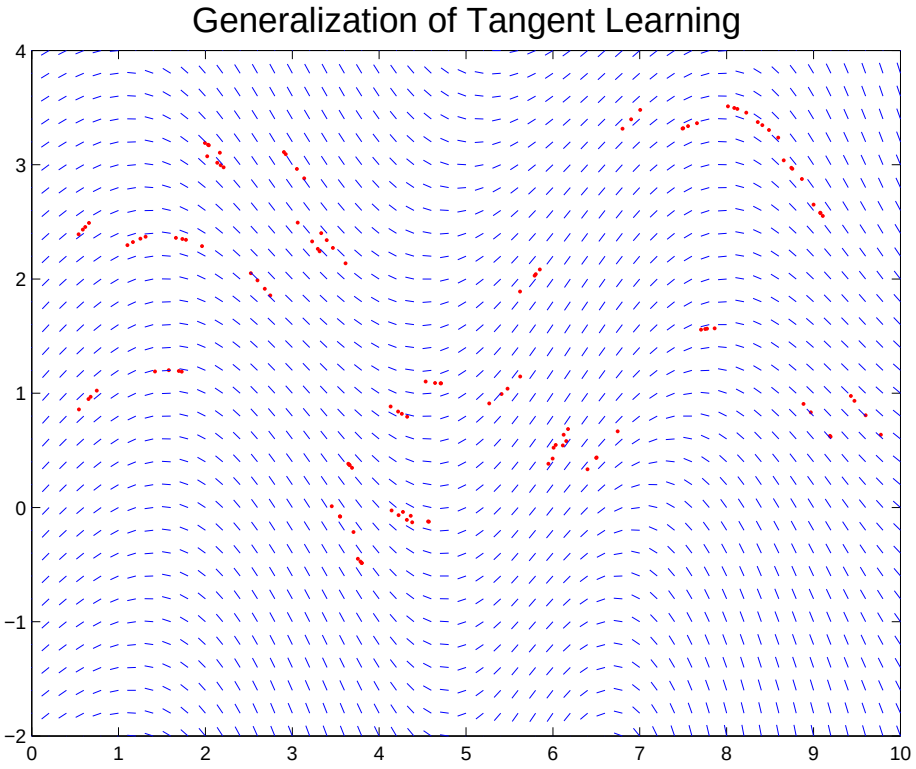

Figure 4: *Task 1, 2-D data with 1-D sinusoidal manifolds: the method indeed captures the tangent planes. The small segments are the estimated tangent planes. Red points are training examples.*

Error Measurement In addition to visualizing the results for the low-dimensional data, we measure performance by considering how well the algorithm learns the local tangent distance, as measured by the normalized projection error of nearest neighbors (eq. 3). We compare the errors of four algorithms, always on test data not used to estimate the tangent plane: (a) **analytic** (using the true manifold's tangent plane at x computed analytically), (b) **tangent learning** (using the neural-network trained tangent plane predictor $F(x)$, trained using the $k \geq d$ nearest neighbors in the training set of each training set example), (c) **Dim-NN** (using the d nearest neighbors of x in the training set), (d) **Local PCA** (using the d principal components of the k nearest neighbors of x in the training set).

In all the experiments we found that all the randomly initialized neural networks converged to similarly good solutions. The number of hidden units was not optimized, although preliminary experimentation showed phenomena of overfitting and underfitting due to too small or too large number hidden units was possible.

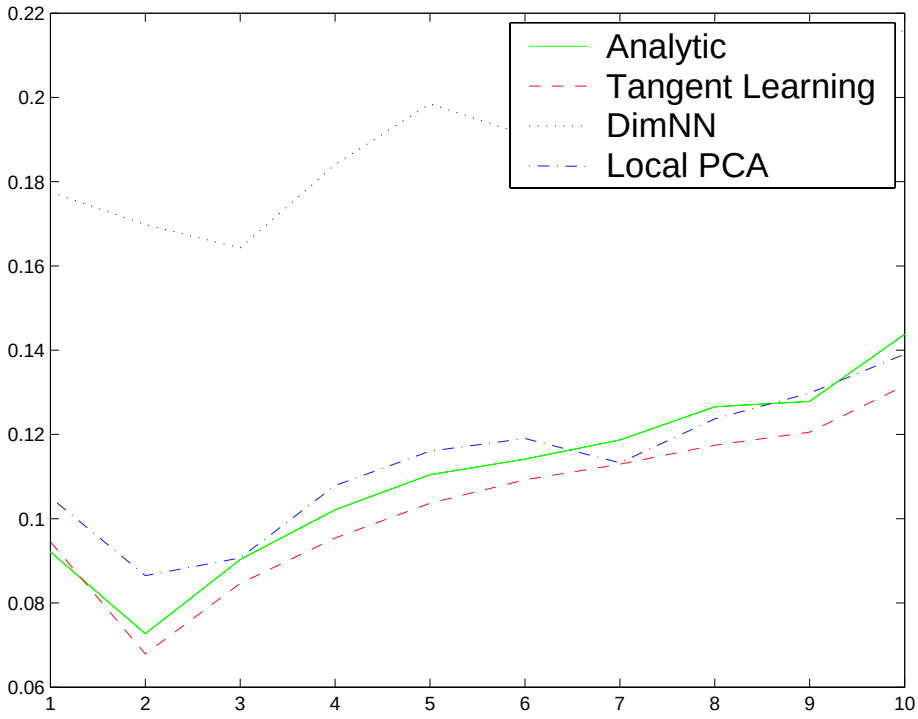

Figure 5: *Comparative results on task 2. Relative projection error for k-th nearest neighbor, w.r.t. k from 1 to 5, for the four compared methods.*

Task 1 We first consider a low-dimensional but multi-manifold problem. The data $\{x_i\}$ are in two dimensions and coming from a set of 40 1-dimensional manifolds. Each manifold is composed of 4 near points obtained from a randomly based sinus, i.e $\forall i \in 1..4, x_i = (a + t_i, sin(a + t_i) + b$, where a, b, and t_i are randomly chosen. Four neighbors were used for training both the Tangent Learning algorithm and the benchmark local non-parametric estimator (local PCA of the 4 neighbors). Figure 4 shows the training set and the tangent planes recovered, both on the training example and generalization away from the data. The neural network has 10 hidden units. This problem is particularly difficult for local manifold learning, which does very poorly here: the out-of-sample relative prediction error are respectively 0.09 for the **analytic** plane, 0.25 for **tangent learning**, 0.88 for **Dim-NN**, and 0.81 for **local PCA**.

Task 2 This is a higher dimensional manifold learning problem, with 41 dimensions. The data are generated by sampling Gaussian curves. Each curve is of the form $x(i) = e^{t_1 - (-2+i/10)^2/t_2}$ with $i \in \{0, 1, \ldots, 40\}$. The manifold coordinates are t_1 and t_2, sampled uniformly, respectively from $(-1, 1)$ and $(.1, 3.1)$. Normal noise (standard deviation = 0.001) is added to each point. 100 example curves were generated

testing on MNIST digits	Average relative projection error
analytic tangent plane	0.27
tangent learning	0.43
Dim-NN or **Local PCA**	1.50

Table 1: Average relative projection error on the 2000 test digit images, for the algorithms compared, as well as for the analytic tangent plane (of image rotations).

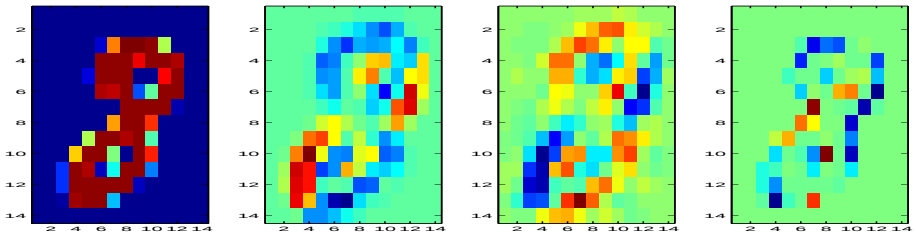

Figure 6: From left to right: a test image, its analytic rotation tangent vector, the tangent vector predicted by the neural network, the tangent vector predicted by local PCA. Red means positive and blue means negative.

for training and 200 for testing. The neural network has 100 hidden units. Figure 5 shows the relative projection error for the four methods on this task, for the k-th nearest neighbor, for increasing values of k. First, the error decreases because of the effect of noise (near noisy neighbors may form a high angle with the tangent plane). Then, it increases because of the curvature of manifold (further away neighbors form a larger angle). Note that in this case, the algorithm learned a bit more the closed curvature than the tangent plane. That's why the "Tangent Learning" error is better that the analytic one for $k = 2$.

Task 3 This is a high-dimensional multi-manifold task, involving **digit images** to which we have applied slight rotations, in such a way as to have the knowledge of the analytic formulation of the manifolds. There is one rotation manifold for each instance of digit from the database, but only two examples for each manifold: one real image from the MNIST dataset and one slightly rotated image. 1000×2 examples are used for training and 1000×2 for testing. In this context we use $k = 1$ nearest neighbor only and the number of manifold dimensions is $d = 1$.

The average relative projection error for the nearest neighbor are given in table 1. The tangent learning neural network has 100 hidden units and was trained for 100 stochastic gradient epochs. In the case of images, it is possible to visualize what the model has learned, since the tangent direction also corresponds to an image. The predicted tangent direction for rotation is shown in figure 6. Note that the predicted direction may disagree in sign with the analytic tangent direction but it is very close to it, whereas a local PCA prediction is poorer.

An even more interesting experiment consists in applying the trained predictor on a novel image that comes from a very different distribution but one that shares the same

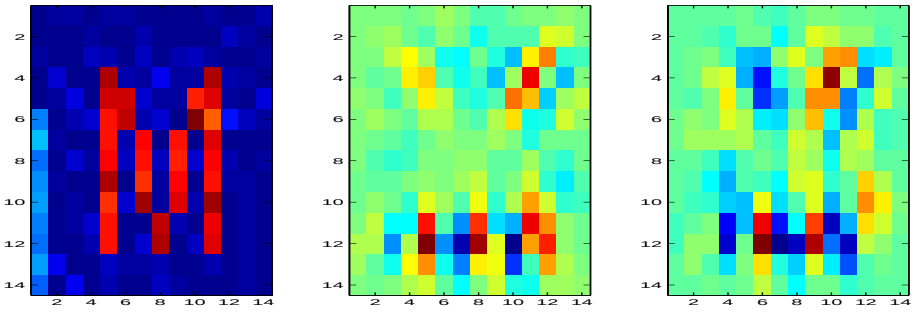

Figure 7: From left to right: a truly out-of-sample test image, the tangent vector predicted by the neural network, and the tangent vector predicted by local PCA. Red means positive and blue means negative.

manifold structure, i.e. images of other characters that are not digits.

As a representative example, we show the predicted tangent direction for rotation on the image of character 'M', in figure 7.

To the trained eye it is clear from figure 7 that the neural network prediction is quite good, whereas the local PCA predictor is much poorer. To make that clearer we have used the predicted tangent planes to follow the manifold by small steps (this is very easy to do in the case of a one-dimensional manifold). Figure 8 shows the effect of a few such steps and a larger number of steps, both for the neural network predictor and for the local PCA predictor.

This example illustrates the crucial point that non-local tangent plane learning is able to generalize to truly novel cases, where local manifold learning fails.

5 Conclusion

The central claim of this paper is that there are fundamental problems with non-parametric local approaches to manifold learning, due to the presence of noise around the manifold, due to the curvature of the manifold, its dimensionality, and the presence of several disjoint manifolds. To address these problems, we propose that learning algorithms should be designed in such a way that they can share information about the tangent structure of the manifold, coming from different regions of space. In this spirit we have proposed a simple learning algorithm based on predicting the tangent plane at x with a function $F(x)$ whose parameters are estimated based on the whole data set. Note that the same four problems are present with non-parametric approaches to semi-supervised learning (e.g. as in (Szummer and Jaakkola, 2002; Chapelle, Weston and Scholkopf, 2003; Belkin and Niyogi, 2003; Zhu, Ghahramani and Lafferty, 2003)), which rely on proper estimation of the manifold in order to propagate label information.

Future work should investigate how to better handle the curvature problem, e.g. by following the manifold (using the local tangent estimates), to estimate a manifold-following path between pairs of neighboring examples. The algorithm can also be

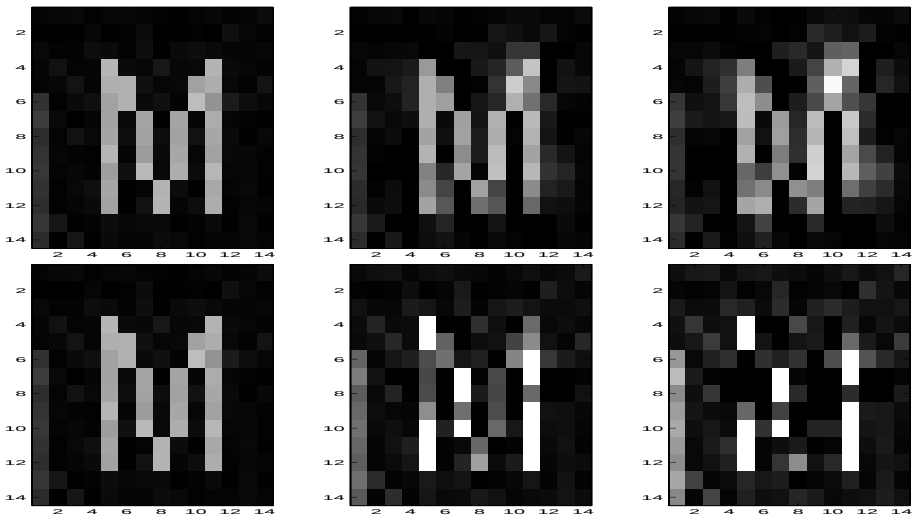

Figure 8: Left column: original image. Middle: applying a small amount of the predicted rotation. Right: applying a larger amount of the predicted rotation. Top: using the estimated tangent plane predictor. Bottom: using local PCA, which is clearly much worse.

extended in a straightforward way to obtain a Gaussian mixture or a mixture of factor analyzers (with the factors or the principal eigenvectors of the Gaussian centered at x obtained from $F(x)$). This view can also provide an alternative criterion to optimize $F(x)$ (the local log-likelihood of such a Gaussian). This criterion also tells us how to estimate the missing information (the variances along the eigenvector directions). Since we can estimate $F(x)$ everywhere, a more ambitious view would consider the density as a "continuous" mixture of Gaussians (with an infinitesimal component located everywhere in space).

Acknowledgments

The authors would like to thank the following funding organizations for support: NSERC, MITACS, IRIS, and the Canada Research Chairs. They authors are also greatful for the feedback and stimulating exchanges that helped to shape this paper, with Olivier Delalleau, Sam Roweis, Pascal Vincent and Léon Bottou.

References

Belkin, M. and Niyogi, P. (2003). Using manifold structure for partially labeled classification. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*, Cambridge, MA. MIT Press.

Bengio, Y., Delalleau, O., Le Roux, N., Paiement, J.-F., Vincent, P., and Ouimet, M. (2004). Learning eigenfunctions links spectral embedding and kernel PCA. *Neural Computation*, to appear.

Brand, M. (2003). Charting a manifold. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*. MIT Press.

Chapelle, O., Weston, J., and Scholkopf, B. (2003). Cluster kernels for semi-supervised learning. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*, Cambridge, MA. MIT Press.

Ghahramani, Z. and Hinton, G. (1996). The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1, Dpt. of Comp. Sci., Univ. of Toronto.

Rao, R. and Ruderman, D. (1999). Learning lie groups for invariant visual perception. In Kearns, M., Solla, S., and Cohn, D., editors, *Advances in Neural Information Processing Systems 11*, pages 810–816. MIT Press, Cambridge, MA.

Roweis, S. and Saul, L. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326.

Schölkopf, B., Smola, A., and Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319.

Simard, P., LeCun, Y., and Denker, J. (1993). Efficient pattern recognition using a new transformation distance. In Giles, C., Hanson, S., and Cowan, J., editors, *Advances in Neural Information Processing Systems 5*, pages 50–58, Denver, CO. Morgan Kaufmann, San Mateo.

Szummer, M. and Jaakkola, T. (2002). Partially labeled classification with markov random walks. In Dietterich, T., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA. MIT Press.

Teh, Y. W. and Roweis, S. (2003). Automatic alignment of local representations. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*. MIT Press.

Tenenbaum, J., de Silva, V., and Langford, J. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323.

Tipping, M. and Bishop, C. (1999). Mixtures of probabilistic principal component analysers. *Neural Computation*, 11(2):443–482.

Vincent, P. and Bengio, Y. (2003). Manifold parzen windows. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*, Cambridge, MA. MIT Press.

Zhou, D., Bousquet, O., Navin Lal, T., Weston, J., and Schölkopf, B. (2004). Learning with local and global consistency. In Thrun, S., Saul, L., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems 16*, Cambridge, MA. MIT Press.

Zhu, X., Ghahramani, Z., and Lafferty, J. (2003). Semi-supervised learning using gaussian fields and harmonic functions. In *ICML'2003*.

Zhu, X., Lafferty, J., and Ghahramani, Z. (2003). Semi-supervised learning: From gaussian fields to gaussian processes. Technical Report CMU-CS-03-175, CMU.