Manifold Learning and its Applications:
Papers from the AAAI Fall Symposium (FS-09-04)

MiPPS: A Generative Model for Multi-Manifold Clustering

Oluwasanmi Koyejo and Joydeep Ghosh
Electrical and Computer Engineering Dept., University Of Texas, Austin
sanmi.k@mail.utexas.edu, ghosh@ece.utexas.edu

Abstract

We propose a generative model for high dimensional data consisting of intrinsically low dimensional clusters that are noisily sampled. The proposed model is a mixture of probabilistic principal surfaces (MiPPS) optimized using expectation maximization. We use a Bayesian prior on the model parameters to maximize the corresponding marginal likelihood. We also show empirically that this optimization can be biased towards a good local optimum by using our prior intuition to guide the initialization phase. The proposed unsupervised algorithm naturally handles cases where the data lies on multiple connected components of a single manifold and where the component manifolds intersect. In addition to clustering, we learn a functional model for the underlying structure of each component cluster as a parameterized hyper-surface in ambient noise. This model is used to learn a global embedding that we use for visualization of the entire dataset. We demonstrate the performance of MiPPS in separating and visualizing land cover types in a hyperspectral dataset.

1. Introduction

Consider a data set consisting of images of various objects from all possible orientations. With all other conditions fixed, there are two degrees of freedom for a given object in this dataset; the elevation and rotation of its corresponding images. This implies that the set of all images in our database is likely to be well represented as some two dimensional non-linear manifold in the image space with multiple connected components corresponding to each object. This property is not unique to object pose images. Other common examples include images of faces, handwritten digits and shapes.

Now, suppose one would like to partition the image data set in an unsupervised manner, for example, to separate the data by object with each cluster containing all poses of the object in question. Special care should be taken when the data is generated from one or more non-linear manifolds. Common methods for data clustering such as k-means or a mixture of Gaussians are unlikely to find the correct partitions as these methods assume that the data clusters are convex; an assumption that is often untrue in complicated datasets. Further, one might be interested in learning a parametric model of each manifold for compression, dimensionality reduction, visualization, generating novel samples or other post-processing. This motivates studying special parametric methods for multi-manifold clustering.

When the data samples are generated from separated clusters on a connected manifold, embedding techniques such as ISOMAP and LLE can be used to *unwrap* the manifold; creating a low dimensional embedding where the data can be clustered more easily (Yankov and Keogh 2006). The quality of the embedding and subsequent clustering found by these methods is very sensitive to noise and parameter selection. Further, ISOMAP and LLE will fail to find to find an embedding when there is a large separation between the clusters[1]. Statistical embedding and data visualization approaches such as the co-ordinated mixture of probabilistic principal component analyzers (PPCA) (Verbeek, Vlassis, and Kröse 2002) and the generative topographic mapping (GTM) (Bishop, Svensen, and Williams 1998b) are more robust to noise. In both methods, the data can be mapped to a low dimensional latent space. The mapped points can then be partitioned to estimate the data clusters. This technique loosely corresponds to a generative model. However, the resulting heuristic does not necessarily optimize the model.

Tino and Nabney describe a supervised visualization model based on hierarchical mixtures of GTM (Tino and Nabney 2002). At each layer, the user initializes the local GTM using *interesting* points. However, the model is not designed to learn data clusters in an unsupervised manner. Further, our experiments suggest that our base model is superior to the GTM as a parametric non-linear manifold model in terms of modelling and clustering performance. The algorithm most related to ours is the k-manifolds algorithm proposed by (Souvenir 2006) for estimating and clustering data generated from intersecting non-linear manifolds. The algorithm uses approximate geodesic distances similar to ISOMAP, then a node-weighted MDS mapping is used to estimate local embeddings. These local embeddings are used to update weights for each data point proportional to the distance between the point and the estimated manifold. The weights and the embeddings are computed iter-

[1]This failure mode is addressed to some extent by (Hadid and Pietikinen 2003) and others.

atively until convergence. The estimation of geodesic distances fails when the clusters are widely separated. For this reason, k-manifolds is primarily motivated for intersecting manifolds. In this paper, we focus on the non-linear manifold case. For this reason, we will not discuss linear multi-manifold methods.

A good model for a dataset generated from a connected low dimensional manifold is the principal surface (Hastie 1984); a non-parametric non-linear manifold that passes through the *middle* of the data. The principal surface can also be described as a non-linear generalization of the principal subspace. However, computing principal surfaces of more than one dimension is computationally expensive. To address this difficulty, (Chang and Ghosh 2001) proposed the probabilistic principal surface (PPS); the basis of our model.

In this paper, we seek a principled algorithm for simultaneously learning a soft partition of high dimensional data and a local parametric model for the clusters where each cluster is a low dimensional non-linear manifold. In our proposed model (MiPPS), each cluster is described using a PPS and the entire dataset is modeled as a probabilistic mixture of these components. By analyzing the resulting complete log-likelihood, we learn soft cluster assignments and parametric models for each cluster.

2. Model Description

Our model is based on the assumption that the generating manifold of each cluster has a dimensionality $Q_m \leqslant D$ where D is the dimensionality of the data space $\mathcal{Y}$. We assume that this manifold can be described as the image of a smooth function $f_m(\mathbf{x})$ that maps a latent space $\mathcal{X}_m \mapsto \mathcal{Y}$. This assumption is valid for a large class of manifolds. A new sample is generated in four steps:

- One of the M clusters is selected with a probability $p(m) = \nu_m$.
- A sample $\mathbf{x}$ is generated from the selected latent space $\mathcal{X}_m$ where $\mathbf{x} \sim p(\mathbf{x}|m)$.
- The latent sample is mapped to the data space via some non-linear smooth function $f_m(\mathbf{x})$.
- The resulting data sample is corrupted by noise ϵ_m.

Using zero mean Gaussian noise, the distribution of each data sample $\mathbf{y}_n$ given a cluster selection m, the corresponding model parameters $\boldsymbol{\theta}_m$ and a latent sample $\mathbf{x}$ is:

$$p(\mathbf{y}_n|\mathbf{x}, \boldsymbol{\theta}_m) = \mathcal{N}(\mathbf{y}_n|f_m(\mathbf{x}), \boldsymbol{\Sigma}_m(\mathbf{x})), \tag{1}$$

where $\mathcal{N}(\mathbf{y}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ represents a Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$.

The PPS uses a radial basis function (RBF) network $f_m(\mathbf{x}) = \mathbf{W}_m\boldsymbol{\phi}(\mathbf{x})$ to define the mapping between the latent space and the data space. The weight matrix, $\mathbf{W}_m$ has dimension $D \times L$ and $\boldsymbol{\phi}(\mathbf{x}) = [1\ \phi_1(x) \cdots \phi_{L-1}(x)]^T$ is a vector of L latent basis functions: $\phi_l(x) : \mathbb{R}^{Q_m} \mapsto \mathbb{R}$. We use isotropic Gaussian radial basis functions; $\phi_l(x) = \exp(\frac{-\|\mathbf{x}-\mathbf{c}_l\|^2}{\sigma})$ given some centroid $\mathbf{c}_l$ and a length scale σ which we fix for all l. The topographic constraints enforced by this model are discussed in (Bishop, Svensen, and Williams 1998b).

The latent variable distribution $p(\mathbf{x}|m)$ must be propagated to the data space through the non-linear mapping $f_m(\mathbf{x})$ to describe the data distribution. The resulting data distribution is computed by marginalizing (1) over the latent space. Unfortunately, for general non-linear mappings, this integration is intractable. One solution is a discrete approximation of the integration by sampling over a uniform grid of K_m points in the latent space. The resulting distribution is a constrained mixture:

$$p(\mathbf{y}_n|\boldsymbol{\theta}_m) = \sum_{k=1}^{K_m} p(\mathbf{y}_n|\mathbf{x}_k, \boldsymbol{\theta}_m)p(\mathbf{x}_k|m). \tag{2}$$

The approximation can be computed to arbitrary accuracy by increasing the number of latent points with a corresponding increase in the computational cost.

The PPS model separates the noise variance into subspaces parallel and orthogonal to the data manifold with variances given by a_m and b_m respectively, where:

$$a_m = \frac{\alpha_m}{\beta_m}$$
$$b_m = \frac{D - \alpha_m Q_m}{\beta_m(D - Q_m)}.$$

The parameter $\alpha_m \in (0, D/Q_m)$ controls the amount of manifold alignment in the noise covariance while β_m controls the noise variance. When $\alpha_m \in (0, 1)$ the noise has a higher variance orthogonal to the manifold. As $\alpha_m \rightarrow 0$, the model satisfies the self consistency property for principal surfaces (Chang and Ghosh 2001). When $\alpha_m \in (1, D/Q_m)$, the noise has a higher variance parallel to the manifold, corresponding to the manifold aligned GTM model (Bishop, Svensen, and Williams 1998a). This is useful when one suspects that much of the sampling noise is tangential to the manifold. Finally, when $\alpha_m = 1$, the noise variance is isotropic and the GTM with $\boldsymbol{\Sigma}_m(\mathbf{x}) = \beta_m^{-1}\mathbf{I}$ is recovered. Chang and Ghosh showed that the PPS learns a more representative model than the GTM in many cases.

The subspaces parallel and orthogonal to the manifold are given by the orthonormal matrices $\mathbf{E}_{//m}(\mathbf{x})$ and $\mathbf{E}_{\perp m}(\mathbf{x})$ respectively. These can easily be computed using the partial derivatives of the the manifold mapping with respect to each latent dimension:

$$\mathbf{E}_{//m}(\mathbf{x}) = \left[\frac{\partial f_m(\mathbf{x})}{\partial x_1} \cdots \frac{\partial f_m(\mathbf{x})}{\partial x_{Q_m}}\right] = \mathbf{W}_m\mathbf{T}_m(\mathbf{x}),$$
$$\mathbf{E}_{\perp m}(\mathbf{x}) = [\mathbf{E}_{//m}(\mathbf{x})]^{\perp} = \mathbf{W}_m\mathbf{O}_m(\mathbf{x}), \tag{3}$$

where $\mathbf{T}_m(\mathbf{x}) = [\frac{\partial\boldsymbol{\phi}(\mathbf{x})}{\partial x_1} \cdots \frac{\partial\boldsymbol{\phi}(\mathbf{x})}{\partial x_{Q_m}}] \in \mathbb{R}^{L\times Q_m}$. The matrices $\mathbf{T}_m(\mathbf{x})$ and $\mathbf{O}_m(\mathbf{x})$ are constant and only need to be computed at initialization. Further, because $[\mathbf{T}_m(\mathbf{x})\ \mathbf{O}_m(\mathbf{x})]$ are a basis for $\mathbb{R}^D$, only $\mathbf{T}_m(\mathbf{x})$ is required to completely describe the model. Unfortunately, neither the rows of $\mathbf{W}_m$ nor the columns of $\mathbf{T}_m(\mathbf{x})$ are guaranteed to be orthonormal and so a further orthonormalization step such as the Gram-Schmidt procedure must be applied to $\mathbf{E}_{//m}(\mathbf{x})$. Now, the oriented covariance can be computed as:

$$\mathbf{\Sigma}_m(\mathbf{x}) = a_m \mathbf{E}_{//m}(\mathbf{x})\mathbf{E}_{//m}^T(\mathbf{x}) + b_m \mathbf{E}_{\perp m}(\mathbf{x})\mathbf{E}_{\perp m}^T(\mathbf{x}). \tag{4}$$

Note that the total noise variance measured as the sum of the eigenvalues of $\mathbf{\Sigma}_m(\mathbf{x})$ remains constant equal to $\frac{D}{\beta_m}$ regardless of varying values of α_m or the local covariance orientation.

For the remainder of this paper, we assume that all the data points are independent and identically distributed. The corresponding log likelihood is then a product over the likelihood of each data point:

$$\begin{aligned}\mathcal{L}_D(\mathbf{\Theta}) &= \ln \prod_{n=1}^{N} p(\mathbf{y}_n|\mathbf{X},\mathbf{\Theta})p(\mathbf{X}|\mathbf{\Theta}) \\ &= \sum_n \ln \sum_{m,k} p(m,\mathbf{x}_k)\mathcal{N}\left(\mathbf{y}_n|\mathbf{W}_m\boldsymbol{\phi}(\mathbf{x}_k), \mathbf{\Sigma}_m(\mathbf{x}_k)\right), \end{aligned} \tag{5}$$

where $\mathbf{\Theta} = \{\boldsymbol{\theta}_m\}_{m=1}^{M}$. This assumption can be relaxed to model dependent realizations such as a time series.

We also introduce a Bayesian prior over the weight parameter $\mathbf{W}_m$. For simplicity, we use a radially symmetric Gaussian prior of the form:

$$\begin{aligned} p(\mathbf{W}|\{\lambda_m\}) &= \sum_{i=1}^{M} p(\mathbf{W}_m|\lambda_m)\delta(i-m), \quad \text{where} \\ p(\mathbf{W}_m|\lambda_m) &= \left(\frac{\lambda_m}{2\pi}\right)^{MD/2} \exp\left\{-\frac{\lambda_m}{2}\|\mathbf{w}_m\|^2\right\}. \end{aligned} \tag{6}$$

λ_m is a hyper-parameter and $\mathbf{w}_m$ is the vectorized form of $\mathbf{W}_m$ (i.e. the matrix stacked column-wise into a single vector). Intuitively, this means we place an independent prior over each $\mathbf{W}_m$. This prior enforces our smoothness assumptions on the weight matrix by effectively regularizing the matrix norm. In summary, the full set of model parameters are: $\{\boldsymbol{\theta}_m\} = \{\mathbf{W}_m, \lambda_m, \alpha_m, \beta_m, \nu_m\}$.

3. Optimizing the Model Parameters

The form of the mapping and the likelihood cost function suggest optimization using the expectation maximization (EM) optimization technique; a co-ordinate ascent algorithm for maximizing data likelihood (Dempster, Laird, and Rubin 1977). The optimization proceeds by alternately computing the expected complete log-likelihood (E-step) then maximizing the expected complete log-likelihood with respect to the parameter selections (M-step) till convergence. The EM algorithm is guaranteed to increase the objective function at each iteration unless it is at a local optimum.

E-step

For the E step, the expectation of the complete log-likelihood is computed with respect to the distribution of the hidden variables given the data. Given a current parameter estimate $\mathbf{\Theta}$, the expected log-likelihood is given by , $\langle\mathcal{L}_c(\mathbf{\Theta})\rangle = \langle\mathcal{L}_D(\mathbf{\Theta})\rangle + \langle\mathcal{L}_W(\mathbf{\Theta})\rangle$ where $\mathcal{L}_D$ is the data likelihood and $\mathcal{L}_W$ is the weight parameter log-likelihood given by the log of (6). $\langle\mathcal{L}_D(\mathbf{\Theta})\rangle$ can be expressed as:

$$\sum_{n,m,k} r_{m,k}^n \left\{\ln p(\mathbf{y}_n|\boldsymbol{\theta}_m,\mathbf{x}_k) + \ln \nu_m + \ln p(\mathbf{x}_k|m)\right\}, \tag{7}$$

where $r_{m,k}^n$ is posterior probability (also called the responsibility) that the data sample $\mathbf{y}_n$ is generated from cluster m and latent variable $\mathbf{x}_k$. Using Bayes theorem,

$$\begin{aligned} r_{m,k}^n &= p(m,\mathbf{x}_k|\mathbf{y}_n) \\ &= \frac{p(m,\mathbf{x}_k)p(\mathbf{y}_n|\boldsymbol{\theta}_m,\mathbf{x}_k)}{\sum_{m',k'} p(m',\mathbf{x}_{k'})p(\mathbf{y}_n|\boldsymbol{\theta}_{m'},\mathbf{x}_{k'})}. \end{aligned} \tag{8}$$

In order to compute these probabilities, the determinant and inverse of the corresponding Gaussian covariances must be computed. The PPS model simplifies this computation ((Chang and Ghosh 2001), Proposition 2):

$$\begin{aligned} |\mathbf{\Sigma}_m(\mathbf{x})| &= a_m^{Q_m} b_m^{D-Q_m} \\ \mathbf{\Sigma}_m^{-1}(\mathbf{x}) &= \frac{1}{b_m}\mathbf{I} - \frac{(a_m - b_m)}{b_m a_m}\mathbf{E}_{//m}(\mathbf{x})\mathbf{E}_{//m}^T(\mathbf{x}). \end{aligned}$$

Hence, the cost of computing $|\mathbf{\Sigma}_m(\mathbf{x})|$ is reduced to a single product, while the cost of computing $\mathbf{\Sigma}_m^{-1}(\mathbf{x})$ is reduced from $\mathcal{O}(D^3)$ to $\mathcal{O}(Q_m D^2)$.

M-step

The M-step involves computing the parameter values $\mathbf{\Theta}^*$ that maximize the expected log-likelihood. This maximization generally involves the derivatives of the expected log-likelihood with respect to each of the parameters. We discuss the parameters in the order that they are updated.

Updating ν : First, differentiating the log likelihood with respect to ν_m and setting it to zero, ν_m is updated as:

$$\nu_m^* = \frac{1}{N}\sum_{n,k} r_{m,k}^n, \tag{9}$$

Updating $p(\mathbf{x}_k|m)$ (Optional): Differentiating the log likelihood with respect to $p(\mathbf{x}_k|m)$ and setting it to zero:

$$p(\mathbf{x}_k|m)^* = \frac{1}{N}\sum_n r_{m,k}^n. \tag{10}$$

In our implementation, we assume that the manifold is uniformly sampled. For this reason, we uniformly sample the latent space as well by fixing $p(\mathbf{x}_k|m) = \frac{1}{K_m}$ without update. This helps to avoid overfitting.

Updating W: We approximate $\mathbf{W}_m^*$ as the weight matrix that locally maximizes the complete log likelihood. The derivative $\frac{\partial\langle\mathcal{L}_C(\mathbf{\Theta})\rangle}{\partial\mathbf{W}_m}$ is:

$$\begin{aligned} \sum_{n,k} r_{m,k}^n \Bigg\{ & \left[\frac{1}{b_m}\mathbf{I} - \frac{a_m - b_m}{b_m a_m}\mathbf{W}_m\mathbf{\Psi}_k^m\mathbf{W}_m^T\right]\mathbf{q}_{m,k}^n\boldsymbol{\phi}(\mathbf{x}_k)^T \\ & + \frac{a_m - b_m}{b_m a_m}\left[\mathbf{q}_{m,k}^n\mathbf{q}_{m,k}^{n\,T}\mathbf{W}_m\mathbf{\Psi}_k^m\right] - \lambda_m\mathbf{W}_m \Bigg\}, \end{aligned} \tag{11}$$

where $\mathbf{q}^n_{m,k} = \mathbf{y}_n - \mathbf{W}_m\boldsymbol{\phi}(\mathbf{x}_k)$, $\boldsymbol{\Psi}^m_k = \mathbf{T}_m(\mathbf{x}_k)\mathbf{T}_m(\mathbf{x}_k)^T$. This derivative is a non-linear function of the weight matrix. For this reason, the updated weight matrix cannot be computed in closed form. Here we provide two approximations as suggested by (Chang and Ghosh 2001). The first option is to iteratively update $\mathbf{W}_m$ using steepest ascent with learning rate η to find a local maximum. At each iteration i,

$$\mathbf{W}^{i+1}_m = \mathbf{W}^i_m + \eta\frac{\partial\langle\mathcal{L}_C(\boldsymbol{\Theta})\rangle}{\partial\mathbf{W}_m}. \tag{12}$$

This formulation fits into the generalized EM (GEM) framework, and convergence to a local optimum is guaranteed (Dempster, Laird, and Rubin 1977). However, there is a high computational cost involved. The second option is to approximate the weighting matrix in the M-step by the weighting matrix obtained using a GTM i.e. assuming that the covariance: $\boldsymbol{\Sigma}_m(\mathbf{x}) = \beta_m^{-1}\mathbf{I}$ for the weight update. Setting $\alpha_m = 1$, (11) simplifies considerably to:

$$\sum_{n,k} r^n_{m,k}\left\{-\beta_m\left[\mathbf{W}_m\boldsymbol{\phi}(\mathbf{x}_k) - \mathbf{y}_n\right]\boldsymbol{\phi}(\mathbf{x}_k)^T - \lambda_m\mathbf{W}_m\right\}. \tag{13}$$

This can be solved for a fixed point to obtain the updated weight matrices $\{\mathbf{W}^*_m\}^M_{m=1}$ by least squares:

$$\left(\boldsymbol{\Phi}^T\mathbf{G}_m\boldsymbol{\Phi} + \frac{\lambda_m}{\beta_m}\mathbf{I}\right)\mathbf{W}^{*T}_m = \boldsymbol{\Phi}\mathbf{R}_m\mathbf{Y}. \tag{14}$$

where $\mathbf{Y}$ is the $N\times D$ data matrix with each row $\mathbf{Y}_{n,:} = \mathbf{y}^T_n$, $\boldsymbol{\Phi}$ is a $K_m \times L$ matrix with elements $\boldsymbol{\Phi}_{k,l} = \phi_l(\mathbf{x}_k)$, $\mathbf{R}_m$ is a $K_m \times N$ matrix with elements $\mathbf{R}_m(k', n') = r^{n'}_{m,k'}$ corresponding to the responsibility matrix for a fixed cluster m, and $\mathbf{G}_m$ is a $K_m \times K_m$ diagonal matrix with elements $\mathbf{G}_m(k', k') = \sum^N_{n=1} r^n_{m,k'}$. This approximation is no longer guaranteed to increase the likelihood, but it yields good results in practice.

Log Evidence Approximation: Incorporating the Bayesian prior over the weights (6), the marginal data likelihood is computed by integrating over the weight matrix:

$$p(\{\mathbf{y}_n\}|\boldsymbol{\Theta}') = \int p(\{\mathbf{y}_n\}|\boldsymbol{\Theta})p(\mathbf{W}|\{\lambda_m\}). \tag{15}$$

Where $\boldsymbol{\Theta}' = \{\boldsymbol{\theta}'\}^M_{m=1}$ is the set of parameters without $\{\mathbf{W}_m\}$. This integration is intractable, however, we employ a quadratic approximation as suggested by (MacKay 1991). The idea is to approximate the full posterior using a Gaussian approximation about some mode, $\mathbf{W}^*$. We define the function:

$$S(\boldsymbol{\theta}'_m, \mathbf{w}) = -\log\{p(\{\mathbf{y}_n\}|\boldsymbol{\theta}_m)p(\mathbf{W}_m|\lambda_m)\}. \tag{16}$$

This can be used to approximate (15) as:

$$\begin{aligned} p(\{\mathbf{y}_n\}|\boldsymbol{\theta}'_m) &= \int e^{\{-S(\boldsymbol{\theta}'_m,\mathbf{w})\}}d\mathbf{w} \\ &\approx e^{\{-S(\boldsymbol{\theta}'_m,\mathbf{w}^*)\}}\int e^{\left\{-\frac{1}{2}(\mathbf{w}_m-\mathbf{w}^*_m)^T\mathbf{H}_m(\mathbf{w}_m-\mathbf{w}^*_m)\right\}}d\mathbf{w} \\ &= e^{\{-S(\boldsymbol{\theta}'_m,\mathbf{w}^*)\}}(2\pi)^{MD/2}|\mathbf{H}_m|^{-\frac{1}{2}} \end{aligned} \tag{17}$$

using a second order Taylor expansion of the log of the integrand. The Hessian matrix $\mathbf{H}_m$ is given by the second derivative of S with respect to $\mathbf{w}_m$. The derivative involves computing terms that are quadratic in $\mathbf{w}_m$ which makes the computation expensive. For this reason, we employ a GTM approximation once more as described in (Bishop, Svensen, and Williams 1998a) . The resulting $\mathbf{H}_m$ matrix is block diagonal with each block of the form $\beta_m\boldsymbol{\Phi}^T\mathbf{G}_m\boldsymbol{\Phi}$. Now the log evidence, $\log p(\{\mathbf{y}_n\}|\boldsymbol{\theta}_m)$ for each m can be computed as:

$$\mathcal{L}_D(\boldsymbol{\theta}_m)|_{\mathbf{w}^*_m} - \frac{\lambda_m}{2}\|\mathbf{w}^*_m\|^2 - \frac{1}{2}|\mathbf{H}_m| + \frac{MD}{2}\log\lambda_m. \tag{18}$$

Updating λ and β: Next, we use (18) to update the parameters λ_m and β_m by finding fixed points of the corresponding derivatives. Assuming some constant $\mathbf{w}^*_m$, the updates are given by:

$$\lambda_m = \frac{\gamma}{\|\mathbf{w}^*_m\|^2}, \tag{19}$$

$$\frac{1}{\beta^*_m} = \frac{1}{\nu^*_m}\frac{1}{ND-\gamma}\sum_{n,k} r^n_{m,k} \left\{\frac{D-Q_m}{(D-\alpha_m Q_m)}\mathbf{d}^{n,1}_{m,k} - \frac{(\alpha_m-1)D}{\alpha_m(D-\alpha_m)Q_m}\mathbf{d}^{n,2}_{m,k}\right\}, \tag{20}$$

where $\mathbf{d}^{n,1}_{m,k} = \|\mathbf{q}^n_{m,k}\|^2$, $\mathbf{d}^{n,2}_{m,k} = \mathbf{q}^{n\,T}_{m,k}\mathbf{W}^*_m\boldsymbol{\Psi}^m_k\mathbf{W}^{*T}_m\mathbf{q}^n_{m,k}$. The parameter γ can be interpreted as the *effective* number of parameters and is given by:

$$\gamma = \sum^{MD}_{i=1}\frac{\zeta_i - \alpha_m}{\zeta_i}, \tag{21}$$

where ζ_i are the eigenvalues of $\mathbf{H}_m$.

Updating α: The derivative of the log likelihood with respect to α_m leads to a cubic function in α_m given by:

$$\alpha^3_m x_3 + \alpha^2_m x_2 + \alpha_m x_1 + x_0 = 0, \tag{22}$$

with coefficients:

$$\begin{aligned} x_3 &= \nu^*_m N Q^2_m D, \\ x_2 &= -\nu^*_m N Q_m D(D+Q_m) \\ &\quad - \beta^*_m(D-Q_m)\mathbf{v}_{1,m} - \beta^*_m D Q_m \mathbf{v}_{2,m}, \\ x_1 &= \nu_m N Q_m D^2 + 2\beta^*_m D Q_m \mathbf{v}_{2,m}, \\ x_0 &= -\beta^*_m D\mathbf{v}_{2,m}, \end{aligned}$$

where $\mathbf{v}_{1,m} = \sum_{n,k} r^n_{m,k}\mathbf{d}^{n,1}_{m,k}$ and $\mathbf{v}_{2,m} = \sum_{n,k} r^n_{m,k}\mathbf{d}^{n,2}_{m,k}$.

Ideally, the likelihood should be evaluated for each possible solution of the cubic polynomial. However, this is expensive. In practice, we pick the solution α^*_mthat is closest 1. By automatically updating α_m, we are able to avoid a potentially more expensive cross validation as used by (Chang and Ghosh 2001).

The E-Step and M-step are alternated until the expected log-likelihood or the parameter estimates do not change significantly over consecutive iterations. At convergence, the data manifolds are given by the learned parameters $\mathbf{\Theta}$ and the soft clustering is encoded in the resulting responsibility (8). A hard clustering can be computed by selecting the most likely cluster:

$$m^* = \underset{m \in \{1 \ldots M\}}{\arg\max} \sum_k r_{m,k}^n \,. \tag{23}$$

One advantage of the EM algorithm is that it can be trained incrementally by feeding data samples in batches or one at a time (Dempster, Laird, and Rubin 1977). This is useful when dealing with large data-sets.

Initialization and parameter selection

The EM algorithm is guaranteed to locally maximize the log-likelihood, however, there is no guarantee of finding a global maximum. For this reason, initialization of the model parameters is critical to the success of the algorithm. In our experiments, we used our prior assumptions to bias the optimization close to a good optimum. We also noticed that the parameters $\{\alpha_m, \lambda_m\}$ that were automatically updated were less sensitive to initialization. Thus we were able to avoid more expensive cross validation.

First we found a hard clustering of the data using our prior assumptions about the form of the final clusters. For instance, when the clusters were expected to be approximately convex, we found that k-means found a good initial cluster estimate. For more complicated clusters, we initialized using hierarchical clustering. Other initial clustering methods that satisfy prior assumptions can also be used. The probability of each cluster, ν_m was then initialized as the quotient of the number of points in cluster m and N.

The manifold estimates were initialized based on the amount of non-linearity assumed in each generating manifold. If the generating manifold was expected to have a low curvature, we initialized our estimate of the manifold to approximate PCA in each cluster by solving a simple least squares problem (Bishop, Svensen, and Williams 1998b). When the manifold was assumed to have a high curvature, each cluster was embedded using some non-linear dimensionality reduction algorithm; in our case, ISOMAP. The weights were then initialized approximate the mapping from the embedded points to the data space.

The noise inverse variance β_m was initialized as the minimum of the inverse variance *lost* in the initial clustering and average spacing of the projected grid. For PCA, the variance lost can be approximated by the $Q_m + 1$ eigenvalue of the data covariance matrix. The parameter α_m was initialized to 1 and λ_m was initialized to 0.01 though in practice, we found that the initial values were not critical.

The latent space was selected as a Q_m dimensional hypercube $[-1, 1]^{Q_m}$. However, when the manifold was expected to be closed, we modelled the latent space as a unit hypersphere $S^{Q_m} \in \mathbb{R}^{Q_m+1}$. K_m latent points were generated by sampling uniformly over this latent space. The number of latent points is only limited by computational constraints. A large K_m improves the modeling performance. On the other hand, if the latent space is insufficiently sampled, the PPS centers in the data space will lie too far away from one another and the mapping loses the topological constraints. The centers of the RBF were sampled on a grid over $\mathcal{X}_m$. We initialized both K_m and L to $0.05N$, suitably rounded. The length scale of the RBF; σ was the only cross validation parameter we used in our experiments. This parameter was set to a small multiple of the distance between the RBF means.

MiPPS assumes that the dimensions of all the local clusters Q_m are known. We used the estimate of Q_m learned by the maximum likelihood method described in (Levina and Bickel 2005) which we found agreed with our prior intuition about the dimensionality of the component manifolds.

4. Local and Global Visualization

A functional model for the component manifolds can be found directly using the learned RBF mapping from a continuous latent space: $\mathbf{W}_m \boldsymbol{\phi}(\mathbf{x}) \; \forall \mathbf{x} \in \mathcal{X}_m$. An alternative is to model the component manifolds using linear interpolation between the mapped latent points $\{\mathbf{W}_m \boldsymbol{\phi}(\mathbf{x}_k)\}_{k=1}^{K_m}$. If desired, splines or other non-linear interpolation techniques can also be used.

If the latent space is one or two dimensional, the data set can be visualized by projections to each local manifold using the posterior distribution:

$$p(\mathbf{x}_k|\mathbf{y}_n, m) = \frac{p(\mathbf{x}_k|m)p(\mathbf{y}_n|m, \mathbf{x}_k)}{\sum_{k'=1}^{K_m} p(\mathbf{x}_{k'}|m)p(\mathbf{y}_n|m, \mathbf{x}_{k'})}. \tag{24}$$

The mean of the projected point is:

$$\langle \mathbf{x}|\mathbf{y}_n, m \rangle = \sum_{k=1}^{K_m} \mathbf{x}_k p(\mathbf{x}_k|\mathbf{y}_n, m). \tag{25}$$

The mode can also be used if the posterior is expected to be multi-modal. This projection method gives m local visualizations.

The PPS can be interpreted as a constrained Mixture of Factor Analyzers (MFA) model (Tipping and Bishop 1999) where each of the MFA means are constrained to lie on some non-linear manifold, and the noise covariances are given by gradients and globally shared parameters. This is in contrast to the general MFA where each local subspace is computed to approximate the local principal components of the data. The effect of this constraint is that the local subspaces are computed using the mapping function as $\mathbf{E}_{//m}(\mathbf{x})$ avoiding several singular vector computations. The noise variance parameters are also shared along the entire manifold. This can be a useful property when the data is high dimensional and sparse as we will discuss in Section 5.The use of gradients to define the subspaces also naturally aligns the subspaces avoiding the degeneracies addressed by (Verbeek, Vlassis, and Kröse 2002). We will exploit this MFA interpretation to define a global embedding.

Teh and Roweis developed an alignment algorithm for manifolds described using several local linear projections called the local linear co-ordination (LLC) algorithm (Teh and Roweis 2003). First the model learns a mixture of factor analyzers to describe the data manifold. Next the algorithm finds a global embedding of these local representations using only linear transformations of the local projections and the computed responsibilities. By interpreting the learned MiPPS model as a MFA, the LLC algorithm can be applied directly to find a global embedding for visualization. MiPPS local co-ordinates are given by the projection of the data to each local subspace. $\mathbf{z}_{n,k,m} = E_{//m}(x_k)\mathbf{y}_n$ and the responsibilities are given by (8). We omit the details of the algorithm implementation due to space constraints. We also note that LLC does not require that the local projections $\mathbf{z}_{n,k,m}$ be two dimensional in order to find a global two-dimensional embedding. Intuitively, MiPPS and LLC can be used to learn an unsupervised embedding of the dataset guided by the clustering. This leads to a visualization that respects the topology of the data, but is also biased to preserve the learned clusters.

5. Results

We now present results applying MiPPS to three toy data sets. In all the synthetic data experiments, the manifolds are sampled in Gaussian noise and are represented by linearly interpolating the MiPPS means. MiPPS was used to learn the partitioned S-curve randomly sampled in high noise (Fig. 1). The algorithm was initialized using k-means to find two clusters. The weights were initialized to approximate PCA on each cluster. The noisy double helix (Fig. 2)

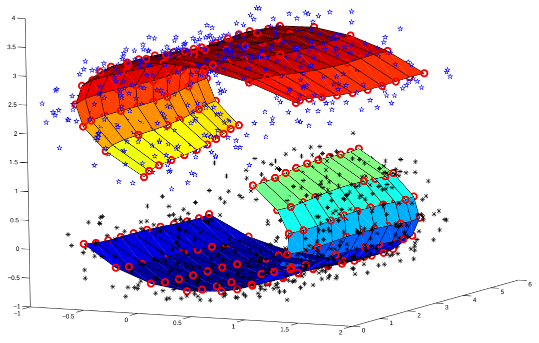

Figure 1: Clustering the partitioned S-curve.

was more challenging because of the intertwined manifolds. Initialization using k-means led to the top and bottom halves being separated and learned as the component manifolds and MiPPS was not able to escape this local maximum. Instead, we used hierarchical clustering to learn a more representative partition. The weights were initialized to approximate the ISOMAP embedding. We also attempted to learn two intersecting manifolds for the intersecting spirals data set (Souvenir 2006). Here, the prior information given was that the manifolds intersect. An initial clustering step could be used to estimate the intersecting clusters. However, we found that the clustering step was not necessary for good results. The model shown in Fig. 3 was learned when

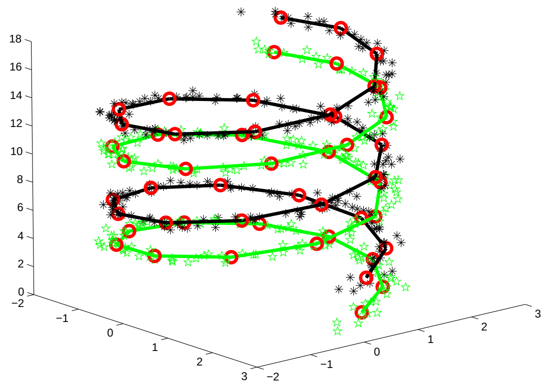

Figure 2: Clustering the double helix.

the weights were initialized to approximate each of the two principal components of the entire data set.

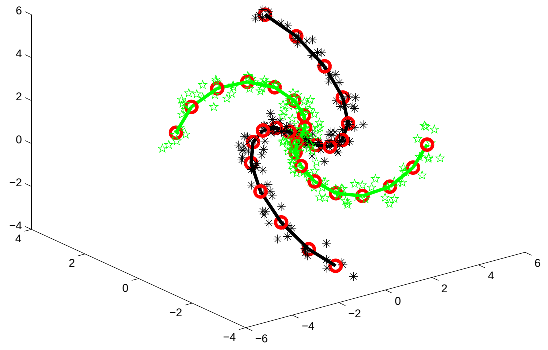

Figure 3: Clustering two intersecting spirals.

Hyperspectral Data

Hyperspectral data consists of sensor measurements by satellites or measurement aircraft measured over various locations in some region. Each feature vector consists of hundreds of frequency bands providing fine spectral information about the measured area. In our experiments, we used Hyperion data acquired by the NASA EO-1 satellite over the Okavango Delta, Botswana in May 2001. Each data point consists of $D = 145$ bands. The land cover is classified into 9 types by domain experts.

We selected three of the most populous classes as shown in Table 1. The total number of data samples was $N = 1239$ so this dataset is high dimensional but sparse. The measured classes are known to vary over location, so intuitively, there

Class	Number of samples
Primary floodplain	437
Riparian	448
Firescar.	354

Table 1: Class Descriptions.

should be two degrees of freedom in the data corresponding to the location of the measurement. The maximum likelihood dimension estimate (Levina and Bickel 2005) of the data was 2.3589 matching our intuition. Correspondingly, we set $Q_m = 2$ for each cluster. MiPPS was initialized using the clusters found by a mixture of PPCA. The weights were learned to approximate PCA in each cluster.

We separated the data randomly into training/test pairs with the percentage of training data between 10% and 100%. We recorded the performance of each model on the test set over 200 iterations using test error and average negative conditional log likelihood (ANCLL) as performance metrics. ANCLL is computed as $-\frac{1}{N}\sum \log p(\hat{c}|y_n)$ where $\hat{c}$ is the true class of y_n. The ANCLL is a good metric for probabilistic algorithms as it gives a sense of how far the model strays from the correct cluster selection. For 100% *training data*, we report the results from the training data only. For both metrics, lower values indicate better performance. The RBF length scale parameter was selected by cross validation using 6 values over a logarithmic scale from $10^{-0.5}$ to $10^{0.8}$ multiplied by the average distance between the latent nodes. We report the results for the length scale with the lowest training error averaged over all iterations.

We compared the performance of MiPPS to related unsupervised algorithms. For co-located measurements, the multivariate Gaussian distribution is often used by domain experts as a good class dependent feature model for hyperspectral data (Manolakis et al. 2001). However, as the measurement locations are varied, the Gaussian model becomes less accurate. This motivated the use of a Gaussian mixture model as the base model for unsupervised clustering of the data. Unfortunately, we were unable to learn a Gaussian mixture model as the data is sparse. Instead, we used a mixture of two dimensional factor analyzers. We also compared the algorithm performance to a mixture of GTM initialized identically to MiPPS. This was achieved by fixing each $\alpha_m = 1$. We attempted to cluster the dataset using k-manifolds algorithm. Unfortunately, the training process failed to converge due to widely separated cluster manifolds. This suggests that the clusters did not clearly intersect as assumed by k-manifolds.

All three algorithms performed well in clustering the data set using test error and ANCLL as the performance metrics. The results reported are the average performance over 200 iterations. As shown in Fig. 4 and Fig. 5, MiPPS outperformed the MFA and mixture of GTM algorithm. It was also clear that the mixture of GTM algorithm gave a high probability to the correct clusters while the MFA model was less certain about the selections. Surprisingly, the MFA outperformed the mixture of GTM in terms of test error until about 60% of the data was used for training.

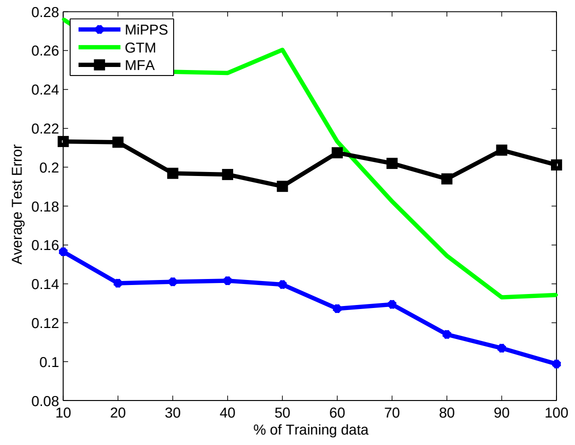

Figure 4: Average Test error

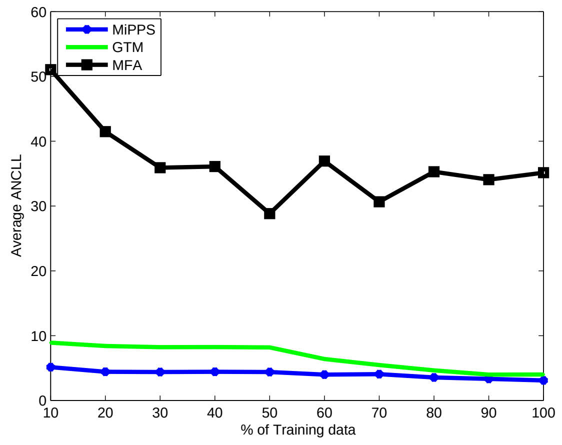

Figure 5: Average ANCLL

Finally, we applied the LLC algorithm to find a global embedding. LLC requires a nearest neighbor parameter which we set as 10 though we found that the parameter did not have much effect on the global embedding learned. Fig. 6 shows one embedding learned for the dataset. The embedding shows good separation by class. The model learned had a likelihood of -1.04×10^6, a test error of 0.03 and an ANCLL of 1.638 with $\alpha = [.6144\ .6315\ .6374]$ suggesting that much of the noise was orthogonal to the estimated generating manifold.

In contrast, the LLC algorithm initialized by MFA learned the embedding shown in Fig. 7. The LLC visualization was unable to separate the *Primary Floodplain* and *Firescar* classes. We were only able to fit up to 20 MFA's to the data due to sparsity. This highlights another advantage of the MiPPS model combined with LLC for data visualization as it can facilitate learning a detailed model even when the data is too sparse for less constrained models.

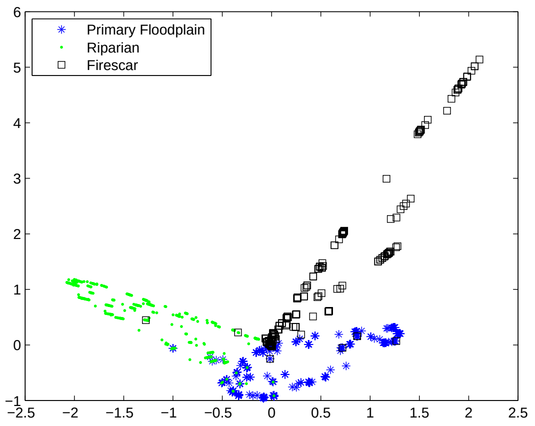

Figure 6: Global embedding learned by MiPPS and LLC.

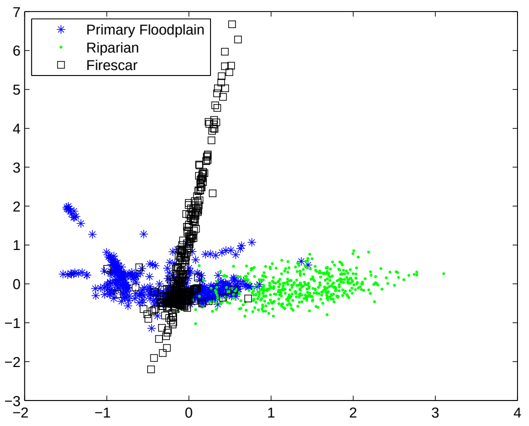

Figure 7: Global embedding learned by MFA and LLC.

6. Conclusion

In this paper, we discussed the multi-manifold clustering problem. We described MiPPS: a principled probabilistic technique for manifold clustering. We derived the data likelihood and showed that this model can be trained using an EM algorithm. We showed the use of a Bayesian prior and showed that the resulting model can be optimized using simple approximations. We also showed how one can bias the algorithm towards a good optimum by initializing based on prior intuition on the form of the data clusters while minimizing the amount of cross validation required to choose parameters.

We explored the versatility of the model to show that we can model a large class of manifolds in a principled manner. The use of a probabilistic model allowed us to learn in uncertainty such as when the component manifolds intersected or were sampled in high noise. We also explicitly computed a model for the manifold clusters which we used lo learn a global two dimensional visualization of the dataset. We demonstrated this technique in separating and visualizing land cover types in hyperspectral data.

Acknowledgments: This research was supported in part by NSF (Grant IIS-0705836).

References

Bishop, C. M.; Svensen, M.; and Williams, C. K. I. 1998a. Developments of the generative topographic mapping. *Neurocomputing* 21:203–224.

Bishop, C. M.; Svensen, M.; and Williams, C. K. I. 1998b. GTM: The generative topographic mapping. *Neural Computation* 10:215–234.

Chang, K., and Ghosh, J. 2001. A unified model for probabilistic principal surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23:22–41.

Dempster, A. P.; Laird, N. M.; and Rubin, D. B. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society* B(39):1–38.

Hadid, A., and Pietikinen, M. 2003. Efficient locally linear embeddings of imperfect manifolds. In *Machine Learning and Data Mining in Pattern Recognition*, volume 2734 of *Lecture Notes in Computer Science*.

Hastie, T. 1984. *Principal curves and Surfaces*. Ph.D. Dissertation, Department of Statistics, Stanford University.

Levina, E., and Bickel, P. J. 2005. Maximum likelihood estimation of intrinsic dimension. In *NIPS 17*. Cambridge, MA: MIT Press. 777–784.

MacKay, D. J. 1991. Bayesian interpolation. *Neural Computation* 4:415–447.

Manolakis, D. G.; Marden, D.; Kerekes, J. P.; and Shaw, G. A. 2001. Statistics of hyperspectral imaging data. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 4381, 308–316.

Souvenir, R. M. 2006. *Manifold learning for natural image sets*. Ph.D. Dissertation, Department of Computer Science, Washington University.

Teh, Y. W., and Roweis, S. 2003. Automatic alignment of local representations. *NIPS* 15:841–848.

Tino, P., and Nabney, I. 2002. Hierarchical GTM: Constructing localized nonlinear projection manifolds in a principled way. *IEEE Transactions on Pattern Analysis Machine Intelligence* 24(5):639–656.

Tipping, M. E., and Bishop, C. M. 1999. Mixtures of probabilistic principal component analyzers. *Neural Computation* 11:443–482.

Verbeek, J.; Vlassis, N.; and Kröse, B. 2002. Coordinating principal component analyzers. In *Proceedings of the International Conference on Artificial Neural Networks*.

Yankov, D., and Keogh, E. J. 2006. Manifold clustering of shapes. In *Proceedings of the International Conference on Data Mining*.