2012 IEEE 24th International Conference on Tools with Artificial Intelligence

Latent Beta Topographic Mapping

Kirthevasan Kandasamy
Dept. of Electronic & Telecommunication Engineering
University of Moratuwa, Sri Lanka
kirthevasankandasamy@ieee.org

***Abstract*— This paper describes Latent Beta Topographic Mapping (LBTM), a generative probability model for non linear dimensionality reduction and density estimation. LBTM is based on Generative Topographic Mapping (GTM) and hence inherits its ability to map complex non linear manifolds. However, the GTM is limited in its ability to reliably estimate sophisticated densities on the manifold. This paper explores the possibilities of learning a probability distribution for the data on the lower dimensional latent space. Learning a distribution helps not only in density estimation but also in maintaining topographic structure. In addition, LBTM provides useful methods for sampling, inference and visualization of high dimensional data. Experimental results indicate that LBTM can reliably learn the structure and distribution of the data and is competitive with existing methods for dimensionality reduction and density estimation.**

***Keywords*—Non linear dimensionality reduction, Manifold mapping, Density Estimation, Generative Topographic Mapping**

I. Introduction

The curse of dimensionality is a recurrent problem in machine learning and many related fields. Obtaining a compact representation of high dimensional data is often a necessary preprocessing step in many algorithms. Dimensionality reduction algorithms compress data by abstracting patterns and retaining only important information. Though there are efficient methods for linear dimensionality reduction, Non Linear Dimensionality Reduction (NLDR) remains one of the challenging problems in unsupervised learning.

Classical methods such as Principal Components Analysis (PCA), Factor Analysis and Multi Dimensional Scaling can only capture linear structure in the data. Most Neural Network based methods [1], [2] suffer from not having mathematically well defined optimization objectives.

Mixture of Probabilistic Principal Component Analyzers (MPPCA), Tipping and Bishop [3] and Mixture of Factor Analyzers (MFA), Ghahramani and Hinton [4], are generative latent variable models that perform local dimensionality reduction. Local dimensionality reduction is a powerful concept [4]. Both models have been successfully used in many density estimation applications. However, it is important to note that they do not produce a global lower dimensional embedding of all points. This implies, among other things, that they cannot be used as a preprocessing technique to obtain a lower dimensional representation of data.

Non-parametric, non-probabilistic formulations for NLDR such as Locally Linear Embedding (LLE) [5], Laplacian Eigenmaps [6] and Isomap [7] are promising methods that have gained in popularity due to their computational simplicity. A favorable property in these algorithms is that they have a convex optimization objective. Also, unlike MPPCA and MFA they allow studying global structure in data. However, the performance of these models is poor in the presence of noise [8] and multiple separated clusters [9]. In addition, they have two critical shortcomings. First, there is no natural transformation from points in the data space to the latent space and vice versa. Hence while these algorithms can return a lower dimensional embedding of the training set, they cannot handle out of sample points. Second, as they are not probabilistic models, density estimation and data generation are not straight forward. Methods to work around these problems either require the initial data set be present, an expensive eigenvector calculation or an unnatural extension to the model [10], [11].

An interesting formulation for NLDR is the Gaussian Process Latent Variable Model (GPLVM) [12]. GPLVM is a generative model and assumes a transformation from the latent space to the data space. Unlike conventional models the likelihood function in GPLVM is obtained by marginalizing over the parameters and not the latent variables. While this approach may produce coherent lower dimensional embeddings, it provides no inverse function mapping points in the data space to the latent space and hence cannot handle out of sample data. Also, though data may be sampled using a GPLVM, the sampled points will generally not be independent which limits its use as a data generator.

In the light of the discussion above, we list the following as desirable characteristics of an NLDR algorithm. **(a) Global structure:** Producing a global lower dimensional embedding of data. **(b) Bijective mapping:** Should have a natural mapping from the data space to the latent space and vice versa. **(c) Density estimation:** Should be able to learn a density model for the data on the manifold. **(d) Data Generation (Sampling):** After learning one should be able to use the model to sample independent data points that replicate and generalize the distribution of the training set. This property is desirable because it is often needed to generate data (e.g. creating additional training data for learning, approximate inference in complex models). **(e) Convex Objective:** The algorithm should guarantee a globally optimum solution.

Generative Topographic Mapping, Bishop *et al.* [13] is a manifold mapping model that addresses concerns (a) - (d) discussed above. It can learn complex manifolds while preserving topographic structure and is less sensitive to noise

1082-3409/12 $26.00 © 2012 IEEE
DOI 10.1109/ICTAI.2012.27

TABLE I

COMPARISON OF DIFFERENT MODELS OF NLDR

	Global embedding	$f(\mathbf{y})$	$f^{-1}(\mathbf{x})$	Probabilistic
LLE/ Laplacian / Isomap	$\surd$			
MFA/ MPPCA		$\surd^1$		$\surd$
GPLVM	$\surd$	$\surd$		$\surd$
GTM/ LBTM	$\surd$	$\surd$	$\surd^2$	$\surd$

Comparison of properties of different NLDR algorithms. Does the algorithm produce a global embedding of data (Col1)?. Is there a transformation from the latent space to the data space (Col2) and vice versa (Col3)? Is the method probabilistic (Col4)?.

[1] MFA & MPPCA produce multiple latent spaces and a point in any of them can be mapped to the data space.

[2] There is no direct inversion but there exists a function that has a valid interpretation as an inverse mapping.

and outliers. However, the GTM is severely limited because it assumes a uniform distribution on the latent space. As a result it fails as a density estimation technique.

This paper proposes a method to learn a distribution for the latent space in GTM. Learning a distribution also helps in achieving robust topographic ordering. The resulting algorithm, Latent Beta Topographic Mapping can reliably learn non-linear lower dimensional structure of high dimensional data and a distribution for the data on the manifold. LBTM is trained using Expectation Maximization (EM) and is computationally more expensive than the GTM. However, once trained, inference and sampling is simple and efficient. Like most EM formulations, LBTM suffers from having multiple local maxima. However, we take a look at a different initialization technique based on nonparametric methods that ensures a near global optimum at convergence.

This paper begins with a brief review of GTM in Section II. Section III presents the LBTM algorithm in detail. Section IV discusses results on popular datasets.

II. GENERATIVE TOPOGRAPHIC MAPPING

An L-dimensional discrete latent space consists of N_L latent nodes, usually assumed to exist on a regular grid. The GTM maps points in the latent space to the D-dimensional data space ($L < D$). To effect this mapping we use an RBF network of M non-linear basis functions. The basis functions are Gaussians centered at M fixed points in the latent space (also usually taken to lie on a regular grid). A typical choice for a basis function centered on the point ν_r is,

$$\phi_r(y) = \exp\left(-\frac{(y-\nu_r)^2}{2\sigma^2}\right) \tag{1}$$

where σ determines the spread of the basis function. Given M such points $\{\nu_1, \ldots \nu_M\}$ we may accordingly define the following vector valued function $\phi(y) \in \mathbb{R}^M$.

$$\phi(y) = \{\phi_r(y) | r = 1, \ldots, M\} \tag{2}$$

Selecting a suitable number of basis functions M and width parameter σ (which is generally taken to be the same for all ϕ_r) are important. When M is small and/or σ is large, the manifold tends to be smoother. But as the map is reluctant to warp itself, it might fail to capture the lower dimensional structure of the data. Conversely, if M is too large and/or σ is too small then it might result in twisted maps and hence poor topographic ordering. Consequently, lower dimensional coordinates obtained using such maps will not be meaningful representations of the data.

A weight matrix $\mathbf{W} \in \mathbb{R}^{D \times M}$ maps a point y in the latent space to the data space. The mapping is given by,

$$f(y; \mathbf{W}) = \mathbf{W}\phi(y) \tag{3}$$

During data generation all latent nodes are sampled with equal probability. The distribution of a point $\mathbf{x}$ in the data space, conditioned on a latent node $\mathbf{y}$ is a radially symmetric Gaussian distribution centered on $f(\mathbf{y})$ and covariance matrix $\frac{1}{\eta^2}\mathbf{I_D}$. The probability distribution for $\mathbf{x}$ is obtained by summing over all N_L latent nodes. The GTM can hence be viewed as a constrained mixture of isotropic Gaussians.

The parameters $\mathbf{W}$ and η are learned via EM. The log likelihood function $l(\mathbf{W}, \eta)$ and the lower bound on the log likelihood $l_b(\mathbf{W}, \eta)$ constructed by applying Jensen's inequality are given by

$$\begin{aligned} l(\mathbf{W}, \eta) &= \sum_{i=1}^{m} \log \sum_{j=1}^{N_L} p(\mathbf{x}^{(i)}, \mathbf{y}^{(j)}; \mathbf{W}, \eta) \\ l_b(\mathbf{W}, \eta) &= \sum_{i=1}^{m} \sum_{j=1}^{N_L} Q_i(\mathbf{y}^{(j)}) \log \frac{p(\mathbf{x}^{(i)}, \mathbf{y}^{(j)}; \mathbf{W}, \eta)}{Q_i(\mathbf{y}^{(j)})} \end{aligned} \tag{4}$$

where $\{\mathbf{x}^{(i)} | i = 1, 2, \ldots, m\}$ is the training set and $\{\mathbf{y}^{(j)} | j = 1, 2, \ldots, N_L\}$ is the set of latent nodes.

In the E-step we assign the responsibilities $Q_i(\mathbf{y}^{(j)}) = p(\mathbf{y}^{(j)} | \mathbf{x}^{(i)}; \mathbf{W}, \eta)$ based on the current guesses for $\mathbf{W}$ and η. In the M-step we update $\mathbf{W}$ and η as follows,

$$\begin{aligned} (\Phi^T G \Phi)\mathbf{W}_{new}^T &= \Phi^T R X \\ \frac{1}{\eta_{new}} &= \frac{1}{mD} \sum_{i=1}^{m} \sum_{j=1}^{N_L} Q_i(\mathbf{y}^{(j)}) || f(\mathbf{y}^{(j)}) - \mathbf{x}^{(i)} ||^2 \end{aligned} \tag{5}$$

In the above expression Φ is an $N_L \times M$ matrix where $\Phi_{ij} = \phi_j(\mathbf{y}^{(i)})$. G is an $N_L \times N_L$ diagonal matrix where $G_{jj} = \sum_i Q_i(\mathbf{y}^{(j)})$ and R is an $m \times N_L$ matrix where $R_{ij} = Q_i(\mathbf{y}^{(j)})$. X is the $m \times D$ data matrix where each row corresponds to a point in the data set.

A concern with point representations for the latent space is that the number of points (N_L), needed to obtain a good embedding increases exponentially with the latent space dimensionality (L) [13], [14]. However, in [12] it was shown that the difficulty in modeling a probability distribution on a lower dimensional manifold was in propagating this distribution through a nonlinear mapping to the data space. In that context, it was argued that point representations were useful because each point can be easily propagated through the nonlinear mapping. Therefore, despite its drawbacks and computational complexity there is a strong case for the GTM as a model for NLDR. Table I summarizes some of the advantages of using the GTM over other approaches.

One of the shortcomings of the GTM is the assumption that all latent nodes are equally likely during data generation. Generally, this means that we will not be able to learn a faithful density model on the manifold. Our manifold might be able to capture topographic structure in the data and correctly map existing points in the dataset. However, it cannot be used for inference and will be useless as a data generator. It is possible to improve density estimation in the GTM by using more basis functions. This however, results in twisted maps [13].

III. Latent Beta Topographic Mapping

A. Model

In the LBTM, the transformation from the latent space to the data space is effected via the GTM. While the basic GTM framework is necessarily present in LBTM, introducing a probability distribution on the latent space overcomes the limitations of GTM in faithfully estimating densities.

The latent space probability distribution is achieved via a finite mixture model. This way, by varying the number of mixtures we can control the degrees of freedom to ensure that the model only learns credible patterns in the data. Our model posits that during the data generation process, a latent node $\mathbf{y}$ was drawn from one of N_P mixtures depending on a second latent variable $\mathbf{z}$. $\mathbf{x}$ is then drawn from an isotropic Gaussian centered on $f(\mathbf{y})$ just as in the GTM (3).

We choose $p(\mathbf{z})$ to be a multinomial on the N_P-1 simplex. For $p(\mathbf{y}|\mathbf{z})$ we require a discrete probability distribution with finite support on L dimensions. We achieve this by constructing a multi-dimensional beta binomial distribution using L independent beta binomial distributions.

The beta binomial is a discrete probability distribution on the support $\{0, 1, \ldots, N\}$. The positive parameters α and β determine its precision $s = \alpha + \beta$ and mean $N\alpha/s$. Depending on the values of α and β the mass function can assume a variety of different shapes, which is convenient for our purpose. The probability mass function of the beta binomial for $k = 0, 1, \ldots, N$ is given by,

$$p_{bb}(k; \alpha, \beta, N) = \binom{N}{k} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \frac{\Gamma(\alpha+k)\Gamma(\beta+N-k)}{\Gamma(\alpha+\beta+N)} \tag{6}$$

Each mixture in our model consists of L independent beta binomials, one per dimension. If our latent nodes are arranged in a $n_1 \times n_2 \times \cdots \times n_L$ grid (hence $N_L = n_1 n_2 \ldots n_L$), the probability of the latent node u whose coordinates in the grid are given by $(a_1, a_2, \ldots, a_L)$ is

$$p_{BB}(u) = \prod_{l=1}^{L} p_{bb}(a_l; \alpha_l, \beta_l, n_l - 1) \tag{7}$$

Here $a_l \in \{0, 1, \ldots, n_l - 1\}$ and α_l, β_l are the parameters of the beta binomial along the $(l)^{th}$ dimension.

We see from equation (7) that the beta binomials are axis aligned. As the beta binomial along each dimension is independent of the others, it might fail to capture any interesting covariance between the dimensions. While this may be seen as a shortcoming, for the task of density estimation this problem becomes insignificant for large enough N_P.

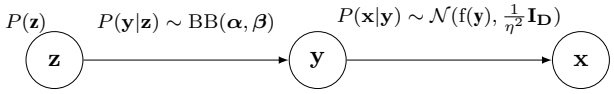

Fig. 1. Bayesian Network representation of LBTM

To summarize, LBTM assumes the following generative process for the data.

1) Choose $\mathbf{z} \sim \omega$, where ω is a multinomial on the $N_P - 1$ simplex.
2) Choose $\mathbf{y} \sim BB(\boldsymbol{\alpha}, \boldsymbol{\beta})$. $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{R}^{N_P \times L}$ such that $\boldsymbol{\alpha}_{ij}$ and $\boldsymbol{\beta}_{ij}$ are the parameters of the $(j)^{th}$ dimensional beta binomial in the $(i)^{th}$ mixture.
3) Choose $\mathbf{x} \sim \mathcal{N}(\mathbf{W}\phi(\mathbf{y}), \frac{1}{\eta^2}\mathbf{I_D})$.

The model is illustrated in Figure 1. Note that $\mathbf{x}$ is independent of $\mathbf{z}$ when $\mathbf{y}$ is known. We shall see later that this assumption of conditional independence reduces the complexity of training.

B. Training and the EM Algorithm

LBTM is trained using EM, [15]. The log likelihood function to be maximized may be written as

$$l(\theta) = \sum_{i=1}^{m} \log \sum_{\mathbf{y},\mathbf{z}} p(\mathbf{x}^{(i)}, \mathbf{y}, \mathbf{z}; \theta) \tag{8}$$

In equation (8), θ refers to the tuple $(\mathbf{W}, \eta, \boldsymbol{\alpha}, \boldsymbol{\beta}, \omega)$. By denoting the responsibilities by $Q_i(\mathbf{y}, \mathbf{z}) = p(\mathbf{y}, \mathbf{z}|\mathbf{x}^{(i)})$, the lower bound $l_b(\theta)$ on the log likelihood is given by,

$$l_b(\theta) = \sum_{i=1}^{m} \sum_{\mathbf{y},\mathbf{z}} Q_i(\mathbf{y}, \mathbf{z}) \log \left(\frac{p(\mathbf{x}^{(i)}|\mathbf{y}, \mathbf{z}) p(\mathbf{y}|\mathbf{z}) p(\mathbf{z})}{Q_i(\mathbf{y}, \mathbf{z})} \right) \tag{9}$$

A naive implementation of EM based on equation (9) will require $O(mN_L N_P)$ time and space complexity to compute the responsibilities as $Q_i(\mathbf{y}, \mathbf{z})$ needs to be computed for each $\mathbf{x}, \mathbf{y}$ and $\mathbf{z}$. Fortunately, the conditional independence assumption gives rise to simpler computation. To derive this implementation we expand equation (9)

$$\begin{aligned} l_b(\theta) = \sum_{i=1}^{m} \sum_{\mathbf{y},\mathbf{z}} Q_i(\mathbf{y}, \mathbf{z}) \Big\{ & \log p(\mathbf{x}^{(i)}|\mathbf{y}, \mathbf{z}) + \log p(\mathbf{y}|\mathbf{z}) \\ & + \log p(\mathbf{z}) - \log Q_i(\mathbf{y}, \mathbf{z}) \Big\} \end{aligned} \tag{10}$$

and note the following.

- $p(\mathbf{x}^{(i)}|\mathbf{y}, \mathbf{z}) = p(\mathbf{x}^{(i)}|\mathbf{y})$ as $(\mathbf{x} \perp \mathbf{z}|\mathbf{y})$.
- $\sum_{\mathbf{z}} Q_i(\mathbf{y}, \mathbf{z}) = \sum_{\mathbf{z}} p(\mathbf{y}, \mathbf{z}|\mathbf{x}^{(i)}) = p(\mathbf{y}|\mathbf{x}^{(i)})$.
  Let us denote $p(\mathbf{y}^{(j)}|\mathbf{x}^{(i)})$ by $Q_i^x(\mathbf{y}^{(j)})$.
- $\sum_i Q_i(\mathbf{y}, \mathbf{z}) = \sum_i p(\mathbf{y}, \mathbf{z}|\mathbf{x}^{(i)}) = \sum_i \frac{p(\mathbf{x}^{(i)}|\mathbf{y}) p(\mathbf{y}|\mathbf{z}) p(\mathbf{z})}{p(\mathbf{x}^{(i)})} = \frac{p(\mathbf{y}|\mathbf{z}) p(\mathbf{z})}{p(\mathbf{y})} \sum_i \frac{p(\mathbf{x}^{(i)}|\mathbf{y}) p(\mathbf{y})}{p(\mathbf{x}^{(i)})} = p(\mathbf{z}|\mathbf{y}) w(\mathbf{y})$ where $w(\mathbf{y}) = \sum_i \frac{p(\mathbf{x}^{(i)}|\mathbf{y}) p(\mathbf{y})}{p(\mathbf{x}^{(i)})} = \sum_i p(\mathbf{y}|\mathbf{x}^{(i)}) = \sum_i Q_i^x(\mathbf{y})$.
  Let us denote $p(\mathbf{z}^{(k)}|\mathbf{y}^{(j)}) w(\mathbf{y}^{(j)})$ by $Q_j^y(\mathbf{z}^{(k)})$.

The above results propose a scheme where $Q_i(\mathbf{y}, \mathbf{z})$ need not be explicitly computed - thus reducing the complexity. Incorporating the above, we may rewrite $l_b(\theta)$ as follows.

$$
\begin{aligned}
l_b(\theta) &= s_1(\mathbf{W}, \eta) + s_2(\boldsymbol{\alpha}, \boldsymbol{\beta}) + s_3(\omega) + s_4. \quad \text{where} \\
s_1(\mathbf{W}, \eta) &= \sum_{i=1}^{m} \sum_{j=1}^{N_L} Q_i^x(\mathbf{y}^{(j)}) \log p(\mathbf{x}^{(i)} | \mathbf{y}^{(j)}; \mathbf{W}, \eta) \\
s_2(\boldsymbol{\alpha}, \boldsymbol{\beta}) &= \sum_{j=1}^{N_L} \sum_{k=1}^{N_P} Q_j^y(\mathbf{z}^{(k)}) \log p(\mathbf{y}^{(j)} | \mathbf{z}(k); \boldsymbol{\alpha}, \boldsymbol{\beta}) \\
s_3(\omega) &= \sum_{j=1}^{N_L} \sum_{k=1}^{N_P} Q_j^y(\mathbf{z}^{(k)}) \log p(\mathbf{z}^{(k)}; \omega) \\
s_4 &= -\sum_{i=1}^{m} \sum_{j=1}^{N_L} \sum_{k=1}^{N_P} Q_i(\mathbf{y}^{(j)}, \mathbf{z}^{(k)}) \log Q_i(\mathbf{y}^{(j)}, \mathbf{z}^{(k)})
\end{aligned}
\tag{11}
$$

An EM procedure based on equation (11) is more efficient due to two reasons. First, the complexity of computing the responsibilities Q_i^x and Q_j^y has been reduced to $O(mN_L + N_L N_P)$. Second, as the parameters $\mathbf{W}, \eta, \boldsymbol{\alpha}, \boldsymbol{\beta}, \omega$ have been decoupled in the expressions for s_1, s_2 and s_3 we may maximize $l_b(\theta)$ by individually maximizing each component.

In maximizing $l_b(\theta)$ first note that s_4 is just a constant and does not affect the optimum point. The expression in equation (4) for the lower bound of the log likelihood in the GTM can be reduced to s_1 in equation (11). Hence the updates for the parameters $\mathbf{W}$ and η obtained by maximizing s_1 is the same as those given by equation (5).

Optimizing s_3 is also easily done. The optimization objective for the multinomial ω is, $\omega = \arg\max_\omega \sum_j \sum_k Q_j^y(\mathbf{z}^{(k)}) \log \omega_k$ subject to the constraint $\sum_k \omega_k = 1$. It is easily derived by constructing the Lagrangian and then applying certain results arrived at before that the update for ω is

$$
\omega_k := \frac{1}{m} \sum_{j=1}^{N_L} Q_j^y(\mathbf{z}^{(k)})
$$

This leaves us with s_2. By observing equations (6), (7) and (11) we construct the following expression for s_2.

$$
\begin{aligned}
s_2(\boldsymbol{\alpha}, \boldsymbol{\beta}) &= \sum_{j,k} Q_j^y(\mathbf{z}^{(k)}) \log \left\{ \prod_{l=1}^{N_L} p_{bb}(\mathbf{y}^{(j)}; \boldsymbol{\alpha}_{kl}, \boldsymbol{\beta}_{kl}, n_l - 1) \right\} \\
&= \sum_{j=1}^{N_L} \sum_{k=1}^{N_P} Q_j^y(\mathbf{z}^{(k)}) \sum_{l=1}^{L} \log \left\{ \frac{(n_l - 1)!}{a_l^{(j)}! b_l^{(j)}!} \right. \\
&\qquad \left. \frac{\Gamma(\boldsymbol{\alpha}_{kl} + \boldsymbol{\beta}_{kl})}{\Gamma(\boldsymbol{\alpha}_{kl})\, \Gamma(\boldsymbol{\beta}_{kl})} \frac{\Gamma(\boldsymbol{\alpha}_{kl} + a_l^{(j)})\, \Gamma(\boldsymbol{\beta}_{kl} + b_l^{(j)})}{\Gamma(\boldsymbol{\alpha}_{kl} + \boldsymbol{\beta}_{kl} + n_l - 1)} \right\}
\end{aligned}
\tag{12}
$$

Here each latent node $\mathbf{y}^{(j)}$ is represented by the coordinates $(a_1^{(j)}, a_2^{(j)}, \ldots, a_L^{(j)})$ on the grid. For notational convenience we have taken $b_l^{(j)} = n_l - 1 - a_l^{(j)}$.

In order to optimize the above expression for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ we use Newton's method. The approach is similar to the procedure outlined in [16] for estimating a Dirichlet distribution. Inverting the Hessian is generally a cubic operation. However we saw before that in each mixture the beta binomials along each dimension were independent. As the optimization of $\boldsymbol{\alpha}_{kl}$ is coupled only to $\boldsymbol{\beta}_{kl}$ and vice versa Newton's method would converge in linear time.

By partially differentiating equation (12) we arrive at the first and second derivatives of s_2 with respect to $\boldsymbol{\alpha}_{hd}$.

$$
\begin{aligned}
\frac{\partial s_2}{\partial \boldsymbol{\alpha}_{hd}} &= \sum_{j=1}^{N_L} Q_j^y(\mathbf{z}^{(k)}) \Big\{ \Psi(\boldsymbol{\alpha}_{hd} + \boldsymbol{\beta}_{hd}) + \Psi(\boldsymbol{\alpha}_{hd} + a_d) \\
&\qquad - \Psi(\boldsymbol{\alpha}_{hd} + \boldsymbol{\beta}_{hd} + n_d - 1) - \Psi(\boldsymbol{\alpha}_{hd}) \Big\} \\
\frac{\partial^2 s_2}{\partial \boldsymbol{\alpha}_{hd}^2} &= \sum_{j=1}^{N_L} Q_j^y(\mathbf{z}^{(k)}) \Big\{ \Psi'(\boldsymbol{\alpha}_{hd} + \boldsymbol{\beta}_{hd}) + \Psi'(\boldsymbol{\alpha}_{hd} + a_d) \\
&\qquad - \Psi'(\boldsymbol{\alpha}_{hd} + \boldsymbol{\beta}_{hd} + n_d - 1) - \Psi'(\boldsymbol{\alpha}_{hd}) \Big\}
\end{aligned}
\tag{13}
$$

In equation (13) Ψ and Ψ' refer to the digamma and trigamma functions respectively[1]. We may similarly construct expressions for $\frac{\partial s_2}{\partial \boldsymbol{\beta}_{hd}}$, $\frac{\partial^2 s_2}{\partial \boldsymbol{\beta}_{hd}^2}$ and $\frac{\partial^2 s_2}{\partial \boldsymbol{\alpha}_{hd} \partial \boldsymbol{\beta}_{hd}}$.

Now that we have computed the first and second derivatives we may proceed to apply Newton's method. As we noted earlier, we may apply the updates individually to each pair $(\boldsymbol{\alpha}_{hd}, \boldsymbol{\beta}_{hd})$. The Newton's method update is given by,

$$
\begin{aligned}
\begin{pmatrix} \boldsymbol{\alpha}_{hd} \\ \boldsymbol{\beta}_{hd} \end{pmatrix} &:= \begin{pmatrix} \boldsymbol{\alpha}_{hd} \\ \boldsymbol{\beta}_{hd} \end{pmatrix} - \mathbf{H}_{\mathbf{hd}}^{-1}\, \mathbf{g}_{\mathbf{hd}}, \quad \text{where} \\
\mathbf{H}_{\mathbf{hd}} &= \begin{pmatrix} \frac{\partial^2 s_2}{\partial \boldsymbol{\alpha}_{hd}^2} & \frac{\partial^2 s_2}{\partial \boldsymbol{\alpha}_{hd} \boldsymbol{\beta}_{hd}} \\ \frac{\partial^2 s_2}{\partial \boldsymbol{\alpha}_{hd} \boldsymbol{\beta}_{hd}} & \frac{\partial^2 s_2}{\partial \boldsymbol{\beta}_{hd}^2} \end{pmatrix}, \quad \mathbf{g}_{\mathbf{hd}} = \begin{pmatrix} \frac{\partial s_2}{\partial \boldsymbol{\alpha}_{hd}} \\ \frac{\partial s_2}{\partial \boldsymbol{\beta}_{hd}} \end{pmatrix}
\end{aligned}
\tag{14}
$$

We may construct the Hessian $(\mathbf{H}_{\mathbf{hd}})$ and the gradient $(\mathbf{g}_{\mathbf{hd}})$ using equation (13) and then use (14) to iteratively update $\boldsymbol{\alpha}_{hd}$ and $\boldsymbol{\beta}_{hd}$. s_2 in equation (11) is thus optimized.

The EM algorithm for LBTM is summarized in Table II. The formulae describe an implementation that can be efficiently vectorized. L_2-regularization has been incorporated to the updates derived thus far. λ and μ are the regularization coefficients for the updates for $\mathbf{W}$ [13] and $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ respectively. The complexity of each iteration as described in the table is $O(mN_L + N_L N_P + N_P L)$.

C. Initialization and Model Hyper-parameters

Like most EM formulations, the performance of LBTM depends critically on initialization. Randomly initializing $\mathbf{W}$ will almost always converge to a very poor local optimum. To alleviate this problem the GTM is initialized using a PCA based procedure in [13]. But a linear approximation may not always be the best initialization - especially for datasets with non-trivial manifolds. Non parametric models for NLDR such as LLE, Laplacian Eigenmaps and Isomap can produce good lower dimensional embeddings. Initializing our manifold using an embedding thus produced might improve our chances of obtaining a near-global optimum. Here we outline a simple method to achieve such an initialization.

[1] $\Psi(x) = \frac{d}{dx} \log(\Gamma(x))$ is the digamma function. The trigamma function $\Psi'(x)$ is the derivative of the digamma function

Given an L-dimensional representation returned by a suitable nonparametric algorithm $Y_e \in \mathbb{R}^{m\times L}$, $\{y_e^{(1)}, \ldots y_e^{(m)}\}$ we initialize $\mathbf{W}$ to minimize the least square error of transforming Y_e to X (see equation 3). i.e. set

$$\mathbf{W} = X^T P (P^T P)^{-1}$$

Where $P \in \mathbb{R}^{m\times M}$ such that $P_{ij} = \phi_j(y_e^{(i)})$. η is set to the inverse of the average reconstruction error when using $\mathbf{W}$ as initialized above. In the experiments reported the initialization which gave the best results was chosen.

$\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and ω are initialized using simple heuristics.

- Uniformly initialize the latent space distribution.
- Compute the inverse mapping ([13]) of all points in the training set to obtain $\hat{Y} \in \mathbb{R}^{m\times L}$.
- Initialize N_P seeds $\{c^{(k)} | k = 1, 2, \ldots, N_P\}$ on the latent space using $\hat{Y}$ via seeding [17], k-means clustering or any appropriate procedure. Initialize N_P unity precision multidimensional beta binomials whose means are given by the N_P seeds. I.e. set $\boldsymbol{\alpha}_{kl} = c_l^{(k)}/d_l$, $\boldsymbol{\beta}_{kl} = (d_l - c_l^{(k)})/d_l$, where d_l is the length of the $(l)^{th}$ dimension in the latent space.
- We now have N_P mixtures. Initialize ω by setting $\omega_k = \frac{1}{m}\sum_i \frac{p(\mathbf{x}^{(i)}|\mathbf{z}^{(k)})}{\sum_z p(\mathbf{x}^{(i)}|\mathbf{z})}$.

For the transformation via the GTM we need to choose N_L, M and σ. A discussion on the choice of these parameters is available in [13], [14]. For the LBTM we should choose N_P. If there is prior knowledge on the nature of clustering of the data then we may set N_P to the number of expected clusters. Otherwise we may choose N_P using heuristics such as a validation set likelihood.

D. Transformations, Inversions and Data Generation

Ideally, we would prefer an NLDR model to have a bijective mapping from points in the data space to the latent space. While most linear dimensionality reduction methods have this property nonlinear methods do not [12]. For the GTM/ LBTM we have a transformation from the latent space to the data space (equation 3). Though there is no direct inverse function, the Inverse Mean Mapping (IMM) [13] has a valid interpretation as an inverse transformation for the GTM. For the LBTM, the slightly modified definition of the IMM $\hat{y}_{mean}$ of a point x in the data space is

$$\begin{aligned} \hat{y}_{mean}(x) &= \sum_j p(\mathbf{y}^{(j)}|x)\cdot \mathbf{y}^{(j)} \quad \text{where} \\ p(\mathbf{y}^{(j)}|x) &\propto p(x|\mathbf{y}^{(j)}) \sum_k p(\mathbf{y}^{(j)}|\mathbf{z}^{(k)}) p(\mathbf{z}^{(k)}) \end{aligned} \tag{15}$$

As a parametric density model the LBTM can also be used to generate data (Figure 1). After learning, the LBTM samples data that faithfully replicates the original distribution of the training set. As a data generator the LBTM outperforms the GTM and other latent variable models.

A multinomial distribution for the latent space: We conclude this section with a slight digression. Instead of learning a mixture of beta binomials we could have just learned a multinomial on the latent nodes - an approach which we shall refer to as the M-GTM. It is easily shown that the E-Step updates to compute the responsibilities $Q_i^x(\mathbf{y}^{(j)})$ and the M-step updates for $\mathbf{W}$ and η are as described in section II. The M-step update for the multinomial on the latent nodes is given by $\pi_j = \frac{1}{m}\sum_i Q_i^x(\mathbf{y}^{(j)})$ where π_j refers to the probability of the $(j)^{th}$ node. While this method may be computationally cheaper, it has certain inherent disadvantages. First, the model will have too many degrees of freedom and will be susceptible to overfitting. The resulting probability distribution on the latent space will be very unsmooth (see Figures 2, 3d). Though this can be smoothed (Laplacian smoothing, convolution) there are no principled methods to choose the smoothing parameters (k for Laplacian smoothing, kernel for convolution). Second, storing the model would require memory linear in N_L (exponential in L). Reducing N_L to avoid these problems is not an option as that would be compromising on the quality of the mapping.

TABLE II
EM ALGORITHM FOR LATENT BETA TOPOGRAPHIC MAPPING

E-step

$$Q_i^x(\mathbf{y}^{(j)}) = \frac{p(\mathbf{x}^{(i)}|\mathbf{y}^{(j)})p(\mathbf{y}^{(j)})}{p(\mathbf{x}^{(i)})} \quad \forall i = 1,\ldots,m\ j = 1,\ldots,N_L$$

$$w(\mathbf{y}^{(j)}) = \sum_{i=1}^{m} Q_i^x(\mathbf{y}^{(j)}) \quad \forall j = 1,\ldots,N_L$$

$$Q_j^y(\mathbf{z}^{(k)}) = w(\mathbf{y}^{(j)})\frac{p(\mathbf{y}^{(j)}|\mathbf{z}^{(k)})p(\mathbf{z}^{(k)})}{p(\mathbf{y}^{(j)})} \quad \forall j = 1,\ldots,N_L\ k = 1,\ldots,N_P$$

M-step

$$\mathbf{W} := \{(\Phi^T G \Phi + \tfrac{\lambda}{\eta} I)^{-1}\Phi^T R X\}^T$$

$$\eta := \Big\{\frac{1}{mD}\sum_{i=1}^{m}\sum_{j=1}^{N_L} Q_i^x(\mathbf{y}^{(j)})||\mathbf{W}\phi(\mathbf{y}^{(j)}) - \mathbf{x}^{(i)}||^2\Big\}^{-1}$$

Update until convergence $\quad \forall h \in \{1,\ldots,N_P\}, \forall d \in \{1,\ldots,L\}$:

$$\xi_{hd} := \sum_{j=1}^{n_d} Q_j^y(\mathbf{z}^{(h)})\Big\{\Psi(\boldsymbol{\alpha}_{hd}+\boldsymbol{\beta}_{hd}) - \Psi(\boldsymbol{\alpha}_{hd}+\boldsymbol{\beta}_{hd}+n_d-1)\Big\}$$

$$\xi'_{hd} := \sum_{j=1}^{n_d} Q_j^y(\mathbf{z}^{(h)})\Big\{\Psi'(\boldsymbol{\alpha}_{hd}+\boldsymbol{\beta}_{hd}) - \Psi'(\boldsymbol{\alpha}_{hd}+\boldsymbol{\beta}_{hd}+n_d-1)\Big\}$$

$$\gamma_{hd} := \sum_{j=1}^{n_d} Q_j^y(\mathbf{z}^{(h)})\Big\{\Psi(\boldsymbol{\alpha}_{hd}+a_d) - \Psi(\boldsymbol{\alpha}_{hd})\Big\} - 2\mu\boldsymbol{\alpha}_{hd}$$

$$\delta_{hd} := \sum_{j=1}^{n_d} Q_j^y(\mathbf{z}^{(h)})\Big\{\Psi(\boldsymbol{\beta}_{hd}+b_d) - \Psi(\boldsymbol{\beta}_{hd})\Big\} - 2\mu\boldsymbol{\beta}_{hd}$$

$$\gamma'_{hd} := \tfrac{\partial\gamma_{hd}}{\partial\boldsymbol{\alpha}_{hd}},\ \delta'_{hd} := \tfrac{\partial\delta_{hd}}{\partial\boldsymbol{\beta}_{hd}},\ \Delta_{hd} := \xi'_{hd}(\gamma'_{hd}+\delta'_{hd}) + \gamma'_{hd}\delta'_{hd}$$

$$\boldsymbol{\alpha}_{hd} := \boldsymbol{\alpha}_{hd} - \tfrac{1}{\Delta_{hd}}\left(\xi'_{hd}(\gamma_{hd}-\delta_{hd}) + \delta'_{hd}(\xi_{hd}+\gamma_{hd})\right)$$

$$\boldsymbol{\beta}_{hd} := \boldsymbol{\beta}_{hd} - \tfrac{1}{\Delta_{hd}}\left(\xi'_{hd}(\delta_{hd}-\gamma_{hd}) + \gamma'_{hd}(\xi_{hd}+\delta_{hd})\right)$$

$$\omega_k := \frac{1}{m}\sum_{j=1}^{N_L} Q_j^y(\mathbf{z}^{(k)}) \quad \forall k \in \{1,\ldots,N_P\}$$

IV. EXPERIMENTAL RESULTS

A. Toy Datasets

We test the GTM, M-GTM and LBTM on the datasets shown in Figures 2(left) and 3a. The former is a sine curve with a non-uniform distribution along the curve. The latter is the standard S-curve which has an intrinsic 2D manifold but

TABLE III

RESULTS ON THE SINE AND S-CURVE DATASETS

Sine data	GTM	M-GTM	LBTM	MPPCA	GPLVM*
Log L -Train	-1.0818	-0.6894	-0.7656	-0.2276	2.3082
Log L -Valid	-1.2116	-0.9112	-0.7847	-2.8535	-
Training Time	1.76s	1.81s	3.31s	0.45s	169.2s
S-Curve					
Log L -Train	-3.2538	-2.6335	-2.8185	-3.2093	-2.2914
Log L -Valid	-3.3669	-3.0827	-2.8639	-3.2156	-
Training Time	46.1 s	50.5 s	96.1 s	1.6 s	736.9 s

The average training and validation likelihoods on the Sine data and S-curve. Though expensive, LBTM achieves a higher likelihood on the validation set when compared to GTM and M-GTM.

*As the GPLVM cannot handle out of sample data it is not possible to compute a Validation set log likelihood. Also strictly, the training log likelihood of GPLVM has a slightly different interpretation as it is marginalized over the parameters and not the latent variables.

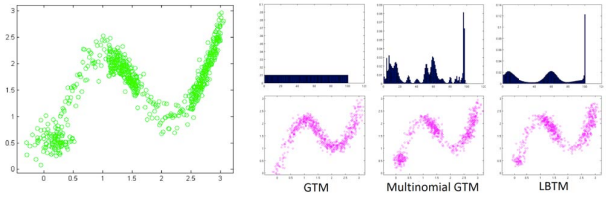

Fig. 2. Left: Sine data in 2 dimensions. Right: The latent space distribution (row 1) and data sampled (row 2) using the GTM, M-GTM and LBTM

is slightly modified to have a non-uniform distribution and contain some spherical Gaussian noise. All 3 models were able to correctly learn the manifold when initialized using the embedding produced by Laplacian Eigenmaps. Figure 3b shows the manifold learned by LBTM for the S-curve projected onto the data space.

Observe the data sampled using the models after learning. In both cases we see that while the GTM has captured the structure of the manifold it has failed to estimate the density properly. The data sampled by the M-GTM and the LBTM both resemble the original dataset. However, the latent space distribution of the M-GTM is very unsmooth (Figures 2(right), 3d). On observing the training and cross validation likelihoods for the M-GTM and the LBTM in Table III it is seen that the former has overfit the data. This highlights the importance of controlling the degrees of freedom when learning. The LBTM, though computationally expensive, outperforms the other 2 versions.

Note the following about the S-Curve test. One of the reasons the manifold resembled a "perfect" S in the data space was that Laplacian Eigenmaps returned a near rectangular embedding whose sides were aligned along the axes. When initialized using Isomap/ LLE where the embedding was not rectangular or not aligned along the axes the manifold did not resemble an S. When initialized using the PCA based procedure [13] performance was very poor. Even though LBTM was able to learn a distribution reasonably well after a "bad" initialization, the log likelihood was less than when initialized using Laplacian Eigenmaps.

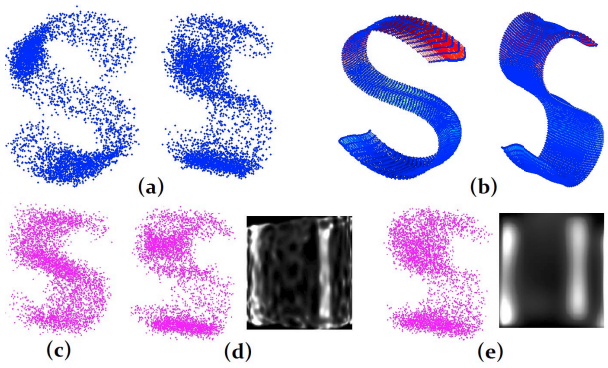

Fig. 3. (a) The S-curve dataset. (b) The manifold learned by LBTM projected to the data space. (c) Sampled data using the GTM. The GTM has a uniform latent space distribution. (d)&(e) Sampled data and the latent space distribution when using the M-GTM and LBTM respectively.

Fig. 4. The first row shows 7 original images from the Isomap face dataset. The images were represented in 3D and then reconstructed using PCA (row 2), GTM (row 3), and LBTM (row 4).

B. Isomap Face Data

The Isomap face dataset consists of 698 grayscale images of size $64 \times 64 (D = 560)$. The images are synthetic faces with different poses and lighting. There are only 3 variants across the dataset (2 for the pose and 1 for the lighting). The task at hand is to recover the 3 intrinsic dimensions and learn a probability model for the data. We compare the performance of LBTM against GTM, MPPCA and MFA.

The dataset was first reduced to its first 240 principal components. A GTM and an LBTM were trained using 80% of the data with 3 latent dimensions, 125 basis functions, 13824 latent points and 400 mixtures. For this dataset, initialization based on PCA gave the best results. Accordingly each image was represented using 3 coordinates. Figure 4 shows images reconstructed from the 3 coordinate representation using PCA, GTM and LBTM. From the reconstructions it is evident that the GTM and LBTM were able to capture nonlinear structure in the data.

Visually, the reconstructions using the LBTM and GTM are not very distinguishable. However, what differentiates the LBTM from the GTM is its ability to faithfully estimate the density of the distribution. The results of density estimation are shown in Table IV. From the Validation set

TABLE IV

RESULTS ON THE ISOMAP FACE DATA

	GTM	M-GTM	LBTM	MPPCA	MFA
Log L	-132.75	-114.34	-105.04	-109.24	-113.25
Percentage	46%	89%	84%	67%	62%

Row 1: Average Log Likelihood on a Validation set consisting of 140 images. Row 2: Percentage of sampled images that resembled a face *(according to the subjective judgment of the author)*. All models were trained using 3 dimensional latent spaces. The best results for MPPCA and MFA were obtained when using 22 and 55 mixtures respectively.

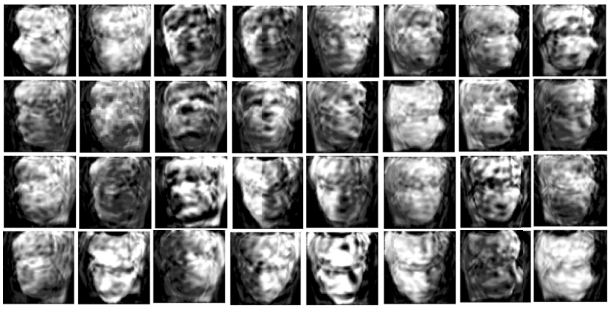

Fig. 5. Faces sampled by LBTM after learning.

log likelihoods we see that the LBTM performs better than GTM, M-GTM, MPPCA and MFA. A better estimation of density meant that data sampled by the LBTM was able to better replicate the original dataset than the GTM. Figure 5 shows some of the images sampled using the LBTM. Though the images are not as neat as the images in Figure 4, the orientation and lighting are very well represented in all images. The structure of the face (including the eyes and nose) is also intact. Faces with orientation and lighting that did not exactly match any of the faces in the dataset were also generated. This indicates that the model was able to adequately generalize the distribution of the data. Note here that the LBTM has estimated the density very well. If not, had we chosen a random point on the 3D latent space that did not belong to this distribution, the reconstruction may not have resembled a face. 84% of the images sampled using the LBTM resembled a face. The corresponding figure for the GTM was 46%.

To see why, observe the point clouds depicted in Figure 7. As expected we see that in the GTM, the sampled data

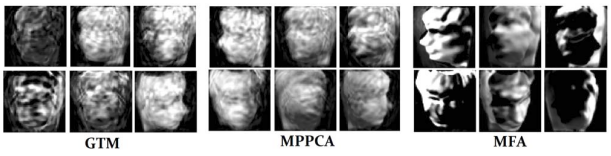

Fig. 6. Faces sampled using the model learned via GTM, MPPCA and MFA. The samples produced by the GTM are of poor visual quality. In the samples produced by the MPPCA even though the orientation and lighting are well represented, the structure of the faces seem to be slightly less pronounced. In most samples produced my MFA the images suffer from having one or more ghost faces in the background.

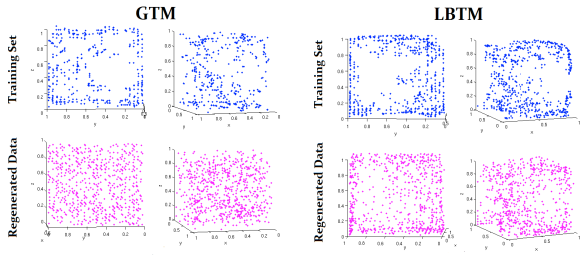

Fig. 7. IMMs of data using the GTM (left), LBTM (right). The first row shows the IMM of the original dataset while the second row shows the corresponding view of the mapping of some sampled data.

is spread uniformly across the latent space even though the original dataset only occupies a subset of the latent space. This confirms the GTM's inability to estimate density and explains why most of the sampled data do not resemble a face (Figure 6). On the other hand, the LBTM seems to have generalized the distribution quite well as the sampled points reasonably resemble the original distribution. Interestingly though, the M-GTM was able to produce a slightly higher fraction of "good" faces. It is important to note that when a model overfits the training set, data sampled using the model tends to exactly reproduce the training set (carefully inspect Figures 2 and 3). Such models have poor generalization performance. Even though the M-GTM produced more "good" faces, most of them closely resembled faces in the original training set as the model failed to interpolate and generalize.

Finally note that though such point clouds cannot be obtained for MPPCA and MFA it is still possible to sample data as they are both generative models. We have shown some of the images generated using MPPCA and MFA in Figure 6. By comparing the quality of the samples in Figure 5 and Figure 6 we see that LBTM has outperformed both MPPCA and MFA.

C. Frey Face Data

The Frey face dataset consists of 1965 grayscale images of size 20×28 ($D = 560$). The dataset was reduced to its first 50 principal components and an LBTM ($L = 2, N_L = 10,000, M = 25, N_P = 100$) was trained using 1572 of the images. For initialization we used the embedding produced by the Isomap algorithm. LBTM was able to successfully learn the manifold and the distribution of this dataset.

Figure 8 shows the IMMs of the original dataset and a sampled dataset. The figure also shows some of the images generated using the model after learning. The images sampled here are less noisy and visually much better than the images generated in the Isomap face experiment. This is not surprising because the Frey face dataset is a much simpler problem than the Isomap face dataset (more data and fewer latent space and data space dimensions).

Also note that the manifold achieves good topographic ordering. The faces shown are the samples produced along different traces in the manifold. When traversing along the curve we see that variations are gradual which indicates that

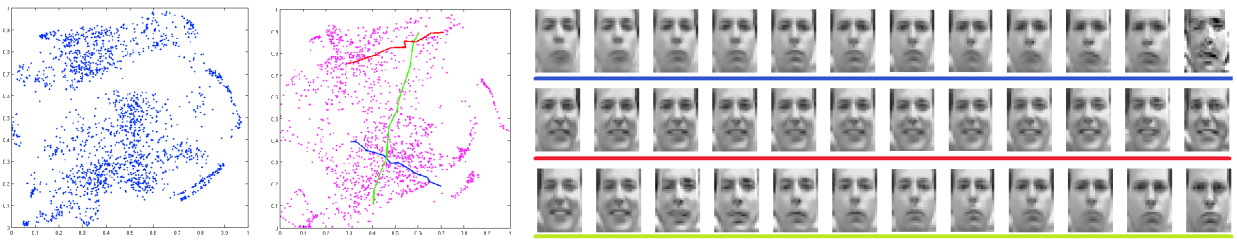

Fig. 8. The scatter plots are the IMMs of the original dataset (blue) and a sampled dataset using LBTM (magenta). The faces shown are images sampled using the model (*not* from the original dataset) that lie along the blue, red and green traces indicated in the sampled data scatter plot. The samples are legitimate faces representative of the original dataset and more importantly respect topographic order.

the LBTM hasn't produced any twisted maps.

V. Conclusion

A. Discussion

It is important to evaluate some of the tests demonstrated above for the LBTM against other NLDR formulations. To compare different density models we generally use validation set log likelihoods. LLE, Laplacian Eigenmaps and Isomap are not density models to begin with. Though GPLVM is a probabilistic model, it has no mechanism to handle out of sample data and hence a validation set log likelihood cannot be computed.

Using most other algorithms discussed above, data cannot be reduced to L-dimensions and then be reconstructed in the data space to assess how much information the lower dimensional representation has retained. Concretely, we cannot carry out the test demonstrated in Figure 4. MPPCA and MFA cannot be used to obtain a global lower dimensional representation in the first place while LLE, Isomap, Laplacian Eigenmaps and GPLVM do not have an inverse function from the data space to the latent space.

As an algorithm that can produce a global lower dimensional embedding of data, map points in the latent space to the data space and vice versa and learn a density model for the data which can be used for inference and data generation, the LBTM can be viewed as a more versatile model for NLDR than the other models discussed above.

B. Summary

This paper proposed a method to learn a latent space probability distribution for the GTM which allows density estimation without compromising on topographic ordering. In addition, we saw how good initialization of the GTM/ LBTM helps avoid local optima. Like popular nonparametric techniques, LBTM provides a global lower dimensional embedding of data. As a generative probability model it outperforms the GTM and powerful local dimensionality reduction techniques such as MPPCA and MFA.

Different variations to the LBTM can be obtained by imposing constraints on the probability distributions of the mixtures. For instance, we may fix ω to be a uniform multinomial. An LBTM learned may increase the probability of a neighborhood in the latent space by peaking the distributions of each beta binomial and/ or grouping more of them together. Another variation would be to fix the precisions for each beta binomial. We may choose the precisions depending on how locally peaked or flat we want our prior probabilities to be. We leave it to future work to build on such formulations.

VI. Acknowledgements

I wish to thank Ranga Rodrigo and the anonymous reviewers for the helpful feedback on my work and the paper.

References

[1] H. Bourlard and Y. Kamp, "Auto-association by multilayer perceptrons and singular value decomposition," *Biological Cybernetics*, vol. 59, no. 4, pp. 291–294, Sep. 1988.
[2] T. Kohonen, "Self Organizing Maps." Springer-Verlag Berlin, 1995.
[3] M. E. Tipping and C. M. Bishop, "Mixtures of Probabilistic Principal Component Analysers," *Neural Computation*, vol. 11, no. 2, pp. 443–482, 1999.
[4] Z. Ghahramani and G. E. Hinton, "The EM algorithm for mixtures of factor analyzers," Tech. Rep., 1996.
[5] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, pp. 2323–2326, 2000.
[6] M. Belkin and P. Niyogi, "Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering," in *Advances in Neural Information Processing Systems 14*. MIT Press, 2001, pp. 585–591.
[7] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction." *Science*, vol. 290, no. 5500, pp. 2319–23, 2000.
[8] M. Balasubramanian and E. L. Schwartz, "The Isomap algorithm and topological stability," *Science*, vol. 295, no. 5552, p. 7, Jan. 2002.
[9] O. Koyejo and J. Ghosh, "MiPPS; A Generative Model for Multi-Manifold Clustering," in *AAAI Fall Symposium on Manifold Learning and Its Applications*, vol. Technical Report FS-09-04. AAAI Press, 2009, pp. 18–25.
[10] L. K. Saul, S. T. Roweis, and Y. Singer, "Think Globally, Fit Locally: Unsupervised learning of low dimensional manifolds," *Journal of Machine Learning Research*, vol. 4, pp. 119–155, 2003.
[11] Y. Bengio, J.-F. Paiement, and P. Vincent, "Out-of-sample extensions for LLE, Isomap, MDS, Eigenmaps, and Spectral clustering," in *Advances in Neural Information Processing Systems*. MIT Press, 2003, pp. 177–184.
[12] N. D. Lawrence, "Probabilistic Non-linear Principal Component Analysis with Gaussian Process Latent Variable Models," *Journal of Machine Learning Research*, vol. 6, pp. 1783–1816, 2005.
[13] C. M. Bishop, M. Svensn, and C. K. I. Williams, "GTM: The Generative Topographic Mapping," *Neural Computation*, vol. 10, pp. 215–34, 1998.
[14] C. Bishop and C. K. I. Williams, "Developments of the Generative Topographic Mapping," *Neurocomputing*, vol. 21, pp. 203–224, 1998.
[15] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Royal Statist. Soc., B*, vol. 39, no. 1, pp. 1–38, 1977.
[16] T. Minka, "Estimating a Dirichlet distribution," 2000.
[17] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Soda*, 2007, pp. 1027–35.