Hierarchical Bayesian Embeddings for Analysis and Synthesis of High-Dimensional Dynamic Data

Haojun Chen, Emily Fox, Jorge Silva, David Dunson and Lawrence Carin

1Electrical and Computer Engineering Department
3Statistics Department
Duke University, Durham, NC, USA

2Department of Statistics, University of Washington
Seattle, WA, USA

Abstract

High-dimensional, time-dependent data are analyzed by developing a dynamic model in an associated low-dimensional embedding space. The proposed approach employs hierarchical Bayesian methods to learn a reversible statistical embedding, allowing one to (i) estimate the latent-space dimension from a set of training data, (ii) discard the training data when embedding new data, and (iii) synthesize high-dimensional data from the embedding space. Properties (i)-(iii) are useful for the general analysis of high-dimensional data, even in the absence of time dependence. For the case of dynamic data, hierarchical Bayesian methods are employed to learn a nonlinear dynamic model in the low-dimensional embedding space, allowing joint analysis of multiple types of dynamic data, sharing strength and inferring inter-relationships in dynamic data. Combined, the overall learned model enables the analysis and synthesis of high-dimensional dynamic data. Example results are presented for statistical embedding, latent-space dimensionality estimation, and analysis and synthesis of high-dimensional (dynamic) motion-capture data.

1. Introduction

Many time series datasets of interest are both high-dimensional and highly-structured. In such high-dimensional domains, classical multivariate time series models are unable to provide an adequate parsimonious representation and hence may have poor performance when data are limited; it is important to flexibly learn a low-dimensional latent structure to reduce dimensionality
while maintaining realism. For example, consider the analysis of human motion from a standard motion capture dataset, an application that has received significant recent attention (Taylor et al., 2007; Wang et al., 2008; Fox et al., 2010). Typical representations of motion-capture data yield time series with well over 50 dimensions, consisting of various joint angle and body position measurements. However, due to both constraints imposed by the geometry of the human body and by similarity of nearby or symmetrically placed sensors (i.e., left/right elbow), the dynamics can be described as evolving on a lower-dimensional manifold. Many high-dimensional time series yield such lower-dimensional representations. We seek a method for discovering the lower-dimensional embedding, and a corresponding latent-space dynamical model. In a similar spirit to (Fox et al., 2010, 2011), our methods also aim to infer relationships between different portions of the time series within a Bayesian nonparametric framework, and to harness this sharing of information.

Our proposed hierarchical Bayesian approach addresses three key challenges: learning the dimension of the latent embedding space, a nonlinear and reversible mapping from the latent to observed space, and a nonlinear latent-space dynamical model. We jointly address the problem of learning the probabilistic reversible mapping and the dimension of the embedding space. Our model employs mixture models to capture the nonlinearities, with one mixture model for the mapping between the latent embedded state \( x_t \) (Tenenbaum et al., 2000; Roweis and Saul, 2000; Coifman and Lafon, 2006) and the corresponding observation \( y_t \), and a separate mixture model for the dynamics mapping between \( x_t \) to \( x_{t+1} \). Specifically, for the embedding, we employ a Bayesian nonparametric variant of a mixture of factor analyzers (MFA) (Ghahramani and Beal, 2000) with a sparsity-inducing structure used to infer the dimensionality of the latent space. The Bayesian nonparametric aspect allows for uncertainty in the number of mixture components. Our dynamic model likewise employs a Bayesian nonparametric mixture model.

Because we learn a dynamic model after embedding the observations, it is crucial that the embedding procedure preserve a measure of proximity. Specifically, for any two observations \( y_t \) and \( y_{t'} \) with corresponding latent embeddings \( x_t \) and \( x_{t'} \), if \( \| y_t - y_{t'} \|_2 \) is small, then we desire that \( \| x_t - x_{t'} \|_2 \) is also small. That is, a differential change in the space of \( y_t \) must be consistent with such in the space of \( x_t \). This is not guaranteed in conventional designs of MFAs (Ghahramani and Beal, 2000). We address this problem by leveraging ideas from diffusion theory.
(Coifman and Lafon, 2006) and other embedding methods. Considering the reverse mapping, by preserving such a measure of proximity, we ensure that dynamics synthesized in latent space are projected to an observation sequence in the full space with an appropriate sense of (informally) smooth dynamics.

The features of the proposed model can be utilized in many ways. First, in contrast to most embedding methods, we can embed new data in the latent space without relying on re-accessing the training data. Second, our learned dynamic model is capable of synthesizing dynamics in the latent space. This represents a key feature of our model and is an implicit benefit of taking a generative approach. Finally, since the proposed mapping to the embedding space is reversible, we may project our synthesized latent dynamics to the full observation space. Using such a procedure, we demonstrate that our model is capable of accurate dynamic-motion synthesis, analogous to (Taylor et al., 2007; Wang et al., 2008; Lawrence and Moore, 2007; Grochow et al., 2004).

The remainder of the paper is organized as follows. In Section 2 we review relevant dynamic models and the MFA approach to latent-space embeddings, and summarize contributions of this paper. Section 3 outlines how we generalize the MFA to achieve useful embeddings and infer the dimension \( d \) of the latent space; Section 4 describes how we model dynamics in the latent space. In Section 5 we discuss how the proposed model is able to (i) synthesize dynamic data in the latent space, (ii) reversibly map from \( x \) back to \( y \) (projecting the synthesized dynamic data to the full observation space), and (iii) embed new data \( y \) into the latent state space \( x \) without having to retain the training data. Results from the analysis of several synthetic and real datasets are presented in Section 7. Conclusions are provided in Section 8.

2. Background

2.1 Switching Linear Dynamical Systems

Let \( y_t \in \mathbb{R}^D \) represent a \( D \)-dimensional observation at time \( t \) and \( x_t \in \mathbb{R}^d \) the latent state (dimension \( D \) is given; we wish to infer an appropriate embedding dimension \( d \)). A linear, time-invariant formulation assumes

\[
 y_t = C x_t + \epsilon_t \\
 x_t = A x_{t-1} + v_t,
\]

(1)
with \( v_t \sim \mathcal{N}(0, Q) \) independent of \( \epsilon_t \sim \mathcal{N}(0, \Sigma) \); \( \mathcal{N}(\mu, \Sigma) \) represents a multivariate Gaussian distribution with mean \( \mu \) and covariance \( \Sigma \). In (1) the latent dynamics evolve linearly according to a fixed dynamic matrix \( A \in \mathbb{R}^{d \times d} \), and these latent dynamics are linearly projected to the measurement space via \( C \in \mathbb{R}^{D \times d} \).

Many phenomena have dynamics that cannot be adequately described using the linear model of (1). For example, consider a person switching between a set of motions such as running, jumping, walking, and so on. In such cases, one can instead consider a switching linear dynamical system (SLDS) in which some underlying discrete-valued dynamic mode \( z_t \in \{1, \ldots, J\} \) determines the linear dynamical system at time \( t \). A common simplifying assumption is that \( z_t \) is a first-order Markov process with transition distributions \( \{\pi_j\} \):

\[
y_t = Cx_t + \epsilon_t
\]

\[
x_t = A_{z_t}x_{t-1} + v_t(z_t)
\]

\[
z_t \mid z_{t-1} \sim \pi_{z_{t-1}}.
\]

where \( \pi_{z_{t-1}} \) is a probability vector over the state \( z_t \), conditioned on \( z_{t-1} \).

Bayesian nonparametric versions of the SLDS of (2) were considered in (Fox et al., 2010, 2011). Specifically, uncertainty in the number of dynamical modes, \( J \), was allowed by employing hierarchical layerings of Dirichlet processes (Ferguson, 1973; Teh et al., 2006).

### 2.2 Mixture of Factor Analyzers

The model in (2) assumes a fixed linear relationship between any \( x_t \) and \( y_t \), via the matrix \( C \). This may be too restrictive for some problems. A natural generalization is

\[
y_t = C_{z_t}x_t + \epsilon_t(z_t)
\]

\[
x_t = A_{z_t}x_{t-1} + v_t(z_t)
\]

\[
z_t \mid z_{t-1} \sim \pi_{z_{t-1}}.
\]

where (3) is a mixture of factor analyzers (MFA) (Ghahramani and Beal, 2000). The MFA uses a mixture model to manifest a nonlinear mapping between \( x_t \) and \( y_t \).

While (3)-(5) constitutes a natural extension of the SLDS model, it has limitations. Specifically, it assumes that the mixture model associated with the dynamics, in (4), is of the same form as that associated with the MFA, (3). These are two distinct processes (one links the latent and
observed spaces, and the other models dynamics in the latent space), and therefore the form of these mixture models is likely to be different (e.g., they may not even have the same number of mixture components). Additionally, joint learning of (possibly distinct) mixture models for these two processes complicates inference significantly.

In this paper we decouple learning the MFA that links $x_t$ and $y_t$, from the separate learning of the dynamic model in the space of $x_t$. The MFA does not explicitly use the time index $t$, and therefore we drop the time index when discussing the MFA. Specifically, given data $\{y_i\}$, we seek to learn the model

$$ y_i = F_{z_i} x_i + \mu_{z_i} + \epsilon_i(z_i) \quad (6) $$

$$ z_i \sim \pi, \quad (7) $$

with $x_i$ characteristic of the latent feature space as before, but without the $t$ subscript; $\pi$ is a probability vector over the mixture components. An advantage of separately learning this MFA is that the mapping $y_i \leftrightarrow x_i$ may be learned for general data, even if there is no time dependence.

For applications for which the data are time evolving, once we have learned the model in (6), it may be employed to efficiently map any $y_t$ into an associated $x_t$. Hence, given time-dependent data $\{y_t\}$, may be mapped into dynamic data in the latent space, $\{x_t\}$. Given this embedded, time-dependent training data, we may then learn the dynamic model

$$ x_t = A_{\hat{z}_i} x_{t-1} + v_t(\hat{z}_t) \quad (8) $$

$$ \hat{z}_t \mid \hat{z}_{t-1} \sim \pi_{\hat{z}_{t-1}}, \quad (9) $$

where we employ $\hat{z}_t$ to distinguish from the $z_i$ used in (6). Since mixture models (6) and (8) are learned separately, we may infer the appropriate model complexity for each (number of mixture components and their composition).

The decoupling of the MFA embedding model and the associated dynamic model simplifies analysis, maintains generality in the embedding and dynamic models, and it yields highly effective results for analysis and synthesis of dynamic data. Further, the MFA component is applicable even in the absence of time-dependent data. This modeling flexibility comes with one important complication, the solution of which is an important contribution of this paper. When learning the MFA in (6), it is essential that we impose that if $y_i$ and $y_j$ are proximate in the original data space, then $x_i$ and $x_j$ are proximate in the embedded space. This is important for connecting
measures of temporal (differential) smoothness between the embedded space and the space of
the original high-dimensional data (differential changes in the space of $y$ need to correspond
to differential changes in the space of $x$). This relationship is not assured in traditional MFA
learning (Ghahramani and Beal, 2000), and in Section 3 we develop procedures to address this
challenge.

2.3 Contributions

This paper makes the following contributions:

- The MFA framework for constituting an embedding of high-dimensional data $\{y_i\}$ into a
  low-dimensional latent space $\{x_i\}$ is generalized such that inter-data relationships in the
  native data space are (approximately) preserved in the latent space. The learned MFA may
  be efficiently employed to embed a new data sample $y_i$ in the latent space $x_i$, without
  having to retain the training data after model learning. The proposed model also explicitly
  allows inference of the latent-space dimension $d$ (dimension of the vectors $\{x_i\}$).

- The above contribution is applicable to general high-dimensional data for which one is
  interested in an embedding (with or without time-dependence in the data). For the case of
  time-dependent data, we develop a new means of modeling the low-dimensional (embedded)
  dynamic data $\{x_t\}$, in which we infer and share dynamic relationships between different
  classes of time-evolving data.

- After learning the aforementioned dynamic model, it may be used to synthesize dynamic data
  in the embedding space. Further, based upon the MFA discussed in the first bullet above, we
  may efficiently map the synthesized dynamic data from the low-dimensional latent space to
  the high-dimensional space, where it may be observed in the native space characteristic of
  the original data. We demonstrate this on several human motion-capture examples.

3. Mixture of Factor Analyzers for Embeddings

3.1 Sparse Nonparametric Mixture of Factor Analyzers

Consider the following MFA

$$y_i = F_{z(i)}A_{z(i)}(x_i - \hat{\mu}_{z(i)}) + \mu_{z(i)} + \epsilon_i.$$  (10)
Note that we use the data index \( i \), rather than time index \( t \), as the approach developed here is applicable to general data, even without time dependence.

We take each factor loadings matrix \( F_j \in \mathbb{R}^{D \times K} \) with \( K \) larger than the anticipated latent dimension \( d \), but smaller than \( D \) (\( K \) is an upper bound on the dimension of \( x_i \), and we learn the subset of components \( d \) that are needed to represent the data). Diagonal matrix \( \Lambda_j \in \mathbb{R}^{K \times K} \) has a sparse diagonal, used to identify the latent dimension \( d \leq K \). Specifically, we take \( \Lambda_j = \text{diag}(\lambda_{1j}b_1, \ldots, \lambda_{Kj}b_K) \) with \( (b_1, \ldots, b_K) \) a sparse binary vector. The following priors are imposed for \( \Lambda_j \):

\[
\lambda_{jk} \sim \mathcal{N}(0, 1/\beta), \quad b_k \sim \text{Bernoulli}(\pi_k), \quad \pi_k \sim \text{Beta}(a/K, b(K-1)/K),
\]

(11)

with a gamma hyper-prior placed on \( \beta \). The elements \( (\lambda_{j1}, \ldots, \lambda_{jK}) \) play the role of singular values, although we do not explicitly impose that these are positive, with the sign absorbed into the factor loading. Note that for large \( K \), imposition of \( \pi_k \sim \text{Beta}(a/K, b(K-1)/K) \) encourages small \( \pi_k \) and hence a sparse diagonal for \( \Lambda_j \).

Additionally, we specify

\[
\hat{\mu}_j \sim \mathcal{N}(0, \xi^{-1}I_d), \quad x_i \sim \mathcal{N}(\hat{\mu}_{z(i)}, \beta_{z(i)}^{-1}I_d)
\]

(12)

with \( \epsilon_i \) drawn as in (13). Gamma hyperpriors are placed on \( \alpha, \xi \) and \( \{\beta_j\}_{j=1}^J \). Note that the introduction of \( \hat{\mu}_j \) is nonstandard in MFA modeling; these offsets in the latent space will be important below for preserving proximity in the data and latent spaces. Finally, the remaining MFA terms are straightforwardly given as

\[
f_{jk} \sim \mathcal{N}(0, 1/D I_D), \quad \mu_j \sim \mathcal{N}(0, \gamma^{-1}I_D), \quad \epsilon_i \sim \mathcal{N}(0, \alpha_0^{-1}I_D).
\]

(13)

Gamma hyper-priors are employed for \( \gamma \) and \( \alpha_0 \).

What remains is to place a prior on the indicator variables \( z(i) \), these yielding a partition of the data. We wish to infer the number of unique clusters (mixture components) needed to represent the data. We consider

\[
z(i) \sim \pi, \quad \pi = \sum_{j=1}^{\infty} w_j \delta_j.
\]

(14)

with \( w_j = V_j \prod_{i<j}(1-V_i) \), with \( V_i \sim \text{Beta}(1, \alpha) \). This corresponds to the “stick-breaking” representation (Sethuraman, 1994) of the DP; the DP “base” measure \( G_0 \) is the factored distribution above from which mixture parameters \( \{F_j, \Lambda_j, \hat{\mu}_j, \mu_j\} \) are drawn (collectively discussed above).
Henceforth we use the simplified notation $\pi \sim \text{Stick}(\alpha)$, and in practice we truncate the sum to $J$ terms (an upper bound on the number of mixture components). In the truncated stick-breaking construction used throughout $V_J = 1$, with $V_l \sim \text{Beta}(1, \alpha)$ for $1 \leq l \leq J - 1$. The accuracy of the truncated stick-breaking approximation, for large $J$, is discussed in (Ishwaran and James, 2001).

3.2 Limitations of Direct MFA Modeling

The above construction is a relatively standard implementation of an MFA (Ghahramani and Beal, 2000). To appreciate the limitations of directly using the model in (10) to perform an embedding (i.e., inferring the low-dimensional representation $x$), consider Figure 1. At top are the $\{y_i\}_{i=1,N}$, in Figure 1 represented in the form of two partially overlapping Gaussians; the colors indicate which of the two Gaussians the data are associated with. For this simple two-Gaussian example, considering (10), the Gaussians in the high-dimensional space $\mathbb{R}^D$ are centered at $\mu_1$ and $\mu_2$; in the latent space the mixtures for $\{x_i\}_{i=1,N}$ in $\mathbb{R}^K$ are centered correspondingly at $\hat{\mu}_1$ and $\hat{\mu}_2$. As constituted thus far, there are no constraints within the model on $\hat{\mu}_1$ and $\hat{\mu}_2$ that impose the alignment at the bottom-right in Figure 1. This implies that if $y_i$ and $y_j$ are from distinct Gaussians, but with small $\|y_i - y_j\|_2$, there are no assurances that $\|x_i - x_j\|_2$ will also be small; consequently, the $x_i$ do not serve as effective low-dimensional features of $y_i$ (differential changes in the space of $y$ do not translate in general to differential changes in the space of $x$).

3.3 Local one-step MFA alignment

Consider a general kernel $\mathcal{K}(y_i, y_j; \Theta)$, where $\Theta$ are kernel-dependent parameters, and $\mathcal{K}(y_i, y_i; \Theta) = 1$ with $\mathcal{K}(y_i, y_j; \Theta) = 0$ in the limit $\|y_i - y_j\|_2 \to \infty$ (and for all $i$ and $j$, $0 \leq \mathcal{K}(y_i, y_j; \Theta) \leq 1$). For example, in our analysis we employ a radial basis function (RBF)

$$\mathcal{K}(y_i, y_j; \sigma^2) = \exp\left[-\frac{1}{\sigma^2}\|y_i - y_j\|_2^2\right]$$

(15)

As in diffusion analysis (Coifman and Lafon, 2006), we define a random-walk matrix for the probability of “walking” from $y_i$ to $y_j$ in a single step,

$$W(i, j) = \mathcal{K}(y_i, y_j; \sigma_i^2) / \sum_{k=1}^{N} \mathcal{K}(y_i, y_k; \sigma_i^2)$$

(16)
We employ a distinct $\sigma_i$ for each $y_i$, as in (Zelnik-Manor and Perona, 2004), such that with high probability we may only walk to a prescribed number of nearest neighbors of $y_i$. The key to most embedding and many semi-supervised-learning models (Krishnapuram et al., 2004) is that $y_i \approx \sum_{l=1}^{N} W(i, l)y_l$, particularly if the RBF kernel parameter $\sigma_i$ is constructed such that $W(i, l)$ only has appreciable amplitude for $y_l$ within a close neighborhood of $y_i$. Hence, $y_i$ is approximately equal to a weighted average of $y_l$ within a nearby neighborhood (assuming sufficient quantity of data $\{y_l\}$ such that the data space in $\mathbb{R}^D$ is well sampled).

Let us define $x_i = \sum_{l=1}^{N} W(i, l)x_i^*$ for any vectors $x_i^* \in \mathbb{R}^K$, $l = 1, \ldots, N$, and $W$ defined as in (16). Based on this specification, if $y_i$ is proximate to $y_j$ in the observation space $\mathbb{R}^D$, $x_i$ will be proximate to $x_j$ in the latent space $\mathbb{R}^K$, which is justified as follows. Assume any isotropic kernel function $K(y_i, y_j; \Theta)$ in (16), such as the RBF function. By definition, the kernel function is continuous and bounded. Therefore, for any $y_i$ and $y_j$ such that $\|y_i - y_j\| < \epsilon$, we have that $K(y_i, y_j; \Theta) \rightarrow K(y_j, y_j; \Theta)$ as $\epsilon \rightarrow 0$. Since the normalization of the kernel is a continuous mapping, we likewise have that $W(i, l) \rightarrow W(j, l)$ as $\epsilon \rightarrow 0$. Thus, assuming $\|x_i^*\|$ is bounded for all $l$, then $\|x_i - x_j\| = \|\sum_{l=1}^{N} (W(i, l) - W(j, l))x_i^*\| \leq \sum_{l=1}^{N} |W(i, l) - W(j, l)|\|x_i^*\|$ tends to 0 as $\epsilon \rightarrow 0$. That is, $x_i \rightarrow x_j$ as $y_i \rightarrow y_j$.

So motivated, we modify the MFA model in the following simple manner. Assume that the mixture means in the latent space $\{\mu_j\}_{j=1,M}$ are drawn as in (12), and we employ a stick-breaking construction for probability vector $w$. We impose the following refined hierarchical construction for $x_i$ in (17):

$$x_i = \sum_{k=1}^{N} W(i, k)x_k^*, \quad x_k^* \sim \mathcal{N}(\hat{\mu}_z(k), \beta_{z(k)}^{-1}I_K), \quad z(k) \sim \sum_{j=1}^{\infty} w_j \delta_j \quad (17)$$

This model preserves the meaning of proximity in the spaces $\{y_i\}$ and $\{x_i\}$, for any $\{x_k^*\}$. The mapping from $x_k^*$ to $x_i$ in (17) and from $x_i$ to $y_i$ in (10) allow for learning $\{x_k^*\}$ that fit the data.

We only use the random-walk matrix $W$ to learn the aligned MFA based on training data $y_{1:1}$. Once the model is so learned, the mapping $y_{N+1} \rightarrow x_{N+1}$ is performed using the learned model (discussed in Section 5.3); $W$ is not updated and the training data $y_{1:1}$ are not needed. This is an important distinction with techniques like diffusion (Coifman and Lafon, 2006), which must augment the random-walk matrix using $y_{1:1}$ to embed a new $y_{N+1}$.

November 2, 2012  DRAFT
3.4 Global MFA alignment and nonlinear spectral regression

The above alignment procedure has been found effective in practice, in that it infers aligned embeddings in the space of \( \{x_i\} \). By this we mean that differential changes in the space of \( y \) translate to differential changes in \( x \). However, it has the limitation of only imposing the local contiguity of the embedding space \( x \) (locally around each \( x_i \)). Therefore, while locally smooth, the overall inferred space \( \{x_i\} \) may be overly complex (compared to traditional embedding procedures, we now leverage).

The alignment procedure in the previous subsection utilized the matrix \( W \), which corresponds to the random-walk matrix in diffusion analysis (Coifman and Lafon, 2006); similar matrices characteristic of the inter-data relationships in \( \{y_i\} \) are used in other embedding methods, such as local linear embedding (LLE) (Roweis and Saul, 2000) and Isomap (Tenenbaum et al., 2000). Any of these methods may be used to map \( \{y_i\} \) into an associated embedding space \( \{x_i\} \).

Rather than inferring \( \{x_i\} \) while developing the MFA, as in Section 3.3, assume one of the aforementioned embedding methods is employed to map \( \{y_i\} \) into \( \{x_i\} \). We now have the joint data \( \{y_i, x_i\} \). The modified MFA employs a Gaussian mixture model (GMM) to model the statistical distribution of \( \{x_i\} \), via (12). A GMM is simultaneously constituted for \( \{y_i\} \): each mixture component in the space of \( \{x_i\} \) is linked to an associated mixture component in the space of \( \{y_i\} \), with the linkage manifested via the associated factor loading \( F_{z(i)} \).

The \( F_{z(i)} \) serve to perform regression between the space of \( \{y_i\} \) and \( \{x_i\} \). We therefore refer to this as nonlinear spectral regression (NSR), with the nonlinearity manifested via the mixture of factor models (mixture of regressions). Importantly, since methods like diffusion analysis, LLE and Isomap define \( \{x_i\} \) in terms of the global manifold properties, such are accounted for within the proposed NSR procedure.

The joint mixture modeling between \( \{y_i\} \) and \( \{x_i\} \) implies that a Gaussian cloud of data in the space of \( x_i \) is mapped to an associated Gaussian cloud in the space of \( y_i \), with this mapping manifested via \( F_{z(i)} \) for mixture component \( z(i) \). This is intimately related to the work in (Singer and Coifman, 2008), in which a small point cloud in the embedding space is related to a corresponding point cloud in the native high-dimensional space. The DP-based mixture model constitutes a means of inferring these relationships between the spaces \( \{y_i\} \) and \( \{x_i\} \), which was not considered in (Singer and Coifman, 2008); in (Singer and Coifman, 2008) it was assumed.
that the clusters of point clouds – mixture components – were known (or could be experimentally constituted), which may be difficult in many applications.

One of the challenges of embedding methods concerns inference of the latent-space dimension \( d \). An advantage of the proposed method is that we have a means of inferring \( d \). When using one of the embedding methods to define \( \{x_i\} \), we may specify an upper bound \( K \) on dimension \( d \). Within \( \Lambda_{z(i)} \) in (10), the binary vector \( b \) in (10) infers how many of the potential \( K \) embedded coordinates are needed to represent the data via the associated subspace defined by \( F_{z(i)} \). The embedded components with associated zero components in \( b \) are attributed to the noise, and modeled via \( \epsilon_i \). As discussed in Section 7, the model is therefore naturally amenable to analyzing noisy data, and proves to be particularly effective in inferring data dimensionality in that case.

In our experiments, both the alignment method in Section 3.3 and NSR of this subsection have worked equally well in the context of modeling and analysis of dynamic high-dimensional data (discussed in the next section). However, because of its preservation of global properties in the embedding space, the NSR approach typically yields simpler and more interpretable embeddings. All results presented in Section 7 are based upon NSR.

4. Modeling Latent-Space Dynamics

When interested in analysis of dynamic data \( \{y_t\} \), it is convenient to develop the dynamic model in the (low-dimensional) space of \( \{x_t\} \), as suggested by (8). Below we develop a dynamic model for \( \{x_t\} \), which permits sharing of statistical strength between related forms of dynamic data, and it therefore infers latent relationships within the dynamic data. Once the dynamic model has been so learned, it may be employed to perform synthesis of dynamic data in the latent space, as discussed in Section 5. In Section 5 we also describe how the MFA from Section 3 may be used to map the synthesized data in the latent space to the original high-dimensional space of \( \{y_t\} \), where it is most naturally visualized. Examples of synthesized dynamic data, for motion of humans, is considered in Section 7, wherein we also depict how the model infers relationships between different types of dynamic data.
4.1 Model learning

Assume access to a set of training data, \( \{x_t\} \) inferred from given high-dimensional dynamic data \( \{y_t\} \) using the procedures in Section 3. We develop a Markovian mixture model to represent the mapping from \( x_t \) to \( x_{t+1} \). Specifically, consider the following hierarchical model:

\[
x_{t+1} = A\hat{z}(t)x_t + \epsilon_{t+1}, \quad x_t \sim \mathcal{N}(m\hat{z}(t), \Sigma\hat{z}(t)), \quad \hat{z}(t) \sim \sum_{j=1}^{\infty} c_j \delta_j \tag{18}
\]

where \( c \sim \text{Stick}(\gamma) \), and \( \epsilon_{t+1} \sim \mathcal{N}(0, \gamma_0^{-1}I_d) \); gamma priors are again employed on \( \gamma \) and \( \gamma_0 \). Completing the model, each column of \( A_j \) is drawn iid from a zero-mean Gaussian distribution with diagonal precision matrix and gamma hyperprior, and each \((m_j, \Sigma_j)\) pair is drawn from a normal inverse-Wishart prior. The model in (18) corresponds to (8)-(9). We emphasize that the mixture model in (8)-(9) accounts for the (nonlinear) temporal dynamics, which is distinct from the MFA mixture model, which accounts for the nonlinear (in general) relationship between \( \{x_t\} \) and \( \{y_t\} \).

The generative process of this model represents each \( x_t \in \mathbb{R}^d \) (assuming we have inferred \( d \), as discussed above) as coming from one Gaussian mixture component, represented by mixture index \( \hat{z}(t) \). For this mixture component, there is an associated matrix \( A_{\hat{z}(t)} \in \mathbb{R}^{d \times d} \) used to regress the latent-space vector \( x_{t+1} \) on \( x_t \). Marginalizing over the latent mixture index \( \hat{z}(t) \), we see that the mapping from \( x_t \) to \( x_{t+1} \) is no longer linear. Note that since the latent \( x_{1:T} \) are assumed known at this stage, learning the Markov mixture model mapping solely relies on interpretation of the sequence \( x_{1:T} \) as a sequence of pairs \( \{x_t, x_{t+1}\} \).

4.2 Hierarchical multi-task dynamic model

There are situations for which we may have multiple distinct forms of dynamic data (e.g., motion-capture data for people walking, running, dancing, etc.); let \( x_{1:T,q}^{(q)} \) represent such data for data class \( q \). Rather than learning dynamic models for each data source in isolation, it is desirable to use all available data to jointly infer dynamic models. Such a procedure allows one to infer relationships between different types of data (e.g., what motions in dance are similar to related motions in running). All available data may be aggregated when inferring statistical embeddings of the form discussed in Sections 3.3 and 3.4.
We jointly analyze the dynamics of $Q$ different sources of dynamic data $\{x^{(q)}_{1:T_q}\}_{q=1,Q}$ as

$$x^{(q)}_{t+1} = A z[q,q(t)] x^{(q)}_t + e^{(q)}_{t+1} \quad (19)$$

$$x^{(q)}_t \sim N[m^{(q)}_{z(q,t)}, \Sigma^{(q)}_{z(q,t)}] \quad (20)$$

$$z(q,t) \sim \sum_{j=1}^{\infty} c^{(q)}_j \delta_j \quad (21)$$

$$\hat{z}(q,j) \sim \sum_{k=1}^{\infty} \nu_k \delta_k \quad (22)$$

The parameters $\{A_k\}$ are drawn using the same Gaussian prior as in Section 4.1, while parameters $\{m^{(q)}_j, \Sigma^{(q)}_j\}$ are drawn from the normal inverse-Wishart prior used in Section 4.1 for the related form of the model. The probability vectors are again drawn from truncated stick-breaking representations, $c^{(q)} \sim Stick(\eta_1)$ and $\nu \sim Stick(\eta_2)$, with gamma priors placed on $\eta_1$ and $\eta_2$. The residuals $e_{t+1}$ are modeled as in (18).

The model in (19)-(22) may appear daunting at initial inspection; however, it builds naturally upon the simpler model in (18). Expressions (20) and (21) are the same as the right two expressions in (18). Equations (20) and (21) impose that each data class $q$ is characterized by an underlying (Markov) state space; $z(q,t)$ denotes the state of class $q$ at time $t$. The $q$-dependent mixture weights $\{c^{(q)}_j\}_{j=1,\infty}$ generalize $c_j$ from (18).

Expression (19) imposes an AR model for the dynamics, as in the left expression in (18). The set of dynamic AR models $\{A_k\}$ are shared across all classes $q = 1, \ldots, Q$. This sharing is motivated by the idea that the $Q$ forms of data may be different from a macroscopic perspective, but they may share localized temporal dynamics. Specifically, $\hat{z}[q,q(t)]$ is an indicator variable, assigning a form of AR model to modality $q$, when it is in state $z(q,t)$. This construction therefore shares local forms of dynamic motion, as reflected by a sharing of local AR models $\{A_k\}$.

One may infer relationships between data $x^{(q)}_{1:T_q}$ and $x^{(q')}_{1:T_{q'}}$, for dynamic data types $q$ and $q'$, by examining the degree of sharing they manifest in the underlying dynamics, reflected in sharing regression matrices $\{A_k\}$. Specifically, dynamics modeled by the same component of the set $\{A_k\}$ are likely to be similar. This is examined when presenting results for the joint analysis of multiple types of dynamic motion-capture data.

November 2, 2012 DRAFT
4.3 Gibbs Sampling

For each of the models discussed above, the conditional posteriors for each model parameter, conditioned on all other parameters and the data, may be expressed in closed form. These models are therefore well suited to computations based upon a Gibbs sampler (see, for example, (Ishwaran and James, 2001)). The Gibbs update equations for each of the models considered here are summarized in the Appendix.

5. Synthesis, Reversible Embeddings and Embedding New Data

5.1 Dynamic synthesis

Based on a learned model from training data (as discussed in Section 4.1), one can perform dynamic synthesis. The synthesis is performed for one of the $Q$ types of motion, using the associated model, even if the model was learned jointly with training data from all $Q$ types of data. Given $x_t$, we wish to now synthesize $x_{t+1}$, and this process may be repeated consecutively to synthesize dynamic data $(x_1, x_2, \ldots)$, starting from an arbitrary $x_1$ associated with a given class of data. Note that although our Bayesian nonparametric model specification allows for infinitely many components, our learned model from training data will yield some finite $J$ components. Then, for synthesis we employ the following construction:

$$x_{t+1} = \sum_{j=1}^{J} w_j(x_t) A_j x_t + \epsilon_{t+1}$$  \hspace{1cm} (23)

$$w_j(x_t) = \frac{c_j N(x_t | m_j, \Sigma_j)}{\sum_{j'=1}^{J} c_{j'} N(x_t | m_{j'}, \Sigma_{j'})}$$  \hspace{1cm} (24)

where the residual is drawn $\epsilon_{t+1} \sim N(0, \gamma_0^{-1} I_d)$, with $\gamma_0$ learned when analyzing the training data. The expression $w_j(x_t)$ denotes the probability that $x_t$ is associated with mixture component $j$; typically $w_j(x_t)$ is peaked for one component $j \in \{1, \ldots, J\}$. In the above representation, we have model parameters $\{A_j, m_j, \Sigma_j\}_{j=1}^{J}$, and there is one such set of parameters for each Gibbs collection sample manifested when performing model learning, as in Section 4. The synthesis is performed by averaging across all collection samples, although similar results are found if (for simplicity) only the maximum-likelihood sample is utilized.

The above synthesis is performed in the latent, low-dimensional space $x_t$. It is often of interest to map the dynamics up into the typically high-dimensional space $y_t$, as this is the space in which
the data are originally observed. This motivates the need for reversible embeddings, discussed next.

5.2 Reverse embeddings

Assume we are given an embedded vector $\mathbf{x} \in \mathbb{R}^d$, where $d \leq K$ is the inferred dimensionality of the latent space. We wish to infer the associated data $\mathbf{y} \in \mathbb{R}^D$ in the original high-dimensional space ($D \gg d$). Based upon the model learned as discussed above, on the training data $\mathcal{D} = \{\mathbf{y}_i\}_{i=1,N}$, for each Gibbs collection sample we infer a statistical distribution for the latent variable $\mathbf{x}$:

$$p(\mathbf{x}|\mathcal{D}) = \sum_{m=1}^{M} a_m \mathcal{N}(\mathbf{x}; \hat{\mu}_m, \Sigma_m)$$  \hspace{1cm} (25)

where here $\{a_m, \hat{\mu}_m, \Sigma_m\}_{m=1,M}$ are associated with a particular collection sample. For this collection sample the expected mapping $\mathbf{x} \rightarrow \mathbf{y}$ is

$$\mathbf{y} = \sum_{m=1}^{M} \frac{a_m \mathcal{N}(\mathbf{x}; \hat{\mu}_m, \Sigma_m)}{\sum_{j=1}^{M} a_j \mathcal{N}(\mathbf{x}; \hat{\mu}_j, \Sigma_j)} \left[ F_m \Lambda_m (\mathbf{x} - \hat{\mu}_m) + \mu_m \right]$$  \hspace{1cm} (26)

corresponding to the mapping from each mixture component alone, weighted by the probability that $\mathbf{x}$ (and hence $\mathbf{y}$) is associated with that mixture component. The final mapping is manifested by averaging across Gibbs collection samples (for simplicity, one may also just use the parameters $\{a_m, \hat{\mu}_m, \Sigma_m\}_{m=1,M}$ associated with the maximum-likelihood sample, which was found to yield very similar results).

To constitute synthesized dynamic data in the high-dimensional space $\mathbf{y}$, we first synthesize dynamic data in the embedding space $\mathbf{x}$, as discussed in Section 5.1. With $\{\mathbf{x}_t\}$ so constituted, the methods in this subsection are used to map each $\mathbf{x}_t$ to $\mathbf{y}_t$.

5.3 Statistical embeddings of new data

While it is often of interest to map synthesized latent data $\mathbf{x}_t$ to the generally high-dimensional space $\mathbf{y}_t$, there are also cases for which we are given new, measured high-dimensional data $\mathbf{y}_t$, and we wish to embed this into the low-dimensional latent space $\mathbf{x}_t$ based upon an existing model like that developed in Section 3.4, without having to relearn the models. Assume we have learned the latent-space mixture model in (25), and assume that we have (mean) values for the corresponding mixture-component parameters. The generative process corresponds to
first drawing a mixture component (shared between the low- and high-dimensional spaces), then drawing a latent vector \( x \), and finally mapping this vector to the high-dimensional space \( y \) via the mixture-component-dependent regression mapping (factor loading). If the components of \( \epsilon_i \) are zero-mean with precision \( \alpha_0 \), then the joint probability of \( y \) and \( x \) is

\[
p(y, x) = \sum_{m=1}^{M} a_m N(y; F_m A_m(x - \hat{\mu}_m) + \mu_m, \alpha_0^{-1} I_D) \mathcal{N}(x; \hat{\mu}_m, \Sigma_m)
\]

where we have marginalized out the latent mixture component \( m \). Because of conjugacy, one may analytically express

\[
p(x|y) = p(x, y) / \int p(x, y) dx = \sum_{m=1}^{M} \tilde{a}_m N(x; \tilde{\mu}_m, \tilde{\Sigma}_m)
\]

with

\[
\tilde{a}_m = \frac{a_m N(y; \mu_m, \alpha_0^{-1} I_D + F_m A_m \Sigma_m A_m F_m^T)}{\sum_{m=1}^{M} a_m N(y; \mu_m, \alpha_0^{-1} I_D + F_m A_m \Sigma_m A_m F_m^T)}
\]

\[
\tilde{\Sigma}_m = (\Sigma_m^{-1} + \alpha_0 A_m F_m A_m)^{-1}
\]

\[
\tilde{\mu}_m = \tilde{\Sigma}_m \left[ \alpha_0 A_m F_m^T (y - \mu_m + F_m A_m \hat{\mu}_m) + \Sigma_m^{-1} \hat{\mu}_m \right]
\]

In the above computations, the following identity for a normal distribution is used: \( \mathcal{N}(y; F_m A_m(x - \hat{\mu}_m) + \mu_m, \alpha_0^{-1} I_D) \mathcal{N}(x; \hat{\mu}_m, \Sigma_m) = \mathcal{N}(x; \hat{\mu}_m, \tilde{\Sigma}_m) \mathcal{N}(y; \mu_m, \alpha_0^{-1} I_D + F_m A_m \Sigma_m A_m F_m^T). \)

When performing the embedding \( y \rightarrow x \) for new \( y \notin \mathcal{D} \), instead of yielding a point estimation for \( x \), we achieve the distribution \( p(x|y) \) from (28). When presenting the results, we use the mean value \( \hat{\mu} \) of the inferred \( x \), where \( \hat{\mu} = \sum_{m=1}^{M} a_m \hat{\mu}_m \), again averaging over Gibbs collection samples (but, again, results from the maximum-likelihood collection sample yielded very similar results).

6. Related Work

Our dynamical models are similar in goal to the Bayesian nonparametric Markov switching processes of (Fox et al., 2010, 2011). Specifically, we employ Bayesian nonparametric methods to flexibly model complex temporal dynamics present in multivariate time series. However, the formulations of (Fox et al., 2010, 2011) assume conditionally linear mappings from \( x_t \) to \( x_{t+1} \) and from \( x_t \) to \( y_t \) based on a Markov-evolving latent dynamical mode. We instead assume that the
mappings are based on mixture models and discover the dimensionality of the latent space. The mixture models are developed using techniques similar to those considered in (Ghahramani and Beal, 2000), but with the added goal not addressed there of preserving the concept of proximity in the spaces \( \{x_t\} \) and \( \{y_t\} \), as discussed in Section 3.2 ((Ghahramani and Beal, 2000) did not consider dynamic data, and therefore such preservation of distances was not an issue).

There has been previous research on development of techniques for “aligning” the latent features of an MFA (Verbeek, 2006; Zhang and Zha, 2004; Teh and Roweis, 2003; Verbeck et al., 2004) with the high-dimensional data, such that proximity relationships between elements of the set \( \{y_t\} \) are preserved among elements of \( \{x_t\} \). The method in (Verbeek, 2006) used a MAP solution that is not directly transferable to the fully Bayesian solution of interest here (which allows us to infer a proper number of mixture components, and the dimensionality of each component). The approach in (Zhang and Zha, 2004) is effective, but the inferred model is not explicitly statistical in nature (and again the latent-space dimensionality is set), and consequently it is not appropriate for the proposed model. There has been much research (Teh and Roweis, 2003; Verbeck et al., 2004) on development of two-step processes, in which one first learns an MFA, and then subsequently aligns it. However, two step approaches are well known to be sub-optimal in treating the results of the first step as fixed and known in implementing the second step. Our one-step alignment procedure is compatible with hierarchical Bayesian formulations of the type developed here, in which the goals of MFA learning and latent-space alignment are addressed simultaneously.

There are numerous existing methods for performing the latent space \( \{x_t\} \) embeddings, such as in (Schölkopf et al., 1998; Tenenbaum et al., 2000; Roweis and Saul, 2000; Coifman and Lafon, 2006). Our approach provides a new means of addressing two problems that have attracted significant attention. First, we develop a Bayesian means of inferring the latent dimension \( d \), and we demonstrate that our results compare favorably with existing methods (see (Levina and Bickel, 2004) and the references therein), particularly in the presence of noise. The second problem we address concerns development of a method for embedding new samples \( y \) into the latent space \( x \) without having to retain all the training data used to learn the original embedding. Most existing methods (Coifman and Lafon, 2006; Lawrence, 2005; Drineas and Mahoney, 2005) require access to the training data \( \{y_i\}_{i=1,N} \) used to learn the embedding if one wishes to embed a new \( y_{N+1} \).
7. Experiments

7.1 Hyperparameter settings

The model may appear to have many hyperparameters, but in practice these parameters are set in a “standard” way (Tipping, 2001), with no tuning. Specifically, for all embedding experiments in this paper, the hyperparameter values are $a_0 = 0.02$, $b_0 = 1$, $\xi = 1$ and all gamma priors are set to $\text{Gamma}(10^{-6}, 10^{-6})$. The hyperparameter values for the dynamic model are as follows: $u_0 = 0.1$, $v_0 = K + 2$, and $m_0$ and $B_0$ are set to the sample mean and sample precision of the training set, respectively. All gamma priors are equal to $\text{Gamma}(10^{-6}, 10^{-6})$, as in the embedding experiments. The model was not found to be sensitive to deviations from these standard settings, hence no attempt was made to optimize the hyperparameters.

7.2 Latent-space dimensionality estimation

The proposed nonlinear spectral regression (NSR) model provides an estimate of the latent-space dimension, by counting the number of nonzero elements of the diagonal matrices $\Lambda_z$. The Gibbs sampler provides an estimate of the full posterior distribution on this number, and one example of such a posterior distribution is presented below. In the following experiments on dimensionality estimation we present the posterior mean as our estimate. When implementing NSR dimensionality estimation, we first analyze the data using Isomap (Tenenbaum et al., 2000), and retain the $K$ most significant features, and use these within the method discussed in Section 3.4; the model infers which subset of the $K$ spectral features are needed to represent the data $y_i$, via the factor loadings, with the remaining spectral features discarded and $\epsilon_i$ accounting for the residual. In these experiments we employed truncation $K = 20$, and we truncate the number of mixture components to $M = 20$. A total of 500 burn-in Gibbs samples were employed, with 500 collection samples (but typically good mean parameter values were inferred after only 50 collection samples).

We compare the NSR dimensionality estimates with those from the following methods: Maximum Likelihood Estimation (MLE), Eigenvalue thresholding, Geodesic Minimum Spanning Tree (GMST) and Correlation Dimension (CorrDim). These methods are all described in (Levina and Bickel, 2004) and references therein.
We have experimented with simulated spheres, balls, cubes and Gaussians of dimensions ranging from two to 25, embedded in 100-dimensional observation space. For brevity, we only show results in Figure 2 for a 6-dimensional ball, a 4-dimensional cube, Gaussians of dimension 8 and 10, a 2-dimensional Swiss roll, and a 2-dimensional torus, the two latter datasets being embedded in three-dimensional observation space; the Swiss-roll example is from (Tenenbaum et al., 2000). We consider additive zero-mean, i.i.d. Gaussian noise with standard deviations $\sigma_{\text{noise}}$ set to 0, 0.01, 0.05 and 0.1. For these examples, these noise levels correspond to the signal-to-noise ratio (SNR) ratios presented in Table I, which are computed according to $\text{SNR} = 10 \log_{10}(\frac{\|x\|_2^2}{D\sigma_{\text{noise}}^2})$, with $\|x\|_2^2$ being the average squared $\ell_2$ norm of the vectors drawn from each manifold. The SNR values are given for completeness, but they are an imperfect measure of noise level; for our purposes greater concern is placed on the noise level at which the data no longer support a dimensionality consistent with the original noise-free data (shown below in an example).

The results are averaged across 100 realizations of the noise and are representative of the wider set of experiments. Note that NSR consistently provides the best dimensionality estimation, particularly at high noise levels.

Note that the NSR method provides highly accurate estimates of the dimensionality of the data, up to a noise standard deviation of 0.1 (SNR as low as -1 dB). To get a sense of when such dimensionality estimation breaks down with increasing noise level, we reconsider the Swiss-roll data, which is readily visualized since it is in three dimensions. In Figure 3 we depict the noise-free data (from (Tenenbaum et al., 2000)), and also show example data draws for noise standard deviation 0.1 and 0.2. All methods, including NSR, fail for $\sigma_{\text{noise}} = 0.2$ (SNR 8 dB). For this noise level, the surfaces of the roll come together, and therefore it is not surprising that the model no longer estimates the data dimension to be two.

7.3 Embeddings

To further demonstrate embedding performance, we consider the following widely studied datasets: the teapot data (Teh and Roweis, 2003), the MNIST digit database, and two face datasets from (Tenenbaum et al., 2000). We again use Isomap to supply the latent coordinates for the NSR formulation. The above results indicate that NSR infers the latent-space (embedding) dimensionality effectively, and therefore an important aspect of this subsection concerns the efficient embedding of new data without having to return to the training data, and also the ability to synthesize new
data. In these examples the truncation level of the latent space is $K = 30$, and the truncation level on the number of mixture components is $M = 60$. We employed 2000 burn-in Gibbs samples, and 500 collection samples.

The teapot dataset is comprised of rotated teapot images. There are 400 RGB images, each one of size $101 \times 76$. Figure 4 shows the NSR embedding, which is smooth, and has no self-intersections and exhibits the expected circular topology. Also for the teapot data, we compare the out-of-sample performance of NSR to that of Isomap with the Nyström approximation. Note that, unlike traditional spectral methods, NSR does not require the Nyström approximation (Drineas and Mahoney, 2005) to embed new samples, because it learns two-way mappings between low and high-dimensional space. As shown in Figure 4, NSR is comparable to Nyström, for 25% and 50% out-of-sample data (5 runs, with different random data partitions); the metric we have adopted is the MSE between the embedding with the full training set and the embedding learned used only the in-sample data. An important distinction between the proposed NSR and Nyström is that the latter requires access to the training data when embedding new samples, while NSR does not (see Section 5.3; a full distribution is available for embedding new data, and here we present the mean). Also shown are the MSE results for embedding the MNIST (400 images per digit from 0-9) and the face datasets with NSR. We do not compare these with the Nyström method due to their more irregular and high-dimensional nature, which makes the Nyström approximation more unstable. To further examine the embedding associated with the MNIST data, example results are shown in Figure 5, with the embedding shown in two dimensions for digits 0-4.

We now present results on the face images considered in (Roweis and Saul, 2000). The images are grayscale with $20 \times 28$ pixels. Figure 6 shows a 2D view of the embedding computed by NSR. The model inferred 21 clusters, and an approximation to the posterior distribution of the latent-space dimensionality is also depicted in Figure 6. Further, in Figure 6 are shown NSR-generated synthesized face images along five selected cuts (A–E) in latent space, depicting smooth evolution of facial expressions. The synthesis of these faces is performed as discussed in Section 5.2; we present the expected synthesized high-dimensional image, based upon the corresponding position in the low-dimensional embedding space. The cuts are straight lines in 12 dimensions (we select the mean dimension from the aforementioned posterior on the dimensionality); the figure depicts
2D projections of the cuts from the 12-dimensional space. The endpoints of each cut are actual images from the training set, and the intermediate images are synthesized.

Considering cut A in Figure 6, which shows the subject with his tongue slowly sticking out, we show comparative results for alternative synthesis methods (top-right in Figure 6). The three alternative methods are: (i) mapping a latent feature vector to its nearest neighbor (NN) from the training set, and using the associated high-dimensional representation as the synthesis (consequently, all “synthesized” data actually come from the training set); (ii) performing SVD on the training data, and using the SVD coordinates as latent features, and the principal components to perform synthesis (denoted SVD); and (iii) linear interpolation (LI) in the high-dimensional space, using the endpoints of the line. Note that LI and NN require access to the high-dimensional data for synthesizing any new image, and LI never actually operates in the low-dimensional embedding space (of interest in the next section, when we consider synthesis of dynamic data). As expected, NN yields a non-continuous discretized synthesis, and SVD loses details in the face (the tongue is blurred). The LI results are comparable to those of the proposed method, NSR, but LI does not achieve our goal of a low-dimensional embedding. By contrast NSR yields effective synthesis from a low-dimensional embedding space, and the high-dimensional training data are not needed after the model is learned.

7.4 Dynamic analysis & synthesis

We consider motion-capture data available from http://people.csail.mit.edu/ehsu/work/sig05stf (termed MIT data) and from http://mocap.cs.cmu.edu (termed CMU data), as in (Taylor et al., 2007; Wang et al., 2008). We obtained the animations by modifying code from http://www.dcs.shef.ac.uk/~neil/mocap. We have used 6 exercise routines from Subject 13 and 14 in the CMU data (as in (Fox et al., 2010)) and 4 sequences of walking and jogging from the MIT data. Each CMU frame is 62-dimensional, while the MIT data is 108-dimensional. The CMU and MIT data sources are analyzed separately; however, in each case all forms of motion are analyzed jointly, as discussed in Section 4.2.

When learning the embedding model, the latent dimension was truncated at $K = 30$ and the number of mixture components was truncated at $M = 60$; 2000 burn-in Gibbs samples were run, with 500 collection samples. For the dynamic model discussed in Section 4, the truncation level on the number of mixture components for both stick-breaking constructions was set at $J = 80$. 

November 2, 2012
For the dynamic model we have considered 2000 Gibbs burn-in iterations and 500 collection iterations. As an example, when analyzing the CMU data, the NSR embedding model infers 50 mixture components and a latent dimensionality of 8, and the associated dynamic model infers 58 mixture components; for the dynamic analysis the NSR embedding was implemented via diffusion (Coifman and Lafon, 2006), with similar results obtained via Isomap. The diffusion was performed using the radial-basis function kernel discussed in Section 3.3.

Each form of motion is characterized by a mixture model (Section 4.2), and the dynamic motion associated with each mixture component is defined by a local linear-regression model. We analyze all forms of dynamic data together, and using the method in Section 4.2 we cluster the different types of dynamic behavior (ideally, each inferred mixture component will represent a different form of dynamic motion). In Figure 7 we depict example dynamic data associated with five of the inferred mixture components; the model effectively learns and clusters different basic forms of motion. These results can be observed in greater detail by viewing the actual video sequences, which have been posted to https://sites.google.com/site/npbnsr. Similar motion-clustering results were presented in (Fox et al., 2010), but the model in (Fox et al., 2010) is not capable of synthesis since the models were learned using only a subset of the $D$ dimensions of the time series.

We now consider dynamic-data synthesis. To provide a quantitative analysis, and to compare with other methods, we removed a contiguous set of 30-frames from the videos of two different subjects before dynamic training, and then computed the root mean-squared error (RMSE) on the held out frames, using the dynamic model to synthesize missing frames; we also computed the maximum RMSE difference between consecutive synthesized frames, to quantify the smoothness of the synthesized data. Results are averaged for 12 different windows (i.e., frames 35-65, 36-66, etc., removed). The two subjects considered in this example corresponded to running motion, but results were similar for all other motions considered. As comparisons, we considered learning an embedding based on a factor-analysis model, rather than the employed NSR mixture model. The purpose of this test is to examine the value of the nonlinear NSR embedding procedure, compared to linear FA (which is essentially statistical PCA). After performing the FA-based embedding, the dynamic motion was modeled exactly as used based on the NSR embedding, to provide a fair comparison. Additionally, we considered direct cubic spline interpolation in the high-dimensional
space, thereby not explicitly learning a dynamic model. As observed in Table II, the proposed method yields excellent performance relative to these alternatives. This is manifested not only in smaller RMSE, but also in smoother and more natural-looking motion (quantified via inter-frame differences); FA has reasonable RMSE but gross motion discontinuities (manifested by larger inter-frame differences), while the spline motion is smooth but very highly distorted relative to truth (high RMSE). We also tried, as a comparison, to do synthesis based on a dynamic model learned directly in the original high-dimensional space, without the intervening step of embedding to a low-dimensional latent space; this failed completely, based upon the limited training data available.

We now employ the inferred dynamic model to generate motion automatically in the latent space, and then to project this back to the high-dimensional space for visualization (i.e., computer-generate dynamic-motion synthesis). In Figure 8 we provide a small example of such synthesized motion, demonstrating the power of modeling the dynamics of multiple types of motion simultaneously (as discussed in Section 4.2). Multiple types of motion have the opportunity to share local linear dynamic models (by sharing associated model parameters within the HDP). To generate the data in Figure 8, we initially synthesized dynamic data by using the learned walking model, and when this model moved into a part of latent space at which it shared dynamics with the running model, we turned over the dynamics to the running model. This allows us to synthesize a walking sequence followed by running; similar types of transitions may be manifested using the other forms of motion, assuming they share local dynamics. Figure 8 shows the synthesized smoothed transition from walking to running.

As a final example, in Figure 9 we show synthesized data for a limping sequence. This shows the range of dynamic motion that may be synthesized in the high-dimensional space, based upon dynamic models learned in the low-dimensional embedding space. The full video for these and other examples is at https://sites.google.com/site/npbnsr.

7.5 Brief discussion of computations

Because of the conjugate-exponential form of all aspects of the hierarchical models, all analysis is performed using analytic Gibbs update equations (see Appendix). Consequently, while the hierarchical models are relatively sophisticated, the detailed collapsed Gibbs inference is relatively routine. All computations were performed in (non-optimized) Matlab and were run efficiently on
A new statistical spectral embedding framework is proposed. This framework provides the ability to discard the training data when embedding new data, while also allowing synthesis of high-dimensional data from the embedding space. This procedure has been used to analyze high-dimensional dynamic data, with the nonlinear dynamics learned in the low-dimensional latent space. Key to handling such data is the fact that our embedding maintains a sense of proximity in that nearby observations yield nearby latent embedded features. The proposed hierarchical dynamic model performs joint learning of dynamics from multiple types of motion, allowing the learning of shared structure. The model has also been demonstrated as a tool for analysis and synthesis of high-dimensional dynamic data. Finally, the method has proven an effective tool for estimating the dimensionality of a dataset, even in the presence of substantial additive noise. The model has been implemented using Gibbs sampling, in which all update equations are analytic, and the models have been found to mix efficiently. Key aspects of the model are that the Dirichlet process is used to infer the number of mixture components in the MFA and in the associated dynamic model, and a variation of the beta-Bernoulli process is used to infer the dimensionality of the low-dimensional latent subspace.

Concerning future research, despite the fact that the computations proved to be efficient for the examples considered here, there may be larger-scale problems for which further computational acceleration may be desired. One may consider other approximate inference engines, such as variational Bayesian analysis (Ghahramani and Beal, 2000). Such an analysis should be possible with analytic update equations, as was possible for Gibbs sampling, wherein we exploit the fact that consecutive terms in the hierarchical model are in the conjugate-exponential family.
Appendix: Summary of Gibbs Update Equations

8.1 Update Equations for the NSR Model

Define \( \{a_0, b_0\}, \{c_0, d_0\}, \{e_0, f_0\}, \{g_0, h_0\} \) and \( \{\tau_{10}, \tau_{20}\} \) as the hyperparameters for \( \pi_k, \alpha_{0j}, \sigma_{jk}, \alpha, \) and \( \zeta, \) respectively; their settings are specified in Section 7. In addition, let \( V_j \) be the stick length as defined in (13), and therefore, \( w_j = V_j \prod_{i=1}^{j-1} (1 - V_i) \). Other variables are defined as in Section 3. The Gibbs sampling inference algorithm for the NSR model can be summarized as follows:

1) Sample the cluster index \( z(i) \) from

\[
p(z(i) = j | -) \propto w_j \mathcal{N}(y_i; \mu_j - \hat{\mu}_j) + \mu_j, \alpha_{0j}^{-1} I_D) \mathcal{N}(x_i; \mu_j, \beta_j^{-1} I_K)
\]

After normalization across \( z, p(z(i) = j | -) \) becomes a Multinomial distribution.

2) Sample the DP concentration parameter \( \alpha \) from

\[
p(\alpha | -) = \text{Gamma} \left( \alpha; g_0 + J - 1, h_0 - \sum_{i=1}^{J-1} \log(1 - V_i) \right).
\]

3) Sample the DP stick length \( V_j \) from

\[
p(V_j | -) = \text{Beta} \left( V_j; 1 + \sum_{i: z(i) = j} 1, \alpha + \sum_{i: z(i) > j} 1 \right), \quad j = 1, 2, \cdots, J - 1 \text{ and } V_J = 1.
\]

Therefore, \( w_j = V_j \prod_{i=1}^{j-1} (1 - V_i) \).

4) \( p(b_j, \lambda_j | -) \propto \text{Bernoulli}(b_j; \pi_k) \mathcal{N}(\lambda_k; 0, 1) \prod_{i=1}^{M} \mathcal{N} \left( y_{ik}^{-k}; F_{z(i)k}(x_{ik} - \hat{\mu}_{z(i)}) b_k \lambda_k, \alpha_{0z(i)}^{-1} I \right) \)

with \( y_{ik}^{-k} \triangleq y_{ik} - \sum_{m \neq k} F_{z(i)m}(x_{ik} - \hat{\mu}_{z(i)}) b_k \lambda_k - \mu_{z(i)} \). Thus we can sample \( b_k \) and \( \lambda_k \) from

\[
p(b_k | -) = \text{Bernoulli}(b_k; \tilde{\pi}_k) \]

\[
p(\lambda_k | b_k, -) = b_k \mathcal{N}(\lambda_k; \delta_k, \gamma_k) + (1 - b_k) \mathcal{N}(\lambda_k; 0, 1)
\]

where

\[
\log \frac{\bar{\pi}_k}{1 - \bar{\pi}_k} = \log \frac{\pi_k}{1 - \pi_k} + \frac{1}{2} \log \gamma_k + \frac{\delta_k^2}{2 \gamma_k}
\]

\[
\gamma_k = \left( 1 + \sum_{i=1}^{M} \alpha_{0z(i)} F_{z(i)k}^T F_{z(i)k} (x_{ik} - \hat{\mu}_{z(i)k})^2 \right)^{-1}
\]

\[
\delta_k = \gamma_k \left( \sum_{i=1}^{M} \alpha_{0z(i)} F_{z(i)k} y_{ik}^{-k} (x_{ik} - \hat{\mu}_{z(i)k}) \right)
\]

5) Sample the Bernoulli parameter \( \pi \) from

\[
p(\pi_k | -) = \text{Beta}(\pi_k; a_0 / K + b_k, b_0 (K - 1) / K + 1 - b_k).
\]
6) Sample the mean vector for each cluster \( \mu_j \) from \( p(\mu_j | -) = \mathcal{N}(\mu_j; \varsigma_j, \Gamma_j) \) where 
\[
\Gamma_j = (\gamma + \alpha_{0j} \sum_{i:z(i)=j} 1)^{-1} I_d; \quad \varsigma_j = \Gamma_j (\alpha_{0j} \sum_{i:z(i)=j} (y_i - \mathbf{F}_j \text{diag}(b \circ \lambda)(x_i - \hat{\mu}_j)))
\]
7) Sample the factor loading matrix \( \mathbf{F}_j \) from 
\[
p(\mathbf{F}_{jk} | -) = \mathcal{N}(\mathbf{F}_{jk}; \eta_{jk}; \Sigma_{jk})
\]
where \( \mathbf{F}_{jk} \) denotes the \( k \)th row of matrix \( \mathbf{F}_j \) and 
\[
\Sigma_{jk} = (\zeta I_K + \alpha_{0j} \text{diag}(b \circ \lambda) \sum_{i:z(i)=j} (x_i - \hat{\mu}_j)(x_i - \hat{\mu}_j)^\top \text{diag}(b \circ \lambda))^{-1}
\]
\[
\eta_{jt} = \Sigma_{jk}(\alpha_{0j} \text{diag}(b \circ \lambda) \sum_{i:z(i)=j} (x_i - \hat{\mu}_j)(y_{ik} - \mu_{jk}))
\]
8) Sample the precision of the column of factor loading \( \zeta \) from \( p(\zeta | -) = \text{Gamma}(\zeta; \tau_1, \tau_2) \) with 
\[
\tau_1 = \tau_{10} + \frac{K J N}{2}; \quad \tau_2 = \tau_{20} + \frac{1}{2} \sum_{j=1}^{J} \sum_{k=1}^{K} \mathbf{F}_{jk}^\top \mathbf{F}_{jk}
\]
9) Sample the precision of the additive noise \( \alpha_{0j} \) from \( p(\alpha_{0j} | -) = \text{Gamma}(\alpha_{0j}; c_j, d_j) \) with 
\[
c_j = c_0 + \frac{D}{2} \sum_{i:z(i)=j} 1; \quad d_j = d_0 + \frac{1}{2} \sum_{k=1}^{D} \sum_{i:z(i)=j} \| y_{ik} - \mathbf{F}_{jk} \text{diag}(b \circ \lambda)(x_i - \hat{\mu}_j) - \mu_{jk} \|^2
\]
10) Sample \( \hat{\mu}_j \) from \( p(\hat{\mu}_j | -) = \mathcal{N}(\hat{\mu}_j; \mu_{\hat{\mu}_j}, \Sigma_{\hat{\mu}_j}) \) \( m = 1, 2, \ldots, N \) with 
\[
\Sigma_{\hat{\mu}_j} = \left( (\alpha_{0j} \text{diag}(b \circ \lambda) \mathbf{F}_j^\top \mathbf{F}_j \text{diag}(b \circ \lambda) + \Sigma_j^{-1}) \sum_{i:z(i)=j} 1 + \xi I_d \right)^{-1}
\]
\[
\mu_{\hat{\mu}_j} = \Sigma_{\hat{\mu}_j} \left( \sum_{i:z(i)=j} \Sigma_j^{-1} x_i - \text{diag}(b \circ \lambda) \mathbf{F}_j^\top \sum_{i:z(i)=j} (y_i - \hat{\mu}_j) \right)
\]
where \( \hat{\mu}_j \triangleq \mu_j + \mathbf{F}_j \text{diag}(b \circ \lambda)x_i \)
11) Sample \( \Sigma_{jk} = \text{diag}(\sigma_{jk1}^{-1}, \sigma_{jk2}^{-1}, \ldots, \sigma_{jkC}^{-1}) \) from 
\[
p(\sigma_{jk} | -) = \text{Gamma}(\sigma_{jk}; c_0 + \frac{1}{2} \sum_{i:z(i)=j} 1, f_0 + \frac{1}{2} \sum_{i:z(i)=j} \| x_{ik} - \hat{\mu}_{jk} \|^2)
\]

### 8.2 Update Equations for the Multi-task Dynamic Model

Define \( \{\alpha_0, b_0\}, \{u_0, v_0, \mathbf{m}_0, \mathbf{B}_0\}, \{e_0, f_0\}, \{\tau_{10}, \tau_{20}\} \) and \( \{\tau_{30}, \tau_{40}\} \) as the hyperparameters for \( \lambda_k, \{\mathbf{m}, \Sigma\}, \gamma q, \eta_1, \) and \( \eta_2, \) respectively; their settings are specified in Section 7. In addition, let \( U_k, V_{qj} \) be the stick lengths as defined in (13), and therefore, \( \nu_k = U_k \prod_{l=1}^{k-1} (1 - U_l) \) and
\[ c_j^{(q)} = V_{qj} \prod_{t=1}^{j-1} (1 - V_{qt}). \] Other variables are defined as in Section 4. The Gibbs sampling inference algorithm for the multi-task dynamic model can be derived as follows:

1) Sample the cluster index \( z_{q,i,t} \) from

\[
p(z_{q,i,t} = j \mid -) \propto c_j^{(q)} N(x_t^{(q,i)}; q_j^{(q)}, (\Sigma_j^{(q)})^{-1}) N(x_{t+1}^{(q,i)}, A_{z_{q,j}}, x_t^{(q,i)}, \gamma_0^{-1} I)
\]

After normalization across \( z \), \( p(z_{q,i,t} = j \mid -) \) becomes a Multinomial distribution.

2) Sample the local DP concentration parameter \( \eta_1 \) from

\[
p(\eta_1 \mid -) = \text{Gamma}\left( \eta_1; \tau_{10} + Q(J - 1), \tau_{20} - \sum_{q=1}^{Q} \sum_{j=1}^{J-1} \log(1 - V_{qj}) \right).
\]

3) Sample the local DP stick length \( V_{qj} \) from

\[
p(V_{qj} \mid -) = \text{Beta}\left( V_{qj}; 1 + \sum_{z_{q,i,t} = j} 1, \eta_1 + \sum_{z_{q,i,t} > j} 1 \right), \quad j = 1, 2, \ldots, J - 1 \text{ and } V_{qJ} = 1. \text{ Therefore, } c_j^{(q)} = V_{qj} \prod_{t=1}^{j-1} (1 - V_{qt}).
\]

4) Sample the cluster index \( \hat{z}_{q,j} \) from

\[
p(\hat{z}_{q,j} = k \mid -) \propto \nu_k N(x_t^{(q,i)}; A_{k} x_t^{(q,i)}, \gamma_0^{-1} I)
\]

After normalization across \( \hat{z} \), \( p(\hat{z}_{q,j} = k \mid -) \) becomes a multinomial distribution.

5) Sample the global DP concentration parameter \( \eta_2 \) from

\[
p(\eta_2 \mid -) = \text{Gamma}\left( \eta_2; \tau_{30} + J' - 1, \tau_{40} - \sum_{k=1}^{J'-1} \log(1 - U_k) \right).
\]

6) Sample the global DP stick length \( U_k \) from

\[
p(U_k \mid -) = \text{Beta}\left( U_k; 1 + \sum_{\hat{z}_{q,i,t} = k} 1, \alpha + \sum_{\hat{z}_{q,i,t} > k} 1 \right), \quad k = 1, 2, \ldots, J' - 1 \text{ and } U_{J'} = 1. \text{ Therefore, } \nu_k = U_k \prod_{l=1}^{k-1} (1 - U_l).
\]

7) Sample \( \lambda_k \) from

\[
p(\lambda_k \mid -) = \text{Gamma}(a_0 + \frac{dJ'}{2}, b_0 + \frac{1}{2} \sum_{l=1}^{d} \sum_{j=1}^{J'} A_{kjl}^2)
\]

where \( A_{kjl} \) is the \( l \)-th element of \( A_{kj} \)

8) Sample \( A_{kj} \) from \( p(A_{kj} \mid -) = N(\mu_{kj}, \Lambda_{kj}) \)

\[
\Lambda_{kj} = \left( \gamma_0 \sum_{\hat{z}_{q,i,t} = j} x_{tk}^{(q,i)} I + \text{diag}(\lambda) \right)^{-1}
\]

\[
\mu_{kj} = \Lambda_{kj} \left( \gamma_0 \sum_{\hat{z}_{q,i,t} = k} x_{tj}^{(q,i)} (x_{t+1}^{(q,i)} - \sum_{i \neq j} A_{ki} x_{ti}^{(q,i)}) \right)
\]
9) Sample the mean vector and covariance matrix for each cluster \( m_j^{(q)}, \Sigma_j^{(q)} \) from

\[
v_{qj} = v_0 + \sum_{z_{q,i,t}=j} 1, \quad u_{qj} = u_0 + \sum_{z_{q,i,t}=j} 1, \quad m_j^{(q)} = \frac{u_0 m_0 + \sum_{z_{q,i,t}=j} x_t^{(q,i)}}{u_{qj}},
\]

\[
(\Sigma_j^{(q)})^{-1} = B_0^{-1} + \sum_{z_{q,i,t}=j} x_t^{(q,i)} x_t^{(q,i)T} + u_0 m_0 m_0^T - \frac{(u_0 m_0 + \sum_{z_{q,i,t}=j} x_t^{(q,i)}) (u_0 m_0 + \sum_{z_{q,i,t}=j} x_t^{(q,i)})^T}{u_{qj}}.
\]

10) Sample the precision of the additive noise \( \gamma_q \) from \( p(\gamma_q|-) = \text{Gamma}(\gamma_q; e_q, f_q) \)

with

\[
e_q = e_0 + \frac{K(T_q - 1)}{2}; \quad f_q = f_0 + \frac{1}{2} \sum_{t=1}^{T_q-1} \|x_t^{(q,i)} - A_k x_t^{(q,i)}\|_2^2.
\]

References


Fig. 1: Schematic of two Gaussians in a mixture model. In the original high-dimensional space $\mathbb{R}^D$ the mixtures partially overlap (top), while this is not guaranteed in the low-dimensional latent space $\mathbb{R}^d$, $d \ll D$. At bottom-right, the high-dimensional mixture components and the latent $x_i$ are aligned, in that if $y_i$ and $y_j$ are proximate, so are the corresponding latent $x_i$ and $x_j$.

**TABLE I:** Signal-to-noise ratio (SNR) ratios for the dimensionality estimation experiments.

<table>
<thead>
<tr>
<th>Noise levels</th>
<th>0.01</th>
<th>0.05</th>
<th>0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>6D ball</td>
<td>19 dB</td>
<td>5 dB</td>
<td>-1 dB</td>
</tr>
<tr>
<td>4D cube</td>
<td>21 dB</td>
<td>7 dB</td>
<td>1 dB</td>
</tr>
<tr>
<td>8D/10D Gaussian</td>
<td>29/30 dB</td>
<td>15/16 dB</td>
<td>9/10 dB</td>
</tr>
<tr>
<td>2D torus</td>
<td>31 dB</td>
<td>17 dB</td>
<td>11 dB</td>
</tr>
<tr>
<td>2D Swiss-roll</td>
<td>34 dB</td>
<td>20 dB</td>
<td>14 dB</td>
</tr>
</tbody>
</table>

**TABLE II: **RMSE for held-out 30-frame windows, and maximum squared norm of the difference between consecutive frames (higher values mean less smoothness). NSR is compared with factor analysis (FA) and spline interpolation.

<table>
<thead>
<tr>
<th></th>
<th>NSR</th>
<th>FA</th>
<th>Spline</th>
<th>NSR</th>
<th>FA</th>
<th>Spline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject 1</td>
<td>31.11</td>
<td>37.68</td>
<td>81.38</td>
<td>53.54</td>
<td>78.01</td>
<td>46.89</td>
</tr>
<tr>
<td>Subject 2</td>
<td>27.57</td>
<td>35.99</td>
<td>75.19</td>
<td>57.08</td>
<td>80.02</td>
<td>48.25</td>
</tr>
</tbody>
</table>
Fig. 2: Latent-space dimensionality estimation on simulated data. We generated data from a 6–dimensional ball, a 4–dimensional cube, Gaussians of dimension 8 and 10, a 2-dimensional Swiss roll and a 2-dimensional torus. In the ball, cube and Gaussian cases, we embedded on 100 dimensions. In the Swiss roll and torus cases we embedded in three dimensions. In all cases, we added Gaussian noise with varying standard deviation (noise level in the plots). We compare our method (nonlinear spectral regression, NSR) with Maximum Likelihood Estimation (MLE), Eigenvalue thresholding (EigValue), Geodesic Minimum Spanning Tree (GMST) and Correlation Dimension (see (Levina and Bickel, 2004)). Results are averaged over 100 runs, with standard deviations depicted.
Fig. 3: Data from a “Swiss roll” manifold, with varying amounts of added Gaussian noise. From left to right, the standard deviation $\sigma_{\text{noise}}$ of the noise is 0, 0.05 and 0.2. The color gradient indicates position, lengthwise, along the manifold. Note how, for the highest noise setting, the manifold structure is lost, due to short-circuits between different folds of the manifold. Any estimation algorithm is unlikely to recover the true intrinsic dimension. The data points here correspond to the original data $\{y_i\}$ under analysis.

Fig. 4: Left: NSR embedding of teapot dataset based on all of the data; the color bar represents the true relative angle. Right: Performance of NSR on inferring latent features of held-out data, on four data sets; for the teapot data, we compare to Nyström’s method (Drineas and Mahoney, 2005). In the left figure, the pictures of the teapot correspond to example $y_i$, and the color-coded points in two-dimensional space correspond to the embedding $x_i$.  

<table>
<thead>
<tr>
<th>Data Set</th>
<th>$50%$ out-of-sample MSE</th>
<th>$25%$ out-of-sample MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teapot (Nyström)</td>
<td>0.0292 ± 0.0152</td>
<td>0.0195 ± 0.0047</td>
</tr>
<tr>
<td>Teapot</td>
<td>0.0376 ± 0.0118</td>
<td>0.0165 ± 0.0069</td>
</tr>
<tr>
<td>MNIST</td>
<td>0.1075 ± 0.0285</td>
<td>0.0844 ± 0.0252</td>
</tr>
<tr>
<td>Frey faces</td>
<td>0.0632 ± 0.0184</td>
<td>0.0429 ± 0.0171</td>
</tr>
<tr>
<td>ISOMAP faces</td>
<td>0.0827 ± 0.0168</td>
<td>0.0582 ± 0.0125</td>
</tr>
</tbody>
</table>
Fig. 5: Embedding for digits "0"–"4" in two dimensions. Note how, for instance, digits "1" and "4" are embedded near each other. The actual figures of the digits correspond to the $y_i$, and their location in the two-dimensional space define the (two-dimensional) embedding coordinates $x_i$. 
Fig. 6: Face embedding and synthesis. **Left:** five straight line cuts (A–E) are shown in latent space, projected down to 2D. The endpoints of each cut (with colored borders) are images from the training set, while the intermediate images are synthesized. **Top right:** comparison of NSR with alternative methods for cut A, where the subject is slowly sticking his tongue out. From top to bottom, we show results using the nearest neighbors in the training set to the sample points on the cut (NN), the linear interpolation in high dimensions between endpoints (LI), SVD–based synthesis using 12 coordinates and our NSR–based synthesis, also using 12 coordinates. **Bottom right:** histogram of inferred dimensionality values from the posterior. The pictures of the faces correspond to $y_i$, and the points (left) correspond to the two-dimensional rendering of $x_i$. 
Cluster 31  Cluster 40  Cluster 42  Cluster 43  Cluster 53
Side twists (R)  Side twists (L)  Jumping jacks  Jog  Kick

Fig. 7: Clustering multiple dynamical behaviors, with dynamic modeling performed jointly. Each plot shows one 16-frame sequence from the training set, with the last frame in a darker color. Each column corresponds to the same most probable cluster. We illustrate five representative clusters, with the cluster index arbitrary. Analysis performed with the CMU data. The figures represent renderings of the data $y_i$, and each $y_i$ has an associated low-dimensional $x_i$ (not shown).

Fig. 8: Fully synthesized motion sequence, with transition from walking (blue) to running (red). Model learning was performed using the MIT data. These data correspond to the synthesized $y_i$, and there is an underlying sequence of $x_i$ (not shown).
Fig. 9: Fully synthesized motion sequence, showing limping behavior. Model learning was performed using the MIT data.