International Journal of Multimedia and Ubiquitous Engineering
Vol.10, No.9 (2015), pp.17-30
http://dx.doi.org/10.14257/ijmue.2015.10.9.03

A Survey of Motion Capture Data Earning as High Dimensional Time Series

Zhou Zhi-Min and Chen Zhong-Wen

Department of Computer Science
Zhejiang University of Water Conservancy and Electric Power, Hangzhou, China
Email: {zhouzhm, chenzwg}@zjweu.edu.cn

Abstract

Constructing effective and generalizable synthesized motions is crucial for creating naturalistic, versatile, and effective virtual characters and robots. High dimensional time series are endemic in applications of machine learning such as robotics (sensor data), computational biology (gene expression data), vision (video sequences) and graphics (motion capture data). Practical nonlinear probabilistic approaches to this data are required. I would like to go through several existing models such as Gaussian Process Dynamic Systems and Deep Belief Networks. I would analyze their strengths and limitations. I would also try to incorporate physical constraints to improve the motion quality. And on the other hand, try to improve the structure of the models or the learning algorithms.

Keywords: *Gaussian process, restricted Boltzmann Machine, variational inference, human motion*

1. Introduction

Nonlinear probabilistic modeling of high dimensional time series data is a key challenge for the machine learning community. A central difficulty in modeling high dimensional time-series data is in determining a model that can capture the nonlinearities of the data without over fitting. The simplest time series models, and the earliest studied, contain no hidden variables. Two members of this class of fully-observed models are the vector autoregressive emodel and the Nth order Markov model. Though elegant in their construction, these models are limited by their lack of memory. To capture long-range structure they must maintain explicit links to observations in the distant past, which results in a blow-up in the number of parameters. The strong regularities present in many time series suggest that a more efficient parameterization is possible.

Linear autoregressive models require relatively few parameters and allow closed-form analysis, but can only model a limited range of systems. In contrast, existing nonlinear probabilistic models can model complex dynamics, but may require large training sets to learn accurate MAP models.

A standard approach is to simultaneously apply a nonlinear dimensionality reduction to the data whilst governing the latent space with a nonlinear temporal prior. The key difficulty for such approaches is that analytic marginalization of the latent space is typically intractable.

Gaussian processes (GPs) (see *e.g.* [1]) are stochastic processes over real-valued functions. GPs offer a Bayesian nonparametric framework for inference of highly

ISSN: 1975-0080 IJMUE

nonlinear latent functions from observed data. They have become very popular in machine learning for solving problems such as nonlinear regression and classification.

Other powerful models, such as the popular hidden Markov model (HMM), introduce a hidden (or latent) state variable that controls the dependence of the current observation on the history of observations. HMMs, however, cannot efficiently model data that is a result of multiple underlying influences since they rely on a single, discrete K-state multinomial to represent the entire history of observations. To model N bits of information about the past, they require 2N hidden states.

2. Related Works

Directed acyclic graphical models (or Bayes nets) are a dominant paradigm in models of static data. Their temporal counterparts, dynamic Bayes nets [2], generalize many existing models such as the HMM and its various extensions. In all but the simplest directed models, inference is made difficult due to a phenomenon known as explaining away where observing a child node renders its parents dependent [3]. To perform inference in these networks, typically one resorts to approximate techniques such as variational inference [4] or Monte Carlo methods which have a significant number of disadvantages [2,5]. An alternative to directed models is to abandon the causal relationship between variables, and instead focus on undirected models. One such model, the restricted Boltzmann machine (RBM) [6], has garnered recent interest due to its desirable property of permitting efficient exact inference. Unfortunately this comes at a cost: Exact Manuscript written August 15, 2014 maximum likelihood learning is no longer possible due to the existence of an intractable normalizing constant called the partition function. However, the RBM has an efficient, approximate learning algorithm, contrastive divergence (CD) [7] that has been shown to scale well to large problems. RBMs have been used in a variety of applications [8–12] and over the last few years their properties have become better understood [13–15]. The CD learning procedure has also been improved [16–18]. With a few exceptions [19,20] the literature on RBMs is confined to modeling static data.

[21] Proposed an alternative class of time series models with distributed (*i.e.*, componential) hidden state. Mixture models such as HMMs generate each observation from a single category. Distributed state models (*e.g.*, products) generate each object from a set of features that each contains some aspect of that objects description. Linear dynamical systems (LDS) have a continuous, and therefore componential hidden state, but in order to make inference in these models tractable, the relationship between latent and visible variables is constrained to be linear. By carefully choosing the right form of nonlinear observation model it is possible to attain tractable, exact inference, yet retain a rich representational capacity that is linear in the number of components. They leverage the desirable properties of an undirected architecture, the RBM, and extend it to model time series. Their observation or emission distribution is an undirected, bipartite graph. This makes inference simple and efficient.

3. Restricted Boltzmann Machine

The restricted Boltzmann machine (RBM) [6,10] is a deep learning model with a layer of visible units fully connected to a layer of hidden units but no connections within a layer. It assigns a probability to any joint setting of the visible units, v and hidden units, h:

$$p(v, \mathbf{h}) = \frac{\exp(-E(\mathbf{v}, \mathbf{h}))}{Z}$$

Where

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{ij} W_{ij} v_i h_j - \sum_i a_i v_j - \sum_j b_j h_j$$

And Z is a normalization constant called the partition function. It is intractable to compute exactly as it involves a sum over the (exponential) number of possible joint configurations:

$$Z = \sum_{v', h'} E(\mathbf{v}', \mathbf{h}').$$

Marginalizing over the RBM's hidden units and maximizing the likelihood leads to a very simple maximum likelihood weight update rule:

$$\Delta W_{ij} \propto \langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{model}}$$

Where $\langle \cdot \rangle_{\text{data}}$ is an expectation with respect to the data distribution and $\langle \cdot \rangle_{\text{model}}$ is an expectation with respect to the model's equilibrium distribution. Because of the conditional independence of RBMs, we can obtain an unbiased sample of $\langle \cdot \rangle_{\text{data}}$ by clamping the visible units to a vector in the training data set, and sampling the hidden units in parallel according to

$$p(h_j = 1 | \mathbf{v}) = \frac{1}{1 + \exp(-b_j - \sum_i W_{ij} v_i)}.$$

For some applications like motion capture, the observed data is not binary, where we use mean-field logistic units to model the very non-binary data. The stochastic binary units of RBMs can be generalized to any distribution that falls in the exponential family. We use binary logistic hidden units and real-valued Gaussian visible units. The energy function is now

$$E(\mathbf{v}, \mathbf{h}) = \sum_i \frac{(v_i - a_i)^2}{2\sigma_i^2} - \sum_{ij} W_{ij} \frac{v_i}{\sigma_i} h_j - \sum_j b_j h_j.$$

Where a_i is the bias of visible unit *i*, b_j is the bias of hidden unit *j* and σ_i is the standard deviation of the Gaussian noise of visible unit *i*. The symmetric weight, W_{ij} , connects visible unit *i* to hidden unit *j*.

Any setting of the hidden units makes a linear contribution to the mean of each visible unit:

$$p(v_i | \mathbf{h}) = \mathcal{N}\left(a_i + \sigma_i \sum_j W_{ij} h_j, \sigma_i^2\right).$$

Inference simply uses a scale form of:

$$p(h_j = 1 | \mathbf{v}) = \frac{1}{1 + \exp(-b_j - \sum_i W_{ij} \frac{v_i}{\sigma_i})}.$$

The RBM models static frames of data, but does not incorporate any temporal information. We can model temporal dependencies by treating the visible variables in the previous times slices. We add two types of direct connections: autoregressive connections from the past *N* configurations (time steps) of the visible units to the current visible configuration, and connections from the past *M* configurations of the visible units to the current hidden configuration.

We concatenate the data at *t-1,...,t-N* into a vector, $v_{<t}$. So if v_t is of dimension *D*, then $v_{<t}$ is of dimension *N* . *D*.

The effect of the past on each hidden unit can be viewed as a dynamic bias:

$$\hat{b}_{j,t} = b_j + \sum_k B_{kj} v_{k,<t}$$

Which includes the static bias, b_j, and the contribution from the past. This slightly modifies the factorial distribution over hidden units: b_j is replaced with $\hat{b}_{j,t}$ to obtain

$$p(h_j, t = 1|\mathbf{v}_t, \mathbf{v}_{<t}) = \frac{1}{1 + \exp(\hat{b}_{j,t} - \sum_i W_{ij} v_{i,t})}.$$

The past has a similar effect on the visible units. The reconstruction distribution becomes

$$p(v_{i,t}|\mathbf{h}_t, v_{<t}) = \mathcal{N}\left(\hat{a}_{i,t} + \sigma_i \sum_j W_{ij} h_{j,t}, 1\right).$$

Where ^ai;t is also a dynamically changing bias that is an affine function of the past:

$$\hat{a}_{i,t} = a_i + \sum_k A_{ki} v_{k,<t}.$$

The updates for the directed weights are also based on simple pairwise products. The gradients are now summed over all time steps.

4. Gaussian Process Dynamical Model

4.1. Gaussian Process Latent Variable Model

Let $Y \in \mathbb{R}^{N \times D}$ be the observed data where N is the number of observations and D the dimensionality of each data vector. These data are associated with latent variables $X \in \mathbb{R}^{N \times D}$ where, for the purpose of doing dimensionality reduction, $Q \ll D$. The GP-LVM [22] defines a forward (or generative) mapping from the latent space to observation space that is governed by Gaussian processes. If the GPs are taken to be independent across the features then the likelihood function is written as

$$p(Y|X) = \prod_{d=1}^{D} p(\mathbf{y}_d|X), \tag{1}$$

Where y_d represents the dth column of Y and

$$p(\mathbf{y}_d|X) = \mathcal{N}(\mathbf{y}_d|\mathbf{0}, K_{NN} + \beta^{-1} I_N). \tag{2}$$

Here, K_{NN} is the $N \times N$ covariance matrix defined by the covariance (or kernel) function $k(x, x')$. For the purpose of doing automatic model selection of the dimensionality of latent space, this kernel can be chosen to follow the ARD (see [1]) squared exponential form:

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{1}{2} \sum_{q=1}^{Q} \alpha_q (x_q - x'_q)^2\right). \tag{3}$$

Equation 1 can be viewed as the likelihood function of a multiple-output GP regression model where the vectors of different outputs are drawn independently from the same Gaussian process prior which is evaluated at the inputs X. Since X is a latent variable, we can assign it a prior density given by the standard normal density. More precisely, the prior for X is:

$$p(X) = \prod_{n=1}^{N} \mathcal{N}(\mathbf{x}_n|\mathbf{0}, I_Q), \tag{4}$$

Where each x_n is the *n*th row of *X*. The joint probability model for the GP-LVM is

$$p(Y, X) = p(Y|X)p(X). \tag{5}$$

The hyperparameters of the model are the kernel parameters $\theta = (\sigma_f^2, \alpha_1, \ldots, \alpha_Q)$ and the inverse variance parameter β. For the sake of clarity, these parameters are omitted from the conditioning of the distribution[1]. Currently, the primary methodology for training the GP-LVM model is to find the MAP estimate of X [22] whilst jointly maximizing with respect to the hyperparameters. Here, we develop a variational Bayesian approach to marginalization of the latent variables, X, allowing us to optimize the resulting lower bound on the marginal likelihood with respect to the hyperparameters. The lower bound can also be used for model comparison and automatic selection of the latent dimensionality.

4.2. Variational Inference

We wish to compute the marginal likelihood of the data:

$$p(Y) = \int p(Y|X)p(X)dX. \tag{6}$$

However, this quantity is intractable as X appears nonlinearly inside the inverse of the covariance matrix $K_{NN}+\beta^{-1}I_N$. Instead, we seek to apply an approximate variational inference procedure where we introduce a variational distribution *q(X)* to approximate the

[1]A precise notation is to write $p(Y,X|\beta,\theta) = p(Y|X,\beta,\theta)p(X)$.

true posterior distribution *p(X|Y)* over the latent variables. We take the variational distribution to have a factorized Gaussian form over the latent variables,

$$q(X) = \prod_{n=1}^{N} \mathcal{N}(\mathbf{x}_n|\mu_n, S_n), \tag{7}$$

Where the variational parameters are $\{\mu_n, S_n\}_{n=1}^N$ and, for simplicity, S_n is taken to be a diagonal covariance matrix[1].Using this variational distribution we can express a Jensens lower bound on the *log p(Y)* that takes the form:

$$\begin{aligned} F(q) &= \int q(X) \log \frac{p(Y|X)p(X)}{q(X)} dX \\ &= \int q(X) \log p(Y|X) dX - \int q(X) \log \frac{q(X)}{p(X)} dX \\ &= \tilde{F}(q) - \mathrm{KL}(q||p), \end{aligned} \tag{8}$$

Where the second term is the negative KL divergence between the variational posterior distribution *q(X)* and the prior distribution *p(X)* over the latent variables. This term is computed analytically since both distributions are Gaussians. Therefore, the difficult part when estimating the above bound is the first term:

$$\tilde{F}(q) = \sum_{d=1}^{D} \int q(X) \log p(\mathbf{y}_d|X) dX = \sum_{d=1}^{D} \tilde{F}_d(q), \tag{9}$$

By introducing 1. Thus, Thus, the computation of $\tilde{F}(q)$ breaks down to separate computations of each $\tilde{F}_d(q)$, corresponding to the dth output. Notice that the computation of $\tilde{F}_d(q)$ involves an analytically intractable integration. This arises because $\log p(y_d|X)$ contains X in a highly nonlinear manner inside the inverse of the covariance matrix, $K_{NN}+\beta^{-1}I_N$. Our main contribution is a mathematical tool that allows us to compute a closed-form lower bound for $\tilde{F}_d(q)$. As we will see, the key idea is to apply variational sparse GP regression in an augmented probability model.

4.3. Gaussian Process Dynamics

The Gaussian Process Dynamical Model (GPDM) comprises a mapping from a latent space to the data space, and a dynamical model in the latent space. These mappings are typically nonlinear. The GPDM is obtained by marginalizing out the parameters of the two mappings, and optimizing the latent coordinates of training data.

Assume a multivariate times series dataset$\{\mathbf{y}_n, t_n\}_{(}^{N} n = 1)$, where $\mathrm{y}_n \in \mathbb{R}^D$ is a data vector observed at time $t_n \in R_1$. We are especially interested in cases where each y_n is a high dimensional vector and, therefore, we assume that there exists a low dimensional manifold that governs the generation of the data. Specifically, a temporal latent function $\mathrm{x}(t) \in \mathbb{R}^Q$ (with $Q \ll D$), governs an intermediate hidden layer when generating the data, and the *d*th feature from the data vector y_n is then produced from $\mathrm{x}_n = \mathrm{x}(t_n)$ according to

$$y_{nd} = f_d(\mathbf{x}_n) + \epsilon_{nd}, \epsilon_{nd} \sim \mathcal{N}(0, \beta^{-1}), \tag{10}$$

Where f_d(x) is a latent mapping from the low dimensional space to *d*th dimension of the observation space and β is the inverse variance of the white Gaussian noise. We do not want to make strong assumptions about the functional form of the latent functions (x,f). Instead we would like to infer them in a fully Bayesian non-parametric fashion using Gaussian processes [1]. Therefore, we assume that x is a multivariate Gaussian process indexed by time *t* and f is a different multivariate Gaussian process indexed by x, and we write

$$x_q(t) \sim \mathcal{GP}(0, k_x(t_i, t_j)), q = 1, \ldots, Q, \tag{11}$$

$$f_d(\mathbf{t}) \sim \mathcal{GP}(0, k_f(\mathbf{x}_i, \mathbf{x}_j)), q = 1, \ldots, D. \tag{12}$$

[1]This can be extended to non-diagonal within our framework.

The covariance function k_x determines the properties of each temporal latent function $x_q(t)$, k_f determines the properties of the latent mapping f that maps each low dimensional variable x_n to the observed vector y_n.

Similar to Y , the matrix $Y \in \mathbb{R}^{N \times D}$ will denote the mapping latent variables, *i.e.* $f_{nd} = f_d(x_n)$, associated with observations Y from 10. Analogously, $\in \mathbb{R}^{N \times D}$ will store all low dimensional latent variables $x_{nq} = x_q(t_n)$. Given the latent variables we assume independence over the data features, and given time we assume independence over latent dimensions to give

$$p(Y,F,X|t) = p(Y|F)p(F|X)p(X|\mathbf{t}) = \prod_{d=1}^{D} p(\mathbf{y}_d|\mathbf{f}_d|X) \prod_{q=1}^{Q} p(\mathbf{x}_q|\mathbf{t}), \tag{13}$$

Where $\mathbf{t} \in \mathbb{R}^N$ and $p(y_d|f_d)$ is a Gaussian likelihood function term defined from 10. Further, $p(f_d|X)$ is a marginal GP prior such that

$$p(\mathbf{f}_d|X) = \mathcal{N}(\mathbf{f}_d|\mathbf{0}, K_{NN}), \tag{14}$$

Where KNN = kf (X;X) is the covariance matrix defined by the covariance function kf and similarly p(xqjt) is the marginal GP prior associated with the temporal function xq(t),

$$p(\mathbf{x}_q|\mathbf{t}) = (\mathbf{x}_q|\mathbf{0}, K_t), \tag{15}$$

Where Kt = kx(t; t) is the covariance matrix obtained by evaluating the covariance function kx on the observed times t.

4.4. Predictions

This algorithm models the temporal evolution of a dynamical system. It should be capable of generating completely new sequences or reconstructing missing observations from partially observed data. For generating a novel sequence given training data the model requires a time vector t_ as input and computes a density $p(Y_*|Y,t,\ t_*)$.For reconstruction of partially observed data the time-stamp information is additionally accompanied by a partially observed sequence $Y_*^p \in \mathbb{R}^{N_* \times D_p}$ from the whole $Y_* = (Y_*^p, Y_*^p)$, where p and m are set indices indicating the present (*i.e*. observed) and missing dimensions of Y_ respectively.

4.4.1. Predictions Given Only the Test Time Points

To approximate the predictive density, we will need to introduce the underlying latent function values $F_* \in \mathbb{R}^{N_* \times D}$ (the noisy-free version of Y_*) and the latent variables $X_* \in \mathbb{R}^{Q_* \times D}$. We write the predictive density as

$$\begin{aligned} p(Y_*|Y) &= \int p(Y_*, F_*, X_*|Y) dF_* dX_* \\ &= \int p(Y_*|F_*) p(F_*|X_*, Y) p(X_*|Y) dF_* f X_*. \end{aligned} \tag{16}$$

The term $p(F_*|X_*, Y)$ is approximated by the variational distribution

$$\begin{aligned} p(F_*|X_*) &= \int \prod_{d \in D} p(\mathbf{f}_{*,d}|\mathbf{u}_d, X_*) q(\mathbf{u}_d) d\mathbf{u}_d \\ &= \prod_{d \in D} q(\mathbf{f}_{*,d}|X_*), \end{aligned} \tag{17}$$

Where $q(\mathrm{f}_{*,d}|\mathrm{X}_*)$ is a Gaussian that can be computed analytically, since in our variational framework the optimal setting for $q(\mathrm{u}_d)$ is also found to be a Gaussian. As for the term $p(X_*|Y)$ in eq. 16, it is approximated by a Gaussian variational distribution $q(X_*)$,

$$q(X_*) = \prod_{q=1}^{Q} q(\mathbf{x}_{*,q}) = \prod_{q=1}^{Q} \int p(\mathbf{x}_{*,q}|\mathbf{x}_q) q(\mathbf{x}_q) d\mathbf{x}_q$$
$$= \prod_{q=1}^{Q} \langle p(\mathbf{x}_{*,q}|\mathbf{x}_q) \rangle_{q(\mathbf{x}_q)}, \tag{18}$$

Where $p(x_{*,q}|x_q)$ is a Gaussian found from the conditional GP prior (see [1]) and $q(X)$ is also Gaussian. We can, thus, work out analytically the mean and variance for 18, which turn out to be:

$$\mu_{x_{*,q}} = K_{*N} \bar{\mu}_q \tag{19}$$

$$\text{var}(x_{*,q}) = K_{**} - K_{*N}(K_t + \Lambda_q^{-1})^{-1} K_{N*} \tag{20}$$

Where $K_{*N} = k_x(\mathbf{t}_*, \mathbf{t}); K_{*N} = K_{*N}^{\top}$ and $K_{**} = k_x(\mathbf{t}_*, \mathbf{t}_*)$. Notice that these equations have exactly the same form as found in standard GP regression problems. Once we have analytic forms for the posteriors in 16, the predictive density is approximated as

$$p(Y_*|Y) = \int p(Y_*|F_*) q(F_*|X_*) q(X_*) dF_* dX_*$$
$$= \int p(Y_*|F_*) \langle q(F_*|X_*) \rangle_{q(X_*)} dF_*, \tag{21}$$

Which is a non-Gaussian integral that cannot be computed analytically. However, following the same argument as in [1,23], we can calculate analytically its mean and covariance:

$$\mathbb{E}(F_*) = B^{\top} \Psi_1^* \tag{22}$$
$$\text{Cov}(F_*) = B^{\top}(\Psi_2^* - \Psi_1^*(\Psi_1^*)^{\top})B + \Psi_0^* I - \text{Tr}\left[(K_{MM}^{-1} - (K_{MM} + \beta\Psi_2)^{-1})\Psi_2^*\right] I, \tag{23}$$

Where $B = \beta(K_{MM} + \beta\Psi_2)^{-1}, \Psi_0^* = \langle k_f(X_*, X_*) \rangle, \Psi_1^* = \langle K_{M*} \rangle$ and $\Psi_0^* = \langle K_{M*} K_{*M} \rangle$. All expectations are taken w.r.t. $q(X_*)$ and can be calculated analytically, while K_{M*} denotes the cross-covariance matrix between the training inducing inputs $\tilde{X}$ and X_*. The Ψ quantities are calculated analytically. Finally, since Y_* is just a noisy version of F_*, the mean and covariance of 21 is just computed as: $\mathbb{E}(Y_*) = \mathbb{E}(F_*)$ and $Cov(Y_*) = Cov(F_*) + \beta^{-1} I N_*$.

4.4.2. Predictions Given the Test Time Points and Partially Observed Outputs

The expression for the predictive density $p(Y_*^m|Y_*^p, Y)$ is similar to 16,

$$p(Y_*^m|Y_*^p, Y) = \int p(Y_*^m|F_*^m) p(F_*^m|X_*, Y_*^p, Y)$$
$$p(X_*|Y_*^p, Y) dF_*^m f X_*, \tag{24}$$

And is analytically intractable. To obtain an approximation, we firstly need to apply variational inference and approximate $p(X_*|Y_*^P, Y)$ with a Gaussian distribution. This requires the optimization of a new variational lower bound that accounts for the contribution of the partially observed data Y_*^p. This lower bound approximates the true marginal likelihood $p(Y_*^p, Y)$ and has exactly analogous form with the lower bound

computed only on the training data Y . Moreover, the variational optimization requires the definition of the variational distribution $q(X_*,X)$ which needs to be optimized and is fully correlated across X and X_*. After the optimization, the approximation to the true posterior $p(X_*|Y_*^P, Y)$ is given from the marginal $q(X_*)$. A much faster but less accurate method would be to decouple the test from the training latent variables by imposing the factorization $q(X_*,X) = q(X)\ q(X_*)$. This is not used, however, in our current implementation.

5. Experiment

We consider two different types of high dimensional time series, a human motion capture data set consisting of different walks and high resolution video sequences. The experiments are intended to explore the various properties of the model and to evaluate its performance in different tasks (prediction, reconstruction, generation of data).

5.1. Human Motion Caption Data

We followed [24-25] in considering motion capture data of walks and runs taken from subject 35 in the CMU motion capture database. We treated each motion as an independent sequence. The data set was constructed and preprocessed as described in [25]. This results in 2,613 separate 59-dimensional frames split into 31 training sequences with an average length of 84 frames each. The model is jointly trained, as explained in section 2.3, on both walks and runs, *i.e*. the algorithm learns a common latent space for these motions. At test time we investigate the ability of the model to reconstruct test data from a previously unseen sequence given partial information for the test targets. This is tested once by providing only the dimensions which correspond to the body of the subject and once by providing those that correspond to the legs. We compare with results in [25], which used MAP approximations for the dynamical models, and against nearest neighbor. We can also indirectly compare with the binary latent variable model (BLV) of [24] which used a slightly different data preprocessing. We assess the performance using the cumulative error per joint in the scaled space defined in [14] and by the root mean square error in the angle space suggested by [25]. Our model was initialized with nine latent dimensions. We

Errors obtained for the motion capture dataset considering nearest neighbour in the angle space (nn) and in the scaled space(nn sc.), gplvm, blv and vgpds. Cl / cb are the leg and body datasets as preprocessed in [24], l and b the corresponding datasets from [25]. Sc corresponds to the error in the scaled space, as in taylor *et al*. While ra is the error in the angle space. The best error per column is in bold.

Table 1.

Data	CL	CB	L	L	B	B
Error Type	SC	SC	Sc	RA	SC	RA
BLV	11.7	**8.8**	-	-	-	-
NN sc.	22.2	**20.5**	-	-	-	-
GPLVM ($Q = 3$)	-	-	11.4	3.40	16.9	2.49
GPLVM ($Q = 4$)	-	-	9.7	3.34	21.8	2.78
GPLVM ($Q = 5$)	-	-	13.4	4.10	23.4	2.77
NN sc.	-	-	13.5	4.44	20.8	2.62
NN	-	-	14.0	4.11	30.9	3.20
VGPDS (RBF)	-	-	8.19	3.57	**10.73**	**1.90**
VGPDS (Matérn)	-	-	**6.99**	**1.90**	14.22	2.33

The mean squared error per pixel for vgpds and nn for the three datasets (measured only in the missing inputs). The number of latent dimensions selected by our model is in parenthesis.

Table 2.

	Missa	Ocean	Dog
VGPDS	2.52 ($Q = 12$)	9.36 ($Q = 9$)	4.01 ($Q = 9$)
NN	2.63	9.53	4.15

Performed two runs, once using the Mat´ern covariance function for the dynamical prior and once using the RBF. From table 1 we see that the variational Gaussian process dynamical system considerably outperforms the other approaches. The appropriate latent space dimensionality for the data was automatically inferred by our models. The model which employed an RBF covariance to govern the dynamics retained four dimensions, whereas the model that used the Mat´ern kept only three. The other latent dimensions were completely switched off by the ARD parameters. The best performance for the legs and the body reconstruction was achieved by the VGPDS model that used the Mat´ern and the RBF covariance function respectively.

5.2. Modeling Raw High Dimensional Video Sequences

For our second set of experiments we considered video sequences. Such sequences are typically preprocessed before modeling to extract informative features and reduce the dimensionality of the problem. Here we work directly with the raw pixel values to demonstrate the ability of the VGPDS to model data with a vast number of features. This also allows us to directly sample video from the learned model.

Firstly, we used the model to reconstruct partially observed frames from test video sequences. For the first video discussed here we gave as partial information approximately 50% of the pixels while for the other two we gave approximately 40% of the pixels on each frame. The mean squared error per pixel was measured to compare with the k-nearest neighbor (NN) method, for $k \in (1, \ldots, 5)$ (we only present the error achieved for the best choice of k in each case). The datasets considered are the following: firstly, the 'Missa' dataset, a standard benchmark used in image processing. This is a 103,680-dimensional video, showing a woman talking for 150 frames. The data is challenging as there are translations in the pixel space. We also considered an HD video

of dimensionality $9^{\times}10^5$ that shows an artificially created scene of ocean waves as well as a 230,400-dimensional video showing a dog running for 60 frames. The latter is approximately periodic in nature, containing several paces from the dog. For the first two videos we used the Matérn and RBF covariance functions respectively to model the dynamics and interpolated to reconstruct blocks of frames chosen from the whole sequence. For the 'dog' dataset we constructed a compound kernel $kx = k_{x(rbf)} + k_{x(periodic)}$, where the RBF term is employed to capture any divergence from the approximately periodic pattern. We then used our model to reconstruct the last 7 frames extrapolating beyond the original video. As can be seen in table 2, our method outperformed NN in all cases.

6. Performance Evaluation

The key properties of the CRBM are that it permits rich distributed representations to be learned from time series, and that exact inference is simple and efficient. I would like to follow the contrastive divergence (CD) learning rules for CRBMs and showed how CRBMs can be stacked to form conditional deep belief nets. We demonstrated that a single model can generate many different styles of motion.

Perhaps the two greatest limitations of CRBMs (and RBMs in general) are first, evaluating the quality of trained models, and second, the learning algorithm with which they are trained. Though we have explored different methods of model evaluation, such as *N*-step forward prediction and the subjective assessment of synthesized data, the most natural way to evaluate a generative model is to compute the log-likelihood it assigns to a held-out test set. For all but the smallest models, this is impossible to do exactly due to the intractability of computing the partition function. [14] have successfully applied annealed importance sampling (AIS) to RBMs. However, conditioning changes the partition function which implies that we would need to perform AIS for every possible configuration of *N*-frame histories (where *N* is the order of the CRBM) if we wish to evaluate the likelihood assigned by the model to an arbitrary sequence. Fortunately, to evaluate models we are often interested in computing likelihoods for a fixed test set rather than arbitrary sequences. This means that we need only to concern ourselves with conditioning on all possible *N*-frame histories in the test set. If we are evaluating *M* sequences whose maximum length is *T*, we would need to make on the order of $M(T - N)$ complete AIS estimates.

A major criticism of contrastive divergence learning is that by "pulling up" on the energy of individual reconstructed data points, the algorithm fails to visit regions far away from the training data. Consequently, the bulk of the energy surface is left arbitrarily low. One solution is to abandon CD altogether, and pursue other learning methodologies, such as the sparse energy-based methods" or score matching [26]. The alternative is to improve CD *e.g.*, [17].

7. Conclusions

We have introduced a fully Bayesian approach for modeling dynamical systems through probabilistic nonlinear dimensionality reduction. Marginalizing the latent space and reconstructing data using Gaussian processes results in a very generic model for capturing complex, non-linear correlations even in very high dimensional data, without having to perform any data preprocessing or exhaustive search for defining the models structure and parameters. A promising future direction to follow would be to enhance this formulation with domain-specific knowledge encoded, for example, in more sophisticated covariance functions or in the way that data are being preprocessed. Thus, we can obtain application-oriented methods to be used for tasks in areas such as robotics, computer vision and finance.

Acknowledgment

The authors are grateful to the anonymous referees for their valuable comments and suggestions to improve the presentation of this paper.

References

[1] Rasmussen and W. CS, "Gaussian processes for machine learning", **(2006)**.
[2] Z. Ghahramani, "Learning dynamic bayesian networks", in Adaptive processing of sequences and data structures. Springer, **(1998)**, pp. 168–197.
[3] J. Pearl, "Probabilistic reasoning in intelligent systems: networks of plausible inference", Morgan Kaufmann, **(1988)**.
[4] R. M. Neal and G. E. Hinton, "A view of the em algorithm that justifies incremental, sparse, and other variants", learning in graphical models. Springer, **(1998)**, pp. 355–368.
[5] K. P. Murphy, "Dynamic bayesian networks: representation, inference and learning", Ph.D. dissertation, University of California, Berkeley, **(2002)**.
[6] P. Smolensky, "Information processing in dynamical systems: Foundations of harmony theory", **(1986)**.
[7] G. E. Hinton, "Training products of experts by minimizing contrastive divergence", Neural computation, vol. 14, no. 8, **(2002)**, pp.1771–1800.
[8] M. Welling, M. Rosen-Zvi and G. E. Hinton, "Exponential family harmoniums with an application to information retrieval", in Advances in neural information processing systems, **(2004)**, pp. 1481–1488.
[9] P. V. Gehler, A. D. Holub and M. Welling, "The rate adapting poison model for information retrieval and object recognition", in Proceedings of the 23rd international conference on Machine learning. ACM, **(2006)**, pp. 337–344.
[10] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks", Science, vol. 313, no. 5786, **(2006)**, pp. 504–507.
[11] H. Larochelle, D. Erhan, A. Courville, J. Bergstra and Y. Bengio, "An empirical evaluation of deep architectures on problems with many factors of variation", in Proceedings of the 24th international conference on Machine learning. ACM, **(2007)**, pp. 473–480.
[12] R. Salakhutdinov, A. Mnih and G. Hinton, "Restricted boltzmann machines for collaborative filtering", in Proceedings of the 24th international conference on Machine learning. ACM, **(2007)**, pp. 791–798.
[13] Y. Bengio, P. Lamblin, D. Popovici and H. Larochelle, "Greedy layer-wise training of deep networks", Advances in neural information processing systems, vol. 19, **(2007)**, pp. 153.
[14] R. Salakhutdinov and I. Murray, "On the quantitative analysis of deep belief networks", in Proceedings of the 25th international conference on Machine learning. ACM, **(2008)**, pp. 872–879.

[15] I. Sutskever and G. E. Hinton, "Deep, narrow sigmoid belief networks are universal approximators", Neural Computation, vol. 20, no. 11, **(2008)**, pp. 2629–2636.
[16] M. A. Carreira-Perpinan and G. E. Hinton, "On contrastive divergence learning", in Proceedings of the tenth international workshop on artificial intelligence and statistics. Citeseer, **(2005)**, pp. 33–40.
[17] T. Tieleman, "Training restricted boltzmann machines using approximations to the likelihood gradient", in Proceedings of the 25th international conference on Machine learning. ACM, **(2008)**, pp. 1064–1071.
[18] T. Tieleman and G. Hinton, "Using fast weights to improve persistent contrastive divergence", in Proceedings of the 26th Annual International Conference on Machine Learning. ACM, **(2009)**, pp. 1033–1040.
[19] A. D. Brown, "Spiking boltzmann machines", in Advances in Neural Information Processing Systems 12: Proceedings of the 1999 Conference, vol. 12. MIT Press, **(2000)**, pp. 122.
[20] I. Sutskever and G. E. Hinton, "Learning multilevel distributed representations for high-dimensional sequences", in International Conference on Artificial Intelligence and Statistics, **(2007)**, pp. 548–555.
[21] G. W. Taylor, G. E. Hinton and S. T. Roweis, "Two distributed-state models for generating high-dimensional time series", The Journal of Machine Learning Research, vol. 12, **(2011)**, pp. 1025–1068.
[22] L. N. D, "Probabilistic non-linear principal component analysis with gaussian process latent variable models", Journal of Machine Learning Research, vol. 6, **(2005)**, pp. 1783–1816.
[23] A. Girard, C. E. Rasmussen, J. Q. Candela and R. M. Smith, "Gaussian process priors with uncertain inputs? Application to multiple-step ahead time series forecasting", **(2003)**.
[24] H. Taylor and Roweis, "Modeling human motion using binary latent variables", in Advances in Neural Information Processing Systems, vol. 19, **(2007)**.
[25] L. N. D, "Learning for larger datasets with the gaussian process latent variable model", in Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics, **(2007)**, pp. 243–250.
[26] A. H.¨Arinen, "Estimation of non-normalized statistical models by score matching", in Journal of Machine Learning Research, **(2005)**, pp. 695–709.

Authors

Zhou Zhimin, female, professor, was born in Baoding of Hebei Province in 1966. Her research areas include analysis & design of MIS, machine learning algorithms and application of Linux Operating System.

Chen Zhongwen, male, was born in Cang Zhou of Hebei Province in 1967. His research areas include computer network technology and computer application design and development.

