Gaussian Processes Autoencoder for Dimensionality Reduction

Xinwei Jiang[1], Junbin Gao[2], Xia Hong[3], and Zhihua Cai[1]

[1] School of Computer Science,
China University of Geosciences, Wuhan, 430074, China
ysjxw@hotmail.com, zhcai@cug.edu.cn
[2] School of Computing and Mathematics,
Charles Sturt University, Bathurst, NSW 2795, Australia
jbgao@csu.edu.au
[3] School of Systems Engineering,
University of Reading, Reading, RG6 6AY, UK
x.hong@reading.ac.uk

Abstract. Learning low dimensional manifold from highly nonlinear data of high dimensionality has become increasingly important for discovering intrinsic representation that can be utilized for data visualization and preprocessing. The autoencoder is a powerful dimensionality reduction technique based on minimizing reconstruction error, and it has regained popularity because it has been efficiently used for greedy pretraining of deep neural networks. Compared to Neural Network (NN), the superiority of Gaussian Process (GP) has been shown in model inference, optimization and performance. GP has been successfully applied in nonlinear Dimensionality Reduction (DR) algorithms, such as Gaussian Process Latent Variable Model (GPLVM). In this paper we propose the Gaussian Processes Autoencoder Model (GPAM) for dimensionality reduction by extending the classic NN based autoencoder to GP based autoencoder. More interestingly, the novel model can also be viewed as back constrained GPLVM (BC-GPLVM) where the back constraint smooth function is represented by a GP. Experiments verify the performance of the newly proposed model.

Keywords: Dimensionality Reduction, Autoencoder, Gaussian Process, Latent Variable Model, Neural Networks.

1 Introduction

Dimensionality Reduction (DR) aims to find the corresponding low dimensional representation of data in a high-dimensional space without incurring significant information loss and has been widely utilized as one of the most crucial preprocessing steps in data analysis such as applications in computer vision [15]. Theoretically the commonly-faced tasks in data analysis such as regression, classification and clustering can be viewed as DR. For example, in regression, one

V.S. Tseng et al. (Eds.): PAKDD 2014, Part II, LNAI 8444, pp. 62–73, 2014.

tries to estimate a mapping function from an input (normally with high dimensions) to an output space (normally with low dimensions).

Motivated by the ample applications, DR techniques have been extensively studied in the last two decades. Linear DR models such as Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) may be the most well-known DR techniques used in the settings of unsupervised and supervised learning [2]. These methods aim to learn a linear mapping from high dimensional observation to the lower dimension space (also called latent space). However, in practical applications the high dimensional observed data often contain highly nonlinear structures which violates the basic assumption of the linear DR models, hence various non-linear DR models have been developed, such as Multidimensional Scaling (MDS) [11], Isometric Mapping (ISOMAP) [19], Locally Linear Embedding (LLE) [16], Kernel PCA (KPCA) [17], Gaussian Process Latent Variable Model (GPLVM) [9], Relevance Units Latent Variable Model (RULVM) [6], and Thin Plate Spline Latent Variable Model (TPSLVM) [7].

Among the above mentioned nonlinear DR approaches, the Latent Variable Model (LVM) based DR models attract considerable attention due to their intuitive explanation. LVM explicitly models the relationship between the high-dimensional observation space and the low-dimensional latent space, thus it is able to overcome the out-of-sample problems (projecting a new high-dimensional sample into its low-dimensional representation) or pre-image problems (projecting back from the low-dimensional space to the observed data space). The linear Probabilistic PCA (PPCA) [20] and GPLVM [9] may be the most well-known LVM based DR techniques, where the mapping from the low dimensional latent space (latent variables) to the high dimensional observation space (observed variables) is represented by a linear model and a nonlinear Gaussian Process (GP), respectively. Since the nonlinear DR technique GPLVM performs very well in many real-world data sets, this model has become popular in many applications, such as movement modelling and generating [9]. Meanwhile, many GPLVM extensions have been developed to further improve performance. For instance, the Gaussian Process Dynamical Model (GPDM) [22] allows modelling dynamics in the latent space. The back constraint GPLVM (BC-GPLVM) was proposed in [10] to maintain a smooth function from observed data points to the corresponding latent points thus enforcing the close observed data to be close in latent space. Other extensions, such as Bayesian GPLVM, shared GPLVM and supervised GPLVM which further extend the classical GPLVM to unsupervised and supervised settings, can be referred to [4,21,8].

The autoencoder [3] can be regarded as an interesting DR model, although originally it is a neural network (NN) architecture used to determine latent representation for observed data. The idea of autoencoder is to resolve the latent embedding within the hidden layer by training the NN to reproduce the input observed data as its output. Intuitively this model consists of two parts: the encoder which maps the input observed data to a latent representation, and the decoder which reconstructs the input through a map from the latent representation to the observed input (also called output). Basically, the two mappings

in the encoder and decoder are modelled by neural network (NN). Recently, autoencoder has regained popularity because it has been efficiently used for greedy pre-training of deep neural network (DNN) [1].

The relationship between GP and NN was established by Neal [12], who demonstrated that NN could become GP in the limit of infinite hidden units and the model inference may be simpler. With specific covariance function (NN covariance) [14], the back constraint GPLVM (BC-GPLVM) can be seen as autoencoders[18], where the encoder is NN and the decoder is GP. The superiority of GP over NN lies in small-scale model parameters, easy model inference and training [12,14], and in many real-world applications GP outperforms NN. Motivated by the comparison, we propose the Gaussian Processes Autoencoder Model (GPAM), which can be viewed as BC-GPLVM where GP represents the smooth mapping from latent space to observation space, and also as an autoencoder where both the encoder and decoder are GPs. It is expected that the proposed GPAM will outperform typical GPLVM, BC-GPLVM and autoencoder models.

The rest of the paper is organized as follows. In Section 2 we briefly review the GP, GPLVM, and autoencoder models. The proposed Gaussian Processes Autoencoder Model (GPAM) will be introduced in Section 3. Then, real-world data sets are used to verify and evaluate the performance of the newly proposed algorithm in Section 4. Finally, the concluding remarks and comments are given in Section 5.

2 Related Works

In this and following section, we use the following notations: $X = [\boldsymbol{x}_1, ..., \boldsymbol{x}_N]^T$ are observed (inputs) data in a high dimensional space $\mathcal{R}^D$, i.e., $\boldsymbol{x}_n \in \mathcal{R}^D$; $Y = [\boldsymbol{y}_1, ..., \boldsymbol{y}_N]^T$ are observed (outputs or labels) data with each $\boldsymbol{y}_n \in \mathcal{R}^q$; and $Z = [\boldsymbol{z}_1, ..., \boldsymbol{z}_N]^T$ are the so-called latent variables in a low dimensional space $\mathcal{R}^p$ with $p \ll D$ where each $\boldsymbol{z}_n$ is associated with $\boldsymbol{x}_n$. For the sake of convenience, we also consider X as an $N \times D$ matrix, Y an $N \times q$ and Z an $N \times p$ matrix. We call $\mathcal{D} = \{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{n=1}^N$ (or $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^N$ if no labels (outputs) given) the observed dataset. Data items are assumed to be i.i.d. Let $\mathcal{N}(\mu, \Sigma)$ denote the Gaussian distribution with mean μ and covariance Σ.

2.1 Gaussian Process

Given a dataset $\mathcal{D} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \cdots, (\boldsymbol{x}_N, \boldsymbol{y}_N)\}$ as defined above, the classical Gaussian Process Regression (GPR) is concerned with the case when $q = 1$. It aims to estimate the predictive distribution $p(y|\boldsymbol{x}^*)$ for any test data $\boldsymbol{x}^*$. In the classical GPR model, each sample y_n is generated from the corresponding latent functional variable g with independent Gaussian noise

$$y = g(\boldsymbol{x}) + \epsilon$$

where g is drawn from a (zero-mean) GP which only depends on the covariance/kernel function $k(\cdot,\cdot)$ defined on input space and ϵ is the additive Gaussian noise with zero mean and covariance σ^2.

Given a new test observation $\boldsymbol{x}^*$, it is easy to prove that the predictive distribution conditioned on the given observation is

$$\begin{aligned} g^*|\boldsymbol{x}^*, X, Y \sim & \mathcal{N}(K_{\boldsymbol{x}^*X}(K_{XX}+\sigma^2\mathbf{I})^{-1}Y, \\ & K_{\boldsymbol{x}^*\boldsymbol{x}^*} - K_{\boldsymbol{x}^*X}(K_{XX}+\sigma^2\mathbf{I})^{-1}K_{X\boldsymbol{x}^*}) \end{aligned} \tag{2.1}$$

where Ks are the matrices of the covariance/kernel function values at the corresponding points X and/or $\boldsymbol{x}^*$.

2.2 Gaussian Process Latent Variable Model (GPLVM)

Lawrence introduced GPLVM in [9], including the motivation of proposing the GPLVM and the relationship between PPCA and GPLVM. Here we just review GPLVM from the view of GP straightforwardly.

Given a high dimensional dataset $\mathcal{D} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\} \subset \mathcal{R}^D$ without any given labels or output data. We aim to obtain the latent/unknown variables $\boldsymbol{z}_n \in \mathcal{R}^p$ corresponding to each data item $\boldsymbol{x}_n$ $(n = 1, 2, ..., N)$. GPLVM [9] defines a generative mapping from the latent variables $\boldsymbol{z}_n$ to its corresponding observed variables $\boldsymbol{x}_n$ which is governed by a group of GPs $\boldsymbol{x}_n = \boldsymbol{g}(\boldsymbol{z}_n) + \epsilon$ where $\boldsymbol{g} = [g_1, ..., g_D]^T$ is assumed to be a group of D GPs, and ϵ is an independent Gaussian noise with zero mean and covariance $\sigma^2\boldsymbol{I}$, which means the likelihood of the observations is Gaussian

$$P(X|\boldsymbol{g}, Z) = \prod_{n=1}^{N} \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{g}(\boldsymbol{z}_n), \sigma^2\boldsymbol{I}) \tag{2.2}$$

Suppose that each GP g_i $(i = 1, ..., D)$ has the same covariance function $k(\cdot,\cdot)$, then the data likelihood defined by equation (2.2) can be marginalized with respect to the given GP priors over all g_ds, giving rise to the following overall marginalized likelihood of the observations X

$$P(X|Z) = \frac{1}{(2\pi)^{DN/2}|K|^{D/2}} \exp\left(-\frac{1}{2}\mathrm{tr}(K^{-1}XX^T)\right) \tag{2.3}$$

where $K = K_{ZZ} + \sigma^2\boldsymbol{I}$ is the kernel matrix over latent variables Z.

The model learning is implemented by maximizing the above marginalized data likelihood with respect to the latent variables Z and the parameters of the kernel function k.

Although GPLVM provides a smooth mapping from latent space to the observation space, it does not ensure smoothness in the inverse mapping. This can be undesirable because it only guarantees that samples close in latent space will be close in data space, while points close in data space may be not close in latent space. Besides, due to the lack of direct mapping from observation space

to latent space the out-of-sample problem becomes complicated, meaning that the latent representations of testing data must be optimized conditioned on the latent embedding of the training examples [9]. In order to address this issue, the back constraint GPLVM (BC-GPLVM) was proposed in [10]. The idea behind this model is to constrain latent points to be a smooth function of the corresponding data points, which forces points which are close in data space to be close in latent space. The back constraint smooth function can be written by

$$\boldsymbol{z}_{mn} = f_m(\boldsymbol{x}_n, \boldsymbol{\alpha}) \tag{2.4}$$

where $\boldsymbol{\alpha}$ are parameters of the smooth function. Typically, we can use a linear model, a kernel based regression (KBR) model or a multi-layer perception (MLP) model to represent this function. As the function f_m is fully parameterized in terms of a set of new parameters $\boldsymbol{\alpha}$, the learning process becomes an optimization process aiming at maximizing the likelihood (2.3) w.r.t. the latent variables X and parameters $\boldsymbol{\alpha}$.

2.3 Autoencoder

The autoencoder[3] is based on NN, which will be termed NNAM (NN Autoencoder Model) for short throughout the paper. Basically it is a three-layer NN with one hidden layer where the input and output layers are the observation data. Our goal is to find the latent representation over the hidden layer of the model through minimizing reconstruction errors. The autoencoder model can be separated into two parts: an encoder (mapping the input into latent representation) and a decoder (reproducing the input through a map from the latent representation to input).

With the above notations, let's define the encoder as a function $\boldsymbol{z} = \boldsymbol{f}(\boldsymbol{x}, \theta$, and the decoder as a function $\boldsymbol{x} = \boldsymbol{g}(\boldsymbol{z}, \gamma)$. Given a high dimensional dataset $\mathcal{D} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\} \subset \mathcal{R}^D$, we jointly optimize the parameters of the encoder θ and decoder γ by minimizing the least-squares reconstruction cost:

$$\{\theta, \gamma\} = \underset{\{\theta,\gamma\}}{\operatorname{argmax}} \sum_{n=1}^{N} \sum_{d=1}^{D} \left\{x_n^d - g^d(\boldsymbol{f}(\boldsymbol{x}_n, \theta), \gamma)\right\}^2 \tag{2.5}$$

where $g^d(\cdot)$ is the dth output dimension of $\boldsymbol{g}(\cdot)$. When f and g are linear transformations, this model is equivalent to PCA. However, nonlinear projections show a more powerful performance. This function is also called the active function in NN framework. In this paper we use the sigmoidal function $\boldsymbol{f}(\boldsymbol{x}, \theta) = (1 + \exp(-\boldsymbol{x}^T\theta))^{-1}$ as the active function. The model can be optimized by gradient-based algorithms, such as scaled conjugate gradient (SCG).

3 Gaussian Processes Autoencoder Model

Based on the relationship between GP and NN, we introduce the detailed model inference of Gaussian Processes Autoencoder Model (GPAM). The fundamental

idea of this novel model is to use Gaussian Process (GP) to replace Neural Networks (NN) that was originally used in autoencoder.

Given a high dimensional dataset $\mathcal{D} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\} \subset \mathcal{R}^D$ without any labels or output data, where each sample $\boldsymbol{x}_n$ is assumed to be associated with the latent/unknown variables $\boldsymbol{z}_n \in \mathcal{R}^p$ $(n = 1, 2, ..., N)$. Our goal is to find these latent variables which should clearly show the intrinsic structures of the observation data for data visualization or preprocessing.

The idea behind GPAM is to define a mapping from the observation variables $\boldsymbol{x}_n$ to the corresponding latent variables $\boldsymbol{z}_n$ (encoder) and a mapping from the latent variables $\boldsymbol{z}_n$ to the corresponding observation variables $\boldsymbol{x}_n$ (decoder) by using Gaussian Processes Regressions (GPRs) defined as follows

$$\boldsymbol{z} = \boldsymbol{f}(\boldsymbol{x}, \theta) + \epsilon_1; \quad \boldsymbol{x} = \boldsymbol{g}(\boldsymbol{z}, \gamma) + \epsilon_2 \tag{3.1}$$

where $\boldsymbol{f} = [f^1, ..., f^p]^T$ and $\boldsymbol{g} = [g^1, ..., g^D]^T$ are assumed to be two groups of p and D GPs with hyperparameters θ and γ, respectively, and both ϵ_1 and ϵ_2 are the independent Gaussian noises with zero mean and covariance $\sigma^2\boldsymbol{I}$. Thus it is easy to see that the likelihood of the observations is Gaussian,

$$P(Z|\boldsymbol{f}, X, \theta) = \prod_{n=1}^{N} \mathcal{N}(\boldsymbol{z}_n|f(\boldsymbol{x}_n), \sigma_1^2\boldsymbol{I}); \quad P(X|\boldsymbol{g}, Z, \gamma) = \prod_{n=1}^{N} \mathcal{N}(\boldsymbol{x}_n|g(\boldsymbol{z}_n), \sigma_2^2\boldsymbol{I})$$

Let's further assume that both functions $\boldsymbol{f}$ and $\boldsymbol{g}$ are nonlinearly modelled by GPs

$$P(\boldsymbol{f}|X, \theta) = \mathcal{N}(\boldsymbol{f}|0, K_{X,X}); \quad P(\boldsymbol{g}|Z, \gamma) = \mathcal{N}(\boldsymbol{g}|0, K_{Z,Z}) \tag{3.2}$$

By marginalizing over the unknown functions $\boldsymbol{f}$ and $\boldsymbol{g}$, we have

$$P(Z|X, \theta) = \frac{1}{(2\pi)^{pN/2}|K_X|^{p/2}} \exp\left\{-\frac{1}{2}\text{tr}(K_X^{-1}ZZ^T)\right\} \tag{3.3}$$

$$P(X|Z, \gamma) = \frac{1}{(2\pi)^{DN/2}|K_Z|^{D/2}} \exp\left\{-\frac{1}{2}\text{tr}(K_Z^{-1}XX^T)\right\}$$

with $K_X = K_{X,X} + \sigma_1^2\boldsymbol{I}$ and $K_Z = K_{Z,Z} + \sigma_2^2\boldsymbol{I}$ where $K_{X,X}$ and $K_{Z,Z}$ are the covariance matrices defined over the input data X, and the latent variables Z, respectively.

Furthermore, in order to do model inference let's assume that the input X of encoder function $\boldsymbol{f}$ is different from the output X of decoder function $\boldsymbol{g}$, which is rewritten by X_c. Thus the notation of marginal likelihood $P(X|Z, \gamma)$ can be changed to $P(X_c|Z, \gamma)$. Based on the conditional independence property of graphical model the posterior distribution over latent variables Z given observation (X, X_c) can be derived as follows

$$P(Z|X, X_c, \theta, \gamma) = P(Z|X, \theta)P(X_c|Z, \gamma)/P(X_c|X, \gamma, \theta) \tag{3.4}$$

In order to learn the unknown variables (Z, θ, γ), we maximize the log posterior distribution $P(Z|X, X_c, \theta, \gamma)$ (3.4) w.r.t. (Z, θ, γ)

$$\max_{Z,\theta,\gamma} \Big\{ \log P(Z|X,\theta) + \log P(X_c|Z,\gamma) - \log P(X_c|X,\theta,\gamma) \Big\} \tag{3.5}$$

For the sake of convenience, we simply denote the negative log posterior distribution $P(Z|Y, X, \theta, \gamma)$ by

$$\begin{aligned} L = L^r + L^l &= -\log P(Z|X,\theta) - \log P(X_c|Z,\gamma) \\ &= \frac{1}{2}\Big\{ pN\log 2\pi + p\log|K_X| + \mathrm{tr}(K_X^{-1}ZZ^T) \Big\} \\ &\quad + \frac{1}{2}\Big\{ DN\log 2\pi + D\log|K_Z| + \mathrm{tr}(K_Z^{-1}X_cX_c^T) \Big\} \end{aligned} \tag{3.6}$$

where $P(X_c|X, \theta, \gamma)$ has been omitted because it is irrelevant to Z.

The process of model training is equal to simultaneously optimizing a GPR (corresponding to the encoder distribution $P(Z|X, \theta)$) and a GPLVM (corresponding to the decoder distribution $P(X_c|Z, \gamma)$). To apply a gradient based optimization algorithm like SCG algorithm to learn the parameters of the model, we need to find out the gradient of L w.r.t. the latent variables Z, and the kernel parameter (θ, γ).

Firstly for the part of GPR corresponding to $P(Z|X, \theta)$ we can simply obtain the following gradients

$$\frac{\partial L^r}{\partial Z} = K_X^{-1}Z \tag{3.7}$$

As for the parameter θ in kernel K_X, since we consider the output of the mapping $\boldsymbol{z} = \boldsymbol{f}(\boldsymbol{x}, \theta)$ as the known quantity in the GPR model, the optimization process is identical to the procedure of determining parameters for a typical GPR model from training data. Thus we can derive the partial derivative of the hyperparameter θ by chain rule (refer to Chapter 5 in [14]) $\dfrac{\partial L^r}{\partial \theta} = \dfrac{\partial L^r}{\partial K_X}\dfrac{\partial K_X}{\partial \theta}$.

Subsequently for the second part of GPLVM corresponding to $P(X_c|Z, \gamma)$ it is easy to evaluate the gradients of L^l w.r.t. the latent variables Z

$$\frac{\partial L^l}{\partial Z} = \frac{\partial L^l}{\partial K_Z}\frac{\partial K_Z}{\partial Z} \tag{3.8}$$

where the gradients of log likelihood w.r.t. kernel matrix K_Z is evaluated by

$$\frac{\partial L^l}{\partial K_Z} = K_Z^{-1} - K_Z^{-1}YY^TK_Z^{-1}. \tag{3.9}$$

Similarly the gradient of L^l w.r.t. the hyperparameter γ can be calculated by

$$\frac{\partial L^l}{\partial \gamma} = \frac{\partial L^l}{\partial K_Z}\frac{\partial K_Z}{\partial \gamma} \tag{3.10}$$

and the computation of the derivative of the kernel matrix w.r.t. the latent variable Z and hyperparameter depend on a specific kernel function.

By combining equations (3.7) with equation (3.8) and (3.9), it is quite simple to get the complete gradients of L w.r.t. the latent variables Z ($\partial L/\partial Z$). Once we get all the derivative ready, the derivative based algorithms like SCG can be utilized to iteratively optimize these parameters. However, when we perform experiments, we find that the value of L^r (corresponding to the encoder distribution $P(Z|X,\theta)$) is much smaller than that of L^l (corresponding to the decoder distribution $P(X_c|Z,\gamma)$), leading to very little performance improvement compared to GPLVM. Thus we propose a novel algorithm to train the model based on two-stage optimization; this is to say, we try to asynchronously optimize the model consisting of GPR and GPLVM rather than simultaneously learn it. The algorithm is detailed in Algorithm 1.

Algorithm 1. Train and Test GPAM

Input: High dimensional training inputs $X \subset \mathcal{R}^{D\times N}$, pre-fixed latent dimensionality p, number of training iterations T and testing inputs $X^* \subset \mathcal{R}^{D\times M}$.
Output: $s = \{Z, Z^*, \theta, \gamma\}$.

1. Initialize $Z = \text{PPCA}(X, p)$, θ and γ (depending on specific kernel function);
2. For i = 1:T{
3. Optimize $\{Z^t, \gamma\} = \text{argmax}_{Z,\gamma} \log P(X|Z^{t-1}, \gamma)$;
4. Optimize $\{Z^{t+1}, \theta\} = \text{argmax}_{Z,\theta} \log P(Z^t|X, \theta)$;
5. Check converges: break if Error(Z) $= ||Z^{t+1}(:) - Z^t(:)||^2 \leqq \eta\}$; //end loop
6. Compute latent variables $Z^* = K_{\boldsymbol{x}^* X} K_X^{-1} Z$ with learnt hyperparameters θ for testing data X^*;
7. **return** s

To sum up, there are two ways to view the proposed GPAM. Firstly, it can be seen as the generalization of classic NNAM. While GPAM makes use of GPR model to encode and decode the data, NN is utilized to do encoding and decoding in classic NNAM. Based on the superiority of GP over NN, we believe that the proposed GPAM will outperform typical NNAM. Secondly, the proposed GPAM can also be considered as the BC-GPLVM where the back constrain function is modelled by GPR. Compared to classic BC-GPLVM, such as the KBR or MLP based models, the smooth mapping from the observation space to the latent space in the proposed GPAM is modelled by GPR, which results in better performance than typical KBR and MLP based BC-GPLVM.

4 Experiments

In this section, we compare the proposed GPAM with original GPLVM [9], BC-GPLVM [10] and NNAM [3], in two real-world tasks to show the better performance that GPAM provides. In order to assess the performance of these models in visualizing high dimensional data sets, we perform dimensionality reduction

by using a 2D latent space for visualization. Moreover, the nearest neighbour classification error is tested in the low dimensional latent space to objectively evaluate the quality of visualization for training data. After the DR models are learnt, we further use them as feature extraction, followed by a k-Nearest Neighbour (kNN) classifier for testing data. Of course we can use other classifier such as GP Classifier (GPC) rather than a simple kNN to classify the testing data, but the final goal is to reduce the dimensionality of the observation, and the learnt low-dimensional data would be utilized for other proposes, such as data visualization and compression, so the simple kNN classifier is better to evaluate the quality of DR models. By comparing the classification accuracies in low dimensional latent space for testing data, we demonstrate the improvement of the proposed model again. The experimental results verify that the proposed GPAM is an efficient DR model and outperforms GPLVM, BC-GPLVM and NNAM.

For a fair comparison, we ran 500 iterations for all the models, and the covariance used for GPLVM, BC-GPLVM and GPAM was optimally selected from RBF(ARD), POLY(ARD), and MLP(ARD) in Neil D. Lawrence's MATLAB packages `Kern`. The back constraint function of BC-GPLVM is manually picked from KBR and MLP. The code GPLVM/BC-GPLVM and NNAM are based on Neil D. Lawrence's MATLAB packages `FGPLVM`[1], and R. B. Palm's `Deep Learning Toolbox`[2] [13], respectively. Since the learning rate of NNAM needs to be selected manually, we varied it between 0.1 and 10 optimally with sigmoidal active function.

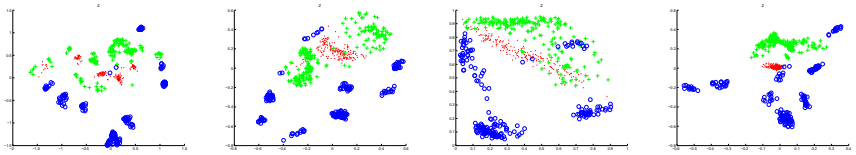
4.1 Oil Flow Data

The oil flow data set [17] consists of 12 dimensional measurements of oil flow within a pipeline. There are 3 phases of flow associated with the data and 1000 samples in the data set. For all four models, we use 600 samples (200 points from each class) to learn the corresponding 2D latent data for the purpose of data visualization, and the remaining 400 samples are the testing data. RBF covariance function is used for GPLVM/BC-GPLVM (MLP back-constraint) and GPAM (RBF covariance for both GPR and GPLVM in the model). As can be seen from Figure 1, the proposed GPAM is superior to GPLVM/BC-GPLVM and NNAM remarkably because the novel model makes the points in the latent space which belong to the same class in the original feature space much closer than the rest of three models.

Furthermore, in order to objectively evaluate the new DR technique we compare the nearest neighbour errors and the classification accuracies based on kNN classifier in the learnt 2D latent space provided by the four models on this data set, respectively. All the four DR models are firstly learnt from training data with the 2D latent space corresponding to the training data where the nearest neighbour errors are evaluated, and then based on the trained four DR models the testing data will be projected to the low dimensional latent/feature space

[1] `http://ml.sheffield.ac.uk/~neil/fgplvm`
[2] `https://github.com/areslp/matlab/tree/master/DeepLearnToolbox-master`

(2D in our experiments) where kNN is performed to compare the testing accuracies ($K = 10$ in kNN). Table I tells us that the proposed GPAM outperforms GPLVM/BC-GPLVM and NNAM in terms of nearest neighbour errors and classification accuracies for training and testing data respectively, which verifies that the novel DR model is better than the other three techniques.



(a) GPLVM **(b)** BC-GPLVM **(c)** NNAM **(d)** GPAM

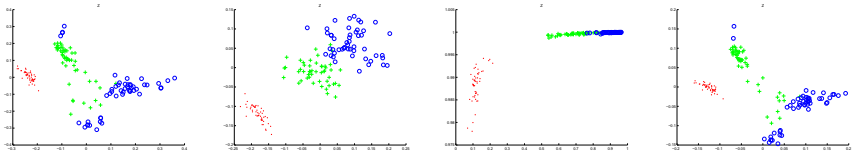
Fig. 1. Oil data set is visualized by GPLVM, BC-GPLVM, NNAM and GPAM

Table 1. The comparisons of nearest neighbor classification errors for training data and kNN classification accuracies for testing data in oil flow data

	GPLVM	BC-GPLVM	NNAM	GPAM
NN Error	8	16	108	5
kNN Accuracy	96.75%	97.00%	90.50%	99.25%

4.2 Iris Data

The Iris data set [5] contains three classes of 50 instances each, where each class refers to a type of iris plant. There are four features for each instance. All 150 data points are utilized to learn the 2D latent space. POLY covariance achieves the best results than the other two covariance functions (RBF and MLP) for GPLVM/BC-GPLVM (MLP back-constraint) and GPAM (POLYARD and POLY covariances for GPR and GPLVM respectively). Figure 2 and Table II show the same conclusion as stated for the oil flow data set. Since there is no more testing data for this data set, the classification comparison for testing is not given.



(a) GPLVM **(b)** BC-GPLVM **(c)** NNAM **(d)** GPAM

Fig. 2. Iris data set is visualized by GPLVM, BC-GPLVM, NNAM and GPAM

Table 2. The comparisons of nearest neighbour classification errors for training data in iris data

	GPLVM	BC-GPLVM	NNAM	GPAM
NN Error	4	5	17	3

As for the model complexity, we have to admit that the training algorithm of GPAM is time-consuming compared to GPLVM, BC-GPLVM and NNAM due to the two-stage optimization. However, in the testing step GPAM is as fast as BC-GPLVM and NNAM without iterative optimization like classical GPLVM.

5 Conclusion

In this paper a novel LVM-based DR technique, termed Gaussian Processes Autoencoder Model (GPAM), has been introduced. It can be seen as the generalization of classic Neural Network Autoencoder Model (NNAM) model by replacing the NNs with GPs, leading to simpler model inference and better performance. Also, we can view the new model as the back constraint GPLVM where the smooth back constraint function is represented by GP, and the model is trained by minimizing the reconstruction error. The experimental results have demonstrated the performance of the newly developed model.

For the future work, inspired by recent works in deep learning [1] we will extend the GP Autoencoder to sparse and denoising GP Autoencoder models, and then we also want to study the deep GP model by stacking the GP Autoencoder.

Acknowledgments. Xinwei Jiang' work is supported by the Fundamental Research Funds for the Central Universities, China University of Geosciences (Wuhan). Junbin Gao and Xia Hong's work is supported by the Australian Research Council (ARC) through the grant DP130100364.

References

1. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H.: Greedy layer-wise training of deep networks. In: Advances in Neural Information Processing Systems (2007)
2. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer (2006)
3. Cottrell, G.W., Munro, P., Zipser, D.: Learning internal representations from grayscale images: An example of extensional programming. In: Conference of the Cognitive Science Society (1987)
4. Ek, C.H.: Shared Gaussian Process Latent Variables Models. Ph.D. thesis, Oxford Brookes University (2009)
5. Frank, A., Asuncion, A.: UCI machine learning repository (2010)
6. Gao, J., Zhang, J., Tien, D.: Relevance units latent variable model and nonlinear dimensionality reduction. IEEE Transactions on Neural Networks 21, 123–135 (2010)

7. Jiang, X., Gao, J., Wang, T., Shi, D.: Tpslvm: A dimensionality reduction algorithm based on thin plate splines. IEEE Transactions on Cybernetics (to appear, 2014)
8. Jiang, X., Gao, J., Wang, T., Zheng, L.: Supervised latent linear gaussian process latent variable model for dimensionality reduction. IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics 42(6), 1620–1632 (2012)
9. Lawrence, N.: Probabilistic non-linear principal component analysis with gaussian process latent variable models. Journal of Machine Learning Research 6, 1783–1816 (2005)
10. Lawrence, N.D., Quinonero-Candela, J.: Local distance preservation in the gp-lvm through back constraints. In: International Conference on Machine Learning (ICML), pp. 513–520. ACM Press (2006)
11. Mardia, K.V., Kent, J.T., Bibby, J.M.: Multivariate analysis. Academic Press, London (1979)
12. Neal, R.: Bayesian learning for neural networks. Lecture Notes in Statistics 118 (1996)
13. Palm, R.B.: Prediction as a candidate for learning deep hierarchical models of data. Master's thesis, Technical University of Denmark (2012)
14. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. The MIT Press (2006)
15. Rosman, G., Bronstein, M.M., Bronstein, A.M., Kimmel, R.: Nonlinear dimensionality reduction by topologically constrained isometric embedding. International Journal of Computer Vision 89, 56–68 (2010)
16. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science 290, 2323–2326 (2000)
17. Scholkopf, B., Smola, A., Muller, K.R.: Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation 10(5), 1299–1319 (1998)
18. Snoek, J., Adams, R.P., Larochelle, H.: Nonparametric guidance of autoencoder representations using label information. Journal of Machine Learning Research 13, 2567–2588 (2012)
19. Tenenbaum, J.B., de Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science 290, 2319–2323 (2000)
20. Tipping, M.E., Bishop, C.M.: Probabilistic principal component analysis. Journal of the Royal Statistical Society, Series B 61, 611–622 (1999)
21. Titsias, M.K., Lawrence, N.D.: Bayesian gaussian process latent variable model. In: International Conference on Artificial Intelligence and Statistics (2010)
22. Wang, J.M., Fleet, D.J., Hertzmann, A.: Gaussian process dynamical models. In: Advances in Neural Information Processing Systems (NIPS), pp. 1441–1448 (2005)