Probabilistic Non-linear Principal Component Analysis with Gaussian Process Latent Variable Models

Neil Lawrence
neil@dcs.shef.ac.uk

13th August 2004

Department of Computer Science, University of Sheffield, Regent Court, 211 Portobello Street, Sheffield, S1 4DP, U.K.

Abstract

Summarising a high dimensional data-set with a low dimensional embedding is a standard approach for exploring its structure. In this paper we provide an overview of some existing techniques for discovering such embeddings. We then introduce a novel probabilistic interpretation of principal component analysis (PCA) that we term dual probabilistic PCA (DPPCA). The DPPCA model has the additional advantage that the linear mappings from the embedded space can easily be non-linearised through Gaussian processes. We refer to this model as a Gaussian process latent variable model (GPLVM).

We develop a practical algorithm for GPLVMs which allow for non-linear mappings from the embedded space giving a non-linear probabilistic version of PCA. We develop the new algorithm to provide a principled approach to handling discrete valued data and missing attributes. We demonstrate the algorithm on a range of real-world and artificially generated data-sets and finally, through analysis of the GPLVM objective function, we relate the algorithm to popular spectral techniques such as kernel PCA and multidimensional scaling.

1 Introduction

Machine learning is often split into three categories: supervised learning, where a data-set is split into inputs and outputs; reinforcement learning, where typically a reward is associated with achieving a set goal, and unsupervised learning where the objective is to understand the structure of a data-set. One approach to unsupervised learning is to represent the data, $\mathbf{Y}$, in some lower dimensional embedded space, $\mathbf{X}$. In a probabilistic model the variables associated with such a space are often known as latent variables. In this paper our focus will be on methods that represent the data in this latent (or embedded, we shall use the terms interchangeably) space.

Our approach is inspired by probabilistic latent variable models. It has roots in previously proposed approaches such as density networks [MacKay, 1995] where a multi-layer perceptron (MLP) is used to provide a mapping from the latent projections, $\mathbf{X}$, to the observed data, $\mathbf{Y}$. A prior distribution is placed over the latent space and the latent space's posterior distribution is approximated by sampling. Density networks made use of the MLP to perform the mapping, Bishop et al. [1996] replaced the MLP with a radial basis function (RBF) network with the aim of decreasing the training time for the model. This model evolved [Bishop et al., 1998] into the generative topographic mapping (GTM) where the latent space was now sampled on a uniform grid, and importance sampling is reinterpreted as the fitting of a mixture model via the expectation-maximisation (EM) algorithm. This allows the points in the latent space to be laid out on a uniform grid[1] (rather than sampled). This grid layout is shared with the self organising

[1] When sampling techniques are used the latent points will be in random positions.

map (SOM) [Kohonen, 1990] and in [Bishop et al., 1997] it was argued that the GTM provides a principled alternative to the self organising map.

The models outlined above are typically designed to embed a data-set in two dimensions, they rely on either importance sampling, or a grid of points in the latent space to achieve this embedding, this causes problems when the dimensionality of the latent space increases. Point representations of the latent space are useful because they allow for non-linear models: each point is easy to propagate through the non-linear mapping to the data-space. These non-linear mappings are designed to address the weaknesses in visualising data-sets that arise when using standard statistical tools that rely on linear mappings, such as principal component analysis (PCA) and factor analysis (FA): with a linear mapping it may not be possible to reflect the structure of the data with a low dimensional embedding.

Principal component analysis seeks a lower dimensional sub-space (typically represented by its orthonormal basis) in which the projected variance of the data is maximised. If a two dimensional sub-space is sought then the projections may be visualised; but it may be necessary to include more latent dimensions to capture the variability (and therefore hopefully, but by no means necessarily the structure) in the data. Principal component analysis also has a latent variable model representation [Tipping and Bishop, 1999] which is strongly related to Factor Analysis (FA) [Bartholomew, 1987, Basilevsky, 1994]. Both are linear-Gaussian latent variable models, but FA allows for a richer noise model than PCA.

Naturally statisticians have not constrained themselves to linear methods when visualising data and in the next section we shall briefly review multidimensional scaling and related techniques that rely on *proximity data*.

1.1 Multidimensional Scaling and Kernel PCA

We have already mentioned several visualisation techniques which rely learning a mapping from a latent space (the embedded space) to the data space. In this section we will briefly review methods that use *proximity data* to obtain a visualisation or embedding. In these methods, rather than observing data directly, information about the data-set is summarised in an $N \times N$ matrix of either similarities or dissimilarities. Examples include distance matrices (a dissimilarity matrix) and kernel matrices (a similarity matrix). Each method we review provides answers to at least one of two questions.

1. How is the proximity matrix compiled?

2. How is the embedding developed from the proximity matrix?

Most of the variants of multidimensional scaling [Mardia et al., 1979] appear to focus on the second question. In classical multidimensional scaling (MDS) an eigendecomposition of the centred similarity matrix[2] is performed. This is sometimes viewed as minimising a particular *stress function* where distances in the visualised space are matched to those in the data space. Attempting to preserve these distances is known as *metric* MDS, in *non-metric* MDS only the ordering of distances is preserved.

There are strong connections between classical MDS and kernel PCA [Schölkopf et al., 1997], some of which are formalised in Williams [2001]. Kernel PCA also provides an answer to the first question — the suggestion is that the proximity data is provided by a positive semi-definite kernel function. The use of this kernel function implies the existence of a non-linear mapping from the data-space to the latent-space (recall that the GTM and density networks perform the non-linear mapping in the opposite direction). The existence of this function is important as it allows data-points which where not in the training set to be mapped to a position in the latent space without recomputing the eigenvalue problem. However, for both kernel PCA and MDS methods it is not obvious how to project back from the latent space to the data-space. Neither is it clear how to

[2]When the data is presented in the form of a distance or dissimilarity matrix a simple conversion may be performed to obtain a similarity matrix.

	Proximity	$\mathbf{X} \to \mathbf{Y}$	$\mathbf{Y} \to \mathbf{X}$	Non-linear	Probabilistic
PCA	I	Y	Y		I
FA		Y	Y		Y
Kernel PCA	Y		Y	Y	
MDS	Y			Y	
Sammon mapping	Y			Y	
Neuroscale	Y		Y	Y	
Spectral clustering	Y			Y	
Density Networks		Y		Y	Y
GTM		Y		Y	Y
GPLVM	I	Y		Y	Y

Table 1: Overview of the relationship between algorithms. A 'Y' indicates the algorithm exhibits that property, an 'I' indicates that there is an interpretation of the algorithm that exhibits the associated property. The characteristics of the algorithm are: *proximity*: is the method based on proximity data? $\mathbf{X} \to \mathbf{Y}$: does the method lead to a mapping from the embedded to the data space? $\mathbf{Y} \to \mathbf{X}$: does the method lead to a mapping from data to embedded space? *Non-linear*: does the method allow for non-linear mappings? *Probabilistic*: does the method have a probabilistic interpretation?

handle missing data[3] as the proximity data matrix cannot normally be computed consistently if a particular attribute is not available.

Sammon mappings [Sammon, 1969] also attempt to match the distances between points in an embedded and the distances in an observed space (therefore they are a form of MDS). They suffer from the same weakness as MDS in that projection of data-points which were not in the original data-set can be computationally demanding, *i.e.* despite their name they do not provide an explicit mapping between the data and latent space. The lack of a mapping issues were addressed by the Neuroscale algorithm of Lowe and Tipping [1996], Tipping [1996] a version of which was also suggested for MDS.

Other recent work of importance which has focussed on forming the proximity matrix includes Isomap [Tenenbaum et al., 2000], where an approximation to geodesic distance is used and spectral clustering (see *e.g.* Shi and Malik, 2000) where the proximity data is derived from a graph.

In Table 1 we have summarised some of the properties of these algorithms/models. we have also included the model that is the subject of this paper, the Gaussian process latent variable model (GPLVM).

In the remainder of this paper we will introduce the GPLVM from the latent variable model perspective, in Section 3 we will cover some of the algorithmic issues that arise with the model. The framework within which our GPLVM is developed makes it straightforward to modify the approach for data for which a Gaussian noise model is not appropriate (such as binary or ordinal), this is discussed in Section 5. Handling of missing data attributes is also straightforward (Section 6). The algorithms characteristics are explored empirically in Section 7. In Section 8 we review the empirical results and elucidate connections to MDS and kernel PCA by deriving a shared objective function, the main body of the paper is then rounded off with a some conclusions.

2 Gaussian Process Latent Variable Models

In this paper we present the Gaussian process latent variable model. As we shall see, the model is strongly related to many of the approaches that we have outlined above. There is a point

[3]Here, by missing data, we mean missing attributes which would normally be used in computing the proximity data matrix. For proximity data methods missing data can also mean elements missing from the proximity matrix, we do not discuss this case.

representation in the latent space (as there was for the GTM and density networks) and we will minimise an objective function that can be related to classical MDS and kernel PCA (see Section 8). Our starting point, however, will be a novel probabilistic interpretation of principal component analysis which we will refer to as dual probabilistic principal component analysis (DPPCA). Dual probabilistic principal component analysis turns out to be a special case of the more general class of models we refer to as GPLVMs.

2.1 Latent Variable Models

Typically we specify a latent variable model relating a set of latent variables, $\mathbf{X} \in \Re^{N\times q}$, to a set of observed variables, $\mathbf{Y} \in \Re^{N\times D}$, through a set of parameters. The model is defined probabilistically, the latent variables are then marginalised and the parameters are found through maximising the likelihood.

Let us consider an alternative approach: rather than marginalising the latent variables and optimising the parameters we marginalise the parameters and optimise the latent variables. We will show how the two approaches can be equivalent: for a particular choice of Gaussian likelihood and prior both approaches lead to a probabilistic formulation of principal component analysis (PCA). In the next section we will review the standard derivation of probabilistic PCA [Tipping and Bishop, 1999], then we will show how an alternative probabilistic formulation may be arrived at (see also Appendix A).

2.2 Probabilistic PCA

Probabilistic PCA (PPCA) is a latent variable model in which the maximum likelihood solution for the parameters is found through solving an eigenvalue problem on the data's covariance matrix [Tipping and Bishop, 1999]. Let's assume that we are given a set of centred D-dimensional data $\mathbf{Y} = [\mathbf{y}_1 \ldots \mathbf{y}_N]^{\mathrm{T}}$. We denote the q-dimensional latent variable associated with each data-point $\mathbf{x}_n$. The likelihood for an individual data-point under the PPCA model is then

$$p(\mathbf{y}_n|\mathbf{W},\beta) = \int p(\mathbf{y}_n|\mathbf{x}_n,\mathbf{W},\beta)\, p(\mathbf{x}_n)\, d\mathbf{x}_n$$

where $p(\mathbf{x}_n)$ is Gaussian distributed[4] with unit covariance,

$$p(\mathbf{x}_n) = N(\mathbf{x}_n|0,\mathbf{I}),$$

and

$$p(\mathbf{y}_n|\mathbf{x}_n,\mathbf{W},\beta) = N\left(\mathbf{y}_n|\mathbf{W}\mathbf{x}_n,\beta^{-1}\mathbf{I}\right) \tag{1}$$

which is dependent on the matrix $\mathbf{W} \in \Re^{D\times q}$ and the scalar β.

The solution for $\mathbf{W}$ can then be found by assuming that $\mathbf{y}_n$ is i.i.d. and maximising the likelihood of the data-set,

$$p(\mathbf{Y}|\mathbf{W},\beta) = \prod_{n=1}^{N} p(\mathbf{y}_n|\mathbf{W},\beta), \tag{2}$$

where

$$p(\mathbf{y}_n|\mathbf{W},\beta) = N\left(\mathbf{y}_n|\mathbf{0},\mathbf{W}\mathbf{W}^{\mathrm{T}}+\beta^{-1}\mathbf{I}\right).$$

The maximum is given when the matrix $\mathbf{W}$ spans the principal sub-space of the data [Tipping and Bishop, 1999].

Marginalising the latent variables and optimising the parameters via maximum likelihood is a standard approach for fitting latent variable models. In this paper we consider the dual approach of marginalising $\mathbf{W}$ and optimising $\mathbf{x}_n$. For a particular choice of prior on $\mathbf{W}$ this probabilistic model also turns out to be equivalent to PCA.

[4] We use the notation $N(\mathbf{z}|\boldsymbol{\mu},\Sigma)$ to denote a Gaussian distribution over $\mathbf{z}$ with mean $\boldsymbol{\mu}$ and covariance Σ.

2.3 Probabilistic PCA through Latent Variable Optimisation

In the Bayesian framework parameters, such as $\mathbf{W}$, are viewed as random variables. The Bayesian methodology requires a suitable choice of prior for $\mathbf{W}$, and then proceeds to treat the parameters as latent variables. A simple choice of prior that is conjugate to (1) would be a spherical Gaussian distribution:

$$p(\mathbf{W}) = \prod_{i=1}^{D} N(\mathbf{w}_i|0, \mathbf{I})$$

where $\mathbf{w}_i$ is the ith row of the matrix $\mathbf{W}$. Unfortunately marginalisation of both $\mathbf{W}$ and $\mathbf{X} = [\mathbf{x}_1 \dots \mathbf{x}_N]^{\mathbf{T}}$ is intractable. If we wish to proceed without turning to approximate methods we are faced with a choice over what to marginalise. The natural choice seems to be to marginalise $\mathbf{X} \in \Re^{N \times q}$ as typically it will be of larger dimension[5] than $\mathbf{W} \in \Re^{D \times q}$. In practice though, it turns out that the two approaches are equivalent.

Marginalisation of $\mathbf{W}$ is straightforward due to our choice of a conjugate prior. The resulting marginalised likelihood takes the form

$$p(\mathbf{Y}|\mathbf{X}, \beta) = \prod_{d=1}^{D} p(\mathbf{y}_{:,d}|\mathbf{X}, \beta), \tag{3}$$

where we use $\mathbf{y}_{:,d}$ to represent the dth column of $\mathbf{Y}$ and

$$p(\mathbf{y}_{:,d}|\mathbf{X}, \beta) = N\left(\mathbf{y}_{:,d}|\mathbf{0}, \mathbf{X}\mathbf{X}^{\mathrm{T}} + \beta^{-1}\mathbf{I}\right).$$

We now look to optimise with respect to the latent variables. As might be expected from the duality of (2) and (3), this optimisation is very similar to that presented in [Tipping and Bishop, 1999]. Our objective function is the the log-likelihood,

$$L = -\frac{DN}{2}\ln 2\pi - \frac{D}{2}\ln|\mathbf{K}| - \frac{1}{2}\mathrm{tr}\left(\mathbf{K}^{-1}\mathbf{Y}\mathbf{Y}^{\mathrm{T}}\right). \tag{4}$$

where

$$\mathbf{K} = \mathbf{X}\mathbf{X}^{\mathrm{T}} + \beta^{-1}\mathbf{I}.$$

The gradients of (4) with respect to $\mathbf{X}$ may be found [Magnus and Neudecker, 1988] as,

$$\frac{\partial L}{\partial \mathbf{X}} = \mathbf{K}^{-1}\mathbf{Y}\mathbf{Y}^{\mathrm{T}}\mathbf{K}^{-1}\mathbf{X} - D\mathbf{K}^{-1}\mathbf{X},$$

some rearrangement of this equation leads to

$$\frac{1}{D}\mathbf{Y}\mathbf{Y}^{\mathrm{T}}\mathbf{K}^{-1}\mathbf{X} = \mathbf{X},$$

which will hold when $\mathbf{X}$ is at a stationary point of (4). In Appendix B we show how the values for $\mathbf{X}$ which maximise the likelihood are given by

$$\mathbf{X} = \mathbf{U}\mathbf{L}\mathbf{V}^{\mathrm{T}}$$

where $\mathbf{U}$ is an $N \times q$ matrix whose columns are the first q eigenvectors of $\mathbf{Y}\mathbf{Y}^{\mathrm{T}}$, $\mathbf{L}$ is a $q \times q$ diagonal matrix whose jth element is $l_j = \left(\lambda_j - \frac{1}{\beta}\right)^{-\frac{1}{2}}$ where λ_j is the eigenvalue associated with the jth eigenvector of $\frac{1}{D}\mathbf{Y}\mathbf{Y}^{\mathrm{T}}$ and $\mathbf{V}$ is an arbitrary $q \times q$ rotation matrix. Here and in what follows we will assume that these eigenvalues are ordered according to magnitude with the largest being placed first. Note that the eigenvalue problem we have developed can easily be shown to be equivalent to that solved in PCA (see Appendix C), indeed the formulation of PCA in this manner is a key step in the development of kernel PCA [Schölkopf et al., 1997] where the matrix

[5] The matrix $\mathbf{W}$ will be of larger than $\mathbf{X}$ unless $D > N$, *i.e.* there are more features than data-points.

of inner products $\mathbf{Y}\mathbf{Y}^{\mathrm{T}}$ is replaced with a kernel (see Tipping [2001] for a concise overview of this derivation). Our probabilistic PCA model shares an underlying structure with Tipping and Bishop [1999] but differs in that where they optimise we marginalise and where they marginalise we optimise.

The marginalised likelihood we are optimising (3) is recognised as the product of D independent Gaussian processes where the (linear) covariance function[6] is given by $\mathbf{K}$. A natural extension of the model is its non-linearisation through the introduction of a non-linear covariance function.

Note that we have not specified any distribution over the latent variables to obtain this solution, it is therefore *not inconsistent* to use this formulation of probabilistic PCA as a pre-processing step for independent component analysis, where the prior distribution over the latent variables is chosen to be non-Gaussian. It then remains to determine the rotation matrix $\mathbf{V}$.

3 Fitting a Non-linear GPLVM

We saw in the previous section how PCA can be interpreted as a Gaussian process relating points in latent space to points in data space. The positions of the points can be determined by maximising the process likelihood with respect to $\mathbf{X}$. We will refer to models of this class as Gaussian process latent variable models (GPLVM). Principal component analysis is a GPLVM where the process prior is based on the $N \times N$ inner product matrix of $\mathbf{X}$. It is natural, therefore, to consider alternative GPLVMs by introducing priors which allow for non-linear processes. The resulting model will then be a probabilistic non-linear version of PCA. There is a wide choice of process priors available, some of which will be reviewed in Section 7.1, to use a particular kernel in the GPLVM we first note that gradients of (4) with respect to the latent points can be found through first taking the gradient with respect to the kernel,

$$\frac{\partial L}{\partial \mathbf{K}} = \mathbf{K}^{-1}\mathbf{Y}\mathbf{Y}^{\mathrm{T}}\mathbf{K}^{-1} - D\mathbf{K}^{-1}, \tag{5}$$

and then combining it with $\frac{\partial \mathbf{K}}{\partial x_{n,j}}$ through the chain rule. As computation of (5) is straightforward and independent of the kernel choice we only require that the gradient of the kernel with respect to the latent points can be computed. These gradients may then be used in combination with (4) in a non-linear optimiser such as scaled conjugate gradients (SCG) (see *e.g.* Nabney [2001]) to obtain a latent variable representation of the data. Furthermore gradients with respect to the parameters of the kernel matrix may be computed and used to jointly optimise $\mathbf{X}$ and the kernel's parameters.

In the previous section we saw for the linear kernel that a closed form solution could be obtained up to an arbitrary rotation matrix. Typically, for non-linear kernels, there will be no such closed form solution and there are likely to be multiple local optima.

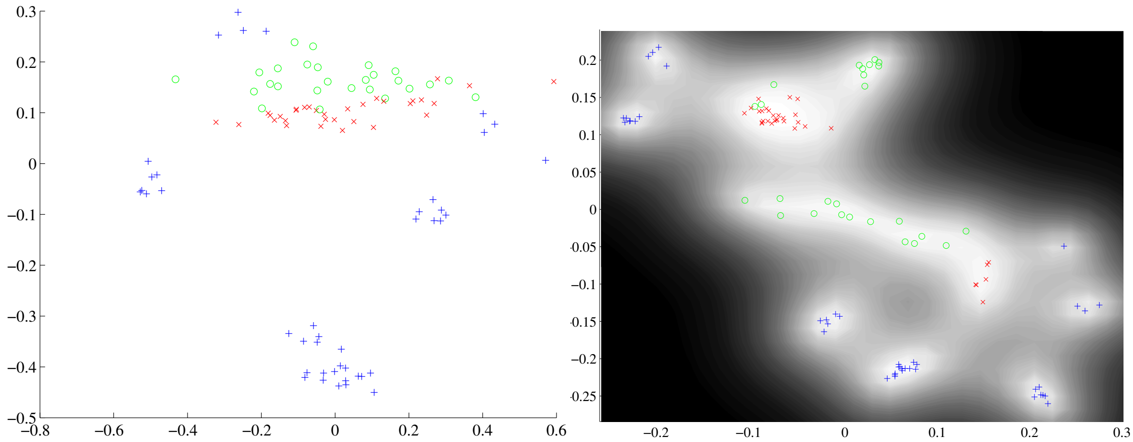
3.1 Illustration of GPLVM via SCG

To illustrate a simple Gaussian process latent variable model we turn to the 'multi-phase oil flow' data [Bishop and James, 1993]. This is a twelve dimensional data-set containing data of three known classes corresponding to the phase of flow in an oil pipeline: stratified, annular and homogeneous. In Bishop et al. [1998], see also Section 7.2.1, this data was used to demonstrate the GTM algorithm. The data-set is artificially generated and therefore is known to lie on a lower dimensional manifold. Here we use a sub-sampled version of the data (containing 100 data-points) to demonstrate the fitting of a GPLVM with a simple RBF kernel.

As we saw in Section 2.3, seeking a lower dimensional embedding with PCA is equivalent to a GPLVM model with a linear kernel,

$$k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j + \beta^{-1}\delta_{ij},$$

[6] In the support vector machine literature the covariance function is known as the kernel. In this paper we shall use the two terms interchangeably.



(a) (b)

Figure 1: Visualisation of the Oil data with (a) PCA (a linear GPLVM) and (b) A GPLVM which uses an RBF kernel. Red crosses, green circles and blue plus signs represent stratified, annular and homogeneous flows respectively. The greyscales in plot (b) indicate the precision with which the manifold was expressed in data-space for that latent point. The optimised parameters of the kernel were $\gamma = 150$, $\alpha = 0.403$ and $\beta = 316$.

where $k(\mathbf{x}_i, \mathbf{x}_j)$ is the element in the ith row and the jth column of the kernel matrix $\mathbf{K}$ and δ_{ij} is the Kronecker delta function. In Figure 1(a) we visualise the first two principal components of the data, this should be compared with the visualisation in Figure 1(b) where a non-linear process model was used. This process model had a kernel of the form

$$k(\mathbf{x}_i, \mathbf{x}_j) = \alpha \exp\left(-\frac{\gamma}{2}(\mathbf{x}_i - \mathbf{x}_j)^{\mathrm{T}}(\mathbf{x}_i - \mathbf{x}_j)\right) + \beta^{-1}\delta_{ij}.$$

This kernel was optimised jointly with respect to the latent positions $\mathbf{X}$ and the parameters α, β and γ. The kernel was initialised using probabilistic PCA to set $\mathbf{X}$ and β and taking $\gamma = \alpha = 1$.

Note that there is a redundancy in the representation between the overall scale of the matrix $\mathbf{X}$ and the value of γ. This redundancy was removed by penalising the log likelihood (4) with half the sum of the squares of each element of $\mathbf{X}$: this implies we were actually seeking a MAP solution with a Gaussian prior for $\mathbf{X}$,

$$p(\mathbf{X}) = \prod_{n=1}^{N} N(\mathbf{x}_n | \mathbf{0}, \mathbf{I}).$$

The likelihood for the RBF kernel was optimised using SCG which converged after 766 iterations (see `http://www.dcs.shef.ac.uk/~neil/gplvm/` for the MATLAB code used).

The gradient based optimisation of the RBF based GPLVM's latent space shows results which are clearly superior (in terms of separation between the different flow phases) to those achieved by the linear PCA model. Additionally the use of a Gaussian process to perform our 'mapping' means that there is uncertainty in the positions of the points in the *data* space. For our formulation of the GPLVM the level of uncertainty is shared across all D dimensions and thus may be visualised in the latent space.

3.1.1 Visualising the Uncertainty

Recall that the likelihood (3) is a product of D separate Gaussian processes. When moving to the log-likelihood in (4) we implicitly assumed that these processes shared the same covariance/kernel function $\mathbf{K}$. This sharing of the covariance function also leads to a shared level of uncertainty. Using different covariance functions for each dimension is also possible (a simple example of this is given by Grochow et al. [2004] with the 'scaled GPLVM'), and may be necessary when the each of the data's attributes have different characteristics, however the more constrained model implemented here allows us to visualise the uncertainty in the latent space, as well has reducing memory and computational requirements, and will be preferred for our empirical studies[7]. In Figure 1 (and subsequently) the uncertainty is visualised by varying the intensity of the background pixels. The lighter the pixel the higher the precision of the mapping.

3.1.2 Computational Complexity

While the quality of the results seem good, a quick analysis of the algorithmic complexity shows that each gradient step requires an inverse of the kernel matrix (see (5)), an $O\left(N^3\right)$ operation, rendering the algorithm impractical for many data-sets of interest. In the next section we will show how a practical algorithm may be developed which circumvents this problem through maximising a sparse approximation to (4).

4 A Practical Algorithm for GPLVMs

So far we have shown that PCA can be viewed probabilistically from two perspectives, the first involves integrating latent variables and the second optimising them. Using the latter perspective we can develop a non-linear probabilistic version of PCA. Unfortunately the optimisation problem we are faced with is then non-linear and high dimensional (Nq interdependent parameters/latent-variables before we consider the parameters of the kernel). In this section we will describe an approximation that relies on a forced 'sparsification' of the model. The resulting computational advantages make visualisation of large numbers of data-points practical. We base our approach on the informative vector machine algorithm [Lawrence et al., 2003]. As we will see in Section 5, this machinery has the added advantage of allowing us to extend our non-linear PCA model to non-Gaussian noise models.

4.0.3 Sparsification

Kernel methods may be sped up through sparsification, *i.e.* representing the data-set by a subset, I, of d points known as the *active set.* The remaining points are denoted by J. We make use of the informative vector machine (IVM) which selects points sequentially according to the reduction in the posterior process's entropy that they induce: implementation details for the IVM algorithm are given in Lawrence et al. [2004].

A consequence of this enforced sparsification is that optimisation of the points in the active set (with $d < N$) proceeds much quicker than the optimisation of the full set of latent variables: the likelihood of the active set is given by

$$p\left(\mathbf{Y}_I\right) = \frac{1}{\left(2\pi\right)^{\frac{D}{2}}\left|\mathbf{K}_{I,I}\right|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\mathrm{tr}\left(\mathbf{K}_{I,I}^{-1}\mathbf{Y}_I\mathbf{Y}_I^{\mathrm{T}}\right)\right), \tag{6}$$

which can be optimised with respect to the kernel's parameters and $\mathbf{X}_I$ with gradient evaluations costing $O\left(d^3\right)$ rather than the prohibitive $O\left(N^3\right)$ which would arise in the full model.

4.0.4 Latent Variable Optimisation

We are interested in visualising all points in the data-set, so while there is a significant speed advantage to selecting an active set, we still need to optimise the *inactive points.* Fortunately, active set selection allows us to optimise each of these points independently as, given a fixed

[7] The two approaches, constaining each data direction to the same kernel and allowing each data dimension to have its own kernel could be considered to be analagous to the difference between probabilistic PCA, where each output data shares a variance, and factor analysis, where each data dimension maintains its own variance.

Algorithm 1 An algorithm for visualisation with a GPLVM.

Require: A size for the active set, d. A number of iterations, T.
Initialise $\mathbf{X}$ through PCA.
for T iterations. **do**
Select an new active set using the IVM algorithm.
Optimise (6) with respect to the parameters of $\mathbf{K}$ (and optionally the latent positions $\mathbf{X}_I$) using scaled conjugate gradients.
Select a new active set.
for each point not in active set j. **do**
Optimise (7) with respect to $\mathbf{x}_j$ using scaled conjugate gradients.
end for
end for

active set, the individual data-points are no longer interdependent. A standard result for Gaussian processes (see *e.g.* Williams [1998]) is that a point from the inactive set, j, can be shown to project into the data space as a Gaussian distribution

$$p(\mathbf{y}_j|\mathbf{x}_j) = N\left(\mathbf{y}_j|\boldsymbol{\mu}_j, \sigma_j^2\mathbf{I}\right) \tag{7}$$

whose mean is

$$\boldsymbol{\mu}_j = \mathbf{Y}^{\mathrm{T}}\mathbf{K}_{I,I}^{-1}\mathbf{k}_{I,j}$$

where $\mathbf{K}_{I,I}$ denotes the kernel matrix developed from the active set and $\mathbf{k}_{I,j}$ made up of rows in I from the jth column of $\mathbf{K}$, and the variance[8] is

$$\sigma_j^2 = k(\mathbf{x}_j, \mathbf{x}_j) - \mathbf{k}_{I,j}^{\mathrm{T}}\mathbf{K}_{I,I}^{-1}\mathbf{k}_{I,j}.$$

Gradients with respect to $\mathbf{x}_j$ do not depend on other data in J, we can therefore independently optimise the likelihood of each $\mathbf{y}_j$ with respect to each $\mathbf{x}_j$. Thus the full set $\mathbf{X}_J$ can be optimised with one pass through the data. The active set is then reselected, and the process is repeated again.

Algorithm 1 summarises the order in which we implemented these steps. We note that for some data-sets (when $N << d$) it may not be necessary to optimise $\mathbf{X}_I$ as the active set is continuously being reselected.

5 Alternative Noise Models

So far we have considered the GPLVM for the particular case where we have Gaussian noise in each dimension with precision β. In this section we consider extensions to this noise model. To this end we firstly reformulate our Gaussian process so that it contains an additional latent variable $\mathbf{F} = [\mathbf{f}_1 \dots \mathbf{f}_N]^{\mathbf{T}}$ between $\mathbf{X}$ and $\mathbf{Y}$.

$$p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\theta}) = \int \prod_{n=1}^{N} p(\mathbf{y}_n|\mathbf{f}_n)\, p(\mathbf{F}|\mathbf{X}, \boldsymbol{\theta})\, d\mathbf{F}. \tag{8}$$

Thus far we have been considering the case where

$$p(\mathbf{y}_n|\mathbf{f}_n) = \prod_{i=1}^{D} N\left(y_{ni}|f_{ni}, \beta^{-1}\right),$$

[8]This fixed variance for all output dimensions is a consequence of sharing the same kernel for each output as was discussed in Section 3.1.1.

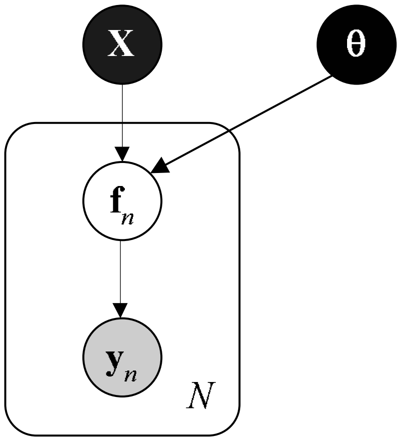


Figure 2: The Gaussian process as a latent variable model.

it is also straightforward to realise the slightly more general case where the precision is dependent on both the data-point and the output dimension,

$$p(\mathbf{y}_n|\mathbf{f}_n) = \prod_{i=1}^{D} N\left(y_{ni}|f_{ni}, \beta_{ni}^{-1}\right). \tag{9}$$

Our approach to different noise models will be to approximate them with a Gaussian noise model of this form (see also Csató [2002], Minka [2001], Lawrence et al. [2004]). The noise models we consider in this paper will be independent across the dimensions,

$$p(\mathbf{y}_n|\mathbf{f}_n) = \prod_{i=1}^{D} p(y_{ni}|f_{ni}),$$

giving approximations of the form

$$p(y_{ni}|f_{ni}) \approx N\left(m_{ni}|f_{ni}, \beta_{ni}^{-1}\right).$$

The approximation to the noise model leads to a Gaussian approximation to the posterior distribution that is built up in a sequential manner, $q(\mathbf{F}) \approx p(\mathbf{F}|\mathbf{X}, \mathbf{Y})$,

$$q(\mathbf{F}) = N\left(\mathbf{f}|\bar{\mathbf{f}}, \Sigma^{-1}\right)$$

where $\mathbf{f}$ is a vector constructed by stacking the columns of $\mathbf{F}$, and $\bar{\mathbf{f}}$ is constructed by stacking the columns of the matrix $\bar{\mathbf{F}} = \left[\bar{\mathbf{f}}_1 \ldots \bar{\mathbf{f}}_N\right]^{\mathrm{T}}$. The covariance matrix has a block diagonal structure[9]

$$\Sigma = \begin{bmatrix} \Sigma_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \Sigma_D \end{bmatrix}.$$

It can be shown [Csató, 2002, Minka, 2001] that the parameters of the approximation are given by

$$\beta_{ni} = \frac{\nu_{ni}}{1 - \nu_{ni}\varsigma_{ni}} \tag{10}$$

[9]For the special case of Gaussian noise with fixed precision β (*i.e.* spherical noise) and shared kernels for each data dimension we find that these blocks are all equal. This leads to computational and memory savings. If the kernels are different or more general noise models are used (such as those described in Appendix D) the blocks will not be equal.

$$m_{ni} = \frac{g_{ni}}{\nu_{ni}} + \bar{f}_{ni} \tag{11}$$

where ς_{ni} is nth diagonal element of Σ_i, $g_{ni} = \frac{\partial}{\partial \bar{f}_{ni}} \log Z_{ni}$ and $\nu_{ni} = g_{ni}^2 - 2\frac{\partial}{\partial \varsigma_{ni}} \log Z_{ni}$ where

$$Z_{ni} = \int p\left(y_{ni}|f_{ni}\right) q\left(\mathbf{F}\right) d\mathbf{F}. \tag{12}$$

To prevent cluttering our notation we have not indicated that the approximation $q(\mathbf{F})$ is typically formed in a sequential manner: its parameters $\bar{\mathbf{F}}$ and Σ change as data-points are incorporated. The approach to approximating the posterior distribution is known as assumed density filtering (ADF).

In Appendix D we outline the use of this approximation for noise models that are appropriate for binary and ordinal categorical data.

6 Missing Values

In many applications attributes are missing for particular data-points. The ability to handle these missing values in a principled way is a desirable characteristic of any algorithm. One motivation behind a probabilistic interpretation of PCA was that the resulting algorithm could handle missing data in a principled manner. This is a characteristic which the Gaussian process latent variable model shares. This should be contrasted with kernel PCA where handling missing values is not so straightforward.

Given the formalism we have described for using different noise models it is straightforward to handle a missing attribute. The corresponding precision parameter from (9), β_{ni}, is simply set to zero.

7 Results

We present a range of empirical evaluations with different data-sets each of which explores a different characteristic of the GPLVM.

So far we have briefly considered two different kernel/covariance functions, before proceeding further we will reconsider these and introduce further kernels which will be used in the experiments that follow.

7.1 Kernels to Be Used

A Gaussian process covariance function can be developed from any positive definite kernel, new kernels can also be formed by adding kernels together. In our experiments we principally make use of three different kernel functions.

7.1.1 Linear Kernel

We have already briefly discussed the linear kernel, it is simply the matrix of inner products,

$$k_{\text{lin}}\left(\mathbf{x}_i, \mathbf{x}_j\right) = \theta_{\text{lin}} \mathbf{x}_i^{\mathrm{T}} \mathbf{x}_j,$$

where we have introduced θ_{lin}, the process variance, which controls the scale of the output functions.

7.1.2 RBF Kernel

We also made use of the popular RBF kernel, it leads to smooth functions that fall away to zero in regions where there is no data.

$$k_{\text{rbf}}\left(\mathbf{x}_i, \mathbf{x}_j\right) = \theta_{\text{rbf}} \exp\left(-\frac{\gamma}{2}\left(\mathbf{x}_i - \mathbf{x}_j\right)^{\mathrm{T}}\left(\mathbf{x}_i - \mathbf{x}_j\right)\right)$$

where γ is the inverse width parameter.

7.1.3 MLP Kernel

The MLP kernel [Williams, 1997] is derived by considering a multi-layer perceptron (MLP) with infinite hidden units,

$$k_{\text{mlp}}\left(\mathbf{x}_i, \mathbf{x}_j\right) = \theta_{\text{mlp}} \sin^{-1}\left(\frac{w\mathbf{x}_i^{\text{T}}\mathbf{x}_j + b}{\sqrt{\left(w\mathbf{x}_i^{\text{T}}\mathbf{x}_i + b + 1\right)\left(w\mathbf{x}_j^{\text{T}}\mathbf{x}_j + b + 1\right)}}\right)$$

where we call w the weight variance and b the bias variance (they have interpretations as the variances of prior distributions in the neural network model). This covariance function also leads to smooth functions, but they have an important characteristic that differentiates them from the RBF kernel: outside regions where the data lies functions will not fall to zero, but tend to remain at a the same value.

7.1.4 The Noise Term

In the experiments in Section 3 we also made use of a 'white noise term'. A white noise process has a kernel of the form

$$k_{\text{white}}\left(\mathbf{x}_i, \mathbf{x}_j\right) = \theta_{\text{white}}\delta_{ij}$$

where δ_{ij} is the Kronecker delta which is zero unless $i = j$ when it takes the value 1. Note that the use of white noise in the kernel is often redundant with some parameters in the noise model, for example with a Gaussian noise model, leaving out the white noise term and setting

$$p\left(y_{in}|f_{in}\right) = N\left(0|f_{in}, \theta_{\text{white}}\right)$$

is equivalent to including the white noise kernel and setting

$$p\left(y_{in}|f_{in}\right) = \lim_{\sigma^2 \to 0} N\left(0|f_{in}, \sigma^2\right).$$

In our experiments we preferred to include the noise term with the kernel as the noise level, θ_{white}, can then be jointly optimised with the kernel parameters and the latent point positions.

7.1.5 Parameter Constraints and Initialisation

All the kernels we have mentioned so far have parameters that need to be constrained be positive. In our experiments this was implemented by reparameterising,

$$\theta = \log\left(1 + \exp\left(\theta'\right)\right).$$

Note that as our transformed parameter $\theta' \to -\infty$ the parameter $\theta \to 0$ and as $\theta' \to \infty$ we see that $\theta \to \theta'$.

We used a consistent initialisation of the parameters for all experiments. This was $\theta_{\text{lin}} = 1$, $\theta_{\text{rbf}} = 1$, $\gamma = 1$, $\theta_{\text{mlp}} = 1$, $w = 10$ and $b = 10$.

7.2 Overview of Experiments

For the experiments that follow we used Algorithm 1 with $T = 15$ iterations and an active set of size $d = 100$. The experiments were run on a 'one-shot' basis, *i.e.* each experiment was only run once with one setting of the random seed and the values of T and d given. If we were producing a visualisation for only one dataset this would leave us open to the criticism that our one-shot result was 'lucky'. However we present several data-sets in what follows and using a one-shot approach in problems with multiple local minima removes the temptation of preferentially selecting 'prettier' results.

The remainder of this section is structured as follows, firstly, in Section 7.2.1 we revisit the oil data first introduced in Section 3.1, but with the revised algorithm which allows us to efficiently visualise all the data-points. We use this data to demonstrate the effect of different prior distributions in the latent space and to form a comparison with the GTM algorithm. For the different algorithms we explore quality of the visualisation in terms of the ease with which the different flow regimes can be separated in the embedded space. In Section 7.3.1 we turn to a much higher (256) dimension data-set of hand-written digits. Again we compare with the GTM and PCA with the GPLVM algorithm by seeing how well the different digits are separated in the latent space. To explore the effects of the two different non-linear kernel functions we considered a greyscale face data-set in Section 7.2.3. All of these first three data-sets made use of the Gaussian noise model, our final experiment with this noise model concerns issues with initialisation. In the data-sets presented above we have no simple 'ground truth' which the algorithm hopes to recover. In Section 7.2.4 we consider the Swiss-roll data Tenenbaum et al. [2000]. For this data the ground truth is known and it turns out that using PCA to initialise the GPLVM it is not recovered, however by initialising using the Isomap algorithm (which is known to give the ground truth) we can recover a probabilistic representation of this data.

In Section 7.3.1 we move on to non-Gaussian data sets. We consider a binary data-set of handwritten 2s. We compare a binary model with a Gaussian model and show that the binary model is more effective at reconstructing twos when pixels are obscured from the model. Finally in Section 7.3.2 we demonstrate the GPLVM's capabilities with a data-set that has ordinal and Gaussian attributes — diagnostic data from horses with cholic.

7.2.1 Oil Flow Data

In this section we return to the twelve dimensional oil data-set that we first introduced in Section 3.1. We now visualise all 1000 of the data-points. For this data-set we are interested in evaluating two different things, the effects of using the different non-linear kernels and the effects of different prior distributions in the latent space[10].

In Figure 3(a) and (b) we present visualisations of the data using the RBF and MLP kernels respectively. For these experiments we used a Gaussian prior over the latent space to remove the redundancy in the scale representation mentioned in Section 3.1. For comparison we also placed a Laplacian prior,

$$p(\mathbf{X}) \propto \prod_{n=1}^{N} \prod_{i=1}^{D} \exp\left(|x_{ni}|\right),$$

(Figure 3(c)) and a 'smoothed uniform' distribution,

$$p(\mathbf{X}) \propto \prod_{n=1}^{N} \prod_{i=1}^{D} \left\{\phi\left(10\left(x_{ni}+1\right)\right) - \phi\left(10\left(x_{ni}-1\right)\right)\right\}.$$

(Figure 3(d)) over the latent space. The Laplacian prior distribution has heavy tails (positive kurtosis) while the smoothed uniform has light tails (negative kurtosis). Using these distributions therefore provides a MAP solution for a non-linear ICA model.

In Figure 3(e) we have recreated the visualisation in [Bishop et al., 1998] which uses the GTM algorithm and finally in (f) we show the full data-set visualised according to the first two principal components. Note the characteristic gridding effect in the GTM's visualisation which arises from the layout of the latent points.

Assessment of the subjective quality of the visualisations is an involved task which is beyond the scope of this paper, however we can obtain a quantitative, objective assessment of the visualisations by using the flow regime labels. These labels were not presented to the algorithms as part of the learning and so can be used as an independent assessment of the visualisation. To this end we used the projected vectors as inputs in an informative vector machine classifier [Lawrence et al., 2004]. We used half of the data to train and half of the data to test, the kernel parameters for the IVM were learnt by type II maximum likelihood. The results are shown in Table 2.

[10] Recall from Section 3 that the latent space is to be 'softly constrained' by a prior distribution to avoid redundancies in the representation between the latent points and the kernel parameters.

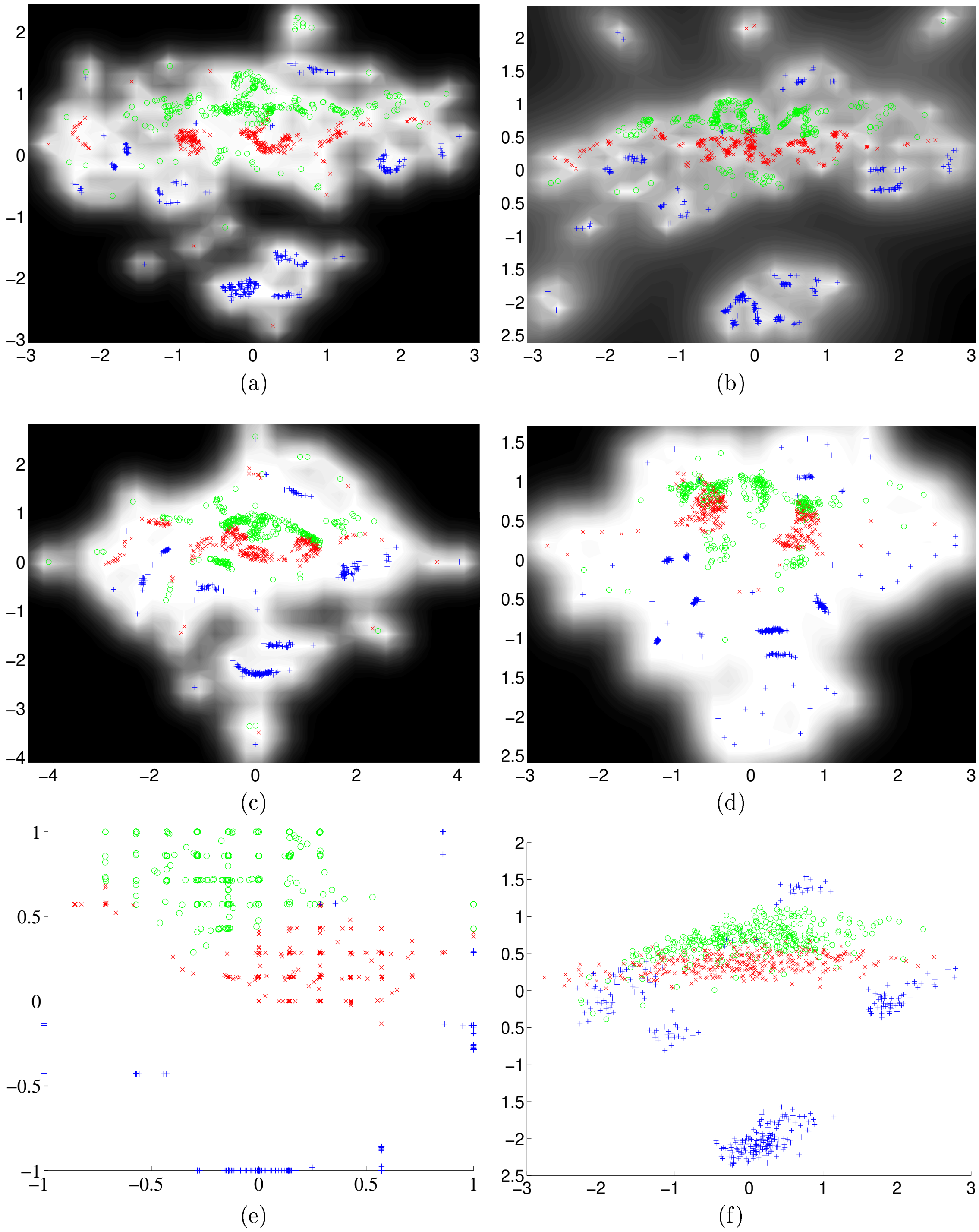


Figure 3: The full oil flow data-set visualised with (a) an RBF based GPLVM, (b) an MLP based GPLVM, (c) an RBF based GPLVM with a Laplace prior on the latent space, (d) an RBF based GPLVM with a smoothed uniform prior on the latent space, (e) GTM with 225 latent points laid out on a 15×15 grid and with 16 RBF nodes and (f) PCA. Notice how the GTM artificially 'discretises' the latent space around the locations of the 225 latent points.

	Model	Error
Kernel	Latent Space Prior	Rate
RBF	Gaussian	4.20%
MLP	Gaussian	3.00%
RBF	Laplacian	4.80%
RBF	Normal-Uniform	11.0%
GTM		2.00%
PCA		13.6%

Table 2: Test error for the oil data for predictions made from the 2-D latent spaces.

	Model	Error
Kernel	Latent Space Prior	Rate
RBF	Gaussian	5.93%
MLP	Gaussian	5.80%
GTM		3.67%
PCA		28.9%

Table 3: Test error for the digit data for predictions made from the 2-D latent spaces.

Under this objective assessment the non-linear methods outperform PCA. For the GPLVM the MLP kernel with Gaussian latent space prior is the best performer, but overall the GTM obtains the best separation between the classes. So despite the rather artificial gridding of the latent space which arises with the GTM it appears to have captured the structure of the data-set better than the GPLVM methods.

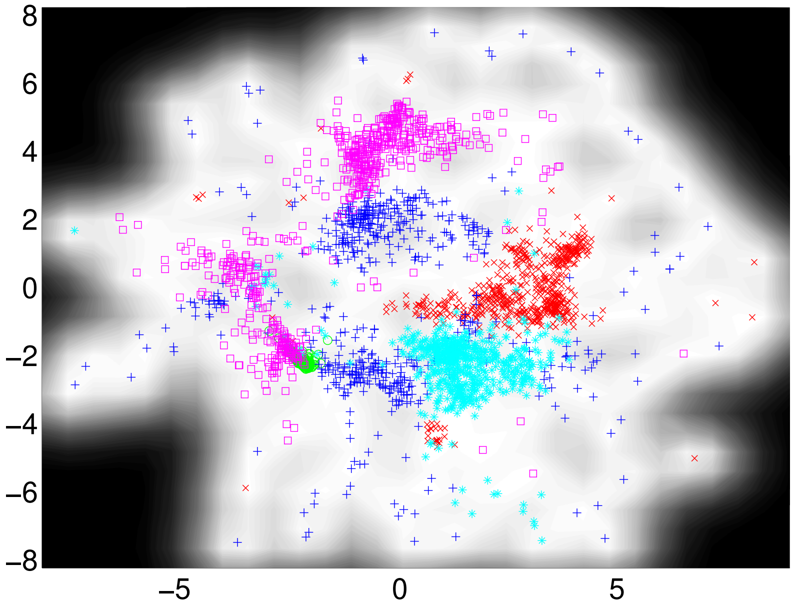
7.2.2 Handwritten Digits

The oil flow data has twelve attributes, twelve dimensions is too many for the structure of the data-set to be visualised without resorting to displaying embedded spaces, but there are many data-sets with much greater dimensionality. One popular data-set for visualisation algorithms has been handwritten digits. We therefore followed Hinton and Roweis [2003] in our 2-D visualisation of a sub-set of 3000 of the digits 0-4 (600 of each digit) from a 16×16 greyscale version of the USPS digit data-set (Figure 4). Again we made use of the RBF and the MLP kernel, but this time only with a Gaussian distribution over the latent space. As well as visualising with the GPLVM we present visualisations from a GTM and PCA (Figure 5).

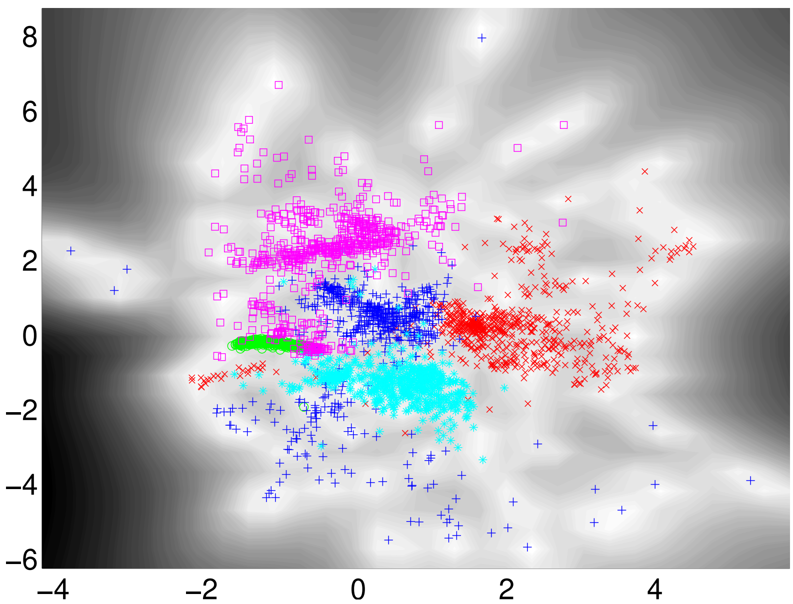
As for the oil data we looked for an objective assessment of the quality of the visualisation by building an IVM classifier that operated on the latent space. The performance benefits associated with the non-linear visualisations are more apparent here than they were for the oil data (Table 3). The GTM once again performs better than the GPLVM under this criterion.

7.2.3 Greyscale Face Images

A key facet of methods which provide a mapping from the latent space to the observed space is the ease with which 'fantasy data' can be generated. We modelled a face data-set [Roweis et al., 2002] consisting of 1965 images from a video sequence digitised at 20×28 using both he RBF and the MLP kernel. As well as demonstrating where the data projects to in the latent space in Figure 6(a) and (b) we show the mean of where the points in the latent space map to in Figure 7(a) and (b). note in particular the different behaviour of the two models in the regions where there is little or no data. The RBF based GPLVM (Figure 7(a)) returns to zero — therefore in these regions the mean face is displayed. The MLP based GPLVM (Figure 7(b)) shows faces which are similar to those in the region where it left the data.

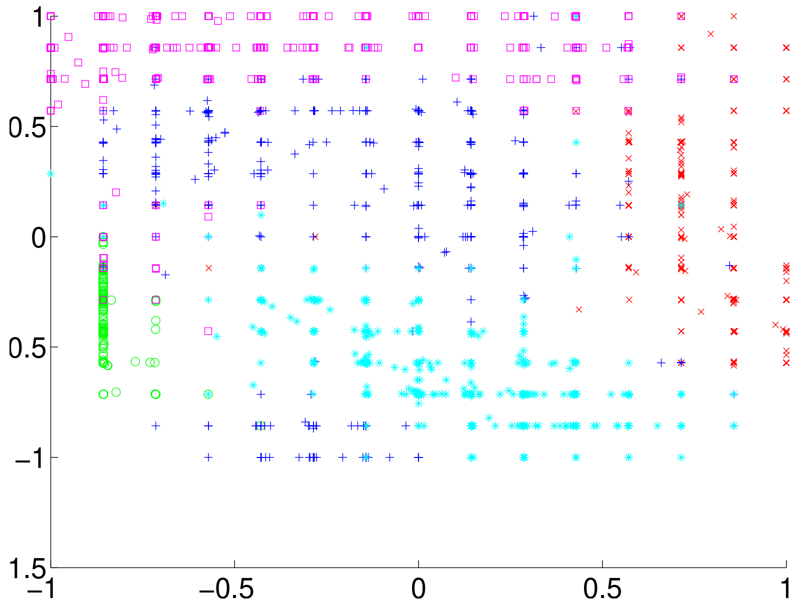


(a)

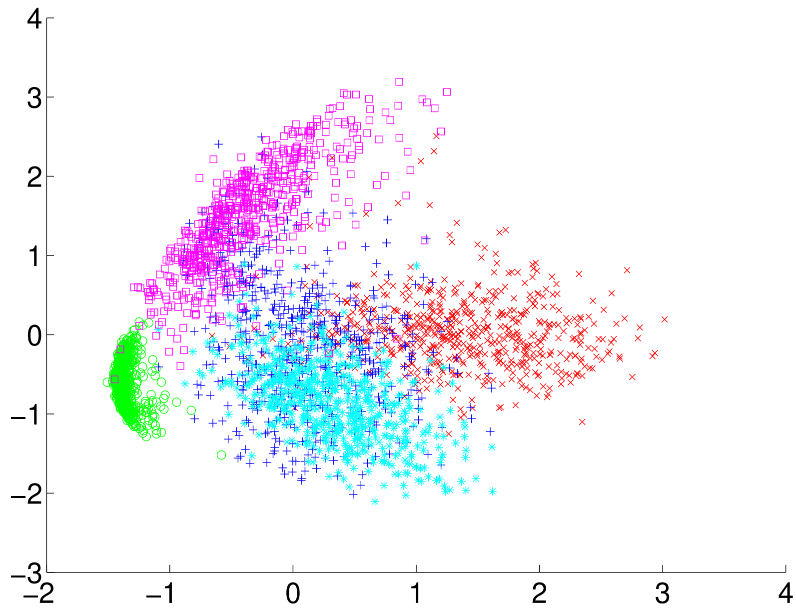


(b)

Figure 4: The digit images visualised in the 2-D latent space. ‘0’ is represented by red crosses; ‘1’: green circles; ‘2’: blue pluses; ‘3’: cyan stars and ‘4’: magenta squares. (a) Visualisation using an RBF kernel. (b) Visualisation using an MLP kernel.

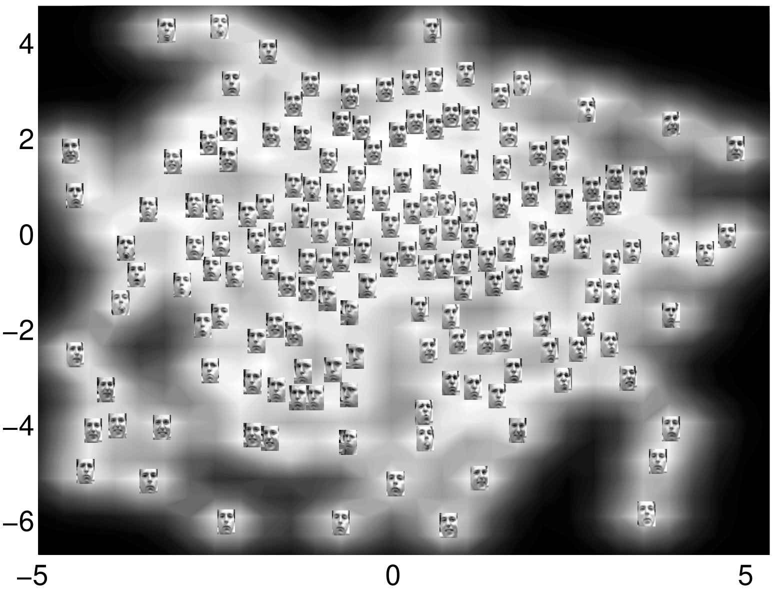


(a)

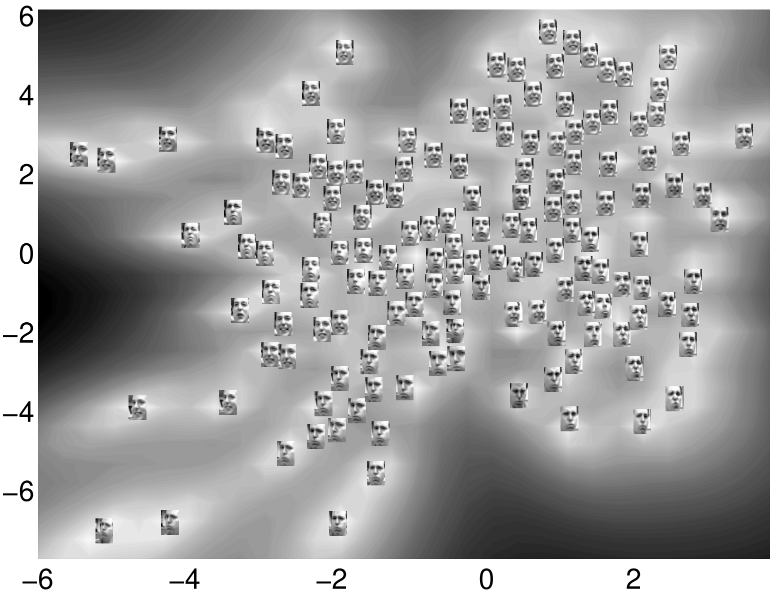


(b)

Figure 5: The digit images visualised in the 2-D latent space. '0' are red crosses, '1' are green circles, '2' are blue pluses, '3' are cyan stars and '4' are magenta squares. (a) Visualisation using the GTM algorithm. (b) Visualisation using PCA.

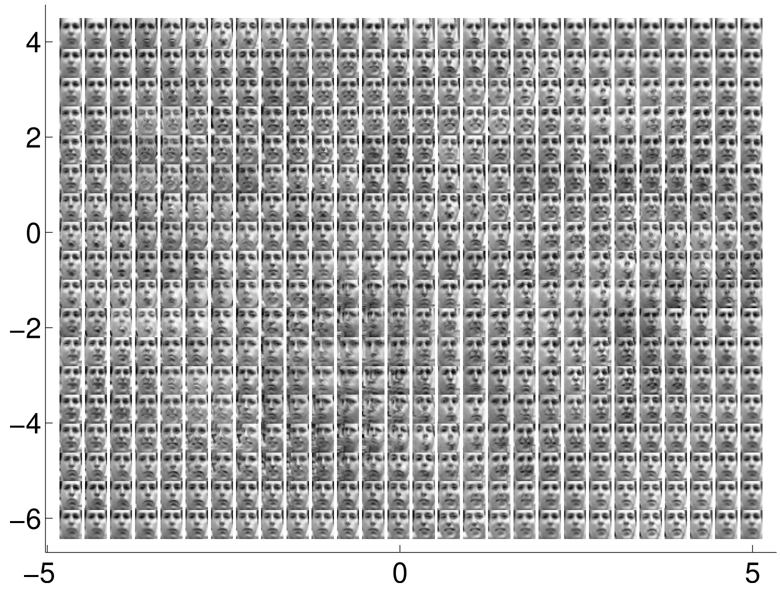


(a)

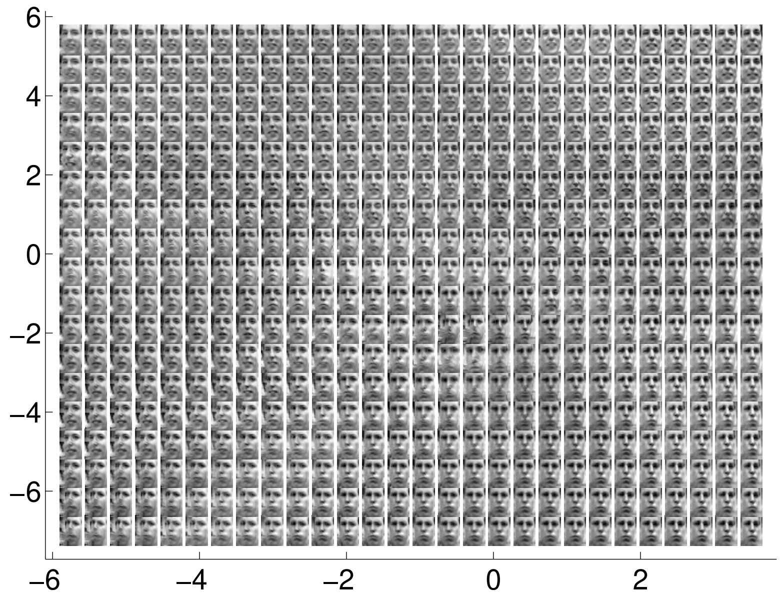


(b)

Figure 6: Visualisation of the face data with (a) the RBF kernel and (b) the MLP kernel.



(a)



(b)

Figure 7: 'Fantasies' from the latent space with (a) the RBF kernel and (b) the MLP kernel.

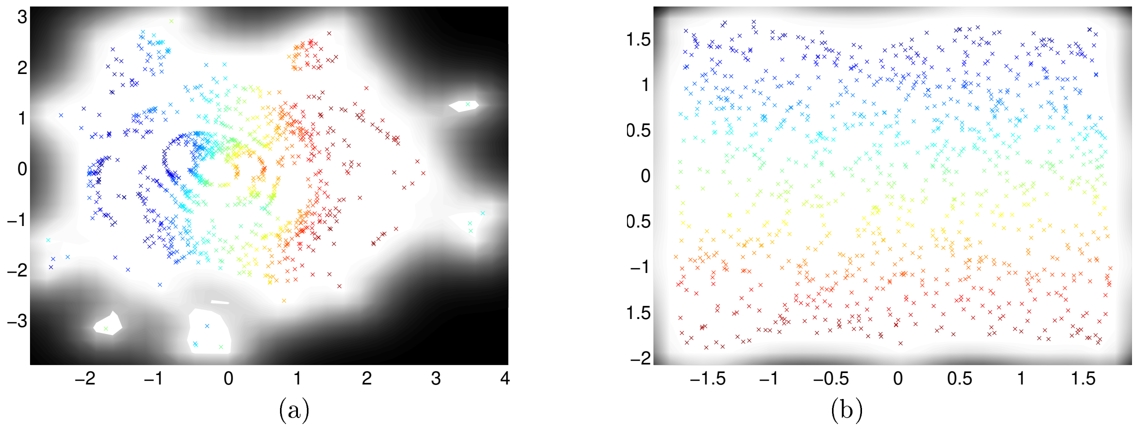


Figure 8: The effect of a poor initialisation. (a) GPLVM initialised using PCA. (b) GPLVM initialised using Isomap.

7.2.4 Initialisation of the Model

In all the experiments so far we used probabilistic PCA to initialise the models, however, there are data-sets for which PCA can provide a poor initialisation, causing the GPLVM to become caught in a local minima. In Figure 8(a) we show a result from modelling the 'swiss roll' data-set [Tenenbaum et al., 2000, data available on line]. For this data the true structure is known — the manifold is a two dimensional square twisted into a spiral along one of its dimensions and living in a three dimensional space. We follow Roweis and Saul [2000] in using colour to show the position along the sheet.

When the GPLVM is initialised with PCA it becomes stuck in an optima that does not recover the true embedded space. However, by initialising using the Isomap algorithm we are able to recover the underlying structure (Figure 8(b)). Isomap extracts the structure of the data and the GPLVM then provides a probabilistic model for the data. In this way we can combine the strengths of the two different approaches — Isomap (and related proximity data based algorithms) provide a unique solution which can recover the structure of the manifold on which the data lies, the GPLVM provides an underlying probabilistic model and an easy way to compute the mapping from the latent to the observed space.

This example also highlights why we should be careful when assessing visualisations. Despite the PCA initialised GPLVM not recovering the 'ground truth' embedded space it highlights that there is some 'circular' structure to the data (which originates from the spiral). These structures are completely missing from the Isomap initialised visualisation.

7.3 Missing Data and Non-Gaussian Noise Models

The examples we have presented so far are for Gaussian noise models. In cases where the data is not continuous a Gaussian noise model is no longer appropriate. Non-Gaussian, *linear*, latent trait models have already been proposed [Bartholomew, 1987, Tipping, 1999], in this section we use the ADF approach described in Section 5 to explore two non-Gaussian data-sets with GPLVM models based around non-Gaussian noise models.

7.3.1 Visualisation of Binary Data

In our first example we follow Tipping [1999] in visualising binary handwritten twos. In Figure 9 we show visualisations from an 8×8 data-set derived from the USPS Cedar CD-ROM. The data contains 700 examples, these examples were taken from the complete data-set of all digits used in Hinton et al. [1995]. For both visualisations an RBF kernel was used in combination with a

Reconstruction method	pixel error rate
GPLVM with Bernoulli noise	23.5%
GPLVM with Gaussian noise	35.9%
Assume pixels are 'not ink'	51.5%

Table 4: Pixel reconstruction error rates.

Gaussian prior over the latent space, however the two visualisations make use of different noise models. In Figure 9(a) a Gaussian noise model was used, in Figure 9(b) the Bernoulli noise model (see Section D.1) was used.

There are certainly differences between the two visualisations in Figure 9, however we again wish to make an objective assessment of the qualities of the embedded spaces. To this end, we turned to a test set containing 400 hundred digits. For each digit in the test set we removed 20% of the pixel values. The digit was then presented to the model and its position in the embedded space optimised. The missing pixels were then filled in by using the mapping from the embedded to the data space. Note that there can be local minima in the embedded space, we therefore optimised the embedded space location ten times with different starting positions and selected that with the largest likelihood. Since we know the original pixel values we can compute the pixel reconstruction error rate. These rates are summarised in Table 4. Results are shown for the Bernoulli noise model, the Gaussian noise model and a baseline approach which is to simply assume that the missing pixels do not contain ink.

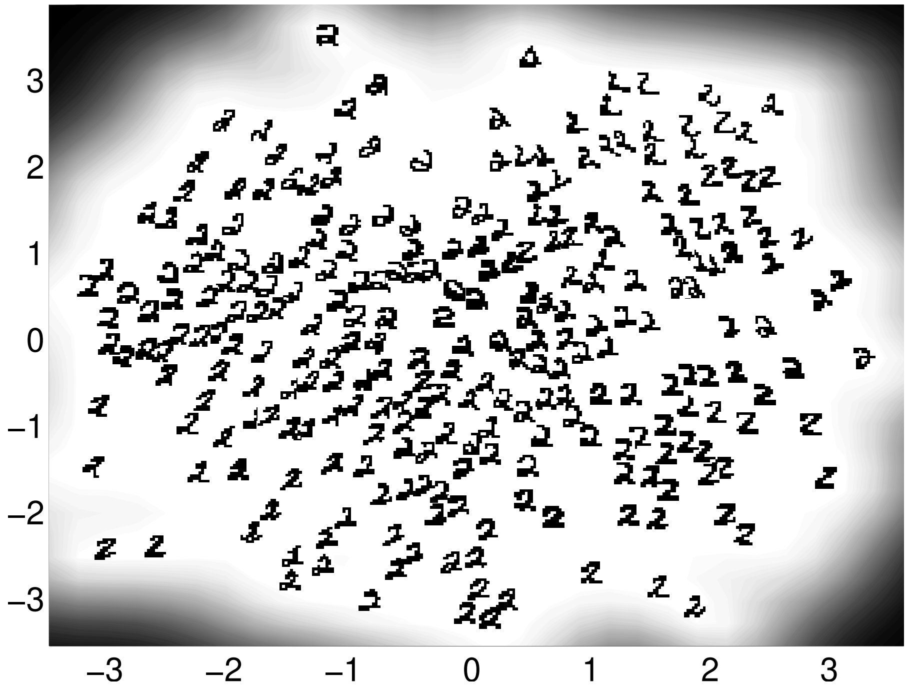
As is expected, both approaches considerably outperform the baseline approach. We also note that using the Bernoulli noise model leads to far better results than the Gaussian noise model. To illustrate the type of mistakes that are made we show some randomly sampled results in Figure 10. For each test digit we present: the original digit, an image showing which pixels are removed and reconstruction using the three methods outlined above. Note that for the GPLVM reconstructions, particularly for the Bernoulli noise model, even when mistakes are made the resulting image often still looks like a handwritten 2.

7.3.2 Ordinal Categories and Missing Data

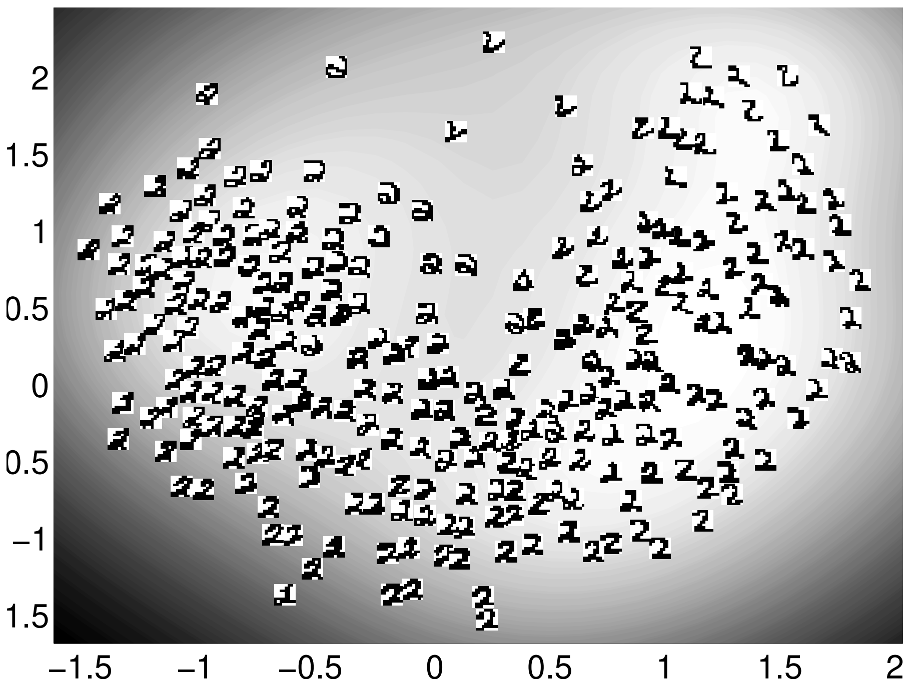
One field where high-dimensional data-sets are common is diagnostic data. A common characteristic of such data-sets is missing values. The variables may also be non-continuous, so the Gaussian noise model may not be appropriate. We reviewed one such data-set from the UCI machine learning repository. The 'horse cholic' data-set contains 366 data-points, each with a maximum of 27 attributes. Of these 27 attributes we used 21 attributes for visualising the data: 7 continuous and 14 either binary, ordinal or nominal. We treated each of the non-continuous attributes as ordinal (binary variables can be viewed as ordinal variables but treating nominal variables as ordinal clearly imposes some constraints on the model).

As is common with diagnostic data, a large proportion (24.8%) of the attributes are missing. The outcome attribute is recorded (death, euthanized or survived). The outcome attribute was one of those withheld from the fitting of the GPLVM but is visualised on the plots. Fitting a Linear GPLVM (equivalent to PCA with ordinal noise models for particular attributes) led to little separation between the different outcomes (Figure 11(a)). However using either the RBF kernel (Figure 11(b)) or the MLP kernel (Figure 11(c)) seems to highlight regions (towards the bottom in both plots) where survival is more likely than natural death. Interestingly the cases which were euthinazed appear to be equally distributed across all regions.

Once again we can obtain an objective assessment of the quality of the visualisations by performing classification in the latent space, indeed this projection to the latent space may be necessary as a pre-processing step due to the large number of missing values. So far when our visualisations have been assessed in this way the GTM algorithm has outperformed the GPLVM. However, one weakness of the GPLVM algorithm is that computational complexity increases exponentially with the dimension of the embedded space. In Figure 12 we show how the error rate



(a)



(b)

Figure 9: The two images visualised in the 2-D latent space. (a) Visualisation using an Gaussian noise model. (b) Visualisation using a Bernoulli noise model.

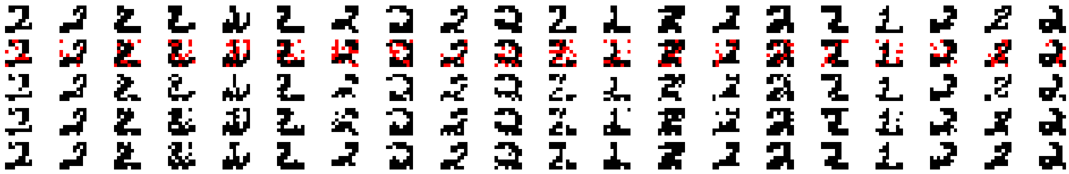


Figure 10: Randomly sampled examples from the test data for the ‘twos’ problem. *Top row*: test images from the data-set of twos, *second row*: pixels removed from the test images are shown in red, *third row*: reconstruction which assumes missing pixels are ‘not ink’, *fourth row*: reconstruction by the Gaussian GPLVM, *fifth row*: reconstruction by the binary noise model.

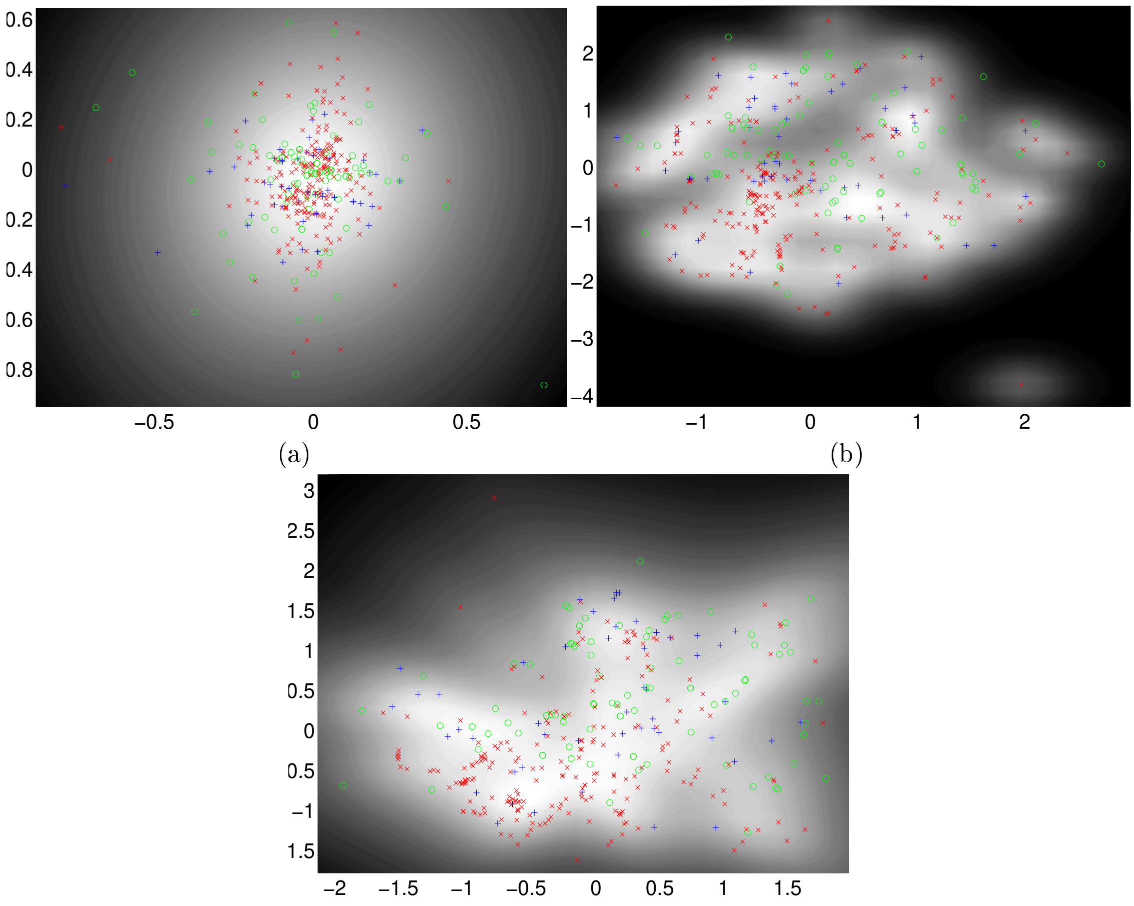


Figure 11: The ‘horse cholic’ data visualised in the 2-D latent space. (a) Visualisation using a linear kernel, (b) RBF kernel, (c) MLP kernel. Horses that died are shown as green circles, those that were euthanized are shown as blue pluses and those that survived are shown as red crosses.

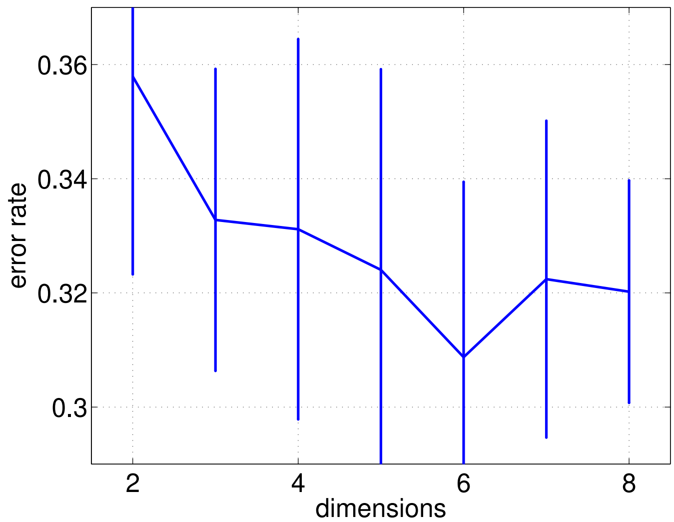


Figure 12: Change in error rate as we increase latent dimensionality for the horse cholic data.The graph shows the mean and error bars at one standard deviation for ten different random partitions of the data into training and test sets.

of classifications performed in the latent space changes as we increase its dimensionality. The plot summarises the mean and standard deviation of ten runs with different 50/50 partitions of training and test data for each run. Note that the error rate seems to be decreasing as the number of dimensions increase to 6 after which point some increase in error rate is observed. This experiment would be prohibitively computationally expensive to perform with the GTM.

This completes our empirical evaluations of the approximate GPLVM algorithm.

8 Discussion

We have presented the Gaussian process latent variable model, which is a non-linear probabilistic extension of PCA. We have reviewed a practical algorithm for fitting the GPLVM [Lawrence, 2004]. The model was extended in a principled manner to take account of missing data and non-continuous data. We assessed the quality of some of the visualisations by performing a classification in the recovered embedded space. Under this criterion, for our examples, the GPLVM was outperformed by the related generative topographic mapping (GTM) algorithm. However the GPLVM does not exhibit the rather artificial 'gridding' effect associated with that algorithm, and it is not as restricted as the GTM in terms of the dimensionality of the embedded space. We saw that a six dimensional space (which would typically be computationally prohibitive for the GTM) was useful in classification of the a diagnostic data-set.

8.1 Proximity Data and the GPLVM

In the introduction we gave a short overview of other visualisation techniques, here we elucidate these connections further. We start from a re-interpretation of the GPLVMs objective function.

8.1.1 A More General Objective Function

We have already mentioned that the eigenvalue problem which provides the maxima of (4) with respect to $\mathbf{X}$ for the linear kernel is exploited in kernel PCA. However the maximum likelihood objective function can then only be interpreted as fitting a probability distribution in a feature space which could be of infinite dimensionality. Rather than deal with the problems encountered with such a model we prefer to take an alternative view of kernel PCA.

Consider that (4) could be written as

$$L = \int N_y\left(\mathbf{z}|0, \mathbf{K}_y\right) \ln N_x\left(\mathbf{z}|0, \mathbf{K}_x\right) d\mathbf{z} \tag{13}$$

where we have introduced a vector, $\mathbf{z}$, of length DN which represents an 'intermediate feature space', $\mathbf{K}_y$ is a positive definite kernel function associated with $\mathbf{Y}$ and we have redefined $\mathbf{K}$ as $\mathbf{K}_x$. The entropy of $N_y\left(\mathbf{z}|0, \mathbf{K}_y\right)$ is constant in $\mathbf{X}$, we therefore may subtract it from L, it is then straightforward to see that maximisng (13) is equivalent to minimising

$$\mathrm{KL}\left(N_y|N_x\right) = -\int N_y\left(\mathbf{z}|0, \mathbf{K}_y\right) \ln \frac{N_x\left(\mathbf{z}|0, \mathbf{K}_x\right)}{N_y\left(\mathbf{z}|0, \mathbf{K}_y\right)} d\mathbf{z}, \tag{14}$$

which is recognised Kullback-Leibler (KL) divergence between the two distributions. With appropriate choice of $\mathbf{K}_y$ and $\mathbf{K}_x$ this is a valid objective function for PCA, kernel PCA and the GPLVM. Note that stochastic neighbor embedding (SNE) [Hinton and Roweis, 2003] also minimises a KL divergence, however for SNE the space $\mathbf{z}$ is discrete.

This KL divergence provides a connection between the proximity data based methods we have outlined in the introduction and the GPLVM. If we have[11] $\mathbf{K}_x = \mathbf{X}\mathbf{X}^{\mathrm{T}} + \sigma_x^2\mathbf{I}$, then as long as $\mathbf{K}_y$ is positive definite the solution for $\mathbf{X}$ is given by

$$\mathbf{X} = \mathbf{U}\mathbf{L}\mathbf{V}^{\mathrm{T}}$$

where $\mathbf{U}$ are the first q eigenvectors of $\mathbf{K}_y$, $\mathbf{L}$ is a $q \times q$ matrix related to the eigenvalues of $\mathbf{K}_y$ and $\mathbf{V}$ is an arbitrary rotation matrix (see Appendix A). In other words (14) is optimised using an eigendecomposition of $\mathbf{K}_y$. If $\mathbf{K}_y$ is derived from a centred distance matrix then this eigendecomposition is known as classical MDS (or principal co-ordinate analysis), if $\mathbf{K}_y$ is simply a positive definite similarity matrix then this eigendecomposition is kernel PCA. In other words (14) provides an objective function for classical MDS and kernel PCA.

Contrast this with the GPLVM: for this case the similarity matrix is $\mathbf{K}_y = \lim_{\sigma_y \to 0}\left[\mathbf{Y}\mathbf{Y}^{\mathrm{T}} + \sigma_y^2\mathbf{I}\right]$, but $\mathbf{K}_x$ is any positive definite kernel matrix. The methods overlap when both $\mathbf{K}_x$ and $\mathbf{K}_y$ are based on inner product matrices (as outlined in the dual probabilistic PCA algorithm in Section 2.3). This provides the interpretation for PCA as a proximity data based method (see also an alternative objective function presented in Mardia et al. [1979] and discussion in Tipping [1996]).

8.2 Implications of KL Divergence as an Objective

Viewing the algorithms within the framework of the KL divergence objective function provokes further discussion, in this section we shall discuss some of the implications.

The KL divergence requires that both $\mathbf{K}_x$ and $\mathbf{K}_y$ are positive definite. In practice we will often encounter positive semi-definite kernel matrices which can be handled by adding constant diagonal terms, the objective function can then be considered in the limit as these diagonals go to zero. However in many MDS applications the similarity matrix may not even be positive semi-definite. Providing a principled explanation for this within the framework of the KL divergence may provide insight into how to deal with the resulting negative eigenvalues that arise in classical MDS.

[11] Here the constant diagonal term ensures that $\mathbf{K}_x$ is positive definite. It has no effect on the resulting eigenvectors $\mathbf{U}$ but gives a constant offset to the eigenvalues that form $\mathbf{L}$. The solution is also valid in the limit as $\sigma_x^2 \to 0$. Similar comments also will apply when this diagonal is used with $\mathbf{K}_y$.

8.2.1 Similarity Matrices with Negative Eigenvalues

The equation (14) represents a KL divergence between two Gaussian distributions for which the similarity matrices are interpreted as covariance matrices or covariance functions[12]. Minimising this KL divergence leads to a matching between the correlations that are implied by the two matrices: explicitly the subspaces spanned by the two similarities should overlap as much as possible.

Minimising the KL divergence encourages an overlap between the sub-spaces, but what if we wish to discourage an overlap between a particular sub-space to obtain a particular embedding? A good example of this is seeking an embedded space for a set of images where we might wish to ignore the effects of lighting.

A simple modification to the objective function in (14) is to minimise with respect to one KL divergence while maximising with respect to another, so our objective becomes

$$\begin{aligned}\delta\mathrm{KL} &= -(1-\alpha)\int N_y\left(\mathbf{z}|0,\mathbf{K}_y^{+}\right)\ln\frac{N_x\left(\mathbf{z}|0,\mathbf{K}_x\right)}{N_y\left(\mathbf{z}|0,\mathbf{K}_y^{+}\right)}d\mathbf{z}\\ &\quad+\alpha\int N_y\left(\mathbf{z}|0,\mathbf{K}_y^{-}\right)\ln\frac{N_x\left(\mathbf{z}|0,\mathbf{K}_x\right)}{N_y\left(\mathbf{z}|0,\mathbf{K}_y^{-}\right)}d\mathbf{z},\end{aligned} \tag{15}$$

where $\mathbf{K}_y^{+}$ encapsulates the subspace with which we want to overlap and $\mathbf{K}_x^{-}$ spans the subspace we wish to avoid, both of these matrices are positive definite. The scalar α provides a weighting between the KL divergences. Dropping terms which do not depend on $\mathbf{K}_x$ we obtain

$$\delta\mathrm{KL} = \frac{N}{2}\ln 2\pi + \frac{1}{2}\ln\left|\mathbf{K}_x\right| + \frac{1}{2}\mathrm{tr}\left(\mathbf{K}_x^{-1}\left((1-\alpha)\mathbf{K}_y^{+}-\alpha\mathbf{K}_y^{-}\right)\right). \tag{16}$$

which is equivalent to our original objective function with $\mathbf{K}_y = (1-\alpha)\mathbf{K}_y^{+} - \alpha\mathbf{K}_y^{-}$. For the positive definite case, it is clear that the largest eigenvalues of $\mathbf{K}_y$ should be retained to minimise the KL divergence (see Appendix B.1.1 and Tipping and Bishop [1999]) however when some of the eigenvalues are negative, it is harder to make such globally applicable statements (see Appendix B.1.2).

8.2.2 Reversing the Kullback-Leibler Divergence

The Kullback-Leibler divergence is asymmetric so it is natural to consider the effect of reversing the role of the distributions and taking expectations under the distribution governed by $\mathbf{K}_x$ rather than that governed by $\mathbf{K}_y$. For this special case, the reversed KL divergence is very similar to the original, only all matrices $\mathbf{K}_x$ and $\mathbf{K}_y$ are now replaced with their inverses. So the new objective function is

$$L = \frac{N}{2}\ln 2\pi + \frac{1}{2}\ln\left|\mathbf{K}_x\right| + \frac{1}{2}\mathrm{tr}\left(\mathbf{K}_x\mathbf{K}_y^{-1}\right),$$

The minima can again be found through an eigenvalue problem, but now the retained eigenvalues from $\mathbf{K}_y$ are the smallest, rather than the largest. In this respect the model is strongly related to minor component analysis.

8.2.3 Twin Kernel Models

All the models we have discussed so far use the inner product matrix for at least one of the two matrices $\mathbf{K}_x$ and $\mathbf{K}_y$. For kernel PCA and classical MDS the matrix $\mathbf{K}_x$ is based on an inner product matrix. For the GPLVM the matrix $\mathbf{K}_y$ is an inner product matrix. However our GPLVM algorithm does not require that $\mathbf{K}_y$ be an inner product matrix to operate, so it may be applied in the case where $\mathbf{K}_y$ is simply a positive definite similarity matrix. Note that the resulting visualisation algorithm would no longer provide a mapping from the embedded space to the data space or vice-versa.

[12] Under the process perspective these would be covariance functions, but the similarity matrix may be derived in such a way that it can only be interpreted as a covariance matrix — in this case the model would be a Gaussian random field rather than a Gaussian process.

8.3 Computing the Likelihood of Test Data

One key advantage of the GPLVM is that it is probabilistic. However computing the likelihood of a previously unseen (test) data-point is not straightforward. In the traditional probabilistic PCA model when a new data-point, $\mathbf{y}_*$, is presented its likelihood under the marginal distribution,

$$p(\mathbf{y}_*|\mathbf{W},\beta) = N\left(\mathbf{y}_*|\mathbf{0},\mathbf{W}\mathbf{W}^{\mathrm{T}} + \beta^{-1}\mathbf{I}\right),$$

is easily computed. Therefore the likelihood of a previously unseen test data-set is straightforward to compute. In the GPLVM the likelihood takes a different form. The new datum has an associated latent variable, $\mathbf{x}_*$. The likelihood of $\mathbf{y}_*$, for the special case where variances over each output direction are constant, is given by

$$p(\mathbf{y}_*|\mathbf{X},\mathbf{x}_*) = N\left(\mathbf{y}_*|\boldsymbol{\mu},\sigma^2\right),$$

where

$$\boldsymbol{\mu} = \mathbf{Y}^{\mathrm{T}}\mathbf{K}_{I,I}^{-1}\mathbf{k}_{I,*}$$

$\mathbf{k}_{I,*}$ being a column vector developed from computing the elements of the kernel matrix between the active set and the new point $\mathbf{x}_*$ and the variance is

$$\sigma^2 = k(\mathbf{x}_*,\mathbf{x}_*) - \mathbf{k}_{I,*}^{\mathrm{T}}\mathbf{K}_{I,I}^{-1}\mathbf{k}_{I,*}.$$

To determine the likelihood of the new point, we first find the MAP solution for this new latent point. Since the posterior over $\mathbf{X}$ can be multi-modal with respect to $\mathbf{x}_*$ this solution will not necessarily be unique. The likelihood is then approximated by computing the probability of the observed data under the distribution given by projecting the MAP solution for $\mathbf{x}_*$ back into data space. In an ideal world, we would integrate out the latent space to determine this marginal likelihood, and the problem with multiple modes would not arise. However given that the integration will not generally be tractable, this MAP approach provides a simple alternative.

8.4 GPLVM in Inverse Kinematics

Work by Grochow et al. [2004] has shown that the GPLVM can be used in achieving realistic models of human motion. Grochow et al. show that styles of human movement can be learnt by a GPLVM using motion capture data. These styles can then be combined with kinematic constraints to quickly animate characters. This is known as inverse kinematics. The probabilistic interpretation of the GPLVM was of particular importance in this work as the likelihood approximation described in the previous section is optimised in real time to simulate the character's movement. Further details on this work are available at `http://grail.cs.washington.edu/projects/styleik/`.

9 Conclusions

We have presented a new class of models for probabilistic modelling and visualisation of high dimensional data. We provided theoretical groundings for the approach by proving that principal component analysis is a special case. On real world data-sets we showed that visualisations provided by the model placed related data-points close to each other. We demonstrated that the model performed well in traditionally difficult domains that involve missing and discrete data.

Our approach is related to density networks and the generative topographic mapping in that these models all provide a non-linear mapping from the embedded space to the observed space. In all these cases the embedded space is treated as a latent variable model and problems of propagating distributions through the non-linear mapping are avoided by using point representations of the data within the latent space. A novel characteristic of the GPLVM is that we can visualise the uncertainty with which the manifold is defined in the data space.

We showed there is a general objective function based on the Kullback-Leibler divergence that connects the Gaussian process latent variable model with proximity data based methods such as kernel PCA and multidimensional scaling. Further analysis of this objective function is expected to provide deeper insights into the behaviour of these algorithms.

Acknowledgements

We thank Aaron Hertzmann and his collaborators for ongoing access to their work on style based inverse kinematics Amos Storkey for pointing out that the GPLVM fails on the spiral data with a PCA initialisation and Michael Tipping for discussions on visualisation techniques.

References

D. J. Bartholomew. *Latent Variable Models and Factor Analysis.* Charles Griffin & Co. Ltd, London, 1987.

A. Basilevsky. *Statistical Factor Analysis and Related Methods.* Wiley, New York, 1994.

S. Becker, S. Thrun, and K. Obermayer, editors. *Advances in Neural Information Processing Systems*, volume 15, Cambridge, MA, 2003. MIT Press.

C. M. Bishop and G. D. James. Analysis of multiphase flows using dual-energy gamma densitometry and neural networks. *Nuclear Instruments and Methods in Physics Research*, A327:580–593, 1993.

C. M. Bishop, M. Svensén, and C. K. I. Williams. A fast EM algorithm for latent variable density models. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 465–471. MIT Press, 1996.

C. M. Bishop, M. Svensén, and C. K. I. Williams. GTM: a principled alternative to the Self-Organizing Map. In *Advances in Neural Information Processing Systems*, volume 9, pages 354–360. MIT Press, 1997.

C. M. Bishop, M. Svensén, and C. K. I. Williams. GTM: the Generative Topographic Mapping. *Neural Computation*, 10(1):215–234, 1998.

L. Csató. *Gaussian Processes — Iterative Sparse Approximations.* PhD thesis, Aston University, 2002.

K. Grochow, S. L. Martin, A. Hertzmann, and Z. Popovic. Style-based inverse kinematics. In *ACM Transactions on Graphics (SIGGRAPH 2004)*, 2004.

G. Hinton and S. Roweis. Stochastic neighbor embedding. In Becker et al. [2003], pages 857–864.

G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal. The wake-sleep algorithm for unsupervised neural networks. *Science*, 268:1158–1161, 1995.

T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

N. D. Lawrence. Gaussian process models for visualisation of high dimensional data. In Thrun et al. [2004], pages 329–336.

N. D. Lawrence, M. Seeger, and R. Herbrich. Fast sparse Gaussian process methods: The informative vector machine. In Becker et al. [2003], pages 625–632.

N. D. Lawrence, M. Seeger, and R. Herbrich. The informative vector machine: A practical probabilistic alternative to the support vector machine. In preparation, expected August/September 04, 2004.

T. K. Leen, T. G. Dietterich, and V. Tresp, editors. *Advances in Neural Information Processing Systems*, volume 13, Cambridge, MA, 2001. MIT Press.

D. Lowe and M. E. Tipping. Feed-forward neural networks and topographic mappings for exploratory data analysis. *Neural Computing and Applications*, 4(83), 1996.

D. J. C. MacKay. Bayesian neural networks and density networks. *Nuclear Instruments and Methods in Physics Research, A*, 354(1):73–80, 1995.

J. R. Magnus and H. Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics.* John Wiley and Sons, 1988.

K. V. Mardia, J. Kent, and M. Bibby. *Multivariate analysis.* Academic Press, London, 1979.

T. P. Minka. *A family of algorithms for approximate Bayesian inference.* PhD thesis, Massachusetts Institute of Technology, 2001.

I. T. Nabney. *Netlab: Algorithms for Pattern Recognition.* Advances in Pattern Recognition. Springer, Berlin, 2001. Code available from `http://www.ncrg.aston.ac.uk/netlab/`.

S. Roweis, L. K. Saul, and G. Hinton. Global coordination of local linear models. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14, pages 889–896, Cambridge, MA, 2002. MIT Press.

S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

J. W. Sammon. IEEE transactions on computers. *A nonlinear mapping for data structure analysis*, C-18(5):401–409, 1969.

B. Schölkopf, A. J. Smola, and K.-R. Müller. Kernel principal component analysis. In *Proceedings 1997 International Conference on Artificial Neural Networks, ICANN'97*, page 583, Lausanne, Switzerland, 1997.

M. Seeger and M. I. Jordan. Fast sparse Gaussian process classification with multiple classes. Submitted to *NIPS* 2004, 2004.

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

J. B. Tenenbaum, V. d. Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

S. Thrun, L. Saul, and B. Schölkopf, editors. *Advances in Neural Information Processing Systems*, volume 16, Cambridge, MA, 2004. MIT Press.

M. E. Tipping. *Topographic Mappings and Feed-Forward Neural Networks.* PhD thesis, Aston University, Aston Street, Birmingham B4 7ET, U.K., 1996.

M. E. Tipping. Probabilistic visualisation of high-dimensional binary data. In M. J. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems*, volume 11, pages 592–598, Cambridge, MA, 1999. MIT Press.

M. E. Tipping. Sparse kernel principal component analysis. In Leen et al. [2001], pages 633–639.

M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, B*, 6(3):611–622, 1999.

C. K. I. Williams. Computing with infinite networks. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, Cambridge, MA, 1997. MIT Press.

C. K. I. Williams. Prediction with Gaussian processes: From linear regression to linear prediction and beyond. In M. I. Jordan, editor, *Learning in Graphical Models*, volume 89 of *Series D: Behavioural and Social Sciences*, Dordrecht, The Netherlands, 1998. Kluwer.

C. K. I. Williams. On a connection between kernel PCA and metric multidimensional scaling. In Leen et al. [2001].

Figure 13: Graphical representation of (a) the standard probabilistic PCA model and (b) its dual representation which also leads to a probabilistic interpretation of PCA. The nodes are shaded to represent different treatments. *Black* shaded nodes are optimised, *white* shaded nodes are marginalised and *gray* shaded nodes are observed variables.

A Probabilistic Interpretations of PCA

The standard probabilistic interpretation of PCA [Tipping and Bishop, 1999] involves a likelihood,

$$p\left(\mathbf{Y}|\mathbf{W},\mathbf{X},\beta\right) = \prod_{n=1}^{N} p\left(\mathbf{y}_n|\mathbf{W},\mathbf{x}_n,\beta\right)$$

which is taken to be Gaussian,

$$p\left(\mathbf{y}_n|\mathbf{W},\mathbf{x}_n,\beta\right) = N\left(\mathbf{y}_n|\mathbf{W}\mathbf{x}_n,\beta^{-1}\mathbf{I}\right),$$

the prior distribution for the latent variables is then taken to be Gaussian,

$$p\left(\mathbf{x}_n\right) = N\left(\mathbf{x}_n|\mathbf{0},\mathbf{I}\right),$$

and is duly marginalised to recover the marginal likelihood for the data,

$$p\left(\mathbf{Y}|\mathbf{W},\beta\right) = \prod_{n=1}^{N} p\left(\mathbf{y}_n|\mathbf{W},\beta\right), \tag{17}$$

where

$$p\left(\mathbf{y}_n|\mathbf{W},\beta\right) = N\left(\mathbf{y}_n|\mathbf{0},\mathbf{W}\mathbf{W}^{\mathrm{T}}+\beta^{-1}\mathbf{I}\right). \tag{18}$$

The structure of this model is shown graphically in Figure 13(a).

The dual representation of probabilistic PCA involves integrating out $\mathbf{W}$ and maximising with respect to $\mathbf{x}_n$

$$p\left(\mathbf{y}_n|\mathbf{W},\beta\right) = \int \prod_{n=1}^{N} p\left(\mathbf{y}_n|\mathbf{x}_n,\mathbf{W},\beta\right) p\left(\mathbf{W}\right) d\mathbf{W}.$$

By first specifying a prior distribution,

$$p\left(\mathbf{W}\right) = \prod_{i} N\left(\mathbf{w}_i|0,\mathbf{I}\right)$$

where $\mathbf{w}_i$ is the ith row of the matrix $\mathbf{W}$, and then integrating over $\mathbf{W}$ we obtain a marginalised likelihood for $\mathbf{Y}$,

$$p\left(\mathbf{Y}|\mathbf{X},\beta\right) = \frac{1}{(2\pi)^{\frac{DN}{2}}\left|\mathbf{K}\right|^{\frac{D}{2}}} \exp\left(-\frac{1}{2}\mathrm{tr}\left(\mathbf{K}^{-1}\mathbf{Y}\mathbf{Y}^{\mathrm{T}}\right)\right), \tag{19}$$

where $\mathbf{K} = \mathbf{X}\mathbf{X}^{\mathrm{T}} + \beta^{-1}\mathbf{I}$ and $\mathbf{X} = \left[\mathbf{x}_1^{\mathrm{T}} \dots \mathbf{x}_N^{\mathrm{T}}\right]^{\mathrm{T}}$. The structure of this model is shown in 13(b). Note that by taking $\mathbf{C} = \mathbf{W}\mathbf{W}^{\mathrm{T}} + \beta^{-1}\mathbf{I}$ we and substituting (18) into (17) as

$$p\left(\mathbf{Y}|\mathbf{X},\beta\right) = \frac{1}{(2\pi)^{\frac{DN}{2}}\left|\mathbf{C}\right|^{\frac{N}{2}}} \exp\left(-\frac{1}{2}\mathrm{tr}\left(\mathbf{C}^{-1}\mathbf{Y}^{\mathrm{T}}\mathbf{Y}\right)\right),$$

which highlights to a greater extent the duality between (19) and (17). Optimisation of (19) is clearly highly related to optimisation of (17). Tipping and Bishop [1999] showed how to optimise (17), in the next section we review this optimisation for DPPCA, but generalise it slightly so that it applies for any symmetric matrix $\mathbf{S}$, rather than only the inner product matrix $\mathbf{YY}^{\mathrm{T}}$. Thereby the derivation also covers the kernel PCA and multidimensional scaling cases outlined in Section 8.1.

B Optimisation of Dual PCA, KPCA and MDS Objective functions

Maximising (19) is equivalent to minimising the negative log likelihood given by

$$L = \frac{N}{2}\ln 2\pi + \frac{1}{2}\ln|\mathbf{K}| + \frac{1}{2}\mathrm{tr}\left(\mathbf{K}^{-1}\mathbf{S}\right), \tag{20}$$

where $\mathbf{S} = \mathbf{YY}^{\mathrm{T}}$. However, $\mathbf{S}$ needn't be constrained to this form, we outlined an objective function (for kernel PCA) in (14) where $\mathbf{S}$ was any positive definite kernel and in (20) the matrix $\mathbf{S}$ could be substituted with any symmetric similarity measure.

The gradient of the likelihood with respect to $\mathbf{X}$ can be found as

$$\frac{\partial L}{\partial \mathbf{X}} = -\mathbf{K}^{-1}\mathbf{S}\mathbf{K}^{-1}\mathbf{X} + \mathbf{K}^{-1}\mathbf{X},$$

setting the equation to zero and pre-multiplying by $\mathbf{K}$ gives

$$\mathbf{S}\left[\beta^{-1}\mathbf{I} + \mathbf{XX}^{\mathrm{T}}\right]^{-1}\mathbf{X} = \mathbf{X}.$$

We substitute $\mathbf{X}$ with its singular value decomposition, $\mathbf{X} = \mathbf{ULV}^{\mathrm{T}}$, giving

$$\mathbf{U}^{\mathrm{T}}\mathbf{SU}\left[\mathbf{L} + \beta^{-1}\mathbf{L}^{-1}\right]^{-1}\mathbf{V}^{\mathrm{T}} = \mathbf{LV}^{\mathrm{T}}$$

Right multiplying both sides by $\mathbf{V}$ (note that the solution is invariant to $\mathbf{V}$) and left multiplying by $\mathbf{U}$ we have, after some rearrangement,

$$\mathbf{SU} = \mathbf{U}\left(\beta^{-1}\mathbf{I} + \mathbf{L}^2\right),$$

which, since $\left(\beta^{-1}\mathbf{I} + \mathbf{L}^2\right)$ is diagonal can be solved by an eigenvalue problem where $\mathbf{U}$ are eigenvectors of $\mathbf{S}$ and $\Lambda = \left(\beta^{-1}\mathbf{I} + \mathbf{L}^2\right)$ are the eigenvalues. This implies that the elements from the diagonal of $\mathbf{L}$ are given by

$$l_i = \left(\lambda_i - \beta^{-1}\right)^{\frac{1}{2}}. \tag{21}$$

B.1 The Retained Eigenvalues

The natural follow up question is which of the N possible eigenvalues/vector pairs should be retained? For convenience let us ignore our previously defined ordering of the eigenvalues in terms of their magnitude and assume that will keep the first q eigenvalues.

B.1.1 Positive Definite S

First note that

$$\mathbf{K}_x = \mathbf{U}\left[\mathbf{L^2} + \beta^{-1}\mathbf{I}\right]\mathbf{U}^{\mathrm{T}}$$

where $\mathbf{U}$ is all the eigenvectors of $\mathbf{K}_y$. The full KL divergence is

$$\begin{aligned}
\mathrm{KL}\left(\mathbf{K}_y||\mathbf{K}_x\right) &= -\frac{N}{2}+\frac{1}{2}\log|\mathbf{K}_x|-\frac{1}{2}\log|\mathbf{K}_y|+\frac{1}{2}\mathrm{tr}\left(\mathbf{K}_x^{-1}\mathbf{K}_y\right)\\
&= \frac{1}{2}\sum_{i=1}^{q}\log\lambda_i-\frac{N-q}{2}\log\beta-\frac{1}{2}\sum_{i=1}^{N}\log\lambda_i+\frac{1}{2}\mathrm{tr}\left(\left[\mathbf{L^2}+\beta^{-1}\mathbf{I}\right]^{-1}\Lambda\right)\\
&= -\frac{1}{2}\sum_{i=q+1}^{N}\log\lambda_i-\frac{N-q}{2}\log\beta-\frac{N-q}{2}+\frac{\beta}{2}\sum_{i=q+1}^{N}\lambda_i
\end{aligned}$$

where we have used the fact that $\mathbf{K}_y=\mathbf{U}\Lambda\mathbf{U}^{\mathrm{T}}$. Differentiating with respect to β and setting the result to zero to obtaine a fixed point equation then gives

$$\beta=\frac{N-q}{\sum_{i=q+1}^{N}\lambda_i}$$

which when substituted back leads to

$$\mathrm{KL}\left(\mathbf{K}_y||\mathbf{K}_x\right)=\frac{N-q}{2}\left(\log\frac{\sum_{i=q+1}^{N}\lambda_i}{N-q}-\frac{1}{N-q}\sum_{i=q+1}^{N}\log\lambda_i\right), \tag{22}$$

which is recognised as the difference between the log ratio of the arithmetic and geometric means of the discarded eigenvalues. This difference will be zero if and only if the discarded eigenvalues are constant (when the arithmetic and geometric means become equal) otherwise it is positive. The difference it is minimised by ensureing that the eigenvalues we discard are adjacent to each other in terms of magnitude.

Which eigenvalues should we then discard? From (21) we note that the retained eigenvalues must be larger than β, otherwise l_i will be complex. The only way this can be true is if we discard the smallest $N-q$ eigenvalues, as retaining any others would force at least one eigenvalue of $\mathbf{X}$ to be negative.

B.1.2 Symmetric S

For the case where $\mathbf{S}$ is not positive definite the difference between the KL divergences is given by

$$\delta\mathrm{KL}-\frac{N}{2}+\frac{1}{2}\log|\mathbf{K}_x|-\frac{1-\alpha}{2}\log\left|\mathbf{K}_y^{+}\right|+\frac{\alpha}{2}\log\left|\mathbf{K}_y^{-}\right|+\frac{1}{2}\mathrm{tr}\left(\mathbf{K}_x^{-1}\mathbf{S}\right),$$

substituting with $\mathbf{K}_x=\mathbf{U}\left(\mathbf{L}^2+\beta^{-1}\mathbf{I}\right)\mathbf{U}^{\mathrm{T}}$, where $\mathbf{U}$ is again all the eigenvectors of $\mathbf{K}_y$, we obtain

$$\delta\mathrm{KL} = \frac{N-q}{2}\left(\frac{1}{N-q}\sum_{i=1}^{q}\log\lambda_i-\log\beta+\frac{1}{N-q}\sum_{i=1}^{N}\log\frac{\left(\lambda_i^{+}\lambda_i^{-}\right)^{\alpha}}{\lambda_i^{+}}-1+\frac{\beta}{N-q}\sum_{i=q+1}^{N}\lambda_i\right) \tag{23}$$

where we assume $\mathbf{S}=\mathbf{K}_y^{+}-\mathbf{K}_y^{-}$, the eigenvalues of $\mathbf{S}$ are given by λ_i, *i.e.* $\mathbf{S}=\mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}}$, and those of $\mathbf{K}_y^{+}$ and $\mathbf{K}_y^{-}$ are given by λ_i^{+} and λ_i^{-} respectively. Solving β again gives

$$\beta=\frac{N-q}{\sum_{i=q+1}^{N}\lambda_i}$$

which when substituted back leads to

$$\delta\mathrm{KL}=\frac{N-q}{2}\left(\frac{1}{N-q}\sum_{i=1}^{q}\log\lambda_i+\log\frac{\sum_{i=q+1}^{N}\lambda_i}{N-q}\right)+\frac{1}{2}\sum_{i=1}^{N}\log\frac{\left(\lambda_i^{+}\lambda_i^{-}\right)^{\alpha}}{\lambda_i^{+}},$$

Algorithm 2 Algorithm for finding a local minima in the objective function.

Include the largest eigenvalue. If the sum of the residual positives is still positive retain it, otherwise discard.
for each positive eigenvalue in descending order of magnitude.
Evaluate objective function for this number of eigenvalues. **do**
  Include the eigenvalue.
  if sum of the residual eigenvalues is still positive **then**
    retain the eigenvalue and evaluate δKL for the current retained eigenvalues.
  else
    discard the eigenvalue.
  end if
end for
Return the eigenvalues with the lowest associated δKL.

Note that the retained eigenvalues, $\{\lambda_i\}_{i=1}^q$, are constrained positive, otherwise the first term in the brackets becomes complex. From the second term in the brackets, the average of the discarded eigenvalues must also be zero or greater. The objective function can be re-written

$$\delta\mathrm{KL} = \frac{N-q}{2}\left(\log\frac{\sum_{i=q+1}^r \lambda_i}{N-q} - \frac{1}{N-q}\sum_{i=q+1}^r \log\lambda_i\right) + \frac{N-q}{2}\log\left(\frac{\sum_{i=r+1}^N \lambda_i}{\sum_{i=q+1}^r \lambda_i}+1\right) + \text{const.}$$

The first term here is lower bounded by zero via Jensen's inequality. Note that as $r \to N$ we recover (22). For $r < N$ the second term is always negative. It is minimised by making the sum of the discarded positive eigenvalues as close as possible to the sum of the discarded negative eigenvalues.

It is now no longer clear which eigenvectors need to be retained to obtain a global maxima, one simple, greedy, approach to optimisation of δKL would be to follow Algorithm 2.

C Equivalence of Eigenvalue Problems

In this section we review the equivalence of the eigenvalue problems associated with DPPCA and PPCA. For DPPCA the eigenvalue problem is of the form

$$\mathbf{Y}\mathbf{Y}^{\mathrm{T}}\mathbf{U} = \mathbf{U}\Lambda.$$

Premultiplying by $\mathbf{Y}^{\mathrm{T}}$ then gives

$$\mathbf{Y}^{\mathrm{T}}\mathbf{Y}\mathbf{Y}^{\mathrm{T}}\mathbf{U} = \mathbf{Y}^{\mathrm{T}}\mathbf{U}\Lambda \tag{24}$$

Since the $\mathbf{U}$ are the eigenvectors of $\mathbf{Y}\mathbf{Y}^{\mathrm{T}}$ (see the previous section) the matrix $\mathbf{U}^{\mathrm{T}}\mathbf{Y}\mathbf{Y}^{\mathrm{T}}\mathbf{U} = \Lambda$, therefore matrix $\mathbf{U}' = \mathbf{Y}^{\mathrm{T}}\mathbf{U}\Lambda^{-\frac{1}{2}}$ is orthonormal. Post multiplying both sides of (24) by $\Lambda^{-\frac{1}{2}}$ gives

$$\mathbf{Y}^{\mathrm{T}}\mathbf{Y}\mathbf{U}' = \mathbf{U}'\Lambda$$

which is recognised as the form of the eigenvalue problem associated with PPCA, where the eigenvectors of $\mathbf{Y}^{\mathrm{T}}\mathbf{Y}$ are given by $\mathbf{U}' = \mathbf{Y}^{\mathrm{T}}\mathbf{U}\Lambda^{-\frac{1}{2}}$ and the eigenvalues are given by Λ (as they were for DPPCA).

D Some useful noise models

The ADF framework allows us to consider an range of noise models. Here we provide details on derivation of the Bernoulli noise model and an ordinal categorical model.

D.1 Bernoulli noise models

For binary data an appropriate likelihood is the Bernoulli distribution. The Bernoulli noise model is widely used in classification problems. It is a special case of the binomial likelihood which, for binary labels given by $y_{ni} \in \{-1, 1\}$, has the form

$$p(y_{ni}|p_{ni}) = p_{ni}^{\frac{1+y_{ni}}{2}} (1 - p_{ni})^{\frac{1-y_{ni}}{2}},$$

where p_{ni} is in the range $[0, 1]$. To convert a continuous value, z_{ni}, to this range, we typically use a 'squashing function', in this paper we use the cumulative Gaussian distribution given by

$$\phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} \exp\left(-\frac{t^2}{2}\right) dt.$$

The likelihood can then be written as,

$$p(y_{ni}|z_{ni}) = \phi(z_{ni})^{\frac{1+y_{ni}}{2}} (1 - \phi(z_{ni}))^{\frac{1-y_{ni}}{2}},$$

which can be further simplified to

$$p(y_{ni}|z_{ni}) = \phi(y_{ni} z_{ni}).$$

If the continuous input parameter is also affected by a slope, λ, and an offset b_i so that $z_{ni} = \lambda(f_{ni} + b_i)$ then we have a likelihood (noise model) of the form

$$p(y_{ni}|f_{ni}) = \phi(y_{ni}\lambda(f_{ni} + b_i)) \tag{25}$$

For the derivations outlined in Section 5 we need the point marginal likelihood. From (12), substituting in (25), we have

$$Z_{ni} = \phi(u_{ni}),$$

where

$$u_{ni} = c_{ni}(\mu_{ni} + b_i)$$

and

$$c_{ni} = \frac{y_{ni}}{\sqrt{\lambda^{-2} + \varsigma_{ni}}}. \tag{26}$$

The form of c_{ni} highlights a redundancy in the representation: the slope λ can always be incorporated into the covariance function by modifying the kernel

$$k'(\mathbf{x}_i, \mathbf{x}_j) = k'(\mathbf{x}_i, \mathbf{x}_j) + \delta_{ij}\lambda^{-2},$$

where δ_{ij} is the Kronecker delta function, as this will have the immediate knock-on effect of making the posterior variance $\varsigma_{ni} \to \varsigma_{ni} + \lambda^{-2}$. Grouping the slope parameter with the kernel parameters is convenient as they may then be jointly optimised, as was discussed for the Gaussian case in Section 7.1.4. This is achieved by considering the noise model to have an infinite slope, such that $\lambda^{-2} \to 0$ giving

$$p(y_{ni}|f_{ni}) = H(y_{ni}(f_{ni} + b_i)),$$

where $H(\cdot)$ is the Heaviside step function. The derivation of the log marginal now leads to

$$c_{ni} = \frac{y_{ni}}{\sqrt{\varsigma_{ni}}}.$$

From Section 5 we know that to implement the noise model we must also find g_{ni} and ν_{ni},

$$g_{ni} = c_{ni}\mathcal{N}(u_{ni})[\phi(u_{ni})]^{-1}, \tag{27}$$

where

$$\mathcal{N}(u_{ni}) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{u_{ni}^2}{2}\right)$$

and

$$\nu_{ni} = g_{ni}\left(g_{ni} + u_{ni}c_{ni}\right). \tag{28}$$

The approximation may then be summarised in terms of site means and precisions by taking

$$m_{ni} = \frac{g_{ni}}{\nu_{ni}} + \mu_{ni} \tag{29}$$

$$\beta_{ni} = \frac{\nu_{ni}}{1 - \nu_{ni}\varsigma_{ni}}. \tag{30}$$

D.2 Ordinal Categorical Models

Taking account of more than two categories is also of interested. How such categories are modelled depends primarily on whether they are ordinal or nominal. We focus on ordinal categories as they are more straightforward to implement, however details of noise models for nominal categories are given in Seeger and Jordan [2004].

For the ordinal model we assume that y_{ni} is an integer in the range $[0, C-1]$. We define a likelihood as

$$p(y_{ni}|f_{ni}) = \left\{ \begin{array}{ccc} H\left(f_{ni} + b_i - \sum_{i=1}^{y_{ni}-1} w_i\right) H\left(-\left(f_{ni} + b_i - \sum_{i=1}^{y_{ni}} w_i\right)\right) & \text{for} & 0 < y_{ni} < C-1 \\ H\left(f_{ni} + b_i - \sum_{i=1}^{C-2} w_i\right) & \text{for} & y_{ni} = C-1 \\ H\left(-\left(f_{ni} + b_i\right)\right) & \text{for} & y_{ni} = 0 \end{array} \right. ,$$

where $H(\cdot)$ is the Heaviside step function (noise can always be introduced to this system by adding a diagonal term to the kernel matrix as we did in the Bernoulli noise model). The marginal likelihood is then given by

$$Z_{ni} = \left\{ \begin{array}{ccc} \phi(u_{ni}) - \phi(u'_{ni}) & \text{for} & 0 < y_{ni} < C-1 \\ \phi(u_{ni}) & \text{for} & y_{ni} = C-1 \\ \phi(-u_{ni}) & \text{for} & y_{ni} = 0 \end{array} \right.$$

where

$$u'_{ni} = c_{ni}\left(\mu_{ni} + b_i - \sum_{i=1}^{y_n - 1} w_i\right)$$

and

$$u_{ni} = c_{ni}\left(\mu_{ni} + b_i - \sum_{i=1}^{y_n} w_i\right)$$

$$c_{ni} = \frac{1}{\sqrt{\varsigma_{ni}}}.$$

Performing the necessary derivatives to obtain g_{ni} and ν_{ni} we have

$$g_{ni} = \left\{ \begin{array}{ccc} c_{ni}\frac{N(u_{ni}) - N(u'_{ni})}{\phi(u_{ni}) - \phi(u'_{ni})} & \text{for} & 0 < y_{ni} < C-1 \\ c_{ni}N(u_{ni})\left[\phi(u_{ni})\right]^{-1} & \text{for} & y_{ni} = C-1 \\ -c_{ni}N(u_{ni})\left[\phi(-u_{ni})\right]^{-1} & \text{for} & y_{ni} = 0 \end{array} \right. ,$$

and

$$\nu_{ni} = \left\{ \begin{array}{ccc} g_{ni}^2 + c_{ni}d_{ni} & \text{for} & 0 < y_{ni} < C-1 \\ g_{ni}\left(g_{ni} + c_{ni}u_{ni}\right) & \text{otherwise} & \end{array} \right. .$$

where

$$d_{ni} = c_{ni} \frac{u_{ni} N\left(u_{ni}\right) - u'_{ni} N\left(u'_{ni}\right)}{\phi\left(u_{ni}\right) - \phi\left(u'_{ni}\right)}.$$

For both this noise model and the Bernoulli noise model some care needs to be taken when computing ratios of the Gaussian to the cumulative Gaussian, as when the arguments have large magnitude the limiting values of these ratios need to be considered.