Deep Gaussian Processes

Andreas C. Damianou  Neil D. Lawrence
Dept. of Computer Science & Sheffield Institute for Translational Neuroscience,
University of Sheffield, UK
{andreas.damianou, n.lawrence}@sheffield.ac.uk

Abstract

In this paper we introduce deep Gaussian process (GP) models. Deep GPs are a deep belief network based on Gaussian process mappings. The data is modeled as the output of a multivariate GP. The inputs to that Gaussian process are then governed by another GP. A single layer model is equivalent to a standard GP or the GP latent variable model (GPLVM). We perform inference in the model by approximate variational marginalization. This results in a strict lower bound on the marginal likelihood of the model which we use for model selection (number of layers and nodes per layer). Deep belief networks are typically applied to relatively large data sets using stochastic gradient descent for optimization. Our fully Bayesian treatment allows for the application of deep models even when data is scarce. Model selection by our variational bound shows that a five layer hierarchy is justified even when modelling a digit data set containing only 150 examples.

1 Introduction

Probabilistic modelling with neural network architectures constitute a well studied area of machine learning. The recent advances in the domain of deep learning [Hinton and Osindero, 2006, Bengio et al., 2012] have brought this kind of models again in popularity. Empirically, deep models seem to have structural advantages that have been shown to improve learning in complicated datasets associated with abstract information [Bengio, 2009]. Many interesting variants of the traditional neural networks have emerged since the development of efficient training methodologies for deep architectures: deep belief networks, (stacked) restricted Boltzmann machines [Hinton, 2010] etc.

Gaussian process models were introduced in the machine learning community as a fully probabilistic substitute for the multilayer perceptron, inspired by the observation [Neal, 1996] that a GP is a multilayer perceptron with infinite basis functions. It seems prudent, therefore, to investigate how these models will perform in deep hierarchies. Inference in deep GP models is analytically intractable. Lawrence and Moore [2007] investigated a maximum a posteriori (MAP) approach for estimation of latent variables. For the MAP approach to work, however, a strong prior was required on the top level of the hierarchy. There are two main contributions in this paper. Firstly, we show how we can approximately marginalize the latent variables variationally. This gives a rigorous lower bound on the marginal log likelihood of a deep GP. We test the resulting approach on a toy problem and then recreate results of Lawrence and Moore [2007] on a motion capture example. Our second contribution is to demonstrate the applicability of deep models even when data is scarce. The variational lower bound gives us an objective measure from which we can select different structures for our deep hierarchy (number of layers, number of nodes per layer). In a simple digits example we find that the best lower bound is given by the model with the deepest hierarchy we applied (5 layers).

The deep GP consists of a cascade of hidden layers of latent variables where each node acts as output for the layer above and as input for the layer below—with the observed outputs being placed in the leaves of the hierarchy. Gaussian processes govern the mappings between the layers.
A single layer of the deep GP is a Gaussian process latent variable model (GPLVM). Inference over the latent variables even in this model is intractable. However, Titsias and Lawrence [2010] have shown that latent variables can be approximately marginalized allowing a variational lower bound on the likelihood to be computed. The appropriate size of the latent space can be computed using automatic relevance determination (ARD) priors [Neal, 1996, MacKay et al., 1992]. Damianou et al. [2011] extended this approach by placing a Gaussian process prior over the latent space giving a Bayesian dynamical GPLVM. Here we extend that approach to allow us to approximately marginalize any number of hidden layers. We demonstrate how a deep hierarchy of Gaussian processes can be obtained by marginalising out the latent variables in the structure, obtaining a fully Bayesian training procedure along with a variational approximation to the true posterior of all the latent variables given the outputs. The resulting model is very flexible, and we discuss a number of potential applications in further work.

2 The Model

We first consider standard approaches to modeling with GPs. We then extend these ideas to deep GPs by considering Gaussian process priors over the inputs to the GP model.

2.1 Standard GP Modelling

In the traditional probabilistic inference framework, we are given a set of training input-output pairs, stored in matrices $X \in \mathbb{R}^{N \times Q}$ and $Y \in \mathbb{R}^{N \times D}$ respectively, and seek to estimate the unobserved, latent function $f = f(x)$, responsible for generating $Y$ given $X$. In this setting, Gaussian processes (GP’s) [Rasmussen and Williams, 2006] can be employed as nonparametric prior distributions over the latent function $f$. More formally, we assume that each datapoint $y_n$ is generated from the corresponding $f(x_n)$ by adding independent Gaussian noise, i.e.

$$y_n = f(x_n) + \epsilon_n, \quad \epsilon_n \sim \mathcal{N}(0, \sigma^2), \quad (1)$$

and $f$ is drawn from a Gaussian process, i.e. $f(x) \sim \mathcal{G}(0, k(x, x'))$. This (zero-mean) Gaussian process prior only depends on the covariance function $k$ operating on the inputs $X$. As we wish to obtain a flexible model, we only make very general assumptions about the form of $f$ and this is reflected in the choice of the covariance function which defines the properties of this mapping. For example, an exponentiated quadratic covariance function $k(x_i, x_j) = (\sigma_{se})^2 \exp\left(-\frac{(x_i - x_j)^2}{2l^2}\right)$, forces the latent functions to be infinitely smooth. We denote any covariance function hyperparameters (such as $(\sigma_{se}, l)$ of the aforementioned covariance function) by $\theta$. The collection of latent function instantiations, denoted by $F = \{f_n\}_n^N$, is normally distributed, allowing us to compute analytically the marginal likelihood $^1$

$$p(Y|X) = \int \prod_{n=1}^N p(y_n|f_n)p(f_n|x_n)dF = \mathcal{N}(Y|0, K_{NN} + \sigma^2_\epsilon I), \quad K_{NN} = k(X, X). \quad (2)$$

Gaussian processes have also been used with success in unsupervised learning scenarios, where the input data $X$ are not directly observed. The Gaussian process latent variable model (GP-LVM) [Lawrence, 2005, 2004] provides an elegant solution to this problem by treating the unobserved inputs $X$ as latent variables, while employing a product of $D$ independent GPs as prior for the latent mapping. The assumed generative procedure now takes the form:

$$y_{nd} = f_d(x_n) + \epsilon_{nd}, \quad (3)$$

where $\epsilon$ is again random Gaussian noise and $F = \{f_d\}_d^D$ with $f_{nd} = f_d(x_n)$. Given a finite dataset, the Gaussian process priors take the form of a Gaussian distribution

$$p(F|X) = \prod_{d=1}^D \mathcal{N}(f_d|0, K_{NN}) \quad (4)$$

which, in turn, allows for general non-linear mappings to be marginalised out analytically to obtain the likelihood $p(Y|X) = \prod_{d=1}^D \mathcal{N}(y_d|X)$, analogously to equation (2).

$^1$All probabilities involving $f$ should also have $\theta$ in the conditioning set, but here we omit it for clarity.
2.2 Deep Gaussian Processes

Our deep Gaussian process architecture corresponds to a graphical model with three kinds of nodes, illustrated in figure 1(a): the leaf nodes \( Y \in \mathcal{R}^{N \times D} \) which are the only observed ones, the intermediate latent spaces \( X_h \in \mathcal{R}^{N \times Q_h}, h = 1, \ldots, H - 1 \), where \( H \) is the number of hidden layers, and the parent latent node \( Z = X_H \in \mathcal{R}^{N \times Q_H} \). In our notation the parent node is separated from the rest of the hidden layers for notational clarity and because it can be constrained with a prior of our choice and, therefore, be involved in different mathematical expressions. In this deep architecture, all intermediate nodes \( X_h \) act as inputs for the layer below (including the leaves) and as outputs for the layer above. For simplicity, consider a structure with only two hidden units, as the one depicted in figure 1(b). The generative process takes the form:

\[
y_{nd} = f_d^Y (x_n) + \epsilon_{nd}, \quad d = 1, \ldots, D, \quad x_n \in \mathcal{R}^Q
\]

\[
x_{nq} = f_q^X (z_n) + \epsilon_{nq}, \quad q = 1, \ldots, Q, \quad z_n \in \mathcal{R}^{Q_H}
\]

and the intermediate node is involved in two Gaussian processes, \( f^Y \) and \( f^X \), playing the role of an input and an output respectively; \( f^Y \sim \mathcal{GP}(0, k^Y(X, X)) \) and \( f^X \sim \mathcal{GP}(0, k^X(Z, Z)) \). This structure can be naturally extended vertically (i.e. deeper hierarchies) or horizontally (i.e. segmentation of each layer into different partitions of the output space), as we will see later in the paper. However, it is already obvious how each layer adds a significant number of model parameters (\( X_h \)) as well as a regularization challenge, since the size of each latent layer is crucial but has to be a priori defined. For this reason, unlike Lawrence and Moore [2007], we seek to variationally marginalise out the whole latent space. Not only this will allow us to obtain an automatic Occam’s razor due to the Bayesian treatment, but also we will end up with a significantly lower number of model parameters, since the variational procedure only adds variational parameters. The first step to this approach, is to define automatic relevance determination (ARD) covariance functions for the GP:

\[
k(x_i, x_j) = \sigma^2 \exp\left(-\frac{1}{2} \sum_{q=1}^{Q} w_q (x_{i, q} - x_{j, q})^2 \right).
\]

This covariance function assumes a different weight \( w_q \) for each latent dimension and this can be exploited in a Bayesian training framework in order to “switch off” irrelevant dimensions by driving their corresponding weight to zero, thus helping towards automatically finding the structure of complex models [Damianou et al., 2012]. However, the nonlinearities introduced by this covariance function make the Bayesian treatment of this model challenging. Nevertheless, following recent non-standard variational inference methods we can define analytically an approximate Bayesian training procedure, as will be explained in the next section.

2.3 Bayesian Training

A Bayesian training procedure requires optimisation of the model evidence:

\[
\log p(Y) = \log \int p(Y|X)p(X|Z)p(Z) dX dZ.
\]

When prior information is available regarding the observed data (e.g. their dynamical nature is known a priori), the prior distribution on the parent latent node can be selected so as to constrain the whole latent space through propagation of the prior density through the cascade. Here we take the general case where \( p(Z) = \mathcal{N}(Z|0, I) \). However, the integral of equation (7) is intractable due to the non-linear way in which \( X \) and \( Z \) are treated through the GP priors \( f^Y \) and \( f^X \). As a first step, we apply Jensen’s inequality to find a variational lower bound on \( \log p(Y) \):

\[
\log p(Y) \geq \mathcal{F}_v = \int Q \log \frac{p(Y|F^Y)p(F^Y|X)p(X|F^X)p(F^X|Z)p(Z)}{Q} dX dZ dF^Y dF^X
\]

where we expanded the likelihood and, additionally, we introduced a variational distribution \( Q \), the form of which will be defined later on. The above integral is still intractable because \( X \) and \( Z \) still appear nonlinearly in the \( p(F^Y|X) \) and \( p(F^X|Z) \) terms respectively. A key result of Titsias and Lawrence [2010] is that expanding the probability space of the GP prior \( p(F|X) \) with extra variables allows for priors on the latent space to be propagated through the nonlinear mapping \( f \). More precisely, we augment the probability space of equation (4) with \( K \) auxiliary pseudo-inputs.
\[ \tilde{X} \in \mathbb{R}^{K \times Q} \text{ and } \tilde{Z} \in \mathbb{R}^{K \times Q_z} \text{ that correspond to a collection of function values } U^Y \in \mathbb{R}^{K \times D} \text{ and } U^X \in \mathbb{R}^{K \times Q} \text{ respectively}. \]

Following this approach, we obtain the augmented GP likelihood:

\[ p(Y | X, \tilde{X}) = p(Y | F^Y)p(F^Y | U^Y, X)p(U^Y | \tilde{X})dF^YdU^Y \tag{9} \]

and similarly for the \( p(X | Z) \) term. The pseudo-inputs \( \tilde{X} \) and \( \tilde{Z} \) are known as inducing points, and will be dropped from our expressions from now on, for clarity. Note that \( F^Y \) and \( U^Y \) are draws from the same GP so that \( p(U^Y) \) and \( p(F^Y | U^Y, X) \) are also Gaussian distributions.

We are now able to define a variational distribution \( Q \) which, when combined with the new expressions for the augmented GP priors, results in a tractable variational bound. Specifically, we have:

\[ Q = p(F^Y | U^Y, X)q(U^Y)q(X)p(F^X | U^X, Z)q(U^X)q(Z). \tag{10} \]

We select \( q(U^Y) \) and \( q(U^X) \) to be free-form variational distributions, while \( q(X) \) and \( q(Z) \) are chosen to be Gaussian, factorised with respect to dimensions:

\[ q(X) = \prod_{q=1}^{Q} \mathcal{N}(\mu^X_q, S^X_q) \quad \text{and} \quad q(Z) = \prod_{q=1}^{Q_z} \mathcal{N}(\mu^Z_q, S^Z_q). \tag{11} \]

By substituting equations (10) and (9) back to (8), we see that the “difficult” terms \( p(F^Y | U^Y, X) \) and \( p(F^X | U^X, Z) \) cancel out in the fraction, leaving a quantity that can be computed analytically:

\[ \mathcal{F}_v = \int Q \log \frac{p(Y | F^Y)p(U^Y)p(X | F^X, U^X)p(U^X)p(Z)}{Q} dXdZdF^YdF^XdU^YdU^X. \tag{12} \]

More specifically, we can break the logarithm in equation (12) by grouping the variables of the fraction in such a way that the bound can be written as:

\[ \mathcal{F}_v = g_Y + r_X + H(q(X)) - KL[q(Z) \| p(Z)] \tag{13} \]

where \( H \) represents the entropy with respect to a distribution, KL denotes the Kullback – Leibler divergence and, using \( \langle \cdot \rangle \) to denote expectations,

\[ g_Y = g(Y, F^Y, U^Y, X) = \langle \log p(Y | F^Y) + \log q(U^Y) \rangle_{p(F^Y | U^Y, X)q(U^Y)q(X)} \tag{14} \]

\[ r_X = r(X, F^X, U^X, Z) = \langle \log p(X | F^X) + \log q(U^X) \rangle_{p(F^X | U^X, Z)q(U^X)q(X)q(Z)} \tag{15} \]

Both terms only involve known Gaussian densities and are, thus, tractable. The \( g_Y \) term is the same as the bound found for the Bayesian GP-LVM [Titsias and Lawrence, 2010]. Since it only involves expectations with respect to Gaussian distributions, the GP output variables are only involved in a quantity of the form \( YY^T \). Further, as can be seen from the above equations, the function \( r(\cdot) \) is similar to \( g(\cdot) \) but it involves expectations with respect to densities of all of its variables. Therefore, \( r_X \) will involve \( X \) (i.e. the outputs of the top layer) in a term \( \langle XX^\top \rangle_{q(X)} = \sum_{q=1}^{Q} \mu^X_q \mu^X_q + S^X_q \).

3 Extending the hierarchy

Although the main calculations were demonstrated in a simple hierarchy, it is easy to extend the model vertically, i.e. by adding more hidden layers, or horizontally, i.e. by considering conditional independencies of the latent variables belonging to the same layer. The first case only requires adding more \( r_X \) functions to the variational bound, i.e. instead of a single \( r_X \) term we will now have a sum: \( \sum_{h=1}^{H-1} r_{X_h} \), where \( r_{X_h} = r(X_{h}, F^{X_{h}}, U^{X_{h}}, X_{h+1}) \), with \( X_H = Z \).

Now consider the horizontal expansion scenario and assume that we wish to break the single latent space \( X_h \), of layer \( h \), to \( M_h \) conditionally independent subsets. As long as the variational distribution \( q(X_h) \) of equation (11) is chosen to be factorised in a consistent way, this is feasible by just breaking the original \( r_{X_h} \) term of equation (15) into the sum \( \sum_{m=1}^{M_h} r_{X_h}^{(m)} \). This follows just from the fact that, due to the independence assumption, it holds that \( \log p(X_h | X_{h+1}) = \)

\(^2\)The number of inducing points \( K \) does not need to be the same for all GPs
\[ \sum_{m=1}^{M_h} \log p(X_h^m | X_{h+1}) \]. Notice that the same principle can also be applied to the leaves by breaking the \( g_Y \) term of the bound. This scenario arises when, for example, we are presented with multiple different observation spaces which, however, we believe they have some commonality. For example, when the observed data are coming from a video and an audio recording of the same event. Given the above, the variational bound for the most general version of the model takes the form:

\[ \mathcal{F}_v = \sum_{m=1}^{M_Y} g_Y^{(m)} + \sum_{h=1}^{H-1} \sum_{m=1}^{M_h} r_X^{(m)} + \sum_{h=1}^{H-1} h_{q(X_h)} - KL (q(Z) \parallel p(Z)). \] (16)

Figure 1(c) shows the association of this objective function’s terms with each layer of the hierarchy. Recall that each \( r_X^{(m)} \) and \( g_Y^{(m)} \) term is associated with a different GP and, thus, is coming with its own set of automatic relevance determination (ARD) weights, which were described in equation (6).

### 3.1 Deep multiple-output Gaussian processes

The particular way of extending the hierarchies horizontally, as presented above, can be seen as a means of performing unsupervised multiple-output GP learning. This only requires assigning a different \( g_Y \) term (and, thus, associated ARD weights) to each vector \( y_d \), where \( d \) indexes the output dimensions. After training our model, we hope that the columns of \( Y \) that encode similar information will be assigned relevance weight vectors that are also similar. This idea can be extended to all levels of the hierarchy, thus obtaining a fully factorised deep GP model.

This special case of our model makes the connection between our model’s structure and neural network architectures more obvious: the ARD parameters play a role similar to the weights of neural networks, while the latent variables play the role of neurons which learn hierarchies of features.

![Diagram](a)

### 3.2 Parameters and complexity

In all graphical model variants shown in figure 1, every arrow represents a generative procedure with a GP prior, corresponding to a set of parameters \( \{ \tilde{X}, \theta, \sigma_e \} \). Further, each layer of latent variables corresponds to a variational distribution \( q(X) \) which is associated with a set of variational means and covariances, as shown in equation (11). The parent node can have the same form as equation (11) or can be constrained with a more informative prior which would couple the points of \( q(Z) \). For example, a dynamical prior would introduce \( Q \times N^2 \) parameters which can, nevertheless, be

![Diagram](b)

![Diagram](c)
reparametrized using less variables [Damianou et al., 2011]. However, as is evident from equations (10) and (12), the inducing points and the parameters of \( q(X) \) and \( q(Z) \) are variational rather than model parameters, something which significantly helps in regularizing the problem. Therefore, adding more layers to the hierarchy does not introduce many more model parameters. Moreover, as in common sparse methods for Gaussian processes [Titsias, 2009], the complexity of each generative GP mapping is reduced from the typical \( O(N^3) \) to \( O(NM^2) \).

4 Demonstration

In this section we demonstrate the deep GP model in toy and real-world datasets. The model is initialised by performing dimensionality reduction in the observations to obtain the first hidden layer and then repeating this process for the next layers. To obtain these stacked initial spaces we experimented with PCA and the Bayesian GP-LVM, but the end result did not vary significantly.

4.1 Toy data

We first test our model in toy data, created by sampling from a three-level hierarchy of GPs. Figure 2 (a) depicts the true hierarchy: from the top latent layer two intermediate latent signals are generated. These, in turn, together generate 10-dimensional observations (not depicted here) through sampling of another GP. These observations are then used to train the following models: a deep GP, a simple stacked Isomap and a simple stacked PCA method, the results of which are shown in figures 2 (b, c, d) respectively. Note that only our method marginalises the latent space and, in contrast to the other two, it is not given any information about the dimensionality of each true signal in the hierarchy. Further, as can be seen in figure 2, our model not only finds the correct dimensionality for each hidden layer, but it also discovers latent signals which are closer to the real ones.

Figure 2: Attempts to reconstruct the real data (fig. (a)) with our model (b), stacked Isomap (c) and stacked PCA (d). Our model can also find the correct dimensionalities automatically.

4.2 Modeling human motion

For our second demonstration we consider the same motion capture dataset used from Lawrence and Moore [2007], taken from the CMU MOCAP database\(^3\). The data, summarised in two 62-dimensional sets of 78 points each, represent two subjects walking towards each other and performing a ‘high-five’. We applied our method with a two-level hierarchy where the two observation sets were taken to be conditionally independent given their parent latent layer. Therefore, we obtained three optimised sets of ARD parameters: one for each modality of the bottom layer (shown with bar graphs having bins of different colours/widths for each modality, in figure 3(b), and one ARD weight set for the top node, as shown in figure 3(c). Our model discovered a common subspace in the intermediate layer, since for dimensions 2 and 6 both ARD sets have a non-zero value. This is expected, as the two subjects perform very similar motions with opposite directions. The ARD weights are also a means of automatically selecting the dimensionality of each layer and subspace. This kind of modelling is impossible for a MAP method like [Lawrence and Moore, 2007] which requires the exact latent structure to be given a priori. The full latent space learned by the aforementioned MAP method is plotted in figure 4 (d,e,f), where fig. (d) corresponds to the top latent space and each of the other two encodes information for each of the two interacting subjects. Our method is not constrained to two dimensional spaces, so for comparison we plot two-dimensional

\(^3\)http://mocap.cs.cmu.edu.
projections of the dominant dimensions of each subspace in figure 4 (a,b,c). The similarity of the latent spaces is obvious. In contrast to Lawrence and Moore [2007], we did not have to constrain the latent space with dynamics in order to obtain results of good quality.

Further, we can sample from these spaces to see what kind of information they encode. Indeed, we observed that the top layer generates outputs which correspond to different variations of the whole sequence, while when sampling from the first layer we obtain outputs which only differ in a small subset of the output dimensions, e.g. those corresponding to the subject’s hand.

Figure 3: Figure (a) shows the deep GP model employed. Figure (b) shows the ARD weights for $f^X$, and $f^Y$ and figure (c) those for $f^X$.

Figure 4: Left (a,b,c): projections of the latent spaces discovered by our model, Right (d,e,f): the full latent space learned for the model of Lawrence and Moore [2007].

4.3 Deep learning of digit images

In our last experiment we demonstrate the ability of our model to learn latent features of increasing abstraction and we demonstrate the usefulness of an analytic bound on the model evidence as a means of evaluating the quality of the data fit for different choices of the overall depth of the hierarchy. We built a dataset consisting of 50 examples for each of the digits 0,1 and 6 taken from the USPS handwritten digit database. Each digit is represented as an image in $16 \times 16$ pixels. We experimented with models of depth ranging from 1 (equivalent to Bayesian GP-LVM) to 5 hidden layers and evaluated each model by measuring the nearest neighbour error in the latent features discovered in each hierarchy. Our finding was that the approximate evidence of the model was increasing with the number of layers and so was the quality of the model in terms of nearest neighbour errors. Indeed, the single-layer model made 5 mistakes even though it automatically decided to use 10 latent dimensions and the quality of the trained models was increasing with the number of hidden layers. Finally, only one point had a nearest neighbour of a different class in the 4-dimensional top level’s feature space of a model with depth 5. A 2D projection of this space is plotted in figure 6(a). The ARD weights for this model are depicted in figure 5.

To further test the nature of the discovered latent features, we generated outputs by sampling from each hidden layer. As can be seen in figure 6(b), the first hierarchical layers encode very local features whereas the higher ones encode much more abstract information.

5 Discussion and future work

We have introduced a framework for efficient Bayesian training of hierarchical Gaussian process mappings for unsupervised learning. Our approach approximately marginalises out the latent space, thus allowing for automatic structure discovery in the hierarchy. The method was able to successfully learn a hierarchy of features which describe natural human motion and the pixels of handwritten

---

4The results were the same when we took into account the Bayesian Information Criterion.
Figure 5: The ARD weights of a deep GP with 5 hidden layers. The top layer is on the far left.

Figure 6: Fig. (a) demonstrates the nearest neighbour class separation test on a deep GP model with depth 5. Fig. (b) shows outputs obtained when sampling from this model. The first two rows (top-down), which were sampled from layers 1 and 2 respectively, encode very local features, e.g. explaining if a zero is a closed circle or not, or how big the circle of a 6 is. We discovered many more local features when we sampled from different dimensions. On the other hand, when we sampled from the two dominant dimensions of the parent latent node (two rows in the bottom) we obtained much more varying outputs. Thus, the higher levels indeed encode much more abstract information.

digits. Our variational lower bound selected a deep hierarchical representation for handwritten digits even though the data in our experiment was relatively scarce (150 data points).

In the future, we wish to test our model on various inference tasks, such as class conditional density estimation. Our method can also be used to improve existing deep algorithms, something which we plan to further investigate by incorporating ideas from past approaches. Indeed, previous efforts to combine GPs with deep structures were successful at unsupervised pre-training [Erhan et al., 2010] or guiding [Snoek et al., 2012] of traditional deep models.

Although the experiments presented here considered only up to five layers in the hierarchy, the methodology is directly applicable to deeper architectures, with which we intend to experiment in the future. The marginalisation of the latent space allows for such an expansion with simultaneous regularisation. The variational lower bound allows us to make a principled choice between models trained using different initializations and with different numbers of layers.

The deep hierarchy we have proposed could also be used with inputs governing the top layer of the hierarchy, leading to a powerful model for regression based on Gaussian processes, but which is not itself a Gaussian process. We expect such a model to have applications in multitask learning (where intermediate layers could learn representations shared across the tasks) and in modelling nonstationary data or data involving jumps. These are both areas where a single layer GP struggles.

Acknowledgments

Research was partially supported by the University of Sheffield Moody endowment fund and the Greek State Scholarships Foundation.

References


