Deep Gaussian Processes for Unsupervised Learning

Simone Rossi [1] **Maurizio Filippone** [1]

Abstract

Deep Gaussian Processes are modern deep probabilistic models that allow to model complex relationships in observed data. Deep Gaussian Processes are extremely complex to train and we recently made some significant advances in making these scalable. The use of Deep Gaussian Processes has been limited to the case of supervised problems. This project will focus on extending the use of Deep Gaussian Processes to unsupervised problems, notably clustering problems.

TODO

1. Introduction

The rapid progress of machine learning in the last few years are largely the result of advances in deep learning, combined with the availability of large datasets, fast GPUs and distributed computing solutions (F. Pace & Michiardi (2016)). For instance, there are models that can recognize images with an accuracy that rivals that of humans (**?**) and this is leading to revolutions in several domains such as autonomous transportation and medical image understanding.

All of these systems, though, currently use **supervised learning** in which the machine is trained with inputs labeled by humans. The challenge is to let machines learn from raw, unlabeled stream of data. This is known as **unsupervised learning**; Artificially Intelligent Systems nowadays do not possess "common sense", which humans acquire by observing the world, acting in it, and understanding the physical constraints of it. Geo Hinton, who is a famous professor of ML at the University of Toronto, has said

> *When were learning to see, nobodys telling us what the right answers are we just look. Every so often, your mother says thats a dog, but thats very little information. Youd be lucky if you got a few bits of information even one bit per second that way. The brains visual system has* 10^{14} *neural connections. And you only live for* 10^9 *seconds. So its no use learning one bit per second. You need more like* 10^5 *bits per second. And theres only one place you can get that much information: from the input itself.*
>
> - Georey Hinton, *1996 (quoted in (Gorder 2006))*

Two of the most interesting problem concerning unsupervised learning are **dimensionality reduction** and **clustering**.

1.1. Related work

TODO

2. Preliminaries

This section will be dedicated to review the main idea behind Gaussian Processes (GP) and a possible deep architecture built on top of them (DGP).

2.1. Gaussian Process

A Gaussian process (GP) is a generalization of the Gaussian probability distribution: whereas a probability distribution describes random variables which are scalars or vectors (for multivariate distributions), a stochastic process governs the properties of functions. Even though there are two different perspectives for looking at a GP, for the following formulation of the DGP, here it's present only the weight-space view.

The weight-space representation of GP for regression is the natural extension of the Bayesian linear regression, where the the number of features for each sample is potentially infinite. Consider a supervised learning problem where a

set of input vectors $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^\top$ is associated with a set of labels $Y = [y_1, \ldots, yn]^\top$, where $\mathbf{x}_i \in R^{D_{\text{in}}}$ and $y_i \in R$.

Stating from the input X, the design matrix $\mathbf{\Phi}$ can be computed applying a set of potentially non linear basis functions. Specifically, we introduce the function $\phi(\mathbf{x})$ which maps a D_{in}-dimensional input vector x into an F dimensional feature space as follow:

$$\Phi_i = \phi(\mathbf{x}_i) \quad \text{with} \quad \phi : R^{D_{\text{in}}} \rightarrow R^F \tag{1}$$

As long as the projections are fixed functions and independent on the parameters, the model is still linear in the parameters, and therefore analytically tractable.

Without entering into details, it can be proved that, for both training and prediction, the model is defined exclusively in terms of inner products in the input space that can be lifted into feature space by replacing occurrences of those product with a function $k(\mathbf{x}_0, \mathbf{x}_1)$, such that:

$$k(\mathbf{x}_0, \mathbf{x}_1) = \phi(\mathbf{x}_0)^\top \phi(\mathbf{x}_1) \tag{2}$$

This technique is particularly useful in situations where it is more convenient to compute the kernel than the feature vectors themselves. Moreover, since both in training and in prediction matrix inversion (which computational cost is $\mathcal{O}(n^3)$) are involved, when the number of features starts increasing, it's more convenient to invert a $n \times n$ matrix than a $F \times F$ one. In addition to that, since kernel functions can be computed without building the feature vectors and the mapping function ϕ, it's also possible to infer an infinite feature space dimension while keeping all the computations feasible.

One of the most famous kernels, also called covariance function, that embeds an infinite feature space dimension is the Radial Basis Function (RBF), defined as follows

$$k_{\text{rbf}}(\mathbf{x}, \mathbf{x}') = \exp\left[-\frac{1}{2}\left\|\mathbf{x} - \mathbf{x}'\right\|^\top\right] \tag{3}$$

It will be used in the following treatise as starting point for making GPs scalable in a deep architecture.

2.2. Deep Gaussian Processes

In this work, we build the deep architecture following the model firstly introduced by K. Cutajar & Filippone (2017). Generally, in deep models, the mapping between inputs and labels is expressed as a composition of functions

$$\mathbf{f}(\mathbf{x}) = \left(\mathbf{f}^{(N_{\text{h}}-1)} \circ \ldots \circ \mathbf{f}^{(0)}\right)(\mathbf{x}), \tag{4}$$

where, in this case, each of the N_{h} layers is finite set of GPs.

In contrast to Deep Neural Networks (DNNs), where each of the hidden layers implements a parametric function of its inputs, in DGPs these functions are assigned a GP prior, and are therefore nonparametric.

2.2.1. Random Fourier Expansion

With reference to the Bochner's theorem, any continuous covariance function $k(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_i - \mathbf{x}_j)$ is positive definite if and only if it can be rewritten as the Fourier transform of a non-negative measure $p(\boldsymbol{\omega})$. Denoting the spectral frequencies by $\boldsymbol{\omega}$, in the case of the RBF covariance in equation 3, this yields:

$$k_{\text{rbf}}(\mathbf{x}_i - \mathbf{x}_j) = \int p(\boldsymbol{\omega}) \exp\left[i(\mathbf{x}_i - \mathbf{x}_j)^\top \boldsymbol{\omega}\right] d\boldsymbol{\omega}. \tag{5}$$

Because the covariance function and the non-negative measures are real, we can drop the unnecessary complex part of the argument of the expectation, keeping $\cos((\mathbf{x}_i - \mathbf{x}_j)^\top \boldsymbol{\omega})$ that can be rewritten as $\cos(\mathbf{x}_i^\top \boldsymbol{\omega}) \cos(\mathbf{x}_j^\top \boldsymbol{\omega}) + \sin(\mathbf{x}_i^\top \boldsymbol{\omega}) \sin(\mathbf{x}_j^\top \boldsymbol{\omega})$.

The importance of the expansion above is that it allows us to interpret the covariance function as an expectation that can be estimated using Monte Carlo. Defining $\mathbf{z}(\mathbf{x}|\boldsymbol{\omega}) = [\cos(\mathbf{x}^\top \boldsymbol{\omega}), \sin(\mathbf{x}^\top \boldsymbol{\omega})]^\top$, the covariance function can be therefore unbisedly approximated as

$$k_{\text{rbf}}(\mathbf{x}_i, \mathbf{x}_j) \approx \frac{1}{N_{\text{RF}}} \sum_{r=1}^{N_{\text{RF}}} \mathbf{z}(\mathbf{x}_i|\tilde{\boldsymbol{\omega}}_r)^\top \mathbf{z}(\mathbf{x}_j|\tilde{\boldsymbol{\omega}}_r), \tag{6}$$

with $\tilde{\boldsymbol{\omega}}_r \sim p(\boldsymbol{\omega})$. This has an important practical implication, as it provides the means to access an approximate explicit representation of the mapping induced by the covariance function that, in the RBF case, we know is infinite dimensional.

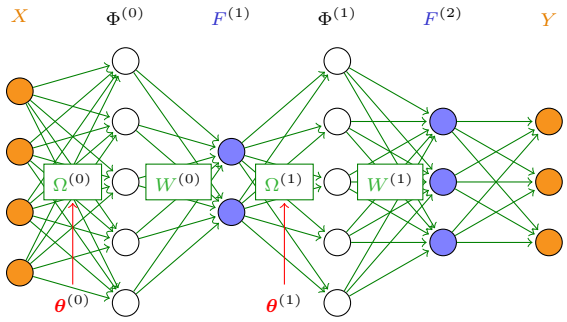

Figure 1. The DGP approximation used for building the deep model. At each hidden layer GPs are replaced by their two-layer weight-space approximation. Random-features $\Phi^{(l)}$ are obtained using a weight matrix $\Omega^{(l)}$. This is followed by a linear transformation parameterized by weights $W^{(l)}$.

2.2.2. Deep Architecture

With reference to K. Cutajar & Filippone (2017) work, the architecture that will be used as starting point for the analysis of the unsupervised case is represented in Figure 2.

Considering a DGP with RBF covariances, using the aforementioned expension, we have that

$$\Phi_{\text{rbf}}^{(l)} = \sqrt{\frac{(\sigma^2)^{(l)}}{N_{\text{RF}}^{(l)}}} \left[\cos\left(F^{(l)}\Omega^{(l)}\right), \sin\left(F^{(l)}\Omega^{(l)}\right)\right], \tag{7}$$

and $F^{(l+1)} = \Phi_{\text{rbf}}^{(l)} W^{(l)}$.

3. Deep models for dimensionality reduction

One of the most important problems in unsupervised learning is to represent the observed data $\mathbf{Y}$ (with $\mathbf{y}_i \in R^{D_{\text{obs}}}$) in some lower dimensional embedded space $\mathbf{X}$ (with $\mathbf{x}_i \in R^{D_{\text{lat}}}$). In a probabilistic model, the $\mathbf{X}$ space is also known as *latent space*, while all the variables $\mathbf{x}_i \in \mathbf{X}$ are called *latent variables*.

In the last decades, a lot of effort has been put in action for building the "optimal" mapping function between $R^{D_{\text{obs}}}$ and $R^{D_{\text{lat}}}$ (see, for instance, SR - Put references to PCA and Kernel PCA). In the next sections, we will review the key idea of Latent Variable Model and its natural extension to GPLVM.

3.1. Dual Probabilistic PCA

Typically, in the context of dimensionality reduction, we specify a latent variable model that relate a set of latent variables $\mathbf{X} \in R^{n \times D_{\text{lat}}}$ to a set of observed variables $\mathbf{Y} \in R^{n \times D_{\text{obs}}}$, through a set of potentially non linear parameters. The model is defined probabilistically, the latent variables are marginalized and the parameters are maximized.

Lawrence (2005) proved that, for a particular choice of Gaussian likelihood, this is the same as marginalize the parameters and maximize the latent variables. Moreover, it can be proved that for a particular choice of prior distribution on the parameters, this probabilistic model is equivalent to the probabilistic PCA (Tipping & Bishop, 1999).

Let's start by specifying a prior for $\mathbf{W}$ and let it be a simple spherical Gaussian distribution:

$$p(\mathbf{W}) = \prod_{i=0}^{D_{\text{obs}}-1} \mathcal{N}(\mathbf{w}_i|\mathbf{0}, \mathbf{I}) \tag{8}$$

The corresponding log-likelihood is

$$\log p(\mathbf{Y}|\mathbf{X}, \sigma^2) = -\frac{D_{\text{obs}}}{2} \ln |\mathbf{K}| - \frac{1}{2} \text{Tr}\left(\mathbf{K}^{-1}\mathbf{Y}\mathbf{Y}^\top\right) + \alpha, \tag{9}$$

where $\mathbf{K} = \mathbf{X}\mathbf{X}^\top + \sigma^2\mathbf{I}$ and $\alpha = -\frac{nD_{\text{obs}}}{2} \ln(2\pi)$ is normalization constant independent with the latents.

Lawrence (2005) proved that the maximization of the likelihood in Equation 9 with respect to the latent variables $\mathbf{X}$ turns out to be the solution of the eigenvalue problem of the classic PCA.

If fact, the solution of this optimization problem can be written in closed form as it follows

$$\mathbf{X} = \mathbf{U}\mathbf{L}\mathbf{V}^\top \tag{10}$$

where $\mathbf{U}$ is an $n \times D_{\text{lat}}$ matrix whose columns are the first D_{lat} eigenvectors of $\mathbf{Y}\mathbf{Y}^\top$, $\mathbf{L}$ is the associated diagonal eigenvalue matrix and $\mathbf{V}$ is eventually a $D_{\text{obs}} \times D_{\text{obs}}$ rotation matrix.

3.2. Gaussian process latent variable model

From the analysis of Equation (9), we can immediately see that the marginal likelihood for dual probabilistic PCA is a product of D_{obs} independent Gaussian processes with linear covariance function.

What will be done next is to replace the inner product kernel with a covariance function so that it allows non-linear functions to obtain a non-linear latent variable model. Due to the close relationship with the linear model, which has an interpretation as probabilistic PCA, such a model can be interpreted as a non-linear probabilistic version of PCA or, as it will be referred to, Gaussian process latent variable model (or GPLVM).

Unfortunately, though, in general there is not closed form solution of the maximization of the likelihood and therefore the resulting models will not be optimizable through an eigenvalue problem. The covariance function that will be used in the following examples is the Squared Exponential in Equation (3). As result, the log-likelihood is a highly non-linear function of the latents and the parameters. We are therefore forced to use automatic optimisation tools, such as the stochastic gradients and ADAM optimizer (Kingma & Ba, 2015) already implemented in TensorFlow.

3.2.1. Extension to deep models

In the previous sections, we showed how the problem of dimensionality reduction can be connected to a GPLVM. Now, we will show that, without great changes in the formulation, this model can be extended to DGPLVM. This is justified by the fact that when the functional relationship between variables is non-stationary, traditional GPs fail, while DGP can better adapt to handle non-stationary functions.

In the DGPLVM, the likelihood associated to the model is optimized using variational inference. Let $p(\mathbf{Y}|\mathbf{X}, \Xi)$ the likelihood of the model, where Ξ is the set of all parameters. As proved by K. Cutajar & Filippone (2017), defining $\mathcal{L}$ the log-likelihood and $\mathcal{E} = \mathrm{E}_{q(\mathbf{W})}\left(\log\left[p\left(\mathbf{Y}|\mathbf{X}, \mathbf{W}, \Xi\right)\right]\right)$, we obtain

$$\mathcal{L} \geq \mathcal{E} - \mathrm{DKL}\left[q(\mathbf{W}) \| p\left(\mathbf{W}\right)\right], \tag{11}$$

where $q(\mathbf{W})$ acts as an approximation to the posterior over all the weights $p(\mathbf{W}|\mathbf{Y}, \mathbf{X}, \Xi)$.

Optimizing the variational lower bound with respect to $q(\mathbf{W})$ and to $\mathbf{X}$, we will obtain a model equivalent to a deep Gaussian process latent variable model (or, as it will be referred to, DGPLVM).

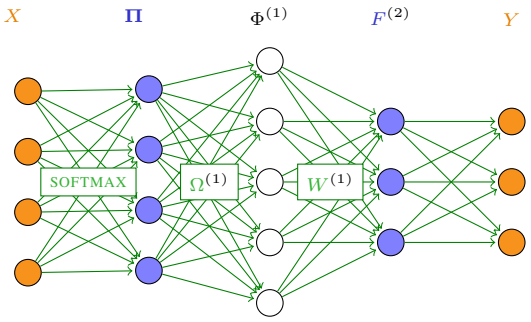

Figure 2. The DGP approximation used for building the deep model. At each hidden layer GPs are replaced by their two-layer weight-space approximation. Random-features $\Phi^{(l)}$ are obtained using a weight matrix $\Omega^{(l)}$. This is followed by a linear transformation parameterized by weights $W^{(l)}$.

4. Deep models for clustering

The second classical problem of unsupervised learning is clustering. SR - Add something general on clustering

4.1. Extended DGPLVM

The architecture already discussed in the previous sections can be also adapted to handle clustering assignment.

Our intent is to build a probabilistic model, such that for each observation $\mathbf{y}_i$ it gives a probability of being assigned to a cluster; formally we want to compute

$$\tilde{k} = \arg\max_k p(x_{ij} = k|\mathbf{y}_i, \Xi) \tag{12}$$

where Ξ is the set of all the parameters and hyperparameters of the model. This can be connected do a latent variable model, since the vector of label probability $\mathbf{x}_i$ is an hidden variable and it's never observed in the training set.

This is achieved keeping the same architecture presented in Figure 2 and replacing the first layer of GPs with a SOFTMAX layer. The SOFTMAX layer simply implements a softmax function on each samples of the latent space $\mathbf{X}$; in practice:

$$\mathbf{\Pi} = \frac{\exp(\mathbf{X})}{\sum_{j=0}^{D\mathrm{lat}-1} \exp(\mathbf{X}_{:,j})} \tag{13}$$

where $\mathbf{\Pi}$ is a $n \times D_{\mathrm{lat}}$ matrix of probabilities and D_{lat} is the number of cluster imposed to the model.

Invariance to rotations and to initialisation

Probabilistic model

5. Experiments

TODO

5.1. Dimensionality reduction for visualisation

TODO

5.1.1. LATENTS INITIALIZATION

TODO

5.2. Clustering

TODO

References

F. Pace, M. Milanesio, D. Venzano D. Carra and Michiardi, P. Experimental performance evaluation of cloud-based analytics-as-a-service. *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, pp. 196–203, 2016.

K. Cutajar, E. V. Bonilla, P. Michiardi and Filippone, M. Random feature expansions for deep gaussian processes. *Proceedings of the 33th International Conference on Machine Learning*, 2017.

Kingma, Diederik P. and Ba, Jimmy. Adam: A method for stochastic optimization. *3rd International Conference for Learning Representations, San Diego*, 2015.

Lawrence, Neil. Probabilistic non-linear principal component analysis with gaussian process latent variable model. *Journal of Machine Learning Research 6 (2005)*, pp. 1783–1816, 2005.

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