Uncertainty Propagation in Gaussian Process Pipelines

Andreas C. Damianou
Dept. of Computer Science and
Sheffield Institute for Translational Neuroscience
University of Sheffield, UK
andreas.damianou@sheffield.ac.uk

Neil D. Lawrence
Dept. of Computer Science and
Sheffield Institute for Translational Neuroscience
University of Sheffield, UK
n.lawrence@dcs.sheffield.ac.uk

1 Introduction and Motivation

Gaussian processes (GPs) constitute ideal candidates for building learning pipelines [1, 2, 3]. Their Bayesian, non-parametric nature reduces the need to cross-validate for specifying kernel parameters and they offer the promise of automatic structure determination through appropriate "pruning" priors, such as ARD or spike and slab. We consider learning pipelines which rely on Gaussian process models to accomplish inference tasks in each stage. Our aim is to allow for the uncertainty obtained in each stage to be propagated across the pipeline. Propagating uncertainty through GPs is challenging due to their non-linearity. To achieve our goal, we generalise a variational inference method [4, 5] that allows for approximately propagating densities throughout the nodes of GP-based directed graphical models. To illustrate the framework we focus on two different instances of learning paradigms that can be reduced to a pipeline of Gaussian processes. The first is semi-supervised learning through data imputation and the second is auto-regressive simulation of states from a dynamical model for domains such as reinforcement learning. Our approach increases automation in a principled way by eliminating the need to sample, cross-validate thresholds or manually detect outliers in between the learning stages. Experiments on simulated and real-world data show the importance of correctly propagating the uncertainty between stages of the pipeline and demonstrate the efficiency of our algorithms.

2 Uncertain Inputs Reformulation of Gaussian Process Models

Assume a dataset of observed input-output pairs stored by rows in matrices $\mathbf{Z} \in \Re^{N \times Q}$ and $\mathbf{Y} \in \Re^{N \times D}$ respectively and assumed to be related according to the following generative model:

$$y_{n,d} = f_d(\mathbf{z}_n) + (\epsilon_f)_{n,d}, \qquad (\epsilon_f)_{n,d} \sim \mathcal{N}\left(0, \beta^{-1}\right) \tag{1}$$

$$\mathbf{z}_n = \mathbf{x}_n + (\boldsymbol{\epsilon}_x)_n, \qquad (\boldsymbol{\epsilon}_x)_n \sim \mathcal{N}\left(\mathbf{z}_n, \mathbf{C}\right), \tag{2}$$

where a single subscript n indexes rows of the corresponding matrix and a double subscript n, d indexes single elements. Here, the vectors $\mathbf{x}_n$ are collected in a matrix $\mathbf{X} \in \Re^{N \times Q}$ and constitute the unobserved noise-free versions of the inputs $\mathbf{Z}$. In standard GP regression we assume GP priors for the mapping f and typically ignore the input noise, so that $\mathbf{C} = \mathbf{0}$. The GP-LVM [6] constitutes the unsupervised version of this setting, where no inputs are given. Now, $\mathbf{X}$ constitutes a *latent space* which must be recovered from the outputs $\mathbf{Y}$. The Bayesian GP-LVM [4] proceeds by placing a prior on the latent space, $p(\mathbf{X})$, and variationally integrating it out:

$$\log p(\mathbf{Y}) \geq \left\langle \log p(\mathbf{Y}|\mathbf{X}) \right\rangle_{q(\mathbf{X})} - \left\langle \log \left({}^{q(\mathbf{X})}/{}_{p(\mathbf{X})} \right) \right\rangle_{q(\mathbf{X})} \tag{3}$$

where $\langle \cdot \rangle_{q(\mathbf{X})}$ denotes an expectation with respect to $q(\mathbf{X})$, with $q(\mathbf{X})$ being a variational distribution which approximates the true posterior $p(\mathbf{X}|\mathbf{Y}, \mathbf{Z})$ (or $p(\mathbf{X}|\mathbf{Y})$ for the unsupervised case). We also select:

$$q(\mathbf{X}) = \prod_{n=1}^{N} q(\mathbf{x}_n), \text{ where: } q(\mathbf{x}_n) = \mathcal{N}\left(\mathbf{x}_n | \boldsymbol{\mu}_n, \mathbf{S}_n\right). \tag{4}$$

Recall equations (1, 2) and notice that a non-linear GP mapping f would not allow for tractable propagation of the input noise $(\boldsymbol{\epsilon}_x)_n$. In this paper we take advantage of the variational framework defined by the Bayesian

GP-LVM to obtain tractability. To achieve this, we equivalently rewrite equation (2) as $\mathbf{x}_n = \mathbf{z}_n + (\boldsymbol{\epsilon}_z)_n$, so that $p(\mathbf{X}|\mathbf{Z}) = \prod_{n=1}^{N} \mathcal{N}(\mathbf{x}_n|\mathbf{z}_n, \boldsymbol{\Sigma}_z)$. This prior only appears in the second expectation of the variational bound of equation (3) and preserves tractability. Alternatively, the input noise can be directly encoded in the approximate posterior of equation (4). Specifically, we can fix each $\boldsymbol{\mu}_n$ to its corresponding $\mathbf{z}_n$ and learn $\mathbf{S}_n$. This choice for $\boldsymbol{\mu}_n$ might not be optimal, but results in a smaller number of parameters to optimise (since we no longer have to estimate $\boldsymbol{\Sigma}_z$) and, in fact, in preliminary experiments this method worked better. We refer to the above model as a *variational GP-LVM*, but underline that it just constitutes a simple reformulation of the Bayesian GP-LVM.

3 Uncertainty Propagation in Auto-regressive Gaussian Processes

Having a method which implicitly models the uncertainty in the inputs of a GP also allows for doing predictions in an autoregressive manner while propagating the uncertainty through the predictive sequence [7, 8]. Specifically, assuming that the given data $\mathbf{Y}$ constitute a multivariate timeseries where the observed time vector $\mathbf{t}$ is equally spaced, we can reformat the data $\mathbf{Y}$ into input-output collections of pairs $\hat{\mathbf{Z}}$ and $\hat{\mathbf{Y}}$. For example, if we have an autoregressive time window of length τ, then to modify the model for autoregressive use the first input to the model, $\hat{\mathbf{z}}_1$, will be given by the stacked vector $\left[\mathbf{y}_1^\top, ..., \mathbf{y}_\tau^\top\right]^\top$ and the first output, $\hat{\mathbf{y}}_1$, will be given by $\mathbf{y}_{\tau+1}$ and similarly for the other data in $\hat{\mathbf{Z}}$ and $\hat{\mathbf{Y}}$. To perform extrapolation we need to train the model on the modified dataset $(\hat{\mathbf{Z}}, \hat{\mathbf{Y}})$. Then, we can perform iterative k-step ahead prediction to find a future sequence $\left[\mathbf{y}_{N+1}^\top, \mathbf{y}_{N+2}^\top, ...\right]^\top$ where, similarly to the approach taken by [7], the predictive variance in each step is accounted for and propagated in the subsequent predictions. For example, if $k = 1$ the algorithm will make iterative 1-step predictions in the future; in the beginning, the output $\mathbf{y}_{N+1}$ will be predicted given the training set. In the next step, the training set will be augmented to additionally include our distribution of predictions over $\mathbf{y}_{N+1}$ as part of the input set. We demonstrate this propagation of uncertainty on the Mackey-Glass chaotic time series, a standard benchmark which was also considered in [7]. Results are shown in Figure 1(b). Note, that it is straightforward to extend this model by introducing functions that map from $\mathbf{y}$ nonlinearly to an observation space, giving a fully nonlinear state space model in the manner of [9]. For our model, uncertainty is encoded in both the states and the nonlinear transition functions. Correct propagation of uncertainty is vital in well calibrated models of future system behavior and automatic determination of the structure of the model (for example the size of the window) can be informative in describing the order of the underlying dynamical system.

4 Uncertainty Propagation in Semi-supervised Gaussian Processes

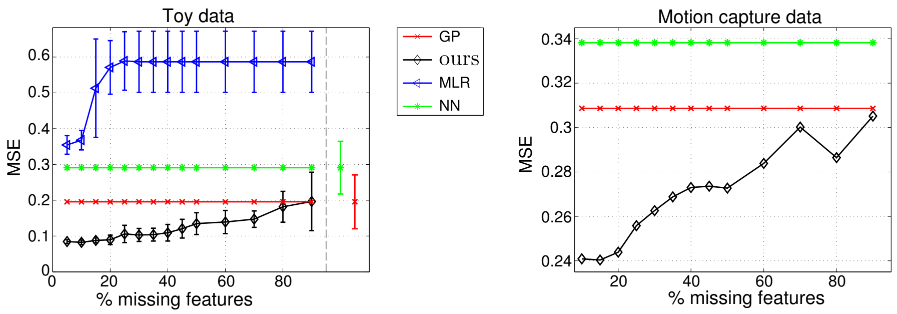
Semi-supervised learning involves making assumptions about the joint distribution of data which, for example, encode the idea of a manifold or data clustering. From a probabilistic perspective, one way of doing this is to describe a joint density over $p(\mathbf{Z}, \mathbf{Y})$ which encodes the relevant assumption. We can then impute missing data to perform semi-supervised regression where either the training inputs or targets may be partially missing. Notationally we consider data to be split into fully and partially observed subsets, e.g. $\mathbf{Z} = (\mathbf{Z}^o, \mathbf{Z}^u)$, where o and u denote fully and partially observed sets respectively. The features missing in $\mathbf{Z}^u$ can be different in number/location for each individual point $\mathbf{z}_n^u$. A standard GP regression model cannot deal with partially observed inputs $\mathbf{Z}^o$ and $\mathbf{Z}^u$. In contrast, in the variational GP-LVM framework we model the joint distribution between $\mathbf{Z}$ and $\mathbf{Y}$, obviating this issue. Implicitly this model then encodes the *manifold assumption* which assumes that the high dimensional data are really generated according to a lower dimensional latent space. From an approximate inference stand point, our main innovation here is to proscribe that the variational posterior distribution for $q(\mathbf{X}^o)$ is itself given by a GP, producing a variational back constraint in the style of [10]. A similar idea is applied by [11] in the context of variational autoencoders. Algorithm 1 summarises the approach.

There are some similarities to self-training [12] but as there are no mechanisms to propagate uncertainty in that domain they rely on boot-strapping to prevent model over-confidence. In contrast to self training we found that our framework, perhaps due to the principled propagation of uncertainty, does not require manual intervention to define thresholds or samples to estimate variances, thereby further automating the learning pipeline.

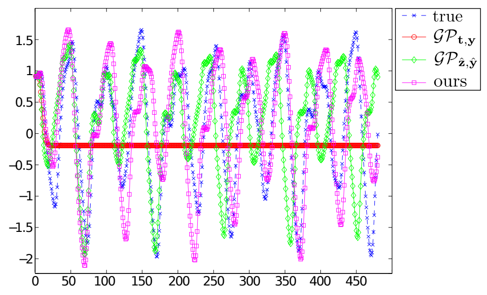
We tested our semi-supervised algorithm in: (a) simulated data, created by sampling inputs $\mathbf{Z}$ from a GP and passing them as inputs to another GP to obtain outputs $\mathbf{Y}$; (b) real-world motion capture data representing a walking motion. We treated the first 20 dimensions of the represented joints as targets and the rest 39 of the dimensions as inputs. In both cases, the goal was to reconstruct test outputs $\mathbf{Y}_*$ given fully observed inputs

Algorithm 1 A semi-supervised GP as a pipeline with automatic uncertainty propagation

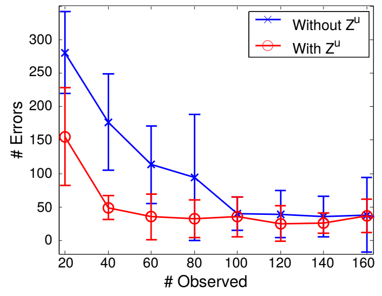
```
1:  *Given*: fully observed data $(\mathbf{Z}^o, \mathbf{Y}^o)$ and partially observed data $(\mathbf{Z}^u, \mathbf{Y}^u)$
2:  Initialize $q(\mathbf{X}^o) = \prod_{n=1}^{N} \mathcal{N}(\mathbf{x}_n^o | \mathbf{z}_n^o, \varepsilon \mathbf{I})$, where: $\varepsilon \to 0$
3:  Fix $q(\mathbf{X}^o)$ in the optimiser                       # (i.e. $q(\mathbf{X}^o)$ has no free parameters)
4:  Train a variational GP-LVM model $\mathcal{M}^o$ given the above $q(\mathbf{X}^o)$ and $\mathbf{Y}^o$
5:  for $n = 1, \cdots, |\mathbf{Y}^u|$ do
6:      Predict $p(\hat{\mathbf{x}}_n^u | \mathbf{y}_n^u, \mathcal{M}^o) \approx q(\hat{\mathbf{x}}_n^u) = \mathcal{N}\left(\hat{\mathbf{x}}_n^u | \hat{\boldsymbol{\mu}}_n^u, \hat{\mathbf{S}}_n^u\right)$
7:      Initialize $q(\mathbf{x}_n^u) = \mathcal{N}(\mathbf{x}_n^u | \boldsymbol{\mu}_n^u, \mathbf{S}_n^u)$ as follows:
8:      for $q = 1, \cdots, Q$ do
9:          if $z_{n,q}^u$ is observed then
10:             $\mu_{n,q}^u = z_{n,q}^u$ and $(S_n^u)_{q,q} = \varepsilon$, where: $\varepsilon \to 0$     # $(S_n^u)_{q,q}$ denotes the q-th diagonal element of $\mathbf{S}_n^u$
11:             Fix $\mu_{n,q}^u, (S_n^u)_{q,q}$ in the optimiser                # (i.e. will not be optimised)
12:         else
13:             $\mu_{n,q}^u = \hat{\mu}_{n,q}^u$ and $(S_n^u)_{q,q} = (\hat{S}_n^u)_{q,q}$
14: Train a variational GP-LVM model $\mathcal{M}^{o,u}$ by initialising $q(\mathbf{X}^o)$ and $q(\mathbf{X}^u)$ as shown above and using data $\mathbf{Y}^o, \mathbf{Y}^u$ (the
    locations that were fixed for the variational distributions will not be optimised).
15: All subsequent predictions in the next stage of the pipeline can be made using model $\mathcal{M}^{o,u}$.
```



(a) MSE for predictions obtained by different methods (missing inputs).



(b) Iterative 1−step ahead prediction.



(c) Semi-supervised learning (missing outputs).

Figure 1: (a) Flat line errors correspond to methods that cannot take into account partially observed inputs: a standard GP and Nearest Neighbour (NN). MLR corresponds to multiple linear regression. The results for simulated data are obtained from 4 trials. For GP and NN, errorbars do not change with x-axis and, for clarity, they are plotted separately on the right of the dashed vertical line (for nonsensical x values). (b) Comparing a standard GP approach ($\mathcal{GP}_{\mathbf{t},\mathbf{Y}}$), an "autoregressive" GP approach which does not propagate uncertainties ($\mathcal{GP}_{\hat{\mathbf{Z}},\hat{\mathbf{Y}}}$) and the variational GP-LVM in an "autoregressive" setting. (c) Using the latent space as features for logistic regression. We compare results of incorporating missing data through our framework versus ignoring them, and we plot the number of incorrectly classified test points as a function of $|\mathbf{Z}^o|$.

$\mathbf{Z}_*$, but training data were only partially observed, i.e. $((\mathbf{Z}^o, \mathbf{Z}^u), (\mathbf{Y}^o, \mathbf{Y}^u))$. For the simulated data we used the following sizes: $|\mathbf{Z}^o| = 40$, $|\mathbf{Z}^u| = 60$ and $|\mathbf{Z}_*| = 100$. The dimensionality of the inputs is 15 and of the outputs is 5. For the motion capture data we used $|\mathbf{Z}^o| = 50$, $|\mathbf{Z}^u| = 80$ and $|\mathbf{Z}_*| = 200$. In Figure 1(a) we plot the MSE obtained for our approach and competing methods for a varying percentage of missing features in $\mathbf{Z}^u$. As can be seen, the semi-supervised GP is able to handle the extra data and make better predictions, even if a very large portion is missing. Indeed, its performance starts to converge to that of a standard GP when there are 90% missing values in $\mathbf{Z}^u$ and performs identically to the standard GP when 100% of the values are missing.

Finally, inspired by [13] we use our semi-supervised GP as a means of extracting features for a subsequent discriminative classifier using all available information. Specifically, assume a classification problem where inputs $\mathbf{Z} = (\mathbf{Z}^u, \mathbf{Z}^o)$ are used to predict class labels $\mathbf{Y} = (\mathbf{Y}^u, \mathbf{Y}^o)$. In contrast to the notation used so far, now (o, u) index rows of the matrices for which the outputs (rather than the inputs) are missing, i.e. the class label is unknown. We make the assumption that in a multilabel scenario, *all* labels are missing (i.e. whole rows of $\mathbf{Y}$) and generalisation is left as future work. We then use the variational GP-LVM to learn a low-dimensional (and less noisy) manifold, represented as a probability $q(\mathbf{X}^u, \mathbf{X}^o)$, from $(\mathbf{Z}^u, \mathbf{Z}^o)$. In the next step of the pipeline, we can learn a discriminative classifier from $q(\mathbf{X}^o)$ to $\mathbf{Y}^o$. As can be seen in Figure 1(c), although the data portion indexed by u comes with no labels, it still helps learning a better probabilistic manifold for the subsequent discriminative task and, thus, results in better predictive performance in cases where labelled data is scarce.

5 Discussion and Future Work

We have presented an inference method and algorithms for propagating the uncertainty between stages of GP-based pipelines with a particular focus on semi-supervised and auto-regressive settings. In contrast to previous approaches, our method does not rely on local approximations (e.g. [8, 7]) and explicitly considers a distribution on the input space. This results in a unified framework for propagating the uncertainty, so that our models are easily extended. Overall, our approach increases pipeline automation and lays the foundation of creating more complex GP-based pipelines. Specifically, we envisage applying our uncertainty propagation algorithms in deep models which use relevance determination techniques [14] to automatically consolidate features (e.g. raw pixels and SIFT), embedded in the deep GP [15] framework that gradually abstracts features to “concepts”. We plan to investigate the application of these models in settings where control [9] or robotic systems learn by simulating future states in an auto-regressive manner and by using incomplete data with miminal human intervention.

Acknowledgments. This research was funded by the European research project EU FP7-ICT (Project Ref 612139 “WYSIWYD”). We thank Michalis Titsias for useful discussions.

References

[1] J. R. Lloyd, D. Duvenaud, R. Grosse, J. B. Tenenbaum, and Z. Ghahramani, “Automatic construction and natural-language description of nonparametric regression models,” *arXiv preprint arXiv:1402.4304*, 2014.

[2] A. G. Wilson and R. P. Adams, “Gaussian process kernels for pattern discovery and extrapolation,” in *ICML* (S. Dasgupta and D. McAllester, eds.), vol. 28 of *JMLR Proceedings*, pp. 1067–1075, JMLR.org, 2013.

[3] A. G. Wilson, *Covariance kernels for fast automatic pattern discovery and extrapolation with Gaussian processes*. PhD thesis, University of Cambridge, 2014.

[4] M. Titsias and N. D. Lawrence, “Bayesian Gaussian process latent variable model,” *Journal of Machine Learning Research - Proceedings Track*, vol. 9, pp. 844–851, 2010.

[5] A. Damianou, M. K. Titsias, and N. D. Lawrence, “Variational Gaussian process dynamical systems,” in *Advances in Neural Information Processing Systems*, vol. 24 of *NIPS ’11 Proceedings*, (Cambridge, MA), MIT, 2011.

[6] N. Lawrence, “Probabilistic non-linear principal component analysis with Gaussian process latent variable models,” *The Journal of Machine Learning Research*, vol. 6, pp. 1783–1816, 2005.

[7] A. Girard, C. E. Rasmussen, J. Quiñonero Candela, and R. Murray-Smith, “Gaussian process priors with uncertain inputs—application to multiple-step ahead time series forecasting,” NIPS, 2003.

[8] N. HajiGhassemi and M. Deisenroth, “Analytic long-term forecasting with periodic Gaussian processes,” *Journal of Machine Learning Research W&CP (Proceedings of AISTATS)*, vol. 33, pp. 303–311, 2014.

[9] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, “Gaussian processes for data-efficient learning in robotics and control,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 99, no. PrePrints, p. 1, 2014.

[10] C. H. Ek, J. Rihan, P. H. Torr, G. Rogez, and N. D. Lawrence, “Ambiguity modeling in latent spaces,” in *Machine Learning for Multimodal Interaction*, pp. 62–73, Springer, 2008.

[11] D. P. Kingma and M. Welling, “Auto-encoding variational Bayes,” *arXiv preprint arXiv:1312.6114*, 2013.

[12] C. Rosenberg, M. Hebert, and H. Schneiderman, “Semi-supervised self-training of object detection models,” in *Application of Computer Vision, 2005. WACV/MOTIONS ’05 Volume 1.*, vol. 1, pp. 29–36, Jan 2005.

[13] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling, “Semi-supervised learning with deep generative models,” *CoRR*, vol. abs/1406.5298, 2014.

[14] A. Damianou, C. Ek, M. Titsias, and N. Lawrence, “Manifold relevance determination,” in *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, ICML ’12, pp. 145–152, Omnipress, July 2012.

[15] A. Damianou and N. Lawrence, “Deep Gaussian processes,” in *Proceedings of the Sixteenth International Workshop on Artificial Intelligence and Statistics (AISTATS-13)*, AISTATS ’13, pp. 207–215, JMLR W&CP 31, 2013.