Training Deep Gaussian Processes using Stochastic Expectation Propagation and Probabilistic Backpropagation

Thang D. Bui, José Miguel Hernández-Lobato, Yingzhen Li, Daniel Hernández-Lobato and Richard E. Turner

UNIVERSITY OF CAMBRIDGE

HARVARD

UNIVERSIDAD AUTÓNOMA DE MADRID

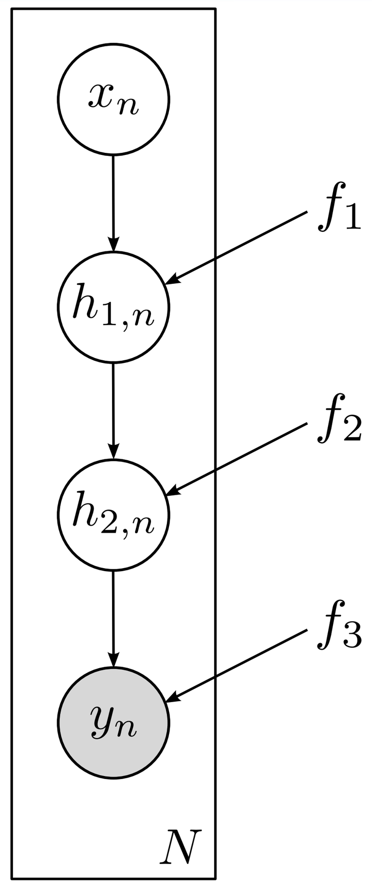
Deep Gaussian processes

Generative model:

$$f_l \sim \mathcal{GP}(0, k(.,.))$$

$$h_{l,n} = f_l(f_{l-1}(\cdots f_1(x_n)))$$

$$y_n = g(x_n) = f_L(f_{L-1}(\cdots f_2(f_1(x_n)))) + \epsilon_n$$



Deep GPs are:
+ multi-layer generalisation of Gaussian processes
+ equivalent to deep neural networks with infinitely wide hidden layers

Advantages:
+ Deep GPs are deep and nonparametric and can,
+ discover useful input warping or dimensionality compression and expansion
  → automatic, nonparametric Bayesian kernel design
+ give a non-Gaussian functional mapping g
+ repair the damage done by using sparse approximations,
+ retain uncertainty over latent mappings and representations.

Open theoretical questions:
+ architecture: number of layers, hidden dimensions, covariance functions,
+ learnability/identifiability/prior knowledge,
+ efficient inference and learning.

Taxonomy of previous approaches

Inducing point approaches

Approximate inference		FITC	Titsias/compression trick
	EP	?*	?*
	approx. EP	our approach (no inter. latent var. h)	
	variational	?**	Damianou et al.
	approx. variational		Hensman et al. (no inter. latent var. h)
	MAP	Lawrence and Moore	

∗ in principle this could be done
∗∗ the lower bound of FITC-Variational is the same as in Damianou et al.

Proposed approach

1. Sparsify the model using the FITC approximation:

Generative model:

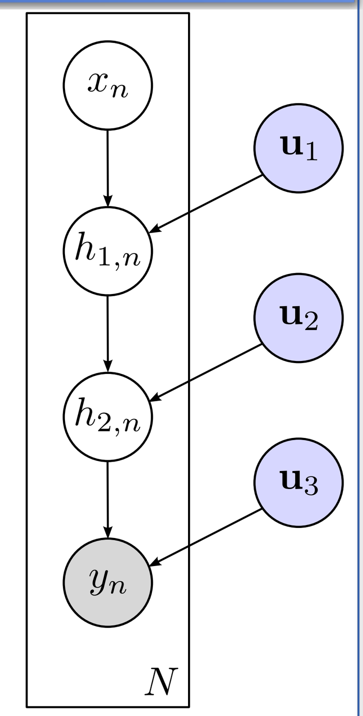
$$p(\mathbf{u}_l) = \mathcal{N}(\mathbf{u}_l; 0, \mathrm{K}_{\mathbf{u}_l\mathbf{u}_l})$$

$$p(h_l|\mathbf{u}_l, h_{l-1}) = \mathcal{N}(h_l; \mathrm{A}(h_{l-1})\mathbf{u}_l, \mathrm{B}(h_{l-1}) + \sigma_l^2)$$

where:

$$\mathrm{A}(h_{l-1}) = \mathrm{K}_{h_{l-1}\mathbf{z}_l}\mathrm{K}_{\mathbf{z}_l\mathbf{z}_l}^{-1}$$

$$\mathrm{B}(h_{l-1}) = \mathrm{K}_{h_{l-1}h_{l-1}} - \mathrm{K}_{h_{l-1}\mathbf{z}_l}\mathrm{K}_{\mathbf{z}_l\mathbf{z}_l}^{-1}\mathrm{K}_{\mathbf{z}_l h_{l-1}}$$



2. Approximate inference using stochastic Expectation Propagation:

	EP	Stochastic EP
Approx. posterior	$q(\theta) \propto p(\theta)\prod_n g_n(\theta)$	$q(\theta) \propto p(\theta)\prod_n g_n(\theta)$
Deletion	$q^{\backslash n}(\theta) \propto q(\theta)/g_n(\theta)$	$q^{\backslash 1}(\theta) \propto q(\theta)/g(\theta)$
Incorporating data	$\tilde{q}(\theta) \propto q^{\backslash n}(\theta)p(y_n\|\theta)$	$\tilde{q}(\theta) \propto q^{\backslash 1}(\theta)p(y_n\|\theta)$
Moment-matching	$\mathrm{KL}(\tilde{q}(\theta)\|\|q(\theta)) \to g_n(\theta)$	$\mathrm{KL}(\tilde{q}(\theta)\|\|q(\theta)) \to \bar{g}(\theta)$
Inclusion	$q(\theta) \propto q^{\backslash n}(\theta)g_n(\theta)$	$q(\theta) \propto q^{\backslash 1}(\theta)\bar{g}(\theta)$
Update		$g(\theta) \leftarrow g(\theta)^{1-\alpha}\bar{g}(\theta)^{\alpha}$
Memory complexity	$\mathcal{O}(NLM^2)$	$\mathcal{O}(LM^2)$

3. Approx. moment-matching using Probabilistic Backpropagation:

Shortcut for the moment maching step:

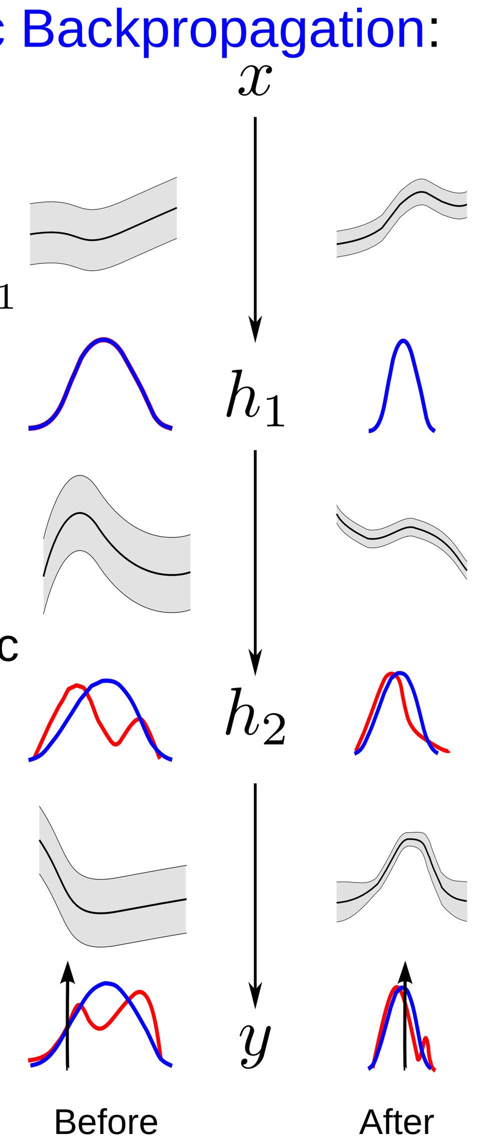
$$\mathbf{m} = \mathbf{m}^{\backslash 1} + \mathbf{V}^{\backslash 1}\frac{\mathrm{d}\log\mathcal{Z}}{\mathrm{d}\mathbf{m}^{\backslash 1}}$$

$$\mathbf{V} = \mathbf{V}^{\backslash 1} - \mathbf{V}^{\backslash 1}\left[\frac{\mathrm{d}\log\mathcal{Z}}{\mathrm{d}\mathbf{m}^{\backslash 1}}\left(\frac{\mathrm{d}\log\mathcal{Z}}{\mathrm{d}\mathbf{m}^{\backslash 1}}\right)^{\mathrm{T}} - 2\frac{\mathrm{d}\log\mathcal{Z}}{\mathrm{d}\mathbf{V}^{\backslash 1}}\right]\mathbf{V}^{\backslash 1}$$

where:

$$\mathcal{Z} = \int_{\mathbf{u}_{1:L}} p(y|\mathbf{x}, \mathbf{u}_{1:L})q^{\backslash 1}(\mathbf{u}_{1:L})$$

We compute $\mathcal{Z}$ and its gradients using the probabilistic backpropagation algorithm, which propagates a moment-matched Gaussian through the network, then computes the gradients using chain rule in the backward step.



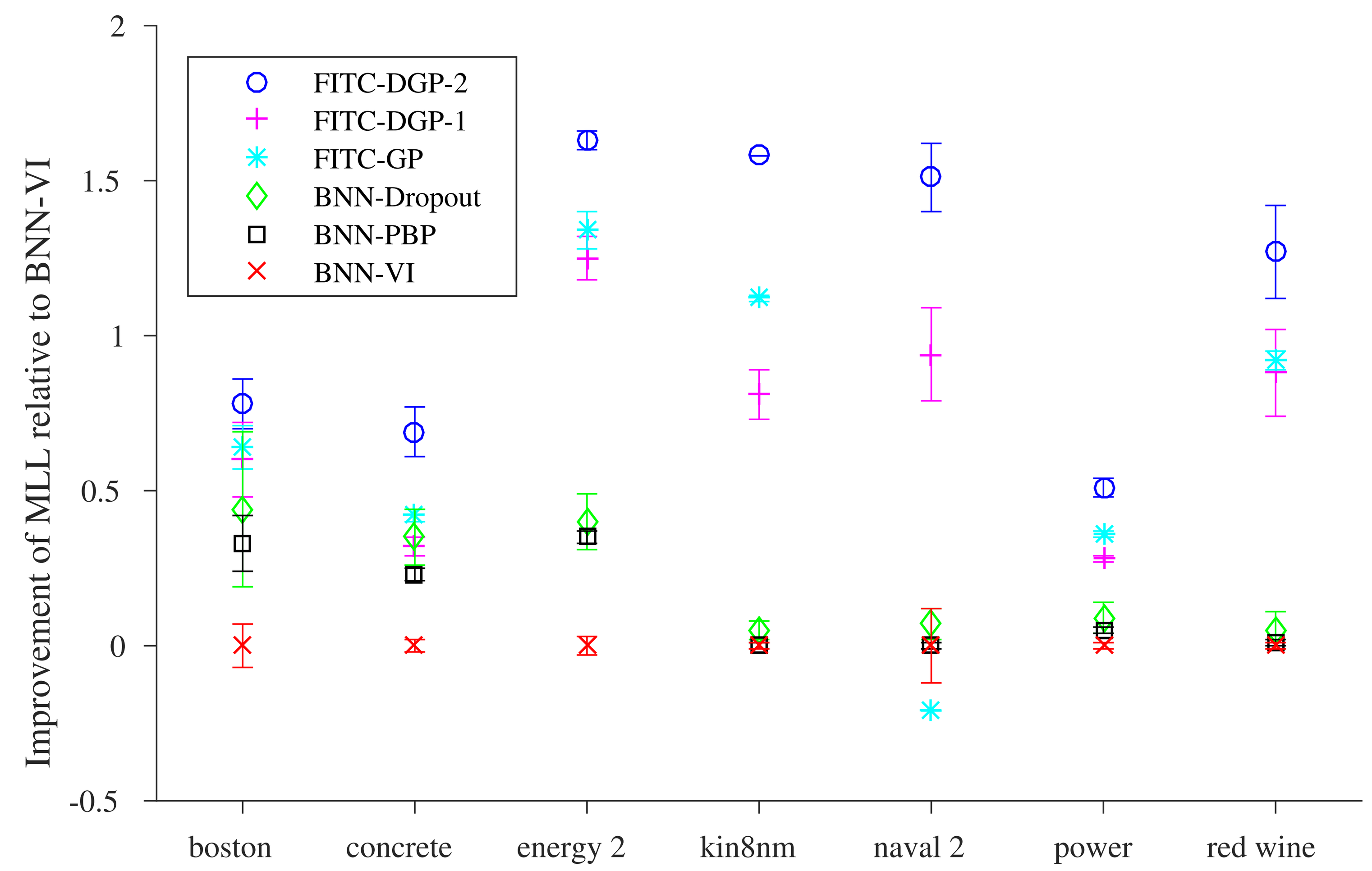
4. Hyperparameter optimisation using stochastic gradients:

+ Optimise the EP energy, but do not wait for EP inner loop to converge
+ Use the median trick and ADF for initialisation
+ Use Theano to compute the gradients of $\mathcal{Z}$
+ Use minibatch-based stochastic optimisation, we use Adam

Experimental results

We compared two different variants of Deep GPs with GPs and Bayesian neural networks on several regression tasks. Deep GPs using our proposed inference technique outperforms other models/methods.

Dataset	N	D	RMSE GP, 50	RMSE DGP, 1, 50	RMSE DGP, 2, 50
boston	506	13	3.09 ± 0.63	2.85 ± 0.65	$\mathbf{2.47 \pm 0.49}$
concrete	1030	8	5.24 ± 0.55	5.91 ± 1.65	$\mathbf{5.21 \pm 0.90}$
energy 1	768	8	0.50 ± 0.10	0.77 ± 0.59	$\mathbf{0.48 \pm 0.05}$
energy 2	768	8	1.60 ± 0.15	1.78 ± 0.43	$\mathbf{1.37 \pm 0.23}$
kin8nm	8192	8	0.04 ± 0.00	0.07 ± 0.04	$\mathbf{0.02 \pm 0.00}$
naval 1	11934	16	0.02 ± 0.01	$\mathbf{0.00 \pm 0.00}$	0.00 ± 0.00
naval 2	11934	16	0.01 ± 0.00	0.00 ± 0.00	$\mathbf{0.00 \pm 0.00}$
power	9568	4	3.19 ± 0.18	3.35 ± 0.20	$\mathbf{2.95 \pm 0.30}$
red wine	1588	11	$\mathbf{0.48 \pm 0.06}$	0.62 ± 0.05	0.54 ± 0.11
white wine	4898	11	0.37 ± 0.04	0.49 ± 0.09	$\mathbf{0.34 \pm 0.07}$
creep	2066	31	95.87 ± 18.03	74.86 ± 13.66	$\mathbf{70.58 \pm 15.55}$



Summary and future work

Our work proposes an approximate inference scheme for Deep GPs, that
+ extends probabilistic backpropagation for Bayesian neural networks
+ combines inducing point based sparse GP approximation with the memory efficient Stochastic Expectation Propagation
+ is fast and easy to implement
+ obtains state of the art regression results.

Current work includes:
+ parallel implementation
+ large scale experiment on big datasets
+ comparison to variational free-energy schemes
+ extending to classification and latent variable models
+ investigate various network architectures.

For more, see http://arxiv.org/abs/1511.03405