

Practical and Scalable Inference for Deep Gaussian Processes



Kurt Cutajar¹



Edwin V. Bonilla²



Pietro Michiardi¹



Maurizio Filippone¹

¹ EURECOM, Sophia Antipolis, France

² University of New South Wales, Sydney, Australia

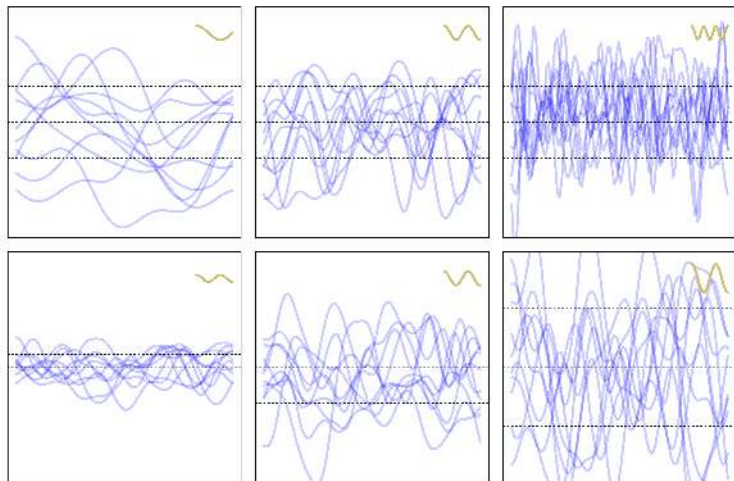
March 23rd, 2017

The Deep Learning Revolution

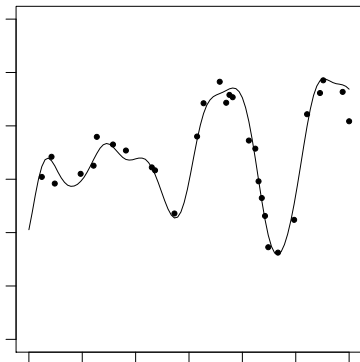
- Large representational power
- Mini-batch-based learning
- Exploit GPU and distributed computing
- Automatic differentiation
- Mature development of regularization (e.g., dropout)
- Application-specific representations (e.g., convolutional)

Gaussian Processes - Priors over Functions

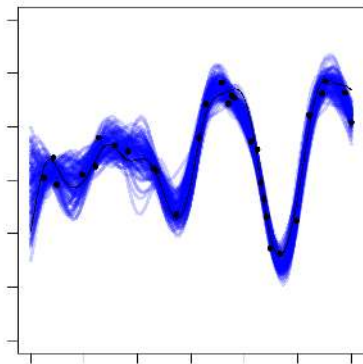
- Infinite Gaussian random variables with parameterized and input-dependent covariance



- Regression example

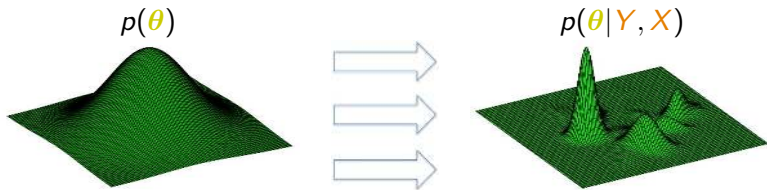


- Regression example



Bayesian Gaussian Processes

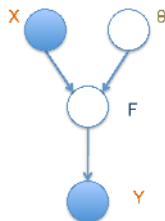
- Inputs = X Labels = Y
- $K = K(X, \theta)$



$$p(\theta|Y, X) = \frac{p(Y|X, \theta)p(\theta)}{\int p(Y|X, \theta)p(\theta)d\theta}$$

Challenges and Limitations

- Can only model stationary functions (shallow model)
- $p(Y|X, \theta)$ might be expensive to compute
- $p(Y|X, \theta)$ might not even be computable!



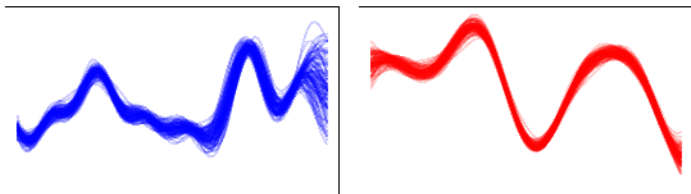
- Marginal likelihood

$$p(Y|X, \theta) = \int p(Y|F, X)p(F|\theta)dF$$

Can we exploit what made Deep Learning successful for practical and scalable learning of Gaussian processes?

Deep Gaussian Processes for Large Representational Power

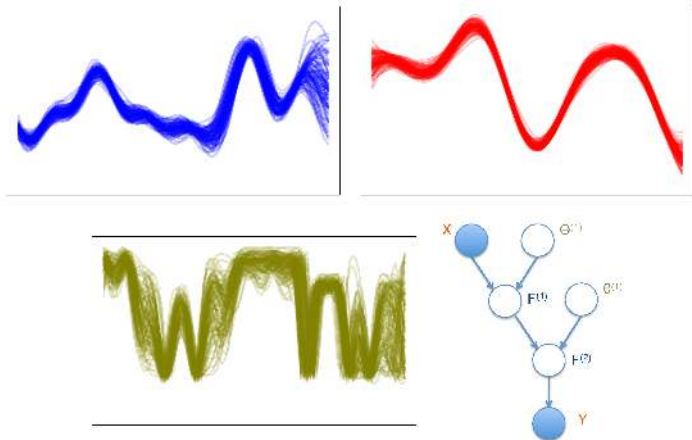
- Composition of processes



$$(f \circ g)(x)??$$

Deep Gaussian Processes for Large Representational Power

- Composition of processes



- Inference requires calculating integrals of this kind:

$$\begin{aligned} p(Y|X, \theta) = & \int p\left(Y|F^{(N_h)}, \theta^{(N_h)}\right) \times \\ & p\left(F^{(N_h)}|F^{(N_h-1)}, \theta^{(N_h-1)}\right) \times \dots \times \\ & p\left(F^{(1)}|X, \theta^{(0)}\right) dF^{(N_h)} \dots dF^{(1)} \end{aligned}$$

- Extremely challenging!

- Continuous shift-invariant covariance function

$$k(\mathbf{x}_i - \mathbf{x}_j | \boldsymbol{\theta}) = \sigma^2 \int p(\boldsymbol{\omega} | \boldsymbol{\theta}) \exp\left(\iota(\mathbf{x}_i - \mathbf{x}_j)^\top \boldsymbol{\omega}\right) d\boldsymbol{\omega}$$

- Continuous shift-invariant covariance function

$$k(\mathbf{x}_i - \mathbf{x}_j | \boldsymbol{\theta}) = \sigma^2 \int p(\boldsymbol{\omega} | \boldsymbol{\theta}) \exp\left(i(\mathbf{x}_i - \mathbf{x}_j)^\top \boldsymbol{\omega}\right) d\boldsymbol{\omega}$$

- Monte Carlo estimate

$$k(\mathbf{x}_i - \mathbf{x}_j | \boldsymbol{\theta}) \approx \frac{\sigma^2}{N_{\text{RFF}}} \sum_{r=1}^{N_{\text{RFF}}} \mathbf{z}(\mathbf{x}_i | \tilde{\boldsymbol{\omega}}_r)^\top \mathbf{z}(\mathbf{x}_j | \tilde{\boldsymbol{\omega}}_r)$$

with

$$\tilde{\boldsymbol{\omega}}_r \sim p(\boldsymbol{\omega} | \boldsymbol{\theta})$$

$$\mathbf{z}(\mathbf{x} | \boldsymbol{\omega}) = [\cos(\mathbf{x}^\top \boldsymbol{\omega}), \sin(\mathbf{x}^\top \boldsymbol{\omega})]^\top$$

- Define

$$\Phi^{(l)} = \sqrt{\frac{\sigma^2}{N_{\text{RFF}}^{(l)}}} \left[\cos \left(F^{(l)} \Omega^{(l)} \right), \sin \left(F^{(l)} \Omega^{(l)} \right) \right]$$

and

$$F^{(l+1)} = \Phi^{(l)} W^{(l)}$$

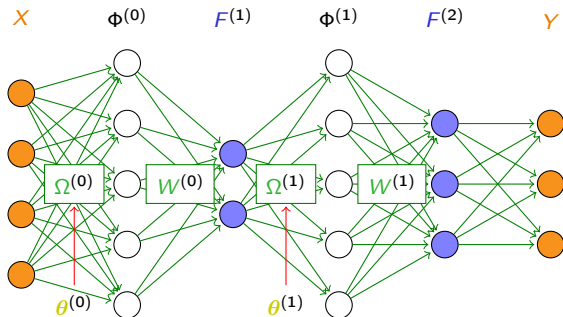
- At each layer, the priors over the weights are

$$p \left(\Omega_{.j}^{(l)} \mid \theta^{(l)} \right) = \mathcal{N} \left(\mathbf{0}, \left(\Lambda^{(l)} \right)^{-1} \right)$$

and

$$p \left(W_{.i}^{(l)} \right) = \mathcal{N} \left(\mathbf{0}, I \right)$$

DGPs with random features become DNNs



- Define $\Psi = (\Omega^{(0)}, \dots, W^{(0)}, \dots)$
- Lower bound for $\log [p(Y|X, \theta)]$

$$E_{q(\Psi)} (\log [p(Y|X, \Psi, \theta)]) - \text{DKL} [q(\Psi) \| p(\Psi|\theta)],$$

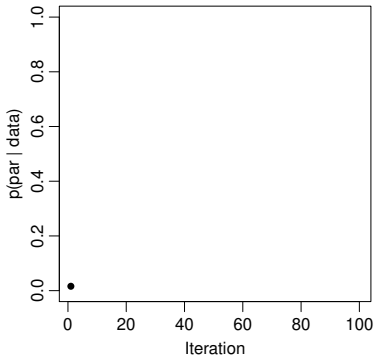
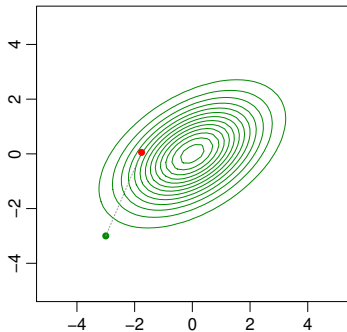
where $q(\Psi)$ approximates $p(\Psi|Y, \theta)$.

- DKL computable analytically if q and p are Gaussian!

Optimize the lower bound wrt the parameters of $q(\Psi)$

Stochastic Variational Inference

$$\text{vpar}' = \text{vpar} + \frac{\alpha_t}{2} \widetilde{\nabla}_{\text{vpar}}(\text{LowerBound}) \quad \alpha_t \rightarrow 0$$



Robbins and Monro, *AoMS*, 1951

- Assume that the likelihood factorizes

$$p(Y|X, \Psi, \theta) = \prod_k p(\mathbf{y}_k | \mathbf{x}_k, \Psi, \theta)$$

- Doubly stochastic **unbiased** estimate of the expectation term
 - Mini-batch

$$E_{q(\Psi)} (\log [p(Y|X, \Psi, \theta)]) \approx \frac{n}{m} \sum_{k \in \mathcal{I}_m} E_{q(\Psi)} (\log [p(\mathbf{y}_k | \mathbf{x}_k, \Psi, \theta)])$$

- Monte Carlo

$$E_{q(\Psi)} (\log [p(\mathbf{y}_k | \mathbf{x}_k, \Psi, \theta)]) \approx \frac{1}{N_{\text{MC}}} \sum_{r=1}^{N_{\text{MC}}} \log [p(\mathbf{y}_k | \mathbf{x}_k, \tilde{\Psi}_r, \theta)]$$

with $\tilde{\Psi}_r \sim q(\Psi)$.

- Reparameterization trick

$$(\tilde{W}_r^{(l)})_{ij} = \sigma_{ij}^{(l)} \varepsilon_{rij}^{(l)} + \mu_{ij}^{(l)}, \quad (1)$$

with $\varepsilon_{rij}^{(l)} \sim \mathcal{N}(0, 1)$

- ... same for Ω
- Variational parameters

$$\mu_{ij}^{(l)}, (\sigma^2)_{ij}^{(l)} \dots$$

... and the ones for Ω

- Optimization with automatic differentiation in TensorFlow

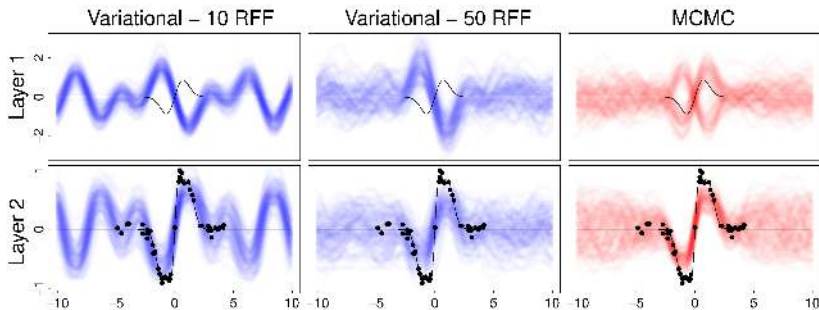
Comparison with MCMC

- Generate data from

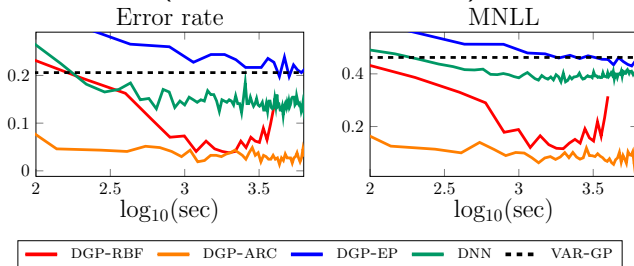
$$\mathcal{N}(y|h(h(x)), 0.01)$$

with

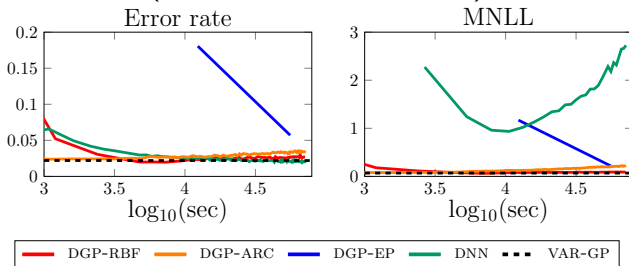
$$h(x) = 2x \exp(-x^2)$$



EEG dataset ($n = 14979$, $d = 14$)

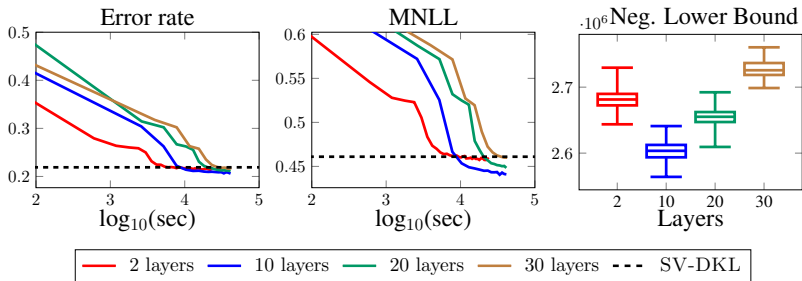


MNIST dataset ($n = 60000$, $d = 784$)



- Variant of MNIST with 8.1M images
- 99+% accuracy!
- Also, check out Krauth et al., arXiv 2016

Airline dataset ($n = 5\text{M}+$, $d = 8$)



- Contributions
 - Novel formulation of DGPs based on random features
 - We study the connections with DNNs
 - Scalable and practical DGPs inference - no inverses!

- Contributions
 - Novel formulation of DGPs based on random features
 - We study the connections with DNNs
 - Scalable and practical DGPs inference - no inverses!
- Ongoing work
 - Large dimensional problems with Fastfood
 - Other random features
 - Improving distributed implementation
 - Adding convolutional layers for image problems
 - Unsupervised learning, Bayesian Optimization, Calibration, ...

References and Acknowledgments

- Reference:

[1] K. Cutajar, E. V. Bonilla, P. Michiardi, and M. Filippone. **Random feature expansions for deep Gaussian processes**, 2016. *arXiv:1610.04386*.

- Code:

github.com/mauriziofilippone/deep_gp_random_features

- Reference:

[1] K. Cutajar, E. V. Bonilla, P. Michiardi, and M. Filippone. **Random feature expansions for deep Gaussian processes**, 2016. *arXiv:1610.04386*.

- Code:

github.com/mauriziofilippone/deep_gp_random_features

Thank you!



AXA
Research Fund