Practical and Scalable Inference for Deep Gaussian Processes

Kurt Cutajar\textsuperscript{1}  \hspace{1cm} Edwin V. Bonilla\textsuperscript{2}  \hspace{1cm} Pietro Michiardi\textsuperscript{1}  \hspace{1cm} Maurizio Filippone\textsuperscript{1}

\textsuperscript{1} EURECOM, Sophia Antipolis, France  \hspace{1cm} \textsuperscript{2} University of New South Wales, Sydney, Australia

March 23\textsuperscript{rd}, 2017
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
Gaussian Processes

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?
Composition of processes

\[(f \circ g)(x)\]
• Composition of processes

Damianou and Lawrence, AISTATS, 2013

Maurizio Filippone  Deep Gaussian Processes
Inference requires calculating integrals of this kind:

\[
p(Y | X, \theta) = \int p \left( Y | F^{(N_h)}, \theta^{(N_h)} \right) \times p \left( F^{(N_h)} | F^{(N_h-1)}, \theta^{(N_h-1)} \right) \times \ldots \times p \left( F^{(1)} | X, \theta^{(0)} \right) \, dF^{(N_h)} \ldots dF^{(1)}
\]

Extremely challenging!
DGPs - Bochner’s theorem

- Continuous shift-invariant covariance function

\[ k(x_i - x_j | \theta) = \sigma^2 \int p(\omega | \theta) \exp \left( \iota(x_i - x_j)^\top \omega \right) d\omega \]
DGPs - Bochner’s theorem

- Continuous shift-invariant covariance function

\[ k(x_i - x_j | \theta) = \sigma^2 \int p(\omega | \theta) \exp \left( (x_i - x_j)^\top \omega \right) d\omega \]

- Monte Carlo estimate

\[ k(x_i - x_j | \theta) \approx \frac{\sigma^2}{N_{\text{RFF}}} \sum_{r=1}^{N_{\text{RFF}}} z(x_i | \tilde{\omega}_r)^\top z(x_j | \tilde{\omega}_r) \]

with

\[ \tilde{\omega}_r \sim p(\omega | \theta) \]

\[ z(x | \omega) = [\cos(x^\top \omega), \sin(x^\top \omega)]^\top \]

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

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

and

$$
p \left( W_{.i}^{(l)} \right) = \mathcal{N} \left( 0, I \right)
$$
DGPs with random features become DNNs

\[ X \xrightarrow{\Phi(0)} F(1) \xrightarrow{\Phi(1)} F(2) \xrightarrow{\Omega(1)} Y \]

\[ \Omega(0) \xrightarrow{W(0)} \Omega(1) \xrightarrow{W(1)} \]

\[ \theta(0) \xrightarrow{\theta(1)} \]

Maurizio Filippone
Deep Gaussian Processes
Define \( \Psi = (\Omega^{(0)}, \ldots, \mathcal{W}^{(0)}, \ldots) \)

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 \( q(\psi) \)
\[ vpar' = vpar + \frac{\alpha_t}{2} \nabla_{vpar}(\text{LowerBound}) \quad \alpha_t \to 0 \]

Robbins and Monro, *AoMS*, 1951
Assume that the likelihood factorizes

\[ p(Y|X, \psi, \theta) = \prod_k p(y_k|x_k, \psi, \theta) \]

Doubly stochastic **unbiased** estimate of the expectation term

- **Mini-batch**

\[ E_{q(\psi)} \left( \log \left[ p(Y|X, \psi, \theta) \right] \right) \approx \frac{n}{m} \sum_{k \in \mathcal{I}_m} E_{q(\psi)} \left( \log \left[ p(y_k|x_k, \psi, \theta) \right] \right) \]

- **Monte Carlo**

\[ E_{q(\psi)} \left( \log \left[ p(y_k|x_k, \psi, \theta) \right] \right) \approx \frac{1}{N_{MC}} \sum_{r=1}^{N_{MC}} \log[p(y_k|x_k, \tilde{\psi}_r, \theta)] \]

with \( \tilde{\psi}_r \sim q(\psi) \).
Reparameterization trick

\[
\tilde{W}_r^{(l)}_{ij} = \sigma_{ij}^{(l)} \epsilon_{rij}^{(l)} + \mu_{ij}^{(l)},
\]

(1)

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

\(\ldots\) same for \(\Omega\)

Variational parameters

\[
\mu_{ij}^{(l)}, (\sigma^2)_{ij}^{(l)} \ldots
\]

\(\ldots\) and the ones for \(\Omega\)

Optimization with automatic differentiation in TensorFlow

Kingma and Welling, *ICLR*, 2014
Comparison with MCMC

Generate data from

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

with

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

EEG dataset

\( n = 14979, \ d = 14 \)

![Graph showing error rate and MNLL for various models](image)

- **Error rate**
  - Log scale on the x-axis: \( \log_{10}(\text{sec}) \)
  - Y-axis: Error rate

- **MNLL**
  - Log scale on the x-axis: \( \log_{10}(\text{sec}) \)
  - Y-axis: MNLL

Models compared:
- DGP-RBF
- DGP-ARC
- DGP-EP
- DNN
- VAR-GP

Maurizio Filippone
Deep Gaussian Processes
Results - Multiclass Classification

MNIST dataset

\( (n = 60000, \, d = 784) \)

![Graph showing error rate and MNLL over log10(sec) for different models: DGP-RBF, DGP-ARC, DGP-EP, DNN, VAR-GP.]
Variant of MNIST with 8.1M images
99+\% accuracy!
Also, check out Krauth et al., arXiv 2016
Airline dataset

\( (n = 5M+, \; d = 8) \)
Conclusions

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, ...
Conclusions

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, ...
Reference:

Code:
github.com/mauriziofilippone/deep_gp_random_features
References and Acknowledgments

Reference:

Code:
github.com/mauriziofilippone/deep_gp_random_features

Thank you!

Maurizio Filippone
Deep Gaussian Processes