Random Feature Expansions 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

February 9\textsuperscript{th}, 2018
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)
Infinite Gaussian random variables with parameteric and input-dependent covariance
Gaussian Processes - Prior over Functions

\[ K = \]

Maurizio Filippone
Deep Gaussian Processes
Gaussian Processes - Prior over Functions

\[ K = n \times n \]

Maurizio Filippone

Deep Gaussian Processes
Gaussian Processes - Prior over Functions

- Regression example
Regression example
Classification example
Classification 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

Inference requires calculating integrals of this kind:

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

Extremely challenging!
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

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

- Monte Carlo estimate

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

with

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

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

Define

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

and

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

We are stacking Bayesian linear models with

\[ p \left( W_{:,i}^{(l)} \right) = \mathcal{N} \left( 0, I \right) \]

Nonlinearity due to trigonometric functions
DGPs with random features become DNNs
Define $\Psi = (\Omega^{(0)}, \ldots, \mathcal{W}^{(0)}, \ldots)$

Lower bound for $\log [p(Y|X, \theta)]$

$$\mathbb{E}_{q(\psi)} \left( \log [p(Y|X, \psi, \theta)] \right) - \text{DKL} [q(\psi) \| p(\psi|\theta)] ,$$

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 Gradient Optimization

$$E\{\widehat{\nabla}_{\text{par}}\text{LowerBound}\} = \nabla_{\text{par}}\text{LowerBound}$$

Robbins and Monro, AoMS, 1951
\[ v_{\text{par}}' = v_{\text{par}} + \frac{\alpha_t}{2} \nabla_{v_{\text{par}}} (\text{LowerBound}) \quad \alpha_t \to 0 \]
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**

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

- **Monte Carlo**

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

with \( \tilde{\psi}_r \sim q(\psi) \).
Stochastic Variational Inference

- Reparameterization trick

\[
(\tilde{W}^{(l)}_{ij}) = \sigma^{(l)}_{ij} \varepsilon^{(l)}_{rij} + \mu^{(l)}_{ij},
\]

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

- ... same for \( \Omega \)

- Variational parameters

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

- ... and the ones for \( \Omega \)

- Optimization with automatic differentiation in TensorFlow

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

- Generate data from

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

with

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

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

Error rate

```
2 2.5 3 3.5
0.1
0.2
log10(sec)
```

MNLL

```
2 2.5 3 3.5
0.2
0.4
log10(sec)
```

- dgp-rbf
- dgp-arc
- dgp-ep
- dnn
- var-gp

Maurizio Filippone
Deep Gaussian Processes
Results - Multiclass Classification

**MNIST dataset**

\[ n = 60000, \quad d = 784 \]

<table>
<thead>
<tr>
<th>log(_{10}(\text{sec}))</th>
<th>Error rate</th>
<th>MNLL</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0.05</td>
<td>1</td>
</tr>
<tr>
<td>3.5</td>
<td>0.07</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>0.09</td>
<td>3</td>
</tr>
<tr>
<td>4.5</td>
<td>0.1</td>
<td></td>
</tr>
</tbody>
</table>

Graph showing comparison of different models (DGP-RBF, DGP-ARC, DGP-EP, DNN, VAR-GP) on MNIST dataset.
Variant of MNIST with 8.1M images
99+\% accuracy!
Also, check out Krauth et al., UAI 2017
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!
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, ...
References and Acknowledgments

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