

Random Feature Expansions for Deep Gaussian Processes

Kurt Cutajar ¹, Edwin V. Bonilla ², Pietro Michiardi ¹, Maurizio Filippone ¹

¹ EURECOM, Sophia Antipolis, France

² University of New South Wales, Sydney, Australia

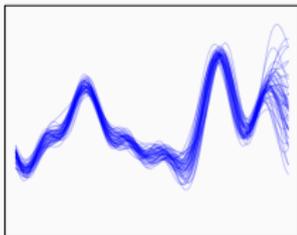
ICML 2017 - August 9th 2017

Deep Gaussian Processes

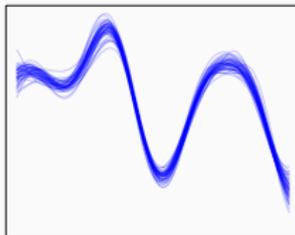
- Deep probabilistic models
- Composition of functions

$$f(\mathbf{x}) = \left(h^{(N_h-1)} \left(\boldsymbol{\theta}^{(N_h-1)} \right) \circ \dots \circ h^{(0)} \left(\boldsymbol{\theta}^{(0)} \right) \right) (\mathbf{x})$$

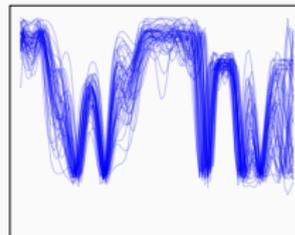
$h^{(0)}(\mathbf{x})$



$h^{(1)}(\mathbf{x})$



$h^{(1)}(h^{(0)}(\mathbf{x}))$



- Inference requires calculating the marginal likelihood:

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

- Very challenging!

- **Variational DGP** (Damianou and Lawrence, 2013)
- **Sequential Inference for DGPs** (Wang et al., 2016)
- **DGP with Expectation Propagation** (Bui et al., 2016)
- **Variational Auto-Encoded DGP** (Dai et al., 2017)
- **Dropout as a Bayesian Approximation** (Gal and Gahramani, 2016)
- **Structured and Efficient Variational Deep Learning with Matrix Gaussian Posteriors** (Louizos and Welling, 2016)

Related Work

- Variational DGP (Damianou and Lawrence, 2013)
- Sequential Inference for DGPs (Wang et al., 2016)
- **DGP with Expectation Propagation** (Bui et al., 2016)
- Variational Auto-Encoded DGP (Dai et al., 2017)
- Dropout as a Bayesian Approximation (Gal and Ghahramani, 2016)
- Structured and Efficient Variational Deep Learning with Matrix Gaussian Posteriors (Louizos and Welling, 2016)

- **Variational DGP** (Damianou and Lawrence, 2013)
- **Sequential Inference for DGPs** (Wang et al., 2016)
- **DGP with Expectation Propagation** (Bui et al., 2016)
- **Variational Auto-Encoded DGP** (Dai et al., 2017)
- **Dropout as a Bayesian Approximation** (Gal and Ghahramani, 2016)
- **Structured and Efficient Variational Deep Learning with Matrix Gaussian Posteriors** (Louizos and Welling, 2016)

Can we develop better DGP models using their connection to DNNs?

1. Faster learning without compromising performance
2. Scalable to very large datasets
3. Extendable to a moderate number of layers

Can we develop better DGP models using their connection to DNNs?

1. Faster learning without compromising performance
2. Scalable to very large datasets
3. Extendable to a moderate number of layers

Can we develop better DGP models using their connection to DNNs?

1. Faster learning without compromising performance
2. Scalable to very large datasets
3. Extendable to a moderate number of layers

Can we develop better DGP models using their connection to DNNs?

1. Faster learning without compromising performance
2. Scalable to very large datasets
3. Extendable to a moderate number of layers

DGP Architecture

DGPs with Random Features

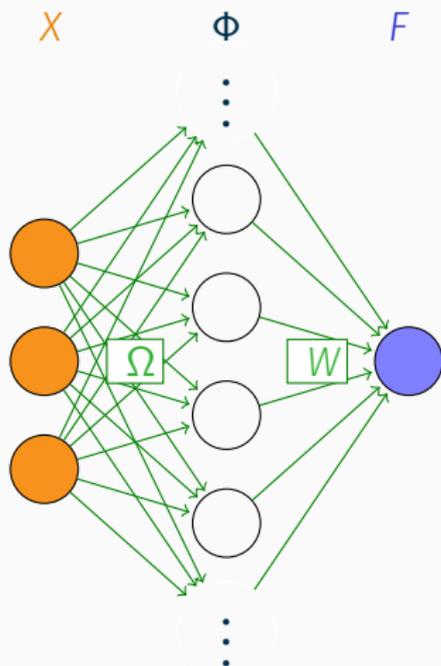
- GPs are single-layered Neural Nets with an infinite number of hidden units

- Weight-space view of a GP

$$F = \Phi W$$

- The priors over the weights are

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



Random Feature Expansion of Kernels

- The RBF kernel
- The first order Arc-Cosine kernel

Random Feature Expansion of Kernels

- The **RBF kernel** can be approximated using **trigonometric functions**

$$\Phi_{\text{RBF}} = \sqrt{\frac{\sigma^2}{N_{\text{RF}}}} [\cos(F\Omega), \sin(F\Omega)] \quad \text{with} \quad p(\Omega_j | \theta) = \mathcal{N}(\mathbf{0}, \Lambda^{-1})$$

allowing for scaling factors σ^2 and $\Lambda = \text{diag}(l_1^2, \dots, l_d^2)$

- The first order **Arc-Cosine kernel** can be approximated using **Rectified Linear Units (ReLU)**

$$\Phi_{\text{ARC}} = \sqrt{\frac{2\sigma^2}{N_{\text{RF}}}} \max(\mathbf{0}, F\Omega) \quad \text{with} \quad p(\Omega_{:,j}|\theta) = \mathcal{N}(\mathbf{0}, \Lambda^{-1})$$

Random Feature Expansion of Kernels

- The **RBF kernel** can be approximated using **trigonometric functions**

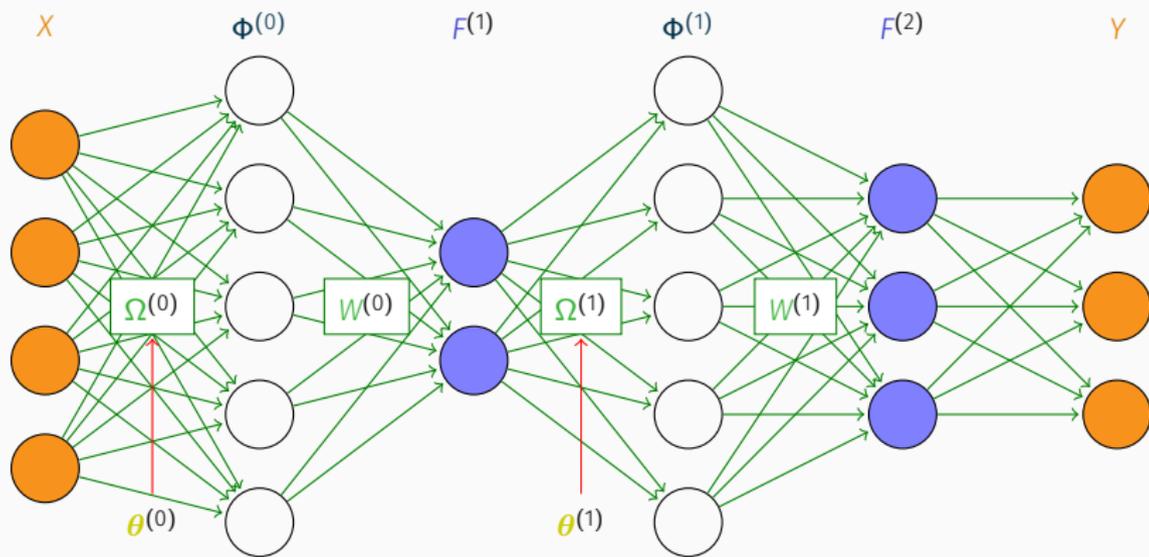
$$\Phi_{\text{RBF}} = \sqrt{\frac{\sigma^2}{N_{\text{RF}}}} [\cos(F\Omega), \sin(F\Omega)] \quad \text{with} \quad p(\Omega_j | \theta) = \mathcal{N}(\mathbf{0}, \Lambda^{-1})$$

allowing for scaling factors σ^2 and $\Lambda = \text{diag}(l_1^2, \dots, l_d^2)$

- The first order **Arc-Cosine kernel** can be approximated using **Rectified Linear Units (ReLU)**

$$\Phi_{\text{ARC}} = \sqrt{\frac{2\sigma^2}{N_{\text{RF}}}} \max(\mathbf{0}, F\Omega) \quad \text{with} \quad p(\Omega_j | \theta) = \mathcal{N}(\mathbf{0}, \Lambda^{-1})$$

DGPs with RFs become DNNs



Approximating the Marginal Likelihood

- Define $\Psi = (\Omega^{(0)}, \dots, \Omega^{(L)}, W^{(0)}, \dots, W^{(L)})$
- Evidence lower bound

$$\log [p(Y|\theta)] \geq \mathbb{E}_{q(\Psi)} (\log [p(Y|\Psi)]) - D_{\text{KL}} [q(\Psi) \| p(\Psi|\theta)]$$

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

- D_{KL} computable analytically if q and p are Gaussian!

DGPs with RFs - Stochastic variational inference

- Assume factorized likelihood:

$$p(Y|X, \Psi, \theta) = \prod_k p(y_k|x_k, \Psi, \theta)$$

- Stochastic **unbiased** estimate of the expectation term
 - **Mini-batch**

$$\mathbb{E}_{q(\Psi)} (\log [p(Y|X, \Psi, \theta)]) \approx \frac{n}{m} \sum_{k \in \mathcal{I}_m} \mathbb{E}_{q(\Psi)} (\log [p(y_k|x_k, \tilde{\Psi}, \theta)])$$

- **Monte Carlo**

$$\mathbb{E}_{q(\Psi)} (\log [p(y_k|x_k, \Psi, \theta)]) \approx \frac{1}{N_{\text{MC}}} \sum_{r=1}^{N_{\text{MC}}} \log [p(y_k|x_k, \tilde{\Psi}_r, \theta)]$$

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

DGPs with RFs - Stochastic variational inference

- Factorized approximate posterior

$$q(\Psi) = \prod_{ijl} q\left(\Omega_{ij}^{(l)}\right) \prod_{ijl} q\left(W_{ij}^{(l)}\right)$$

where

$$q\left(\Omega_{ij}^{(l)}\right) = \mathcal{N}\left(\mu_{ij}^{(l)}, (\sigma^2)_{ij}^{(l)}\right) \quad \text{and} \quad q\left(W_{ij}^{(l)}\right) = \mathcal{N}\left(m_{ij}^{(l)}, (s^2)_{ij}^{(l)}\right)$$

DGPs with RFs - Stochastic variational inference

- Factorized approximate posterior

$$q(\Psi) = \prod_{ijl} q(\Omega_{ij}^{(l)}) \prod_{ijl} q(W_{ij}^{(l)})$$

where

$$q(\Omega_{ij}^{(l)}) = \mathcal{N}(\mu_{ij}^{(l)}, (\sigma^2)_{ij}^{(l)}) \quad \text{and} \quad q(W_{ij}^{(l)}) = \mathcal{N}(m_{ij}^{(l)}, (s^2)_{ij}^{(l)})$$

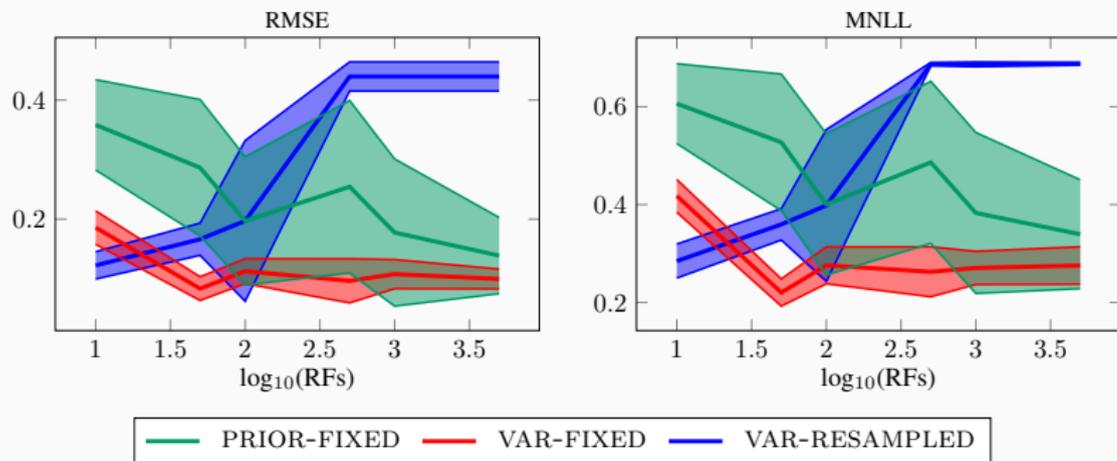
- Reparameterization trick

$$(\tilde{W}_r^{(l)})_{ij} = (s^2)_{ij}^{(l)} \epsilon_{rij}^{(l)} + m_{ij}^{(l)}$$

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

Optimization Strategies for Ω

- Random features can be **fixed**
 - PRIOR-FIXED
- or treated **variationally**
 - VAR-FIXED (with fixed randomness)
 - VAR-RESAMPLED (resampled at each iteration)

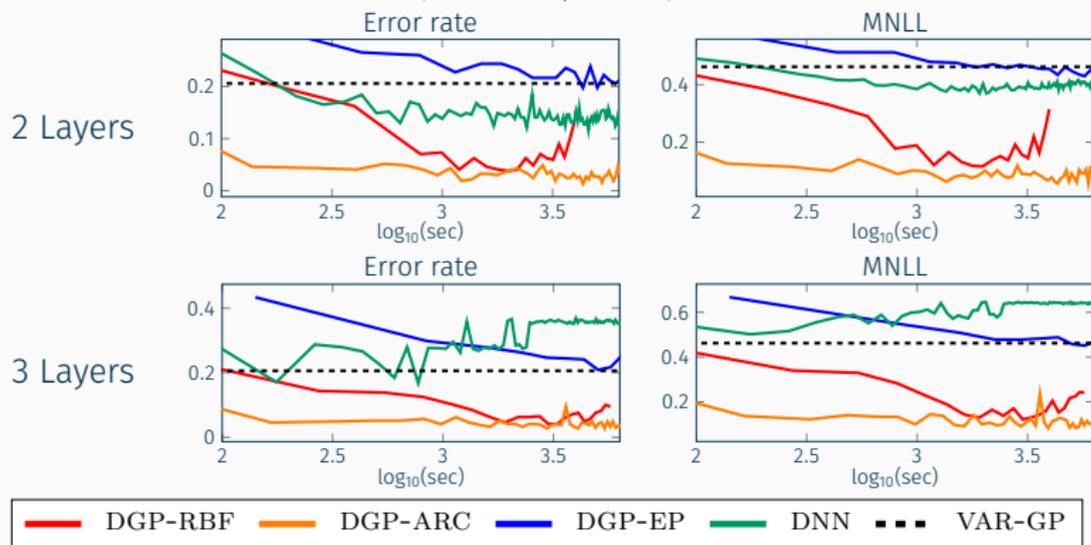


Evaluation

Model Comparison - Binary Classification

EEG Dataset

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

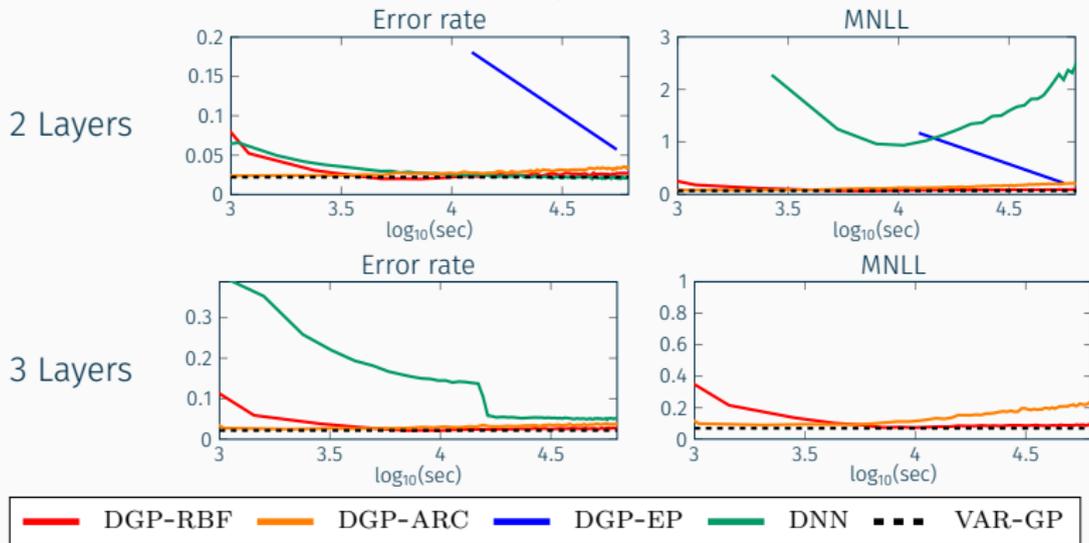


- Consistently outperforms competing techniques both in terms of **speed** and **predictive performance**

Model Comparison - Multiclass Classification

MNIST Dataset

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



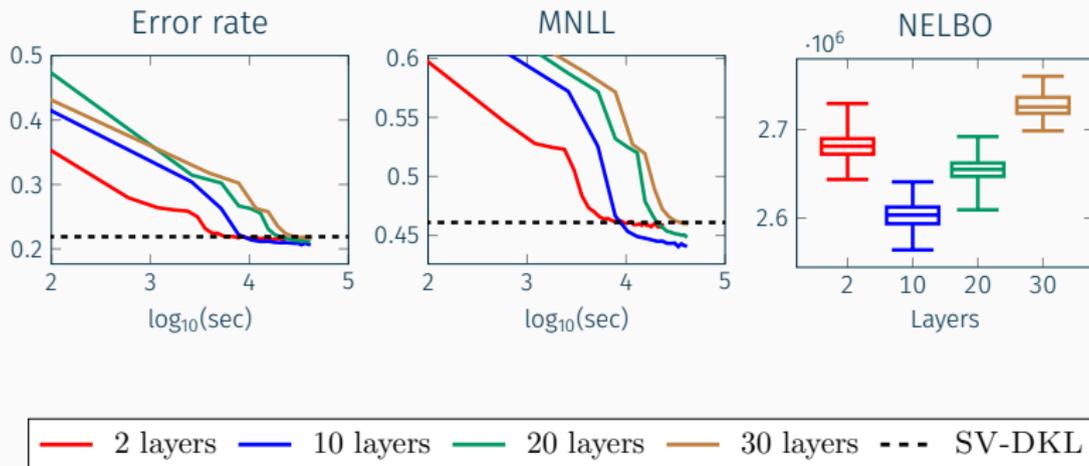
- Model performance remains resilient compared to other techniques

Scalability to Large Datasets

- Over 99% accuracy on variant of MNIST with 8.1 million images!

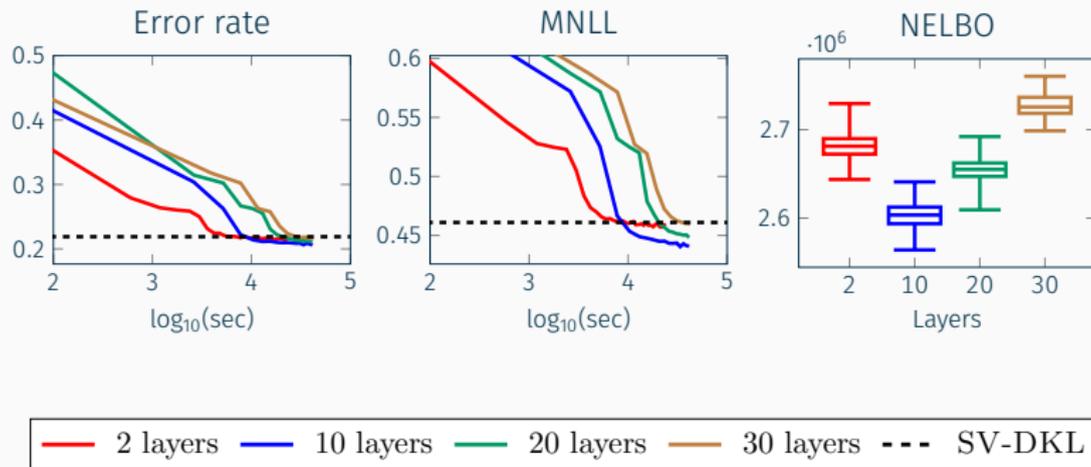
Dataset	Accuracy		MNLL	
	RBF	ARC	RBF	ARC
MNIST8M	99.14%	99.04%	0.0454	0.0465
AIRLINE	78.55%	72.76%	0.4583	0.5335

Performance of Deeper Models - Airline Dataset



- Model converges to an optimal state for every configuration
- NELBO confirmed to be a suitable criteria for model selection
- Includes feed-forward of inputs to each layer

Performance of Deeper Models - Airline Dataset



- Model converges to an optimal state for every configuration
- NELBO confirmed to be a suitable criteria for model selection
- Includes feed-forward of inputs to each layer

Conclusions

- **Contributions**

- A complete evaluation of DGPs inspired by DNNs
- Scalable and practical DGP inference without requiring Cholesky
- A study of various options for optimizing random features
- Distributed implementation using Parameter-Server framework

- **Contributions**

- A complete evaluation of DGPs inspired by DNNs
- Scalable and practical DGP inference without requiring Cholesky
- A study of various options for optimizing random features
- Distributed implementation using Parameter-Server framework

- **Ongoing work**

- Fastfood kernel and orthogonal random features
- Convolutional GP layers for complex image datasets
- Hybrid synchronous/asynchronous distributed approach

Code in TensorFlow:

`github.com/mauriziofilippone/deep_gp_random_features`

Code in TensorFlow:

github.com/mauriziofilippone/deep_gp_random_features

Random Feature Expansions for Deep Gaussian Processes

Karl Cutler*, Edwin V. Bonilla*, Pietro Michiardi*, Mauro Filippone*
* EURECOM, Sophia Antipolis, France † University of New South Wales, Australia

UNSW

Deep Gaussian Processes

- Deep generative models.
- Composition of functions.
$$f(x) = g^{(1)} \circ g^{(2)} \circ \dots \circ g^{(L)}(x)$$
- Inference requires calculating the marginal likelihood.
$$p(y|x) = \int p(y|f(x)) p(f(x)) p(x) dx$$
- Extremely challenging!

Deep GP with Random Features

- GP with random feature-based kernel with an arbitrary number of hidden nodes.
- Using a single layer case of a GP.
$$f(x) = \sum_{i=1}^M \phi_i(x) \beta_i$$
- The prior over the weights is:
$$p(\beta) = \mathcal{N}(\beta | 0, \Sigma)$$
- The RBF kernel:
$$k(x, x') = \sum_{i=1}^M \phi_i(x) \phi_i(x') = \int p(\phi(x)) p(\phi(x')) dx$$

can be approximated using independent functions:
$$k(x, x') \approx \sum_{i=1}^M \phi_i(x) \phi_i(x') \quad \text{with } \phi_i(x) = \mathcal{N}(x | \mu_i, \sigma_i^2)$$

allowing for scaling factors σ_i and μ_i along ϕ_i (from the kernel and the features).
- Moreover, the first order low CoKer kernel
$$k(x, x') = \sum_{i=1}^M \phi_i(x) \phi_i(x') = \int \text{sech}(\sigma_i(x - \mu_i)) \text{sech}(\sigma_i(x' - \mu_i)) dx$$

can be approximated using Rectified Linear Units (ReLU):
$$k(x, x') \approx \sum_{i=1}^M \max(0, \sigma_i(x - \mu_i)) \max(0, \sigma_i(x' - \mu_i))$$

Neural Architectures

- DCPs with fully connected (FC) with low rank weight matrices.

Stochastic Variational Inference

- Use evidence lower bound (ELBO):
$$\log p(y|x) \geq \mathbb{E}_{q(\beta)} [\log p(y|\beta)] - \text{KL}(q(\beta) || p(\beta))$$

where $q(\beta)$ approximates $p(\beta)$.
- Iterational approximation procedure:
$$q(\beta) = \prod_{i=1}^M q(\beta_i | \phi_i)$$

with $q(\beta_i) = \mathcal{N}(\beta_i | \mu_i^i, \sigma_i^i)$ and $q(\phi_i) = \mathcal{N}(\phi_i | \mu_i^i, \sigma_i^i)$.
- Assessing Iterational Method, we can use [redfish](#) stochastic gradient optimization.
$$\text{KL}(q(\beta) || p(\beta)) = \sum_{i=1}^M \text{KL}(q(\beta_i) || p(\beta_i))$$
- The representation can be estimated using Matrix Curls:
$$K(x, x') = \text{tr}(K(x, x') \mathbf{C}^{-1}) = \sum_{i=1}^M \text{tr}(k_i(x, x') \mathbf{C}_i^{-1})$$

with $\mathbf{C}_i = \sigma_i^2 \mathbf{I}$.
- Computationally tractable by low rank matrix multiplication - no tensor required.
- Optimization strategies for ELBO.

Experimental Setup and Results

- Comparing methods:
 - DCP with RBF kernel (DCP-RBF)
 - DCP with the order low CoKer kernel (DCP-ABC)
 - DCP with Expansion Procedure (DCP-EP)
 - Neural GP with functional features (NNGP)
- Classification, Regression, and Multi-Task.
- Early detachable architecture with options for fixed second layers.

Conclusions

- Our contributions:
 - Complete specification and evaluation of DCNs based on random features.
 - Stable and practical DCP inference via tensors.
 - Full Bayesian, hyperparameter, distributed implementation available.
- Chipping away:
 - Fast and other kernels.
 - Combinatorial DCNs (coverage pending).

References

1. Cutler et al. 2016. Deep Gaussian Processes with Random Features. *ICML*.
2. Bonilla et al. 2008. Deep Gaussian Processes. *ICML*.

Poster #126

Thank you!