

Random Feature Expansions for Deep Gaussian Processes

Maurizio Filippone

Workshop on Functional Inference and Machine Intelligence

EURECOM, Sophia Antipolis, France

February 18th, 2020

1 Introduction

2 Gaussian Processes

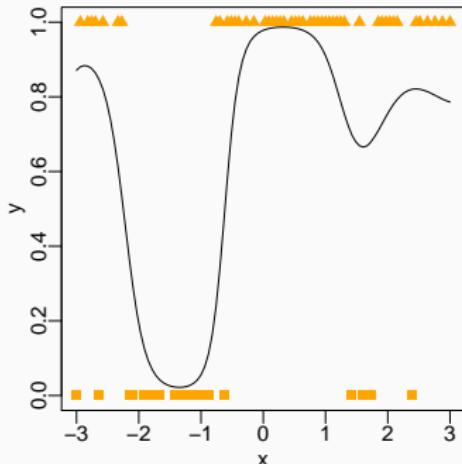
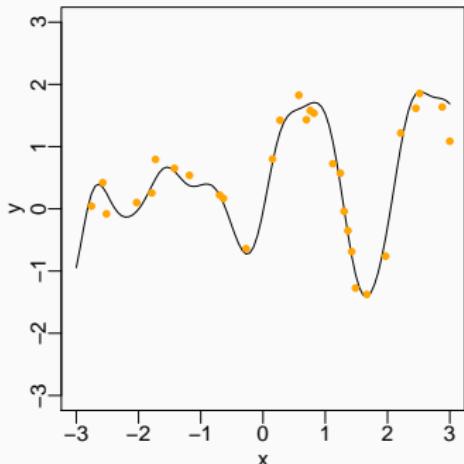
3 Deep Gaussian Processes

4 Conclusions

Introduction

Learning from Data — Function Estimation

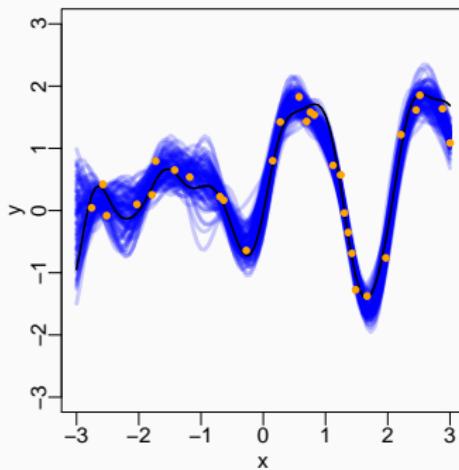
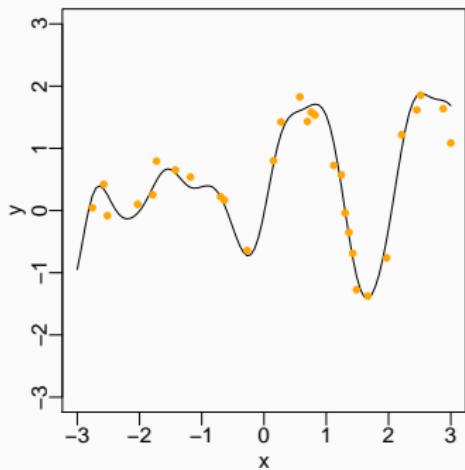
- Take these two examples



- We are interested in estimating a function $f(x)$ from data
- Most problems in Machine Learning can be cast this way!

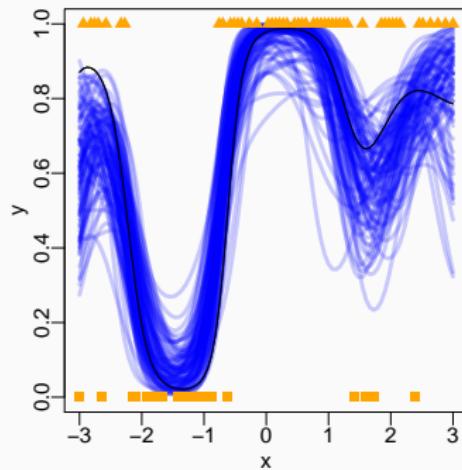
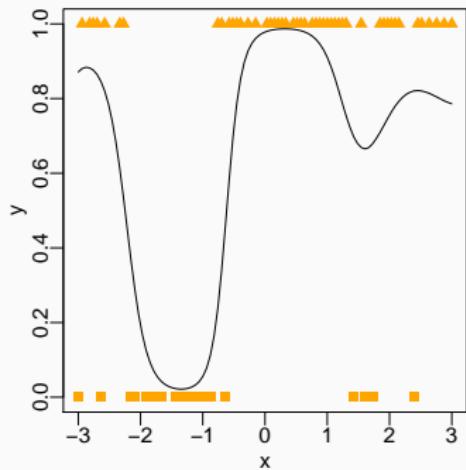
What do Bayesian Models Have to Offer?

- Regression example



What do Bayesian Models Have to Offer?

- Classification example



Linear Models

- Implement a linear combination of basis functions

$$f(\mathbf{x}) = \mathbf{w}^\top \boldsymbol{\varphi}(\mathbf{x})$$

with given

$$\boldsymbol{\varphi}(\mathbf{x}) = (\varphi_1(\mathbf{x}), \dots, \varphi_D(\mathbf{x}))^\top$$

Linear Models

- Implement a linear combination of basis functions

$$f(\mathbf{x}) = \mathbf{w}^\top \boldsymbol{\varphi}(\mathbf{x})$$

with given

$$\boldsymbol{\varphi}(\mathbf{x}) = (\varphi_1(\mathbf{x}), \dots, \varphi_D(\mathbf{x}))^\top$$

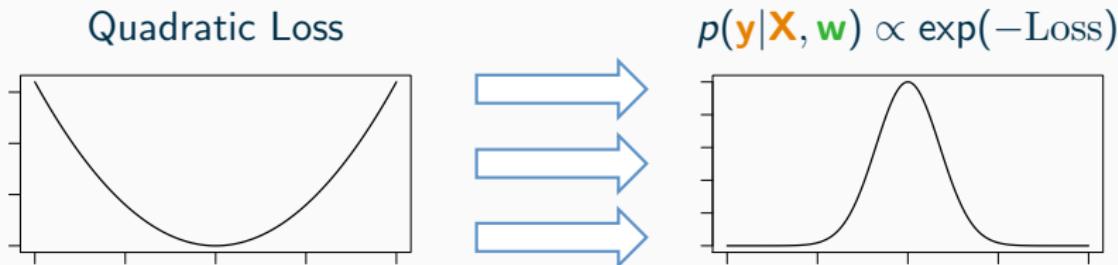
- In regression, for example, observations are modeled as:

$$p(\mathbf{y}|\mathbf{w}, \mathbf{X}, \lambda) = \mathcal{N}(\boldsymbol{\Phi}\mathbf{w}, \lambda\mathbf{I})$$

$$\boldsymbol{\Phi} := \boldsymbol{\Phi}(\mathbf{X}) = \begin{bmatrix} \varphi_1(\mathbf{x}_1) & \dots & \varphi_D(\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ \varphi_1(\mathbf{x}_N) & \dots & \varphi_D(\mathbf{x}_N) \end{bmatrix}$$

Probabilistic Interpretation of Loss Minimization

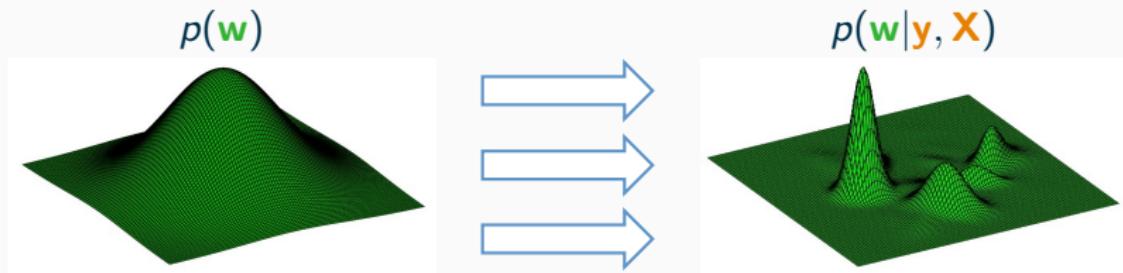
- Inputs : $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)^\top$
- Labels : $\mathbf{y} = (y_1, \dots, y_N)^\top$
- Weights : $\mathbf{w} = (w_1, \dots, w_D)^\top$



- Minimization of a loss function
- ... equivalent as maximizing likelihood $p(\mathbf{y}|\mathbf{X}, \mathbf{w})$

Bayesian Inference

- Inputs : $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)^\top$
- Labels : $\mathbf{y} = (y_1, \dots, y_N)^\top$
- Weights : $\mathbf{w} = (w_1, \dots, w_D)^\top$



$$p(\mathbf{w}|\mathbf{y}, \mathbf{X}) = \frac{p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{w})}{\int p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{w})d\mathbf{w}}$$

Gaussian Processes

Bayesian Linear Regression

- Prior $p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \mathbf{S})$ induces a prior over $\mathbf{f} = \Phi\mathbf{w}$:

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{0}, \mathbf{K})$$

where $\mathbf{K} = \text{cov}(\mathbf{f}) = E[\Phi\mathbf{w}\mathbf{w}^\top\Phi^\top] = \Phi\mathbf{S}\Phi^\top$

Bayesian Linear Regression

- Prior $p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \mathbf{S})$ induces a prior over $\mathbf{f} = \Phi\mathbf{w}$:

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{0}, \mathbf{K})$$

where $\mathbf{K} = \text{cov}(\mathbf{f}) = E[\Phi\mathbf{w}\mathbf{w}^\top\Phi^\top] = \Phi\mathbf{S}\Phi^\top$

- We can solve linear regression thinking about

$$p(\mathbf{y}|\mathbf{w}, \mathbf{X}, \lambda) = \mathcal{N}(\Phi\mathbf{w}, \lambda\mathbf{I}) \quad p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \mathbf{S})$$

or

$$p(\mathbf{y}|\mathbf{w}, \mathbf{X}, \lambda) = \mathcal{N}(\mathbf{f}, \lambda\mathbf{I}) \quad p(\mathbf{f}) = \mathcal{N}(\mathbf{0}, \mathbf{K})$$

Bayesian Linear Regression

- Prior $p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \mathbf{S})$ induces a prior over $\mathbf{f} = \Phi\mathbf{w}$:

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{0}, \mathbf{K})$$

where $\mathbf{K} = \text{cov}(\mathbf{f}) = E[\Phi\mathbf{w}\mathbf{w}^\top\Phi^\top] = \Phi\mathbf{S}\Phi^\top$

- We can solve linear regression thinking about

$$p(\mathbf{y}|\mathbf{w}, \mathbf{X}, \lambda) = \mathcal{N}(\Phi\mathbf{w}, \lambda\mathbf{I}) \quad p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \mathbf{S})$$

or

$$p(\mathbf{y}|\mathbf{w}, \mathbf{X}, \lambda) = \mathcal{N}(\mathbf{f}, \lambda\mathbf{I}) \quad p(\mathbf{f}) = \mathcal{N}(\mathbf{0}, \mathbf{K})$$

- Why is this useful??

Gaussian Processes

- Linear models require specifying a set of basis functions
 - Polynomials, Trigonometric, . . . ??
- We can use the fact that:

$$\mathbf{K} = \Phi \mathbf{S} \Phi^\top$$

to see that \mathbf{K} (actually $k(\cdot, \cdot)$) induces Φ (that is $\varphi(\cdot)$)

Gaussian Processes

- Linear models require specifying a set of basis functions
 - Polynomials, Trigonometric, . . . ??
- We can use the fact that:

$$\mathbf{K} = \Phi \mathbf{S} \Phi^\top$$

- to see that \mathbf{K} (actually $k(\cdot, \cdot)$) induces Φ (that is $\varphi(\cdot)$)
- It is possible to choose $k(\cdot, \cdot)$ such that $\varphi(\cdot)$ infinite dimensional!

- For example, it is possible to show that for

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2}\right)$$

there exists a corresponding $\varphi(\cdot)$ that is infinite dimensional!

- There are other kernels satisfying this property

Bayesian Linear Regression

- Posterior over \mathbf{w} must be Gaussian (simple exercise)

$$p(\mathbf{w}|\mathbf{X}, \mathbf{y}, \lambda) = \mathcal{N}(\boldsymbol{\mu}_{\mathbf{w}}, \boldsymbol{\Sigma}_{\mathbf{w}})$$

- Covariance:

$$\boldsymbol{\Sigma}_{\mathbf{w}} = \left(\frac{1}{\lambda} \boldsymbol{\Phi}^T \boldsymbol{\Phi} + \mathbf{S}^{-1} \right)^{-1}$$

- Mean:

$$\boldsymbol{\mu}_{\mathbf{w}} = \frac{1}{\lambda} \boldsymbol{\Sigma}_{\mathbf{w}} \boldsymbol{\Phi}^T \mathbf{y}$$

- Predictions – with a similar tedious exercise...

$$p(\mathbf{y}_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*, \lambda) = \mathcal{N} \left(\boldsymbol{\varphi}(\mathbf{x}_*)^T \boldsymbol{\mu}_{\mathbf{w}}, \lambda + \boldsymbol{\varphi}(\mathbf{x}_*)^T \boldsymbol{\Sigma}_{\mathbf{w}} \boldsymbol{\varphi}(\mathbf{x}_*) \right)$$

Bayesian Linear Regression - Thinking about \mathbf{f}

- Posterior over \mathbf{f} must be Gaussian

$$p(\mathbf{f} | \mathbf{X}, \mathbf{y}, \lambda) = \mathcal{N}(\boldsymbol{\mu}_{\mathbf{f}}, \boldsymbol{\Sigma}_{\mathbf{f}})$$

- Covariance:

$$\boldsymbol{\Sigma}_{\mathbf{f}} = \mathbf{K} - \mathbf{K}(\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{K}$$

- Mean:

$$\boldsymbol{\mu}_{\mathbf{f}} = \mathbf{K}(\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{y}$$

- Predictions – same as those for \mathbf{w} ... but in terms of $k(\cdot, \cdot)$

$$p(\mathbf{y}_* | \mathbf{X}, \mathbf{y}, \mathbf{x}_*, \lambda) = \mathcal{N} \left(\mathbf{k}_*^\top (\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{y}, \lambda + \mathbf{k}_*^\top (\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{k}_* \right)$$

with $(\mathbf{k}_*)_i = k(\mathbf{x}_i, \mathbf{x}_*)$

Bayesian Linear Regression as a Kernel Machine

Proof sketch to help deriving the previous expressions

- Woodbury identity helps in showing that Bayesian linear regression can be formulated as a kernel machine:

$$(\mathbf{A} + \mathbf{U}\mathbf{C}\mathbf{V})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{C}^{-1} + \mathbf{V}\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}\mathbf{A}^{-1}$$

$$\left[\begin{matrix} & & & & \\ & \text{orange square} & & & \\ & & \ddots & & \\ & & & \text{orange square} & \\ & & & & \ddots \\ & & & & & \text{orange square} \end{matrix} + \begin{matrix} & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{matrix} \right]^{-1}$$

$$\begin{matrix} & & & & \\ & \text{orange square} & & & \\ & & \ddots & & \\ & & & \text{orange square} & \\ & & & & \ddots \\ & & & & & \text{orange square} \end{matrix} - \begin{matrix} & & & & \\ & \text{orange square} & & & \\ & & \ddots & & \\ & & & \text{orange square} & \\ & & & & \ddots \\ & & & & & \text{orange square} \end{matrix} - \begin{matrix} & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{matrix} \left[\begin{matrix} & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{matrix} + \begin{matrix} & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{matrix} \right]^{-1} \begin{matrix} & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{matrix} \left[\begin{matrix} & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{matrix} + \begin{matrix} & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{matrix} \right]^{-1} \begin{matrix} & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{matrix} \left[\begin{matrix} & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{matrix} + \begin{matrix} & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{matrix} \right]^{-1} \begin{matrix} & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{matrix}$$

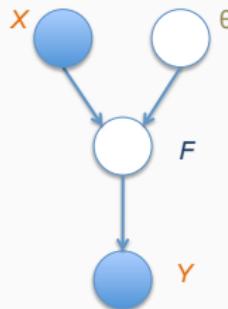
Bayesian Linear Regression as a Kernel Machine

- Working with $\varphi(\cdot)$ costs $O(D^2)$ storage, $O(D^3)$ time
- Working with $k(\cdot, \cdot)$ costs $O(N^2)$ storage, $O(N^3)$ time

Challenges and Limitations

- Kernel design $k(\cdot, \cdot | \theta)$
- GPs might be too expensive (factorize \mathbf{K})
- GP models might not even be tractable! Marginal likelihood computable under conjugacy only

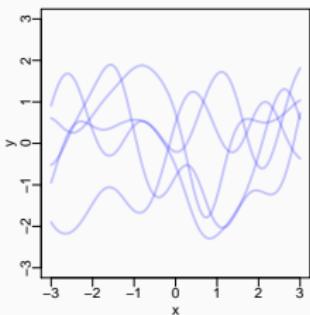
$$p(\mathbf{y}|\mathbf{X}, \theta) = \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{X}, \theta)d\mathbf{f}$$



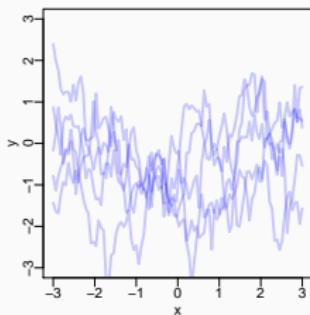
Kernel Design

- The choice of a kernel is critical for good performance
- This encodes any assumptions on the prior over functions

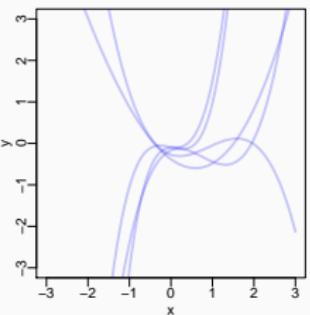
Gaussian



Matérn



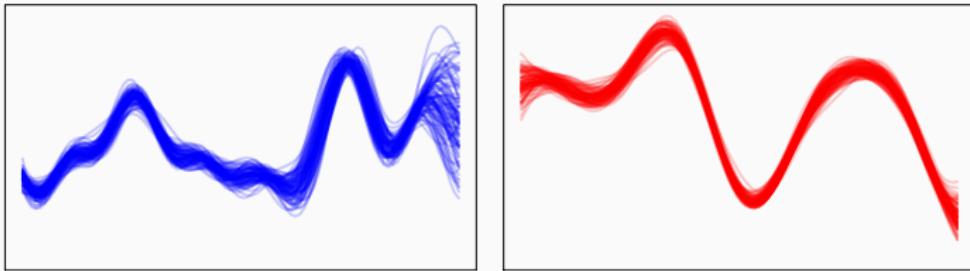
Polynomial



Deep Gaussian Processes

Deep Gaussian Processes for Large Representational Power

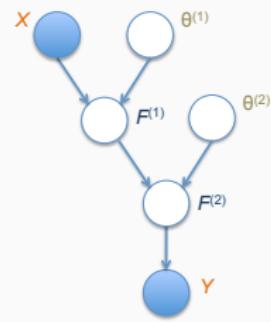
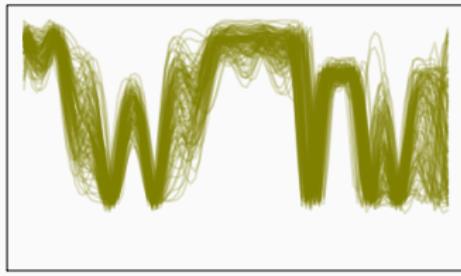
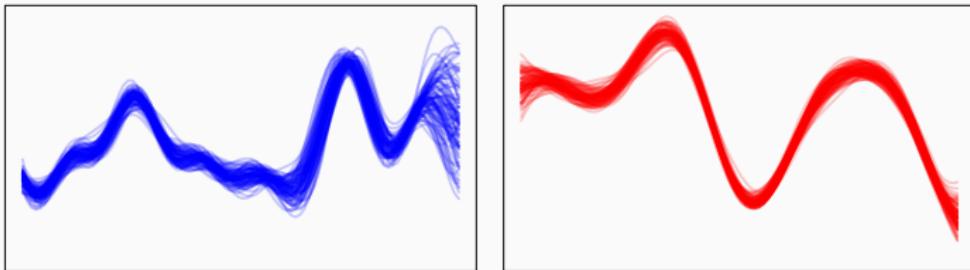
- Bypassing kernel design through **composition** of processes



$$(f \circ g)(\textcolor{orange}{x})??$$

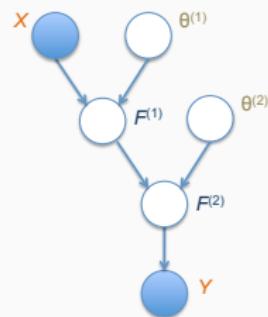
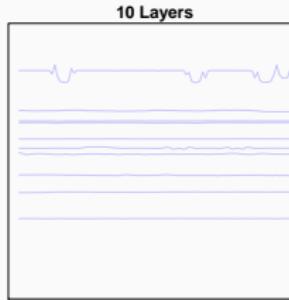
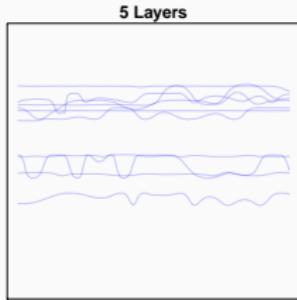
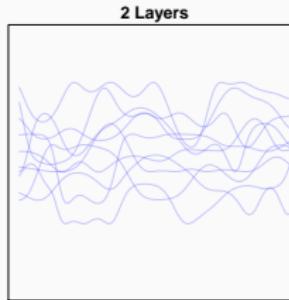
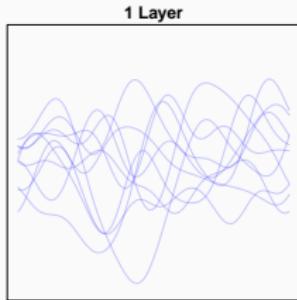
Deep Gaussian Processes for Large Representational Power

- Composition of simple processes yields something much more complex



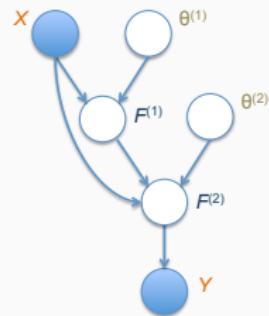
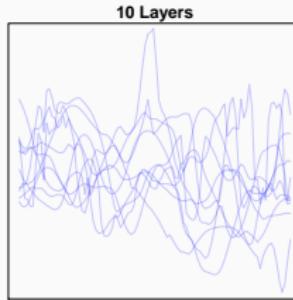
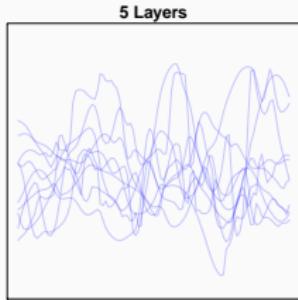
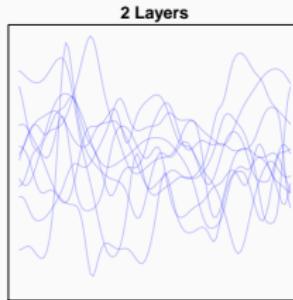
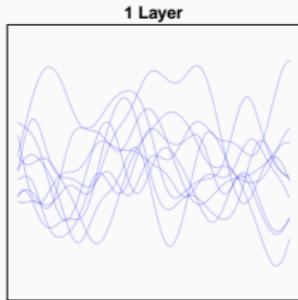
Pathologies of Deep Gaussian Processes

- Deep is not necessarily good!
- Example



Pathologies of Deep Gaussian Processes

- Deep is not necessarily good!
- Feeding input to each layer helps...



Learning Deep Gaussian Processes

- Inference requires calculating integrals of this kind:

$$\begin{aligned} p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) &= \int p\left(\mathbf{y}|\mathbf{F}^{(N_h)}, \boldsymbol{\theta}^{(N_h)}\right) \times \\ &\quad p\left(\mathbf{F}^{(N_h)}|\mathbf{F}^{(N_h-1)}, \boldsymbol{\theta}^{(N_h-1)}\right) \times \dots \times \\ &\quad p\left(\mathbf{F}^{(1)}|\mathbf{X}, \boldsymbol{\theta}^{(0)}\right) d\mathbf{F}^{(N_h)} \dots d\mathbf{F}^{(1)} \end{aligned}$$

- Extremely challenging!

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)

Is There Any Hope for GPs and DGPs?

- Mini-batch training is straightforward when objective factorizes over training points

$$\text{objective} = \sum_i g(y_i, \textcolor{orange}{x}_i, \textcolor{yellow}{\text{par}})$$

Is There Any Hope for GPs and DGPs?

- Mini-batch training is straightforward when objective factorizes over training points

$$\text{objective} = \sum_i g(y_i, \mathbf{x}_i, \text{par})$$

- In GPs latent variables are fully correlated

$$p(\mathbf{f}|\mathbf{X}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K}(\mathbf{X}, \boldsymbol{\theta})) \propto \exp\left(-\frac{1}{2}\mathbf{f}^\top \mathbf{K}^{-1} \mathbf{f}\right)$$

- Naïve mini-batch approaches would still require \mathbf{K}^{-1}

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

Random Feature Expansions for DGPs - Bochner's theorem

- Continuous shift-invariant covariance function

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

Random Feature Expansions for DGPs - Bochner's theorem

- Continuous shift-invariant covariance function

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

- Monte Carlo estimate

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

with

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

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

Random Feature Expansions for DGPs

- Define

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

and

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

- We are stacking Bayesian linear models with

$$p(\mathbf{W}_{\cdot i}^{(l)}) = \mathcal{N}(\mathbf{0}, I)$$

Random Feature Expansions for DGPs

- Define

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

and

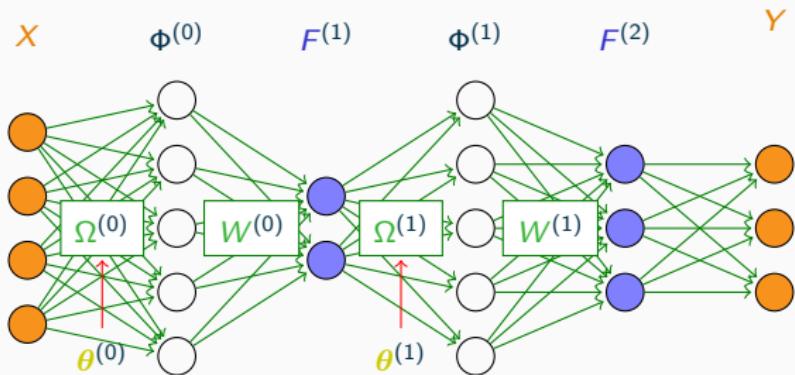
$$\mathbf{F}^{(l+1)} = \Phi^{(l)} \mathbf{W}^{(l)}$$

- We are stacking Bayesian linear models with

$$p(\mathbf{W}_{\cdot i}^{(l)}) = \mathcal{N}(\mathbf{0}, I)$$

- Expansion of arc-cosine kernel yields ReLU activations!

DGPs with random features become DNNs



We can learn the model using Stochastic Variational Inference for Bayesian DNNs!

Stochastic Variational Inference

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

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

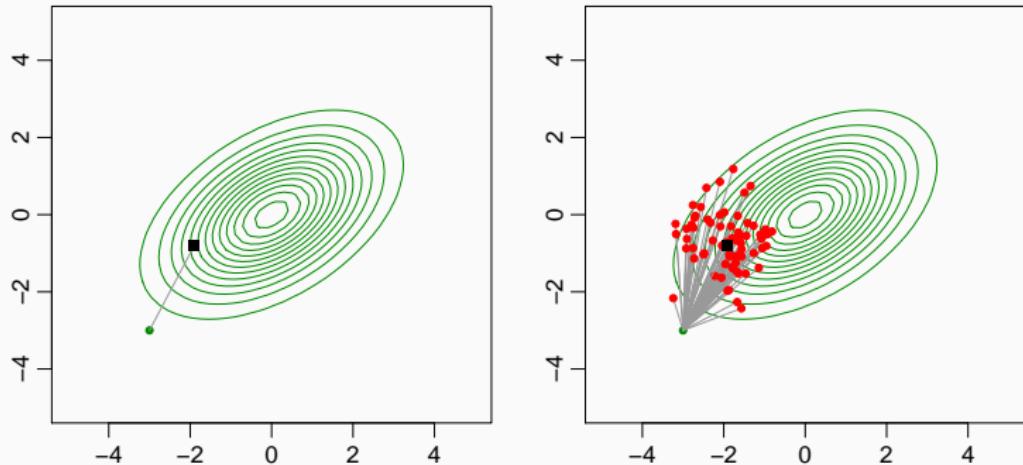
where $q(\Psi)$ approximates $p(\Psi|\mathbf{Y}, \theta)$.

- DKL computable analytically if q and p are Gaussian!

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

Stochastic Gradient Optimization

$$E \left\{ \widetilde{\nabla_{\text{vpar}}} \text{LowerBound} \right\} = \nabla_{\text{vpar}} \text{LowerBound}$$



Stochastic Variational Inference - Illustration

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

Stochastic Variational Inference

- Assume that the likelihood factorizes

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

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

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

- Monte Carlo

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

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

Stochastic variational inference

- Factorized approximate posterior:

$$q(\Psi) = \prod_{ijl} q\left(\Omega_{ij}^{(l)}\right) \prod_{ijl} q\left(W_{ij}^{(l)}\right). \quad (1)$$

with

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

$$q\left(\Omega_{ij}^{(l)}\right) = \mathcal{N}\left(m_{ij}^{(l)}, (s^2)_{ij}^{(l)}\right), \quad (3)$$

- Variational parameters

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

Stochastic Variational Inference

- Reparameterization trick

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

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

- ... same for Ω
- Variational parameters

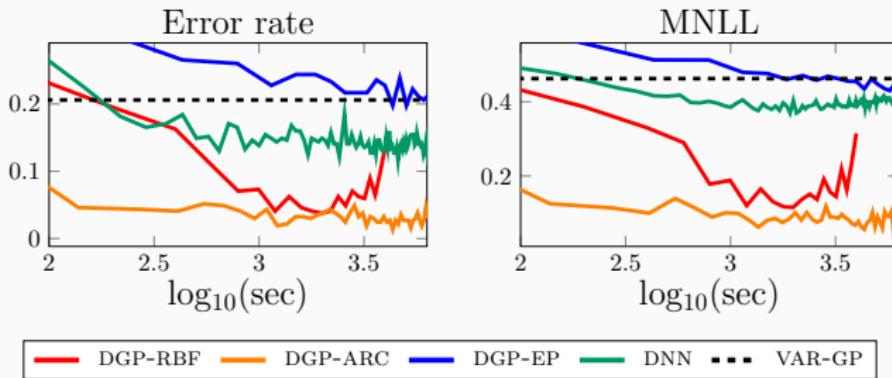
$$\mu_{ij}^{(l)}, (\sigma^2)_{ij}^{(l)} \dots$$

... and the ones for Ω

- Optimization with automatic differentiation

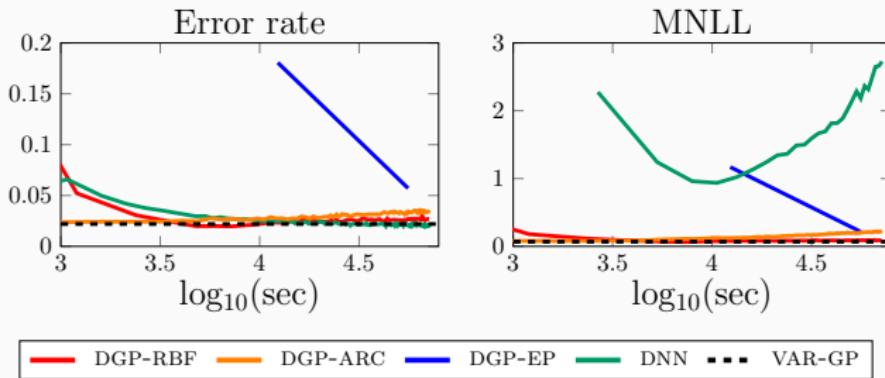
Results - Classification

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



Results - Multiclass Classification

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



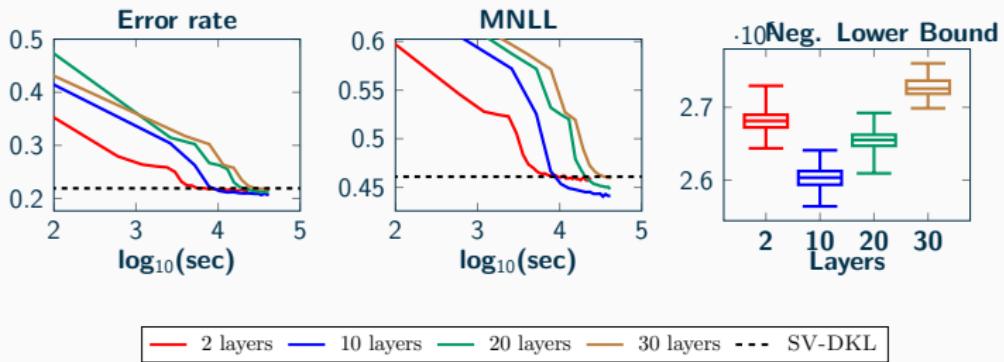
Results - MNIST-8M

- Variant of MNIST with 8.1M images
- 99+% accuracy!
- Also, check out Krauth et al., UAI 2017

Results - Model (Depth) Selection

Airline dataset

($n = 5M+$, $d = 8$)



Other Approximations for DGPs

Gaussian posterior approximations for GPs:

$$p(\mathbf{f}|\mathbf{X}, \mathbf{y}, \boldsymbol{\theta}) \approx q(\mathbf{f})$$

- Laplace Approximation – Barber and Williams, *IEEE TPAMI*, 1998
- Expectation Propagation – Seeger, *Tech. Rep.*, 2002

$$p(\mathbf{f}|\mathbf{X}, \mathbf{y}, \boldsymbol{\theta}) \propto p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{X}, \boldsymbol{\theta}) \approx p(\mathbf{f}|\mathbf{X}, \boldsymbol{\theta}) \prod_i q(\mathbf{y}_i|\mathbf{f}_i)$$

- Variational Bayes – Opper and Archambeau, *Natural Comput.*, 2009

$$\text{KL}[q(\mathbf{f})||p(\mathbf{f}|\mathbf{X}, \mathbf{y}, \boldsymbol{\theta})]$$

- MCMC – Filippone et al., *Mach. Learn.*, 2013

Other Approximations for DGPs

- Sparse GP (Nyström) approximations - introduce $M < N$ new latent variables \mathbf{u} at locations \mathbf{Z} .

$$\mathbf{K} \approx \mathbf{K}_{\mathbf{x}\mathbf{z}} \mathbf{K}_{\mathbf{z}\mathbf{z}}^{-1} \mathbf{K}_{\mathbf{z}\mathbf{x}}$$

- Fully Independent Training Conditionals (FITC)
- Partially Independent Training Conditionals (PITC)
- Titsias, *AISTATS*, 2009 – Variationally Sparse GPs

$$q(\mathbf{f}, \mathbf{u}) \approx p(\mathbf{f}|\mathbf{u})q(\mathbf{u})$$

- Random Feature Expansions
- Kronecker/Toepliz/Tensor structured GPs

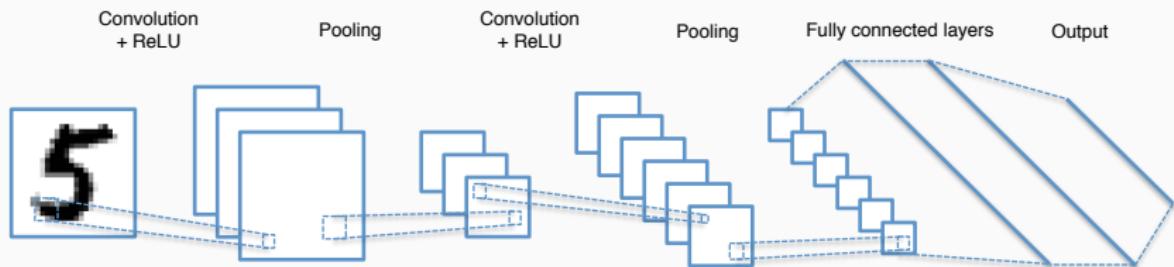
Candela and Rasmussen, *JMLR*, 2005 – Titsias, *AISTATS*, 2009 – Lázaro-Gredilla et al., *JMLR*, 2010 – Wilson and Nickisch., *ICML*, 2015 – Izmailov et al., *AISTATS*, 2018

Inference for DGPs

- Inducing points-based approximations
 - VI+Titsias *AISTATS* 2009 Sparse GP
 - Damianou and Lawrence, *AISTATS*, 2013
 - Hensman and Lawrence, *arXiv*, 2014
 - Salimbeni and Deisenroth, *NIPS*, 2017
 - EP+FITC - Bui et al. *ICML*, 2016
 - MCMC+Titsias *AISTATS* 2009 Sparse GP
 - Havasi et al., *NIPS*, 2018
- VI+Random feature-based approximations
 - Gal and Ghahramani, *ICML* 2016
 - Cutajar et al., *ICML* 2017

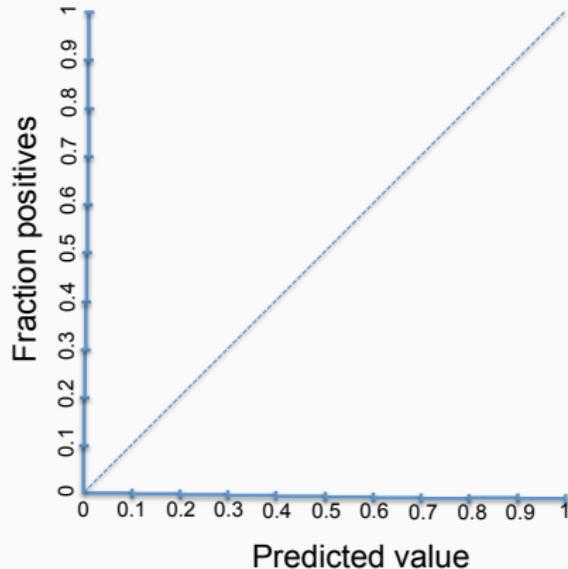
Convolutional Nets

- Convolutional nets are widely used...
- ...but they are known to be overconfident!



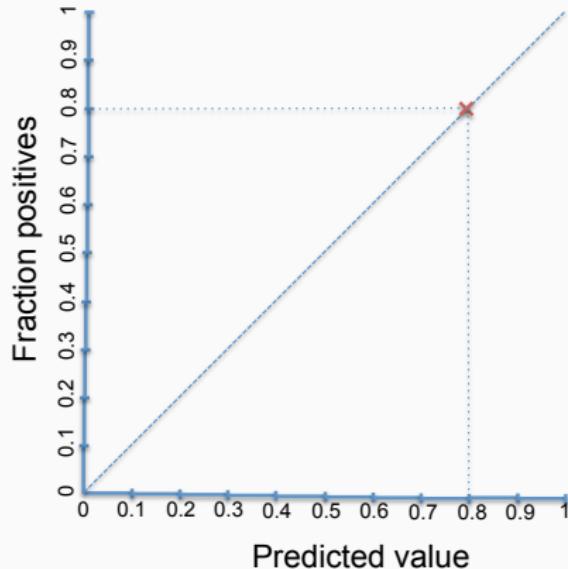
Calibration as a Measure of Quantification of Uncertainty

- Reliability diagrams



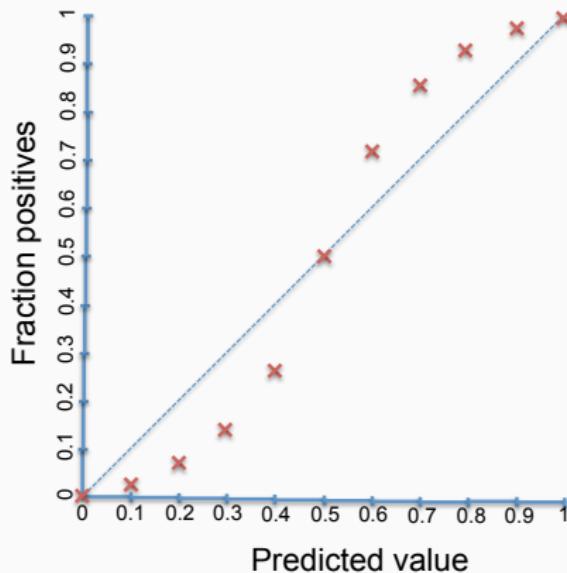
Calibration as a Measure of Quantification of Uncertainty

- Reliability diagrams



Calibration as a Measure of Quantification of Uncertainty

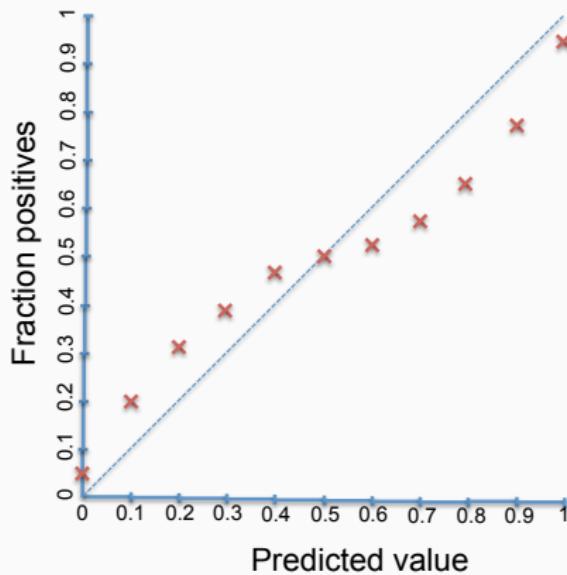
- Reliability diagrams - Under-confident predictions



- We can extract the Expected Calibration Error (ECE) score
- The BRIER score is another measure of calibration

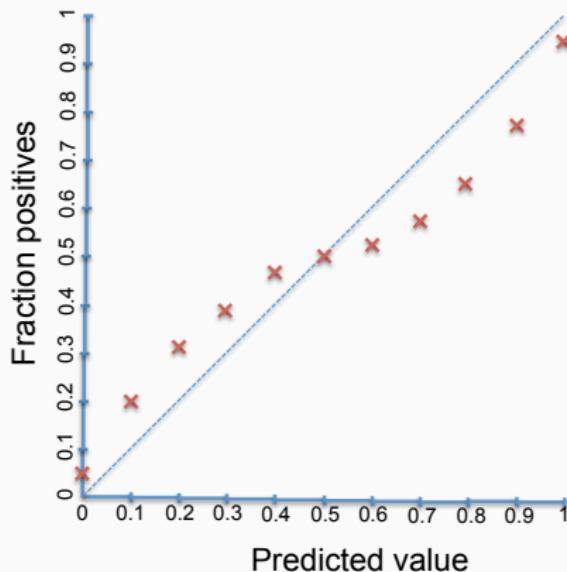
Calibration as a Measure of Quantification of Uncertainty

- Reliability diagrams - Overconfident predictions



Calibration as a Measure of Quantification of Uncertainty

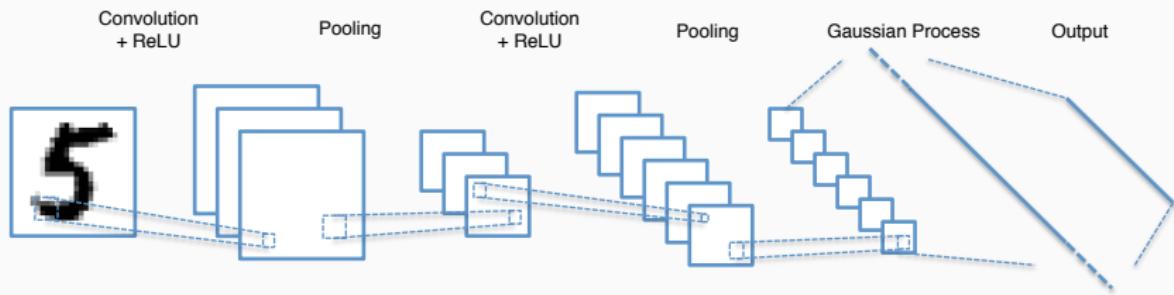
- Reliability diagrams - Overconfident predictions



Reliability diagrams of modern Deep CNNs look like this!
Post-calibration fixes it

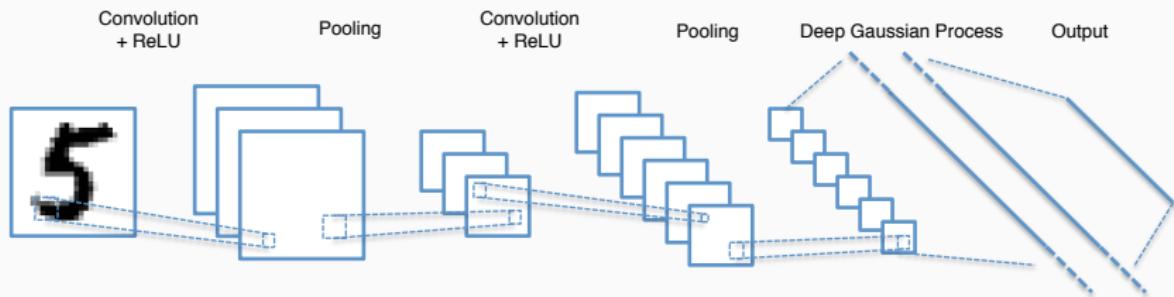
Combining Convolutional Nets with GPs

- There have been attempts to combine CNNs with GPs
- Most popular ones replace fully connected layers with GPs



Combining Convolutional Nets with GPs

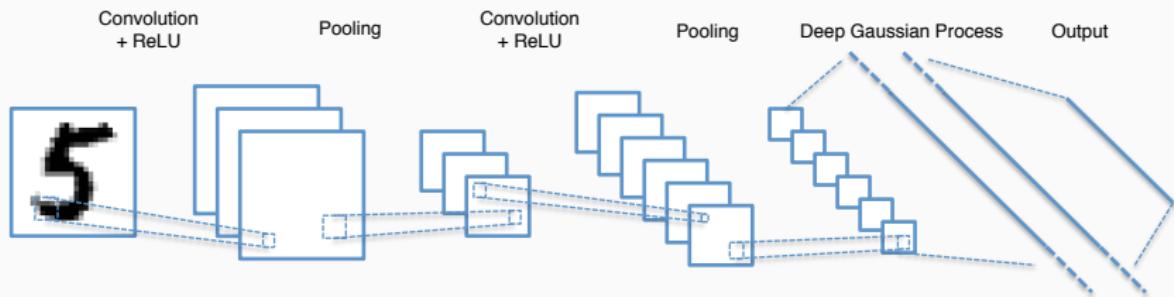
- There have been attempts to combine CNNs with GPs
- Most popular ones replace fully connected layers with GPs



- Better quantification of uncertainty??

Combining Convolutional Nets with GPs

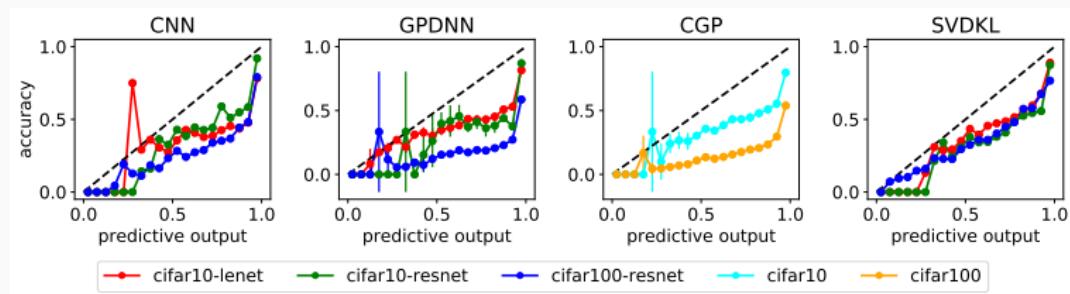
- There have been attempts to combine CNNs with GPs
- Most popular ones replace fully connected layers with GPs



- Better quantification of uncertainty?? **NO!**

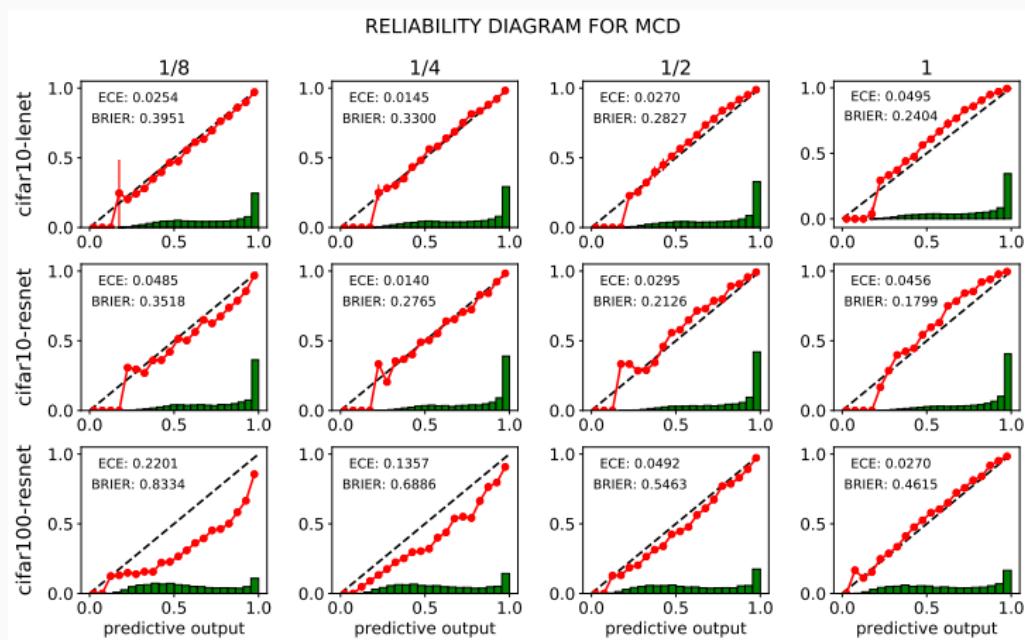
Existing Combinations of CNNs and GPs

- Convolutional Neural Nets - CNN
- Hybrid GPs and DNNs - GPDNN
- Stochastic Variational Deep Kernel Learning - SVDKL
- Convolutional GP - CGP



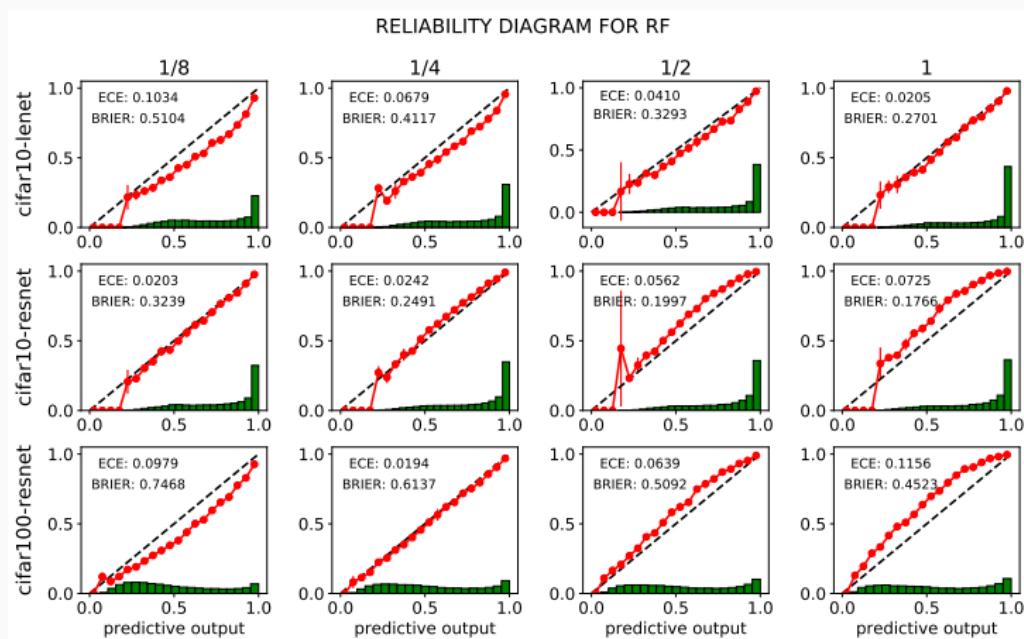
Bayesian CNNs are calibrated

- Inferring parameters of convolutional filter recovers calibration
- Example with Monte Carlo Dropout

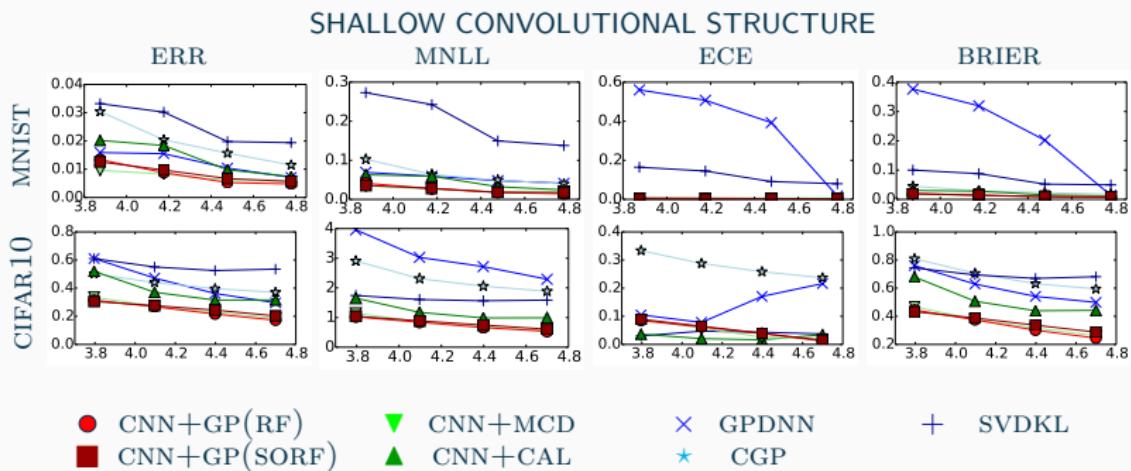


Bayesian CNNs with DGPs with Random Features

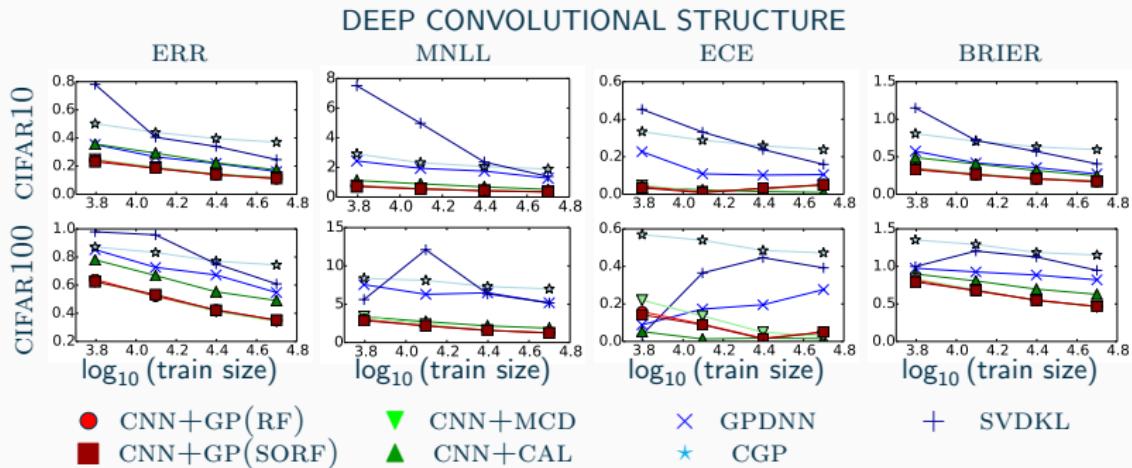
- We extended our work on Random Feature Expansions for DGPs to replace fully connected layers



Comparison with competitors

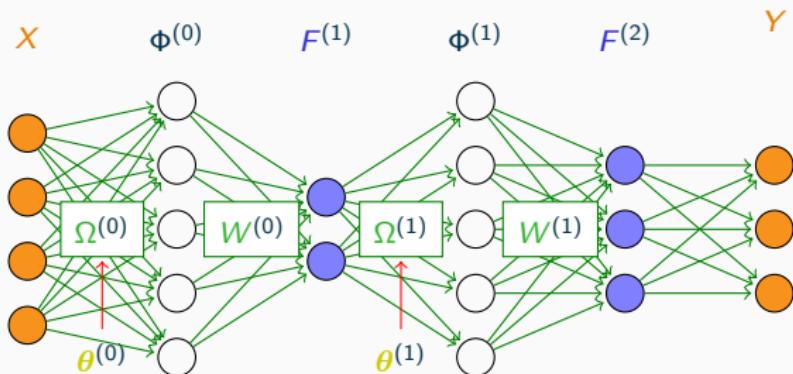


Comparison with competitors



Use Kernel Methods to Improve Bayesian Deep Learning

DGPs with random feature expansions



- Intermediate layers are based on multiplications with random iid normal matrices

$$\Phi^{(l)} = g \left(\mathbf{F}^{(l)} \boldsymbol{\Omega}^{(l)} \right)$$

Use Kernel Methods to Improve Bayesian Deep Learning

- Replace $\mathbf{F}^{(l)} \Omega^{(l)}$ with multiplication with pseudo-random matrices

$$\Omega^{(l)} \approx \mathbf{H} \mathbf{D}_1^{(l)} \mathbf{H} \mathbf{D}_2^{(l)} \mathbf{H} \mathbf{D}_3^{(l)}$$

- This reduces:
 - time complexity from $\mathcal{O}(D^2)$ to $\mathcal{O}(D \log D)$
 - space complexity from $\mathcal{O}(D^2)$ to $\mathcal{O}(D)$

**What about a Bayesian DNN where the posterior over
matrices of weights is parameterized in this way?**

Other Interesting Recent DGP Works

- Autoencoders Dai et al. *ICLR*, 2015 – Domingues et al., *Mach. Learn.*, 2018

Other Interesting Recent DGP Works

- Autoencoders Dai et al. *ICLR*, 2015 – Domingues et al., *Mach. Learn.*, 2018
- DGPs with constrained dynamics Lorenzi and Filippone, *ICML*, 2018

Other Interesting Recent DGP Works

- Autoencoders Dai et al. *ICLR*, 2015 – Domingues et al., *Mach. Learn.*, 2018
- DGPs with constrained dynamics Lorenzi and Filippone, *ICML*, 2018
- DGPs for computer simulators Marmin and Filippone, *arXiv*, 2018

Other Interesting Recent DGP Works

- Autoencoders Dai et al., *ICLR*, 2015 – Domingues et al., *Mach. Learn.*, 2018
- DGPs with constrained dynamics Lorenzi and Filippone, *ICML*, 2018
- DGPs for computer simulators Marmin and Filippone, *arXiv*, 2018
- GPs and DGPs on Optical Processing Units Ohana et al., *ICASSP*, 2020

Other Interesting Recent DGP Works

- Autoencoders Dai et al., *ICLR*, 2015 – Domingues et al., *Mach. Learn.*, 2018
- DGPs with constrained dynamics Lorenzi and Filippone, *ICML*, 2018
- DGPs for computer simulators Marmin and Filippone, *arXiv*, 2018
- GPs and DGPs on Optical Processing Units Ohana et al., *ICASSP*, 2020
- Deep Convolutional GPs Blomqvist et al., *arXiv*, 2018
- Importance Weighted VI for DGPs Salimbeni et al., *ICML*, 2019

Conclusions

Conclusions

- Why DGPs?
 - Sensible priors for Bayesian deep learning
 - Improve our understanding of Bayesian deep learning

Conclusions

- Why DGPs?
 - Sensible priors for Bayesian deep learning
 - Improve our understanding of Bayesian deep learning
- Inference for DGPs is hard
 - Model approximations
 - Approximate inference
- Difficult to assess the impact of these approximations

Conclusions

- We are borrowing/mixing ideas from GPs and deep learning
 - Stochastic-based approximate inference
 - Low-rank process decompositions
 - Algebraic/computational tricks

Conclusions

- Combinations of GPs with CNNs slightly disappointing
 - Quantification of uncertainty not for free...
 - ... regularization of filters is necessary
 - Performance gains are small compared to plain CNNs

Acknowledgments

Thank you!

