Bayesian neural networks: a function space view tour

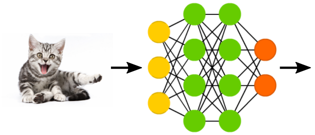
Yingzhen Li

Microsoft Research Cambridge

Neural networks 101

Let's say we want to classify different types of cats

- $\boldsymbol{x}$: input images; $\boldsymbol{y}$: output label
- build a neural network (with param. W):
  $p(\boldsymbol{y}|\boldsymbol{x}, W) = \text{softmax}(f_W(\boldsymbol{x}))$



A typical neural network:

$$f_W(\boldsymbol{x}) = W_L \phi(W_{L-1}\phi(...\phi(W_1\boldsymbol{x} + b_1)) + b_{L-1}) + b_L$$

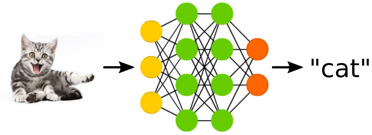
$$\text{for the } l^{th} \text{ layer: } \boldsymbol{h}_l = \phi(W_l \boldsymbol{h}_{l-1} + b_l), \quad \boldsymbol{h}_1 = \phi(W_1\boldsymbol{x} + b_1)$$

Parameters: $W = \{W_1, b_1, ..., W_L, b_L\}$; nonlinearity: $\phi(\cdot)$

Neural networks 101

Let's say we want to classify different types of cats

- $\boldsymbol{x}$: input images; $\boldsymbol{y}$: output label
- build a neural network (with param. W):
  $p(\boldsymbol{y}|\boldsymbol{x}, W) = \text{softmax}(f_W(\boldsymbol{x}))$



Typical deep learning solution:

Training the neural network weights:

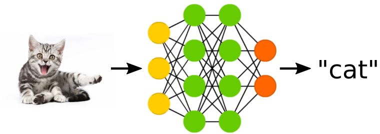
- Maximum likelihood estimation (MLE) given a dataset $\mathcal{D} = \{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{n=1}^N$:

$$W^* = \arg\min \sum_{n=1}^{N} \log p(\boldsymbol{y}_n|\boldsymbol{x}_n, W)$$

Bayesian neural networks 101

Let's say we want to classify different types of cats

- $\boldsymbol{x}$: input images; $\boldsymbol{y}$: output label
- build a neural network (with param. W):
  $p(\boldsymbol{y}|\boldsymbol{x}, W) = \mathrm{softmax}(f_W(\boldsymbol{x}))$



A Bayesian solution:

Put a prior distribution $p(W)$ over W

- compute posterior $p(W|\mathcal{D})$ given a dataset $\mathcal{D} = \{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{n=1}^N$:

$$p(W|\mathcal{D}) \propto p(W) \prod_{n=1}^{N} p(\boldsymbol{y}_n|\boldsymbol{x}_n, W)$$

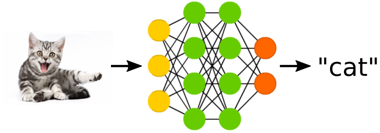
- Bayesian predictive inference:

$$p(\boldsymbol{y}^*|\boldsymbol{x}^*, \mathcal{D}) = \mathbb{E}_{p(W|\mathcal{D})}[p(\boldsymbol{y}^*|\boldsymbol{x}^*, W)]$$

Bayesian neural networks 101

Let's say we want to classify different types of cats

- $\boldsymbol{x}$: input images; $\boldsymbol{y}$: output label
- build a neural network (with param. W): $p(\boldsymbol{y}|\boldsymbol{x}, W) = \mathrm{softmax}(f_W(\boldsymbol{x}))$



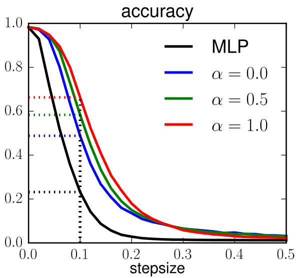
In practice: $p(W|\mathcal{D})$ is intractable

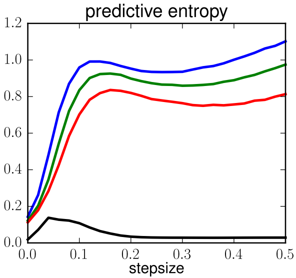
- First find approximation $q(W) \approx p(W|\mathcal{D})$
- In prediction, do Monte Carlo sampling:

$$p(\boldsymbol{y}^*|\boldsymbol{x}^*, \mathcal{D}) \approx \frac{1}{K}\sum_{k=1}^{K} p(\boldsymbol{y}^*|\boldsymbol{x}^*, W^k), \quad W^k \sim q(W)$$

Applications of Bayesian neural networks

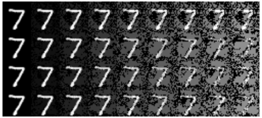
Detecting adversarial examples:





MLP	$\alpha = 0.0$	$\alpha = 0.5$	$\alpha = 1.0$
0.2318	0.4879	0.5832	0.663

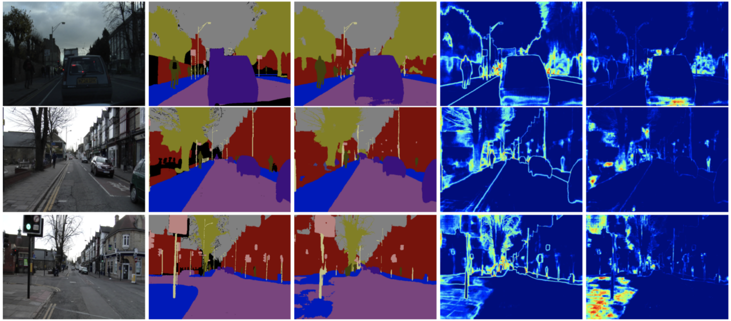
accuracy for stepsize=0.1



Li and Gal 2017

Applications of Bayesian neural networks

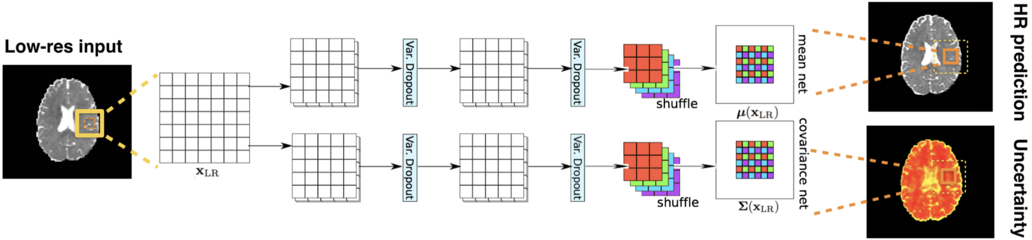
Image segmentation



(a) Input Image (b) Ground Truth (c) Semantic Segmentation (d) Aleatoric Uncertainty (e) Epistemic Uncertainty

Kendall and Gal 2017

Medical imaging (super resolution):



Tanno et al. 2019

Bayesian neural networks vs Gaussian processes

Why learning about BNNs in a summer school about GPs?

- mean-field BNNs have GP limits
- approximate inference on GPs has links to BNNs
- approximate inference on BNNs can leverage GP techniques

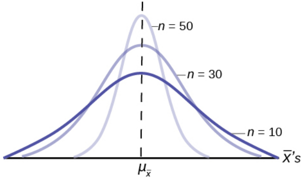
BNN $\rightarrow$ GP

Quick refresher: Central limit theorem

Theorem

Let $\mathbf{x}_1, \ldots, \mathbf{x}_N$ be i.i.d. samples from $p(\mathbf{x})$ and $p(\mathbf{x})$ has mean μ and covariance Σ, then

$$\frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n \xrightarrow{d} \mathcal{N}\left(\mu, \frac{1}{N}\Sigma\right), \quad N \to +\infty$$



Bayesian neural networks $\rightarrow$ Gaussian process [1]

Consider one hidden layer BNN with mean-field prior and bounded non-linearity

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} v_m \phi(\boldsymbol{w}_m^T \boldsymbol{x} + b_m),$$

$$W = \{W_1, \boldsymbol{b}, W_2\}, \quad W_1 = [\boldsymbol{w}_1, ..., \boldsymbol{w}_m]^T, \quad \boldsymbol{b} = [b_1, ..., b_m], \quad W_2 = [v_1, ..., v_m],$$

mean-field prior

$$p(W) = p(W_1)p(\boldsymbol{b})p(W_2), \quad p(W_1) = \prod_m p(\boldsymbol{w}_m), \quad p(\boldsymbol{b}) = \prod_m p(b_m), \quad p(W_2) = \prod_m p(v_m),$$

the same prior for each connection weight/bias:

$$p(\boldsymbol{w}_i) = p(\boldsymbol{w}_j), \quad p(b_i) = p(b_j), \quad p(v_i) = p(v_j), \quad \forall i, j$$

[1] Radford Neal's derivation in his PhD thesis (1994)

Bayesian neural networks $\to$ Gaussian process [1]

Consider one hidden layer BNN with mean-field prior and bounded non-linearity

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} v_m \phi(\boldsymbol{w}_m^T \boldsymbol{x} + b_m),$$

the same prior for each connection weight/bias:

$$p(\boldsymbol{w}_i) = p(\boldsymbol{w}_j), \quad p(b_i) = p(b_j), \quad \forall i, j$$

$\Rightarrow$ the same distribution of the hidden unit outputs:

$$h_i(\boldsymbol{x}) \perp h_j(\boldsymbol{x}), \quad h_i(\boldsymbol{x}) \stackrel{d}{=} h_j(\boldsymbol{x}), \quad h_i(\boldsymbol{x}) = \phi(\boldsymbol{w}_i^T \boldsymbol{x} + b_i)$$

$\Rightarrow$ i.e. $h_1(\boldsymbol{x}), ..., h_M(\boldsymbol{x})$ are i.i.d. samples from some implicitly defined distribution

[1] Radford Neal's derivation in his PhD thesis (1994)

Bayesian neural networks → Gaussian process [1]

Consider one hidden layer BNN with mean-field prior and bounded non-linearity

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} v_m \phi(\boldsymbol{w}_m^T \boldsymbol{x} + b_m),$$

mean-field prior with the same distribution for second layer connection weights:

$$v_i \perp W_1, \boldsymbol{b}, \qquad p(v_i) = p(v_j), \quad \forall i, j$$

$$\Rightarrow v_i h_i(\boldsymbol{x}) \perp v_j h_j(\boldsymbol{x}), \quad v_i h_i(\boldsymbol{x}) \stackrel{d}{=} v_j h_j(\boldsymbol{x})$$

so $f(\boldsymbol{x})$ is a sum of i.i.d. random variables

[1] Radford Neal's derivation in his PhD thesis (1994)

Bayesian neural networks → Gaussian process [1]

Consider one hidden layer BNN with mean-field prior and bounded non-linearity

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} v_m \phi(\boldsymbol{w}_m^T \boldsymbol{x} + b_m),$$

if we make $\mathbb{E}[v_m] = 0$ and $\mathbb{V}[v_m] = \sigma_v^2$ scale as $\mathcal{O}(1/M)$:

$$\mathbb{E}[f(\boldsymbol{x})] = \sum_{m=1}^{M} \mathbb{E}[v_m]\mathbb{E}[h_m(\boldsymbol{x})] = 0$$

$$\mathbb{V}[f(\boldsymbol{x})] = \sum_{m=1}^{M} \mathbb{V}[v_m h_m(\boldsymbol{x})] = \sum_{m=1}^{M} \sigma_v^2 \mathbb{E}[h_m(\boldsymbol{x})^2] \to \sigma_v^2 \mathbb{E}[h(\boldsymbol{x})^2]$$

[1] Radford Neal's derivation in his PhD thesis (1994)

Bayesian neural networks $\to$ Gaussian process [1]

Consider one hidden layer BNN with mean-field prior and bounded non-linearity

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} v_m \phi(\boldsymbol{w}_m^T \boldsymbol{x} + b_m),$$

if we make $\mathbb{E}[v_m] = 0$ and $\mathbb{V}[v_m] = \sigma_v^2$ scale as $\mathcal{O}(1/M)$:

$$\mathrm{Cov}[f(\boldsymbol{x}), f(\boldsymbol{x}')] = \sum_{m=1}^{M} \sigma_v^2 \mathbb{E}[h_m(\boldsymbol{x}) h_m(\boldsymbol{x}')] \to \sigma_v^2 \mathbb{E}[h(\boldsymbol{x}) h(\boldsymbol{x}')]$$

[1] Radford Neal's derivation in his PhD thesis (1994)

Bayesian neural networks $\to$ Gaussian process [1]

Consider one hidden layer BNN with mean-field prior and bounded non-linearity

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} v_m \phi(\boldsymbol{w}_m^T \boldsymbol{x} + b_m),$$

if we make $\mathbb{E}[v_m] = 0$ and $\mathbb{V}[v_m] = \sigma_v^2$ scale as $\mathcal{O}(1/M)$:

$$(f(\boldsymbol{x}), f(\boldsymbol{x}')) \xrightarrow{d} \mathcal{N}(\mathbf{0}, K), \quad K(\boldsymbol{x}, \boldsymbol{x}') = \sigma_v^2 \mathbb{E}[h(\boldsymbol{x})h(\boldsymbol{x}')] \qquad \text{(CLT)}$$

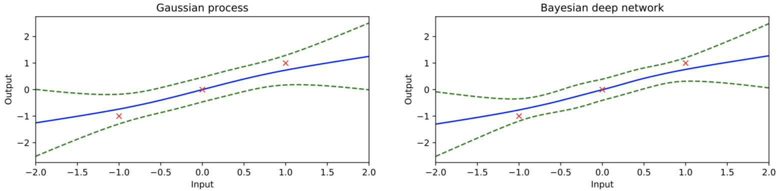
it holds for any $\boldsymbol{x}, \boldsymbol{x}' \Rightarrow f \sim \mathcal{GP}(0, K(\boldsymbol{x}, \boldsymbol{x}'))$

[1] Radford Neal's derivation in his PhD thesis (1994)

Bayesian neural networks $\rightarrow$ Gaussian process

Recent extensions of Radford Neal's result:

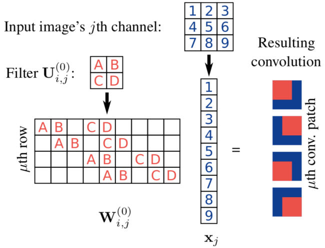
- deep and wide BNNs have GP limits
  - mean-field prior over weights
  - the activation function satisfies $|\phi(x)| \leq c + A|x|$
  - hidden layer widths strictly increasing to infinity



Matthews et al. 2018, Lee et al. 2018

Recent extensions of Radford Neal's result:

- Bayesian CNNs have GP limits
  - Convolution in CNN = fully connected layer applied to different locations in the image
  - # channels in CNN = # hidden units in fully connected NN



Garriga-Alonso et al. 2019, Novak et al. 2019

GP $\rightarrow$ BNN

Exact GP inference can be very expensive:

predictive inference for GP regression:

$$p(\boldsymbol{f}_*|\mathbf{X}_*, \mathbf{X}, \boldsymbol{y}) = \mathcal{N}(\boldsymbol{f}_*; \mathbf{K}_{*n}(\mathbf{K}_{nn} + \sigma^2\mathbf{I})^{-1}\boldsymbol{y}, \mathbf{K}_{**} - \mathbf{K}_{*n}(\mathbf{K}_{nn} + \sigma^2\mathbf{I})^{-1}\mathbf{K}_{n*})$$

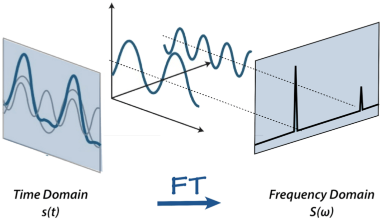
$$(\mathbf{K}_{nn})_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j), \quad \mathbf{K}_{nn} \in \mathbb{R}^{N \times N}$$

Inverting $\mathbf{K}_{nn} + \sigma^2\mathbf{I}$ has $\mathcal{O}(N^3)$ cost!

Quick refresher: Fourier (inverse) transform

$$S(w) = \int s(t)e^{-itw}\,dt$$

$$s(t) = \int S(w)e^{itw}\,dw$$



Bochner's theorem: (Fourier inverse transform)

Theorem

A (properly scaled) translation invariant kernel $K(\boldsymbol{x}, \boldsymbol{x}') = K(\boldsymbol{x} - \boldsymbol{x}')$ *can be represented as*

$$K(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}_{p(\boldsymbol{w})}\left[\sigma^2 e^{i\boldsymbol{w}^T(\boldsymbol{x}-\boldsymbol{x}')}\right]$$

for some distribution $p(\boldsymbol{w})$.

- Real value kernel $\Rightarrow \mathbb{E}_{p(\boldsymbol{w})}\left[\sigma^2 e^{i\boldsymbol{w}^T(\boldsymbol{x}-\boldsymbol{x}')}\right] = \mathbb{E}_{p(\boldsymbol{w})}\left[\sigma^2 cos(\boldsymbol{w}^T(\boldsymbol{x}-\boldsymbol{x}'))\right]$
- $cos(x - x') = 2\mathbb{E}_{p(b)}[cos(x + b)cos(x' + b)], \quad p(b) = \text{Uniform}[0, 2\pi]$

Rahimi and Recht 2007

Bochner's theorem: (Fourier inverse transform)

Theorem

A (properly scaled) translation invariant kernel $K(\boldsymbol{x}, \boldsymbol{x}') = K(\boldsymbol{x} - \boldsymbol{x}')$ *can be represented as*

$$K(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}_{p(\boldsymbol{w})p(\boldsymbol{b})}\left[\sigma^2 cos(\boldsymbol{w}^T\boldsymbol{x} + b)cos(\boldsymbol{w}^T\boldsymbol{x}' + b)\right]$$

for some distribution $p(\boldsymbol{w})$ *and* $p(b) = Uniform[0, 2\pi]$.

- Real value kernel $\Rightarrow \mathbb{E}_{p(\boldsymbol{w})}\left[\sigma^2 e^{i\boldsymbol{w}^T(\boldsymbol{x}-\boldsymbol{x}')}\right] = \mathbb{E}_{p(\boldsymbol{w})}\left[\sigma^2 cos(\boldsymbol{w}^T(\boldsymbol{x} - \boldsymbol{x}'))\right]$
- $cos(x - x') = 2\mathbb{E}_{p(b)}[cos(x + b)cos(x' + b)], \quad p(b) = \text{Uniform}[0, 2\pi]$

Rahimi and Recht 2007

Bochner's theorem: (Fourier inverse transform)

Theorem

A (properly scaled) translation invariant kernel $K(\boldsymbol{x}, \boldsymbol{x}') = K(\boldsymbol{x} - \boldsymbol{x}')$ *can be represented as*

$$K(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}_{p(\boldsymbol{w})p(\boldsymbol{b})}\left[\sigma^2 cos(\boldsymbol{w}^T\boldsymbol{x} + b) cos(\boldsymbol{w}^T\boldsymbol{x}' + b)\right]$$

for some distribution $p(\boldsymbol{w})$ *and* $p(b) = Uniform[0, 2\pi]$.

- Monte Carlo approximation:

$$K(\boldsymbol{x}, \boldsymbol{x}') \approx \tilde{K}(\boldsymbol{x}, \boldsymbol{x}') = \frac{\sigma^2}{M}\sum_{m=1}^{M} cos(\boldsymbol{w}_m^T\boldsymbol{x} + b_m) cos(\boldsymbol{w}_m^T\boldsymbol{x}' + b_m), \quad \boldsymbol{w}_m, b_m \sim p(\boldsymbol{w})p(b_m)$$

Rahimi and Recht 2007

Bochner's theorem: (Fourier inverse transform)

Theorem

A (properly scaled) translation invariant kernel $K(\mathbf{x}, \mathbf{x}') = K(\mathbf{x} - \mathbf{x}')$ *can be represented as*

$$K(\mathbf{x}, \mathbf{x}') = \mathbb{E}_{p(\mathbf{w})p(\mathbf{b})}\left[\sigma^2 cos(\mathbf{w}^T\mathbf{x} + b) cos(\mathbf{w}^T\mathbf{x}' + b)\right]$$

for some distribution $p(\mathbf{w})$ *and* $p(b) = Uniform[0, 2\pi]$.

- Monte Carlo approximation: Define $\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}), ..., h_M(\mathbf{x})], \quad h_m(\mathbf{x}) = cos(\mathbf{w}_m^T\mathbf{x} + b_m), \quad \mathbf{w}_m \sim p(\mathbf{w}), b_m \sim p(b)$

$$\Rightarrow \tilde{K}(\mathbf{x}, \mathbf{x}') = \frac{\sigma^2}{M}\mathbf{h}(\mathbf{x})^T\mathbf{h}(\mathbf{x}')$$

Rahimi and Recht 2007

Approximating the GP kernel with random feature expansions:

$$f \sim \mathcal{GP}(0, K(\boldsymbol{x}, \boldsymbol{x}')), \quad f \approx \tilde{f}, \quad \tilde{f} \sim \mathcal{GP}(0, \tilde{K}(\boldsymbol{x}, \boldsymbol{x}')), \quad \tilde{K}(\boldsymbol{x}, \boldsymbol{x}') = \frac{\sigma^2}{M} \boldsymbol{h}(\boldsymbol{x})^T \boldsymbol{h}(\boldsymbol{x}')$$

Weight space view ⇒ single hidden layer BNN:

$$\tilde{f} \sim \mathcal{GP}(0, \tilde{K}(\boldsymbol{x}, \boldsymbol{x}')) \quad \Leftrightarrow \quad \tilde{f}(\boldsymbol{x}) = \boldsymbol{v}^T \boldsymbol{h}(\boldsymbol{x}), \quad \boldsymbol{v} \sim p(\boldsymbol{v}) = \mathcal{N}(\boldsymbol{0}, \frac{\sigma^2}{M} \mathbf{I})$$

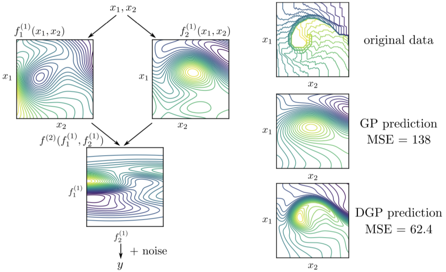
Adding number of components (increase M) → adding hidden units in BNNs

Deep GPs → deep BNNs with bottleneck layers:

Deep Gaussian process:

$$f(\boldsymbol{x}) = f^{(L)} \circ f^{(L-1)} \circ \cdots \circ f^{(0)}(\boldsymbol{x}),$$

$$f^{(i)} \sim \mathcal{GP}(0, K^{(i)}(\boldsymbol{x}, \boldsymbol{x}'))$$



Bui et al. 2016

Recall weight space view: $\tilde{K}(\boldsymbol{x}, \boldsymbol{x}') \approx K(\boldsymbol{x}, \boldsymbol{x}')$

$$\tilde{f} \sim \mathcal{GP}(0, \tilde{K}(\boldsymbol{x}, \boldsymbol{x}')) \quad \Leftrightarrow \quad \tilde{f}(\boldsymbol{x}) = \boldsymbol{v}^T cos(W\boldsymbol{x} + \boldsymbol{b}) \quad W, \boldsymbol{b}, \boldsymbol{v} \sim p(W)p(\boldsymbol{b})p(\boldsymbol{v})$$

Deep GPs $\rightarrow$ deep BNNs with bottleneck layers:

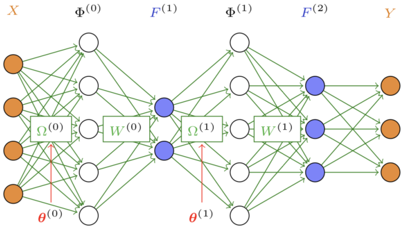
Deep BNN approximation to deep GP:

$$\tilde{f} \approx f, \quad \tilde{f}(\boldsymbol{x}) = \tilde{f}^{(L)} \circ \tilde{f}^{(L-1)} \circ \cdots \circ \tilde{f}^{(0)}(\boldsymbol{x}),$$

$$\tilde{f}^{(i)}(\boldsymbol{x}) = \boldsymbol{v}_i^T cos(W_i \boldsymbol{x} + \boldsymbol{b}_i),$$

$$W_i, \boldsymbol{b}_i, \boldsymbol{v}_i \sim p(W_i)p(\boldsymbol{b}_i)p(\boldsymbol{v}_i),$$

$$\prod_{n=1}^{N} p(\boldsymbol{y}_n | f(\boldsymbol{x}_n))p(\boldsymbol{f}) \approx \prod_{n=1}^{N} p(\boldsymbol{y}_n | \boldsymbol{x}_n, W)p(W)$$



Cutajar et al. 2017

Gaussian process → Bayesian neural networks

Deep GPs → deep BNNs with bottleneck layers:

Approx. infer. for deep GP: random feature expansion + approx. infer. for BNNs:

$$p_{\mathrm{DGP}}(\boldsymbol{y}^*|\boldsymbol{x}^*, \mathcal{D}) \approx p_{\mathrm{BNN}}(\boldsymbol{y}^*|\boldsymbol{x}^*, \mathcal{D}) \approx \frac{1}{K}\sum_{k=1}^{K} p_{\mathrm{BNN}}(\boldsymbol{y}^*|\boldsymbol{x}^*, W^k), \quad W^k \sim q^*(W)$$

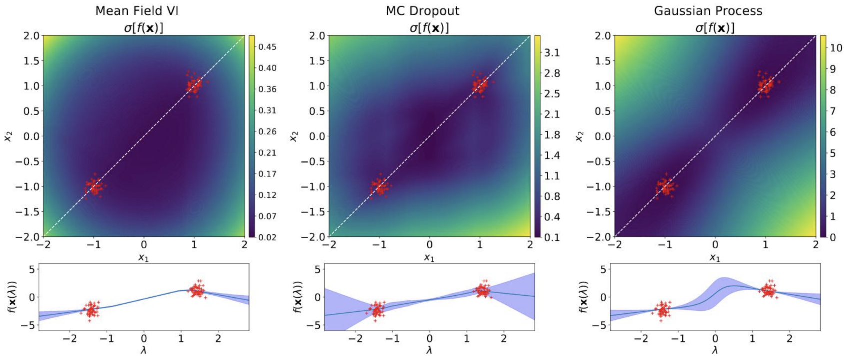
$q^*(W)$ obtained by e.g. variational inference:

$$q^*(W) = \arg\min_{q(W)} \mathbb{E}_{q(W)}\left[\sum_{n=1}^{N} \log p_{\mathrm{BNN}}(\boldsymbol{y}^*|\boldsymbol{x}^*, W)\right] - \mathrm{KL}[q(W)||p(W)]$$

Cutajar et al. 2017

BNN function-space inference

BNN inference in function space?



- weight space approximations can be inefficient
- how to do function space inference for BNNs?

Ma et al. 2019, Foong et al. 2019

Definition: An implicit stochastic process (IP) is a collection of random variables $f(\cdot)$, such that any finite collection $\mathbf{f} = (f(\mathbf{x}_1), ..., f(\mathbf{x}_N))^\top$ has joint distribution implicitly defined by the following generative process:

$$\mathbf{z} \sim p(\mathbf{z}), \quad f(\mathbf{x}_n) = g_\theta(\mathbf{x}_n, \mathbf{z}), \quad \forall\ \mathbf{x}_n \in \mathbf{X}.$$

A function distributed according to the above IP is denoted as $f(\cdot) \sim \mathcal{IP}(g_\theta(\cdot, \cdot), p_{\mathbf{z}})$.

Implicit Stochastic Processes

$\mathbf{z}$ can be finite or infinite dimensional:

- Finite dimensional $\mathbf{z}$:
  prove via Kolmogorov extension theorem
  (marginalisation consistency & permutation invariance)

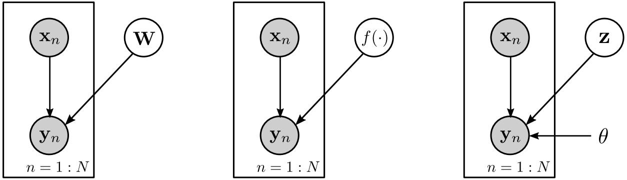
$\mathbf{z}$ can be finite or infinite dimensional:

- Finite dimensional $\mathbf{z}$:
  prove via Kolmogorov extension theorem
  (marginalisation consistency & permutation invariance)
- Infinite dimensional case (here $\mathbf{z} = z(\cdot)$ is a random function):
  sufficient conditions:
  - $z(\cdot) \sim \mathcal{SP}(0, C(\cdot, \cdot))$ is a centered stochastic process on $\mathcal{L}^2(\mathbb{R}^d)$
  - $g(\mathbf{x}, z) = \phi(\int_{\mathbf{x}} \sum_{m=0}^{M} K_m(\mathbf{x}, \mathbf{x}')z(\mathbf{x}')d\mathbf{x}'),\ K_m \in \mathcal{L}^2(\mathbb{R}^d \times \mathbb{R}^d),\ |\phi(x)| \leq A|x|$

  Then $f(\cdot)$ is also a stochastic process.
  Proof: apply Karhunen-Loeve expansion and check convergence in $\mathcal{L}^2(\mathbb{R}^d)$.

Implicit Stochastic Processes

Examples:



Bayesian NN   warped GP   neural sampler

Also include many simulators in physics, ecology, climate science...

Implicit Process Regression

Implicit process regression model:

$$f(\cdot) \sim \mathcal{IP}(g_\theta(\cdot,\cdot), p_{\mathbf{z}}), \ y = f(\mathbf{x}) + \epsilon, \ \epsilon \sim \mathcal{N}(0, \sigma^2).$$

- Similar to GP regression, given dataset $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$, we hope to compute

$$p(\mathbf{f}|\mathbf{X}, \mathbf{y}) \propto p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{X})$$

- Then for predictive inference, compute

$$p(y^*|\mathbf{x}^*, \mathcal{D}) = \int p(y^*|f^*)p(f^*|\mathbf{X}, \mathbf{y})df^*$$

intractable due to the unknown distribution $p(\mathbf{f})$ (cannot use variational inference directly)

Variational Implicit Processes

Generalised wake-sleep applied to implicit processes

- **Sleep phase:** approximate $p_\theta(\mathbf{y}, \mathbf{f}|\mathbf{X}) \approx q(\mathbf{y}, \mathbf{f}|\mathbf{X})$
- **Wake phase:** approximate $\log p_\theta(\mathbf{y}|\mathbf{X}) \approx \log q(\mathbf{y}|\mathbf{X})$ then maximise w.r.t θ
- large-scale learning: spectral approximations lead to a Bayesian linear regression problem

Dayan et al. 1995

Variational Implicit Processes

Sleep phase:

- Define $q_{\mathcal{GP}}(\mathbf{y}, \mathbf{f}|\mathbf{X}) = q(\mathbf{y}|\mathbf{f})q_{\mathcal{GP}}(\mathbf{f}|\mathbf{X})$, $\underbrace{q(\mathbf{y}|\mathbf{f}) = p(\mathbf{y}|\mathbf{f})}_{\text{same likelihood term}}$
- for *any* $\mathbf{X}$, use $(\mathbf{y}, \mathbf{f}) \sim p(\mathbf{y}, \mathbf{f}|\mathbf{X})$ as targets to train q:

$$\min_q \mathrm{D}_{\mathrm{KL}}[p(\mathbf{y}, \mathbf{f}|\mathbf{X})||q_{\mathcal{GP}}(\mathbf{y}, \mathbf{f}|\mathbf{X})]$$

- Reduce to matching mean & covariance functions (with finite function samples):

$$m^\star_{\mathrm{MLE}}(\mathbf{x}) = \frac{1}{S}\sum_s f_s(\mathbf{x}), \quad \mathcal{K}^\star_{\mathrm{MLE}}(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{S}\sum_s \Delta_s(\mathbf{x}_1)\Delta_s(\mathbf{x}_2),$$

$$\Delta_s(\mathbf{x}) = f_s(\mathbf{x}) - m^\star_{\mathrm{MLE}}(\mathbf{x}), \quad f_s(\cdot) \sim \mathcal{IP}(g_\theta(\cdot,\cdot), p_{\mathbf{z}}).$$

$q^\star_{\mathcal{GP}}(\mathbf{f}|\mathbf{X}, m^\star_{\mathrm{MLE}}, \mathcal{K}^\star_{\mathrm{MLE}}, \theta)$ depends on θ

Variational Implicit Processes

Wake phase:

- We want to maximise $\log p_\theta(\mathbf{y}|\mathbf{X})$ w.r.t. θ (intractable)
- Note that in sleep step we are minimising joint KL and

$$\mathrm{D}_{\mathsf{KL}}[p(\mathbf{y},\mathbf{f}|\mathbf{X})||q_{\mathcal{GP}}(\mathbf{y},\mathbf{f}|\mathbf{X})] \geq \mathrm{D}_{\mathsf{KL}}[p(\mathbf{y}|\mathbf{X})||q_{\mathcal{GP}}(\mathbf{y}|\mathbf{X})]$$

- Then we use $\log q^{\star}_{\mathcal{GP}}(\mathbf{y}|\mathbf{X},\theta) \approx \log p_\theta(\mathbf{y}|\mathbf{X})$
- Note that $q^{\star}_{\mathcal{GP}}(\mathbf{y}|\mathbf{X},\theta)$ depends on $\theta \Rightarrow$ just differentiate through

Variational Implicit Processes

Wake phase:

For large dataset GP inference is very expensive ($\mathcal{O}(N^3)$)

Recall the kernel structure

$$\mathcal{K}^{\star}_{\mathrm{MLE}}(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{S}\sum_s \Delta_s(\mathbf{x}_1)\Delta_s(\mathbf{x}_2)$$

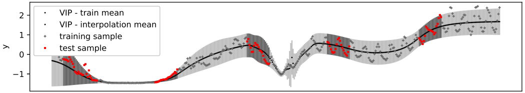
Random feature approximation:

$$\log q^{\star}_{\mathcal{GP}}(\mathbf{y}|\mathbf{X}, \theta) \approx \log \int \prod_n q^{\star}(y_n|\mathbf{x}_n, \mathbf{a}, \theta)p(\mathbf{a})d\mathbf{a},$$

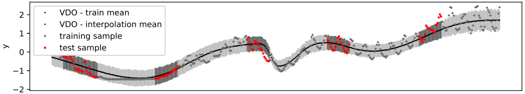
$$q^{\star}(y_n|\mathbf{x}_n, \mathbf{a}, \theta) = \mathcal{N}\left(y_n; m^{\star}_{\mathrm{MLE}}(\mathbf{x}_n) + \frac{1}{\sqrt{S}}\sum_s \Delta_s(\mathbf{x}_n)a_s, \sigma^2\right), \quad p(\mathbf{a}) = \mathcal{N}(\mathbf{a}; 0, \mathbf{I}),$$

Bayesian linear regression (BLR) on top of function samples

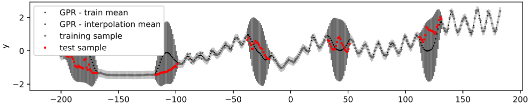
Some Experimental Results



(a) VIP-BNN



(b) Variational dropout (VDO-BNN)

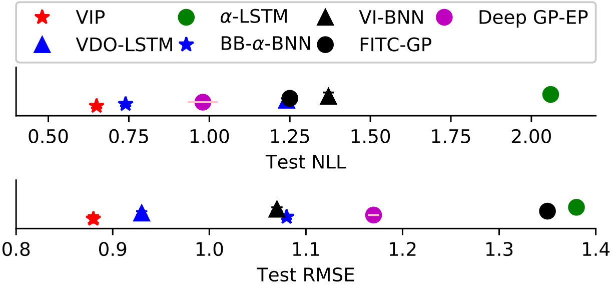


(c) GP regression (GPR)

Solar irradiance prediction:

- methods: VIP, VDO, GPR
- Capturing the predictive mean: VIP $>$ GPR;
- Uncertainty estimates: VIP $>$ VDO;

Some Experimental Results



VIP applied to Bayesian LSTM:

- CEP Data: >1 million datapoints, each $\mathbf{x}$ is a string represneting a molecule;
- Goal: predict power conversion efficiency
- Baselines: (deep) GP, BNN (hand-crafted features) & Bayesian LSTM (directly raw features), with different inference methods;
- VIP works significantly better for both NLL and RMSE.

What we have covered today...

BNNs and GPs are good friends:

- mean-field BNNs have GP limits
- approximate inference on GPs has links to BNNs
- approximate inference on BNNs can leverage GP techniques



Thank you!

References

Neal 1994. Bayesian Learning for Neural Networks. PhD thesis

Dayan et al. 1995. The Helmholtz machine. Neural Computation, 1995.

Rahimi and Recht 2007. Random Features for Large-Scale Kernel Machines. NeurIPS 2007

Lázaro-Gredilla et al. 2010. Sparse spectrum Gaussian process regression. JMLR 2010

Bui et al. 2016. Deep Gaussian Processes for Regression using Approximate Expectation Propagation. ICML 2016

Li and Gal 2016. Dropout inference in Bayesian neural networks with alpha-divergences. ICML 2017

Kendall and Gal 2017. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? NeurIPS 2017

Cutajar et al. 2017. Random Feature Expansions for Deep Gaussian Processes. ICML 2017

Matthews et al. 2018. Gaussian Process Behaviour in Wide Deep Neural Networks. ICLR 2018

Lee et al. 2018. Deep Neural Networks as Gaussian Processes. ICLR 2018

Ma et al. 2019. Variational Implicit Processes. ICML 2019

Tanno et al. 2019. Uncertainty Quantification in Deep Learning for Safer Neuroimage Enhancement. arXiv:1907.13418

Foong et al. 2019. Pathologies of Factorised Gaussian and MC Dropout Posteriors in Bayesian Neural Networks. arXiv:1909.00719