Inference Networks for Gaussian Process Models

Jiaxin Shi

Tsinghua University

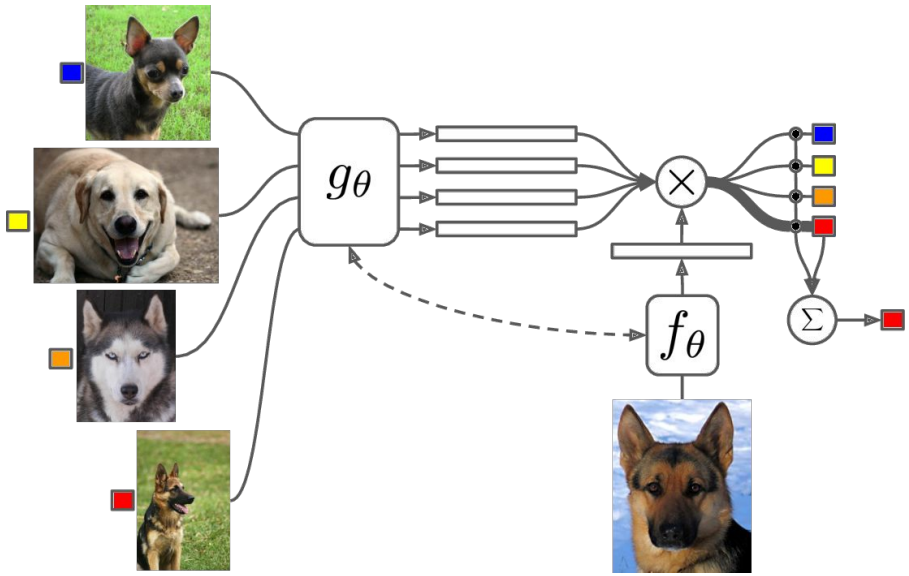
Probabilistic Inference over Functions

Why care?

- Learning is about fitting functions
- A compact way to represent experience - understanding of the world
- Neural networks or nearest neighbors?
- Real-world experience is sparse, costly
- Optimal decision making under this case

Probabilistic Inference over Functions

Few-shot learning: Matching Networks



Soft nearest neighbors:

$$y^* = \sum_{i=1}^{k} a(x^*, x_i) y_i$$

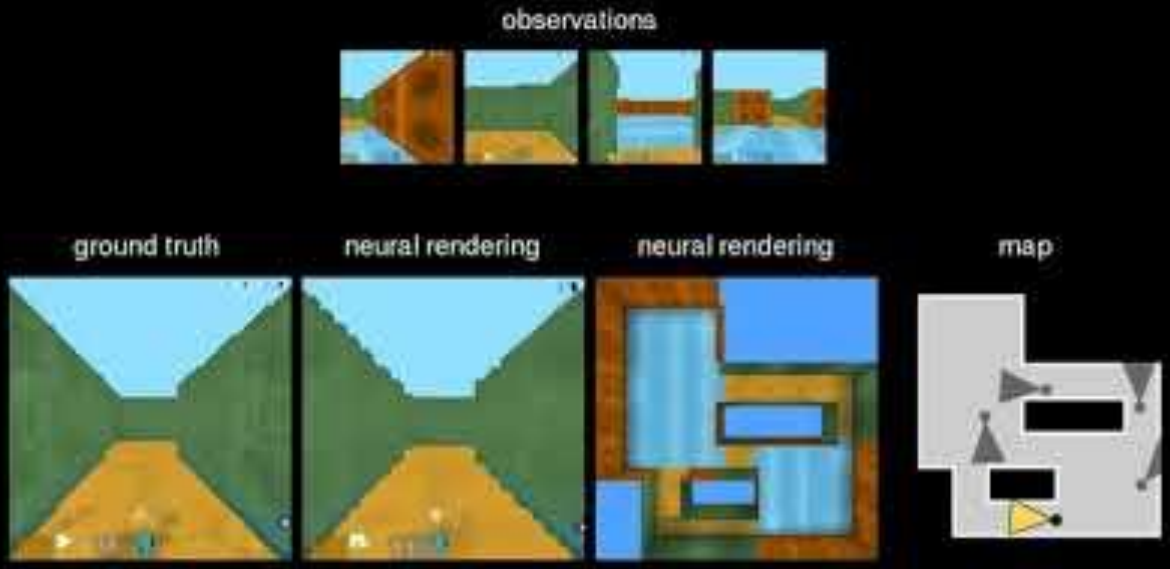
[Vinyals et al., 16]

Model	Matching Fn	Fine Tune	5-way Acc		20-way Acc	
			1-shot	5-shot	1-shot	5-shot
PIXELS	Cosine	N	41.7%	63.2%	26.7%	42.6%
BASELINE CLASSIFIER	Cosine	N	80.0%	95.0%	69.5%	89.1%
BASELINE CLASSIFIER	Cosine	Y	82.3%	98.4%	70.6%	92.0%
BASELINE CLASSIFIER	Softmax	Y	86.0%	97.6%	72.9%	92.3%
MATCHING NETS (OURS)	Cosine	N	**98.1%**	**98.9%**	**93.8%**	98.5%
MATCHING NETS (OURS)	Cosine	Y	97.9%	98.7%	93.5%	**98.7%**

Table 1: Results on the Omniglot dataset.

Probabilistic Inference over Functions

Natural scene representation: Generative Query Network (GQN)



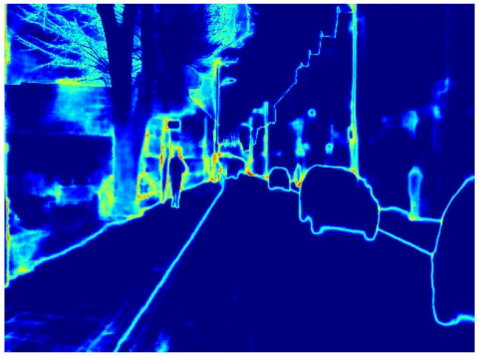
[Eslami et al., 2018]

Probabilistic Inference over Functions

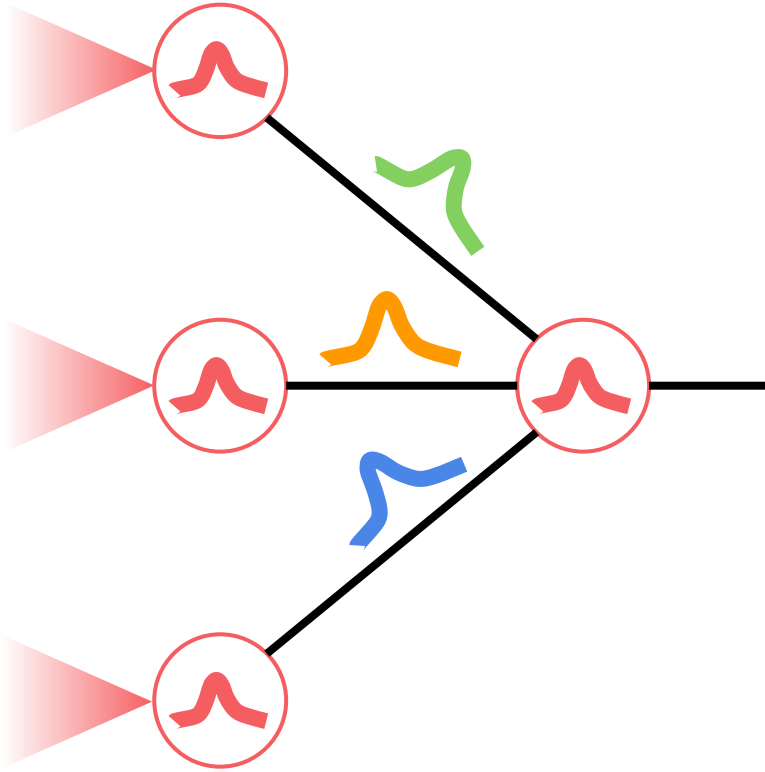
Uncertainty in neural networks

Scene

Uncertainty of depth estimates



[Kendall et al., 17]



- Bayesian learning of neural networks [Neal, 95]

Posterior inference: $$p(\mathbf{w}|\mathcal{D}) \propto p(\mathbf{w}) \prod_{n=1}^{N} p(y_n|\mathbf{x}_n, \mathbf{w})$$

Prediction with uncertainty: $$p(y^*|\mathbf{x}^*, \mathcal{D}) = \int p(y^*|\mathbf{x}, \mathbf{w}) p(\mathbf{w}|\mathcal{D}) d\mathbf{w}$$

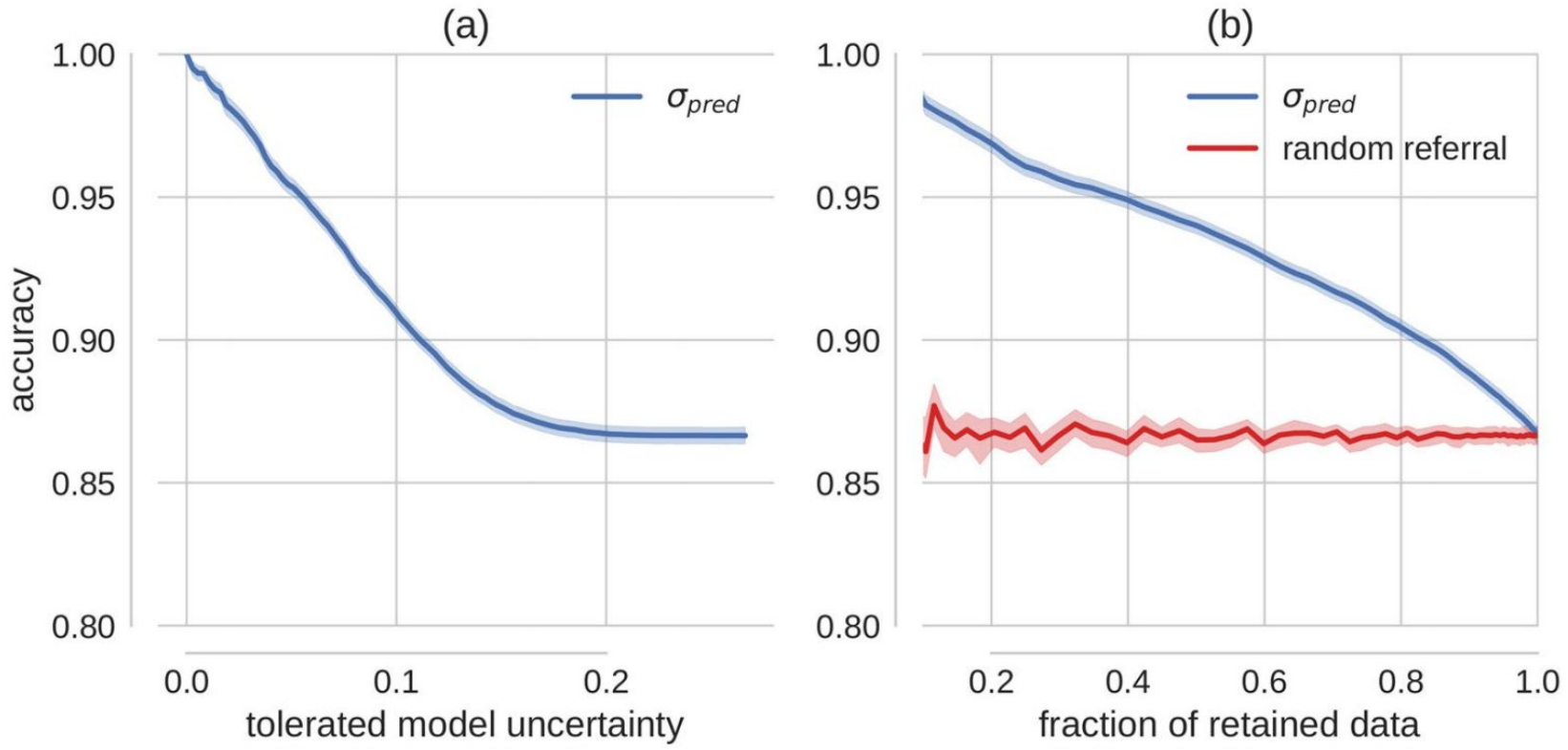
OPEN

[Leibig et al., 17]

Leveraging uncertainty information from deep neural networks for disease detection

Received: 24 July 2017

Christian Leibig[1], Vaneeda Allken[1], Murat Seçkin Ayhan[1], Philipp Berens[1,2] & Siegfried Wahl[1,3]



Probabilistic Inference over Functions

Bayesian decision making

Given the predictive distribution, how to predict?

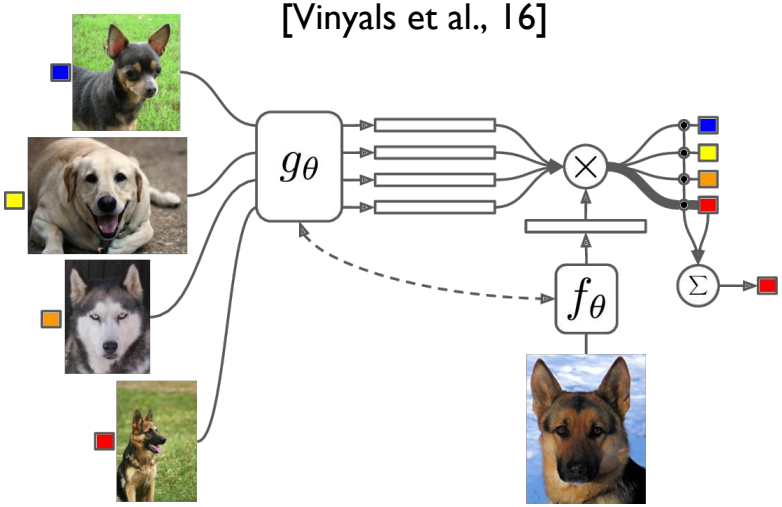
- Define risk/-utility/-reward: $R(\hat{y}^*, y^* | \mathbf{x}^*)$
- Minimize expected risk: $\mathbb{E}_{y^*|\mathbf{x}^*,\mathcal{D}} R(\hat{y}^*, y^* | \mathbf{x}^*)$
- Application-dependent risk
  - Healthcare
  - Automated driving
  - Quantitative investment
- Sequential decision making
  - Thompson sampling

Cost of Incorrect Diagnosis	Prediction	True	Utility Function
£0	Healthy	Healthy	2.0
£0	Mild	Mild	2.0
£0	Severe	Severe	2.0
£30	Severe	Mild	1.4
£35	Mild	Severe	1.3
£40	Mild	Healthy	1.2
£45	Severe	Healthy	1.1
£50	Healthy	Mild	1.0
£100	Healthy	Severe	0.0

[Cobb et al., 18]

Probabilistic Inference over Functions

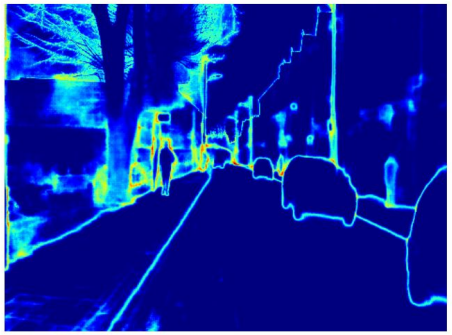
[Vinyals et al., 16]



$$y^* = \sum_{i=1}^{k} a(x^*, x_i) y_i$$

Scene

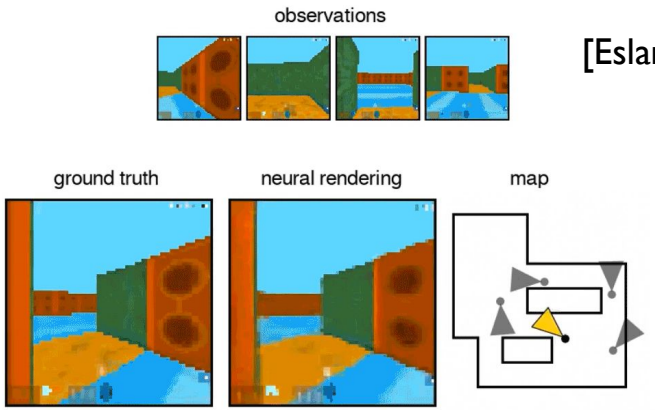
Uncertainty of depth estimates



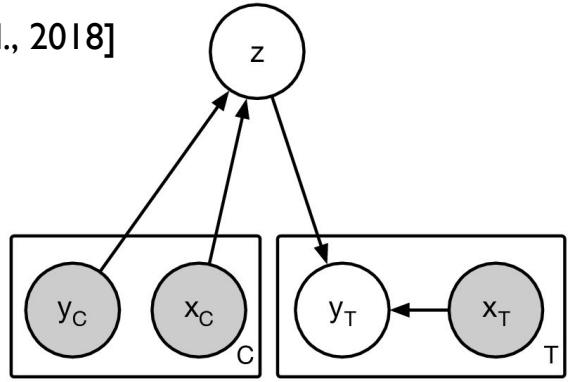
[Kendall et al., 17]

$$p(\mathbf{w}|\mathcal{D}) \propto p(\mathbf{w}) \prod_{n=1}^{N} p(y_n|\mathbf{x}_n, \mathbf{w})$$

$$p(y^*|\mathbf{x}^*) = \int p(y^*|\mathbf{x}^*, \mathbf{w}) p(\mathbf{w}|\mathcal{D}) d\mathbf{w}$$



[Eslami et al., 2018]



Gaussian Processes

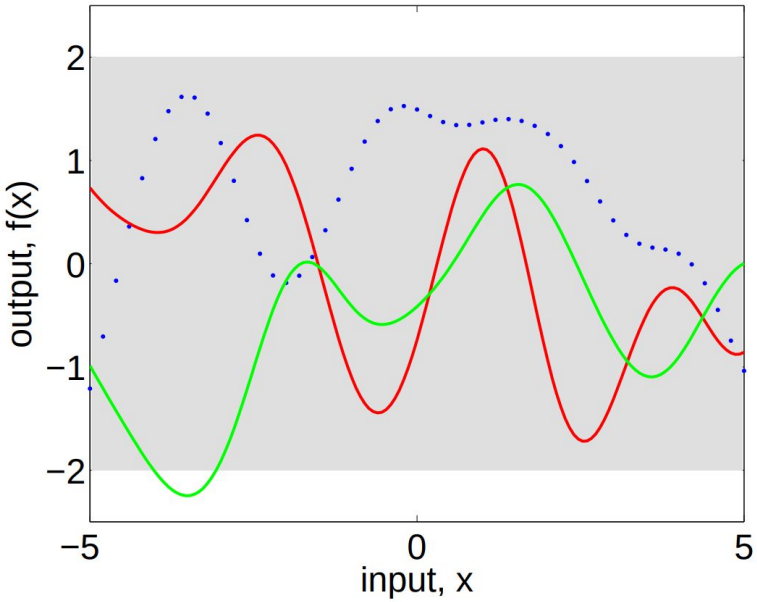
Definition

$$f(\mathbf{x}) \sim \mathcal{GP}\left(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\right)$$

mean function

covariance function / kernel

[GPML, Rasmussen & Williams]



- For any finite number of input locations, the marginal distribution is a multivariate Gaussian

$$\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^\top \qquad \mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N)]^\top \qquad \mathbf{f} \sim \mathcal{N}\left(\mathbf{m}_\mathcal{D}, \mathbf{K}_{\mathcal{D},\mathcal{D}}\right)$$

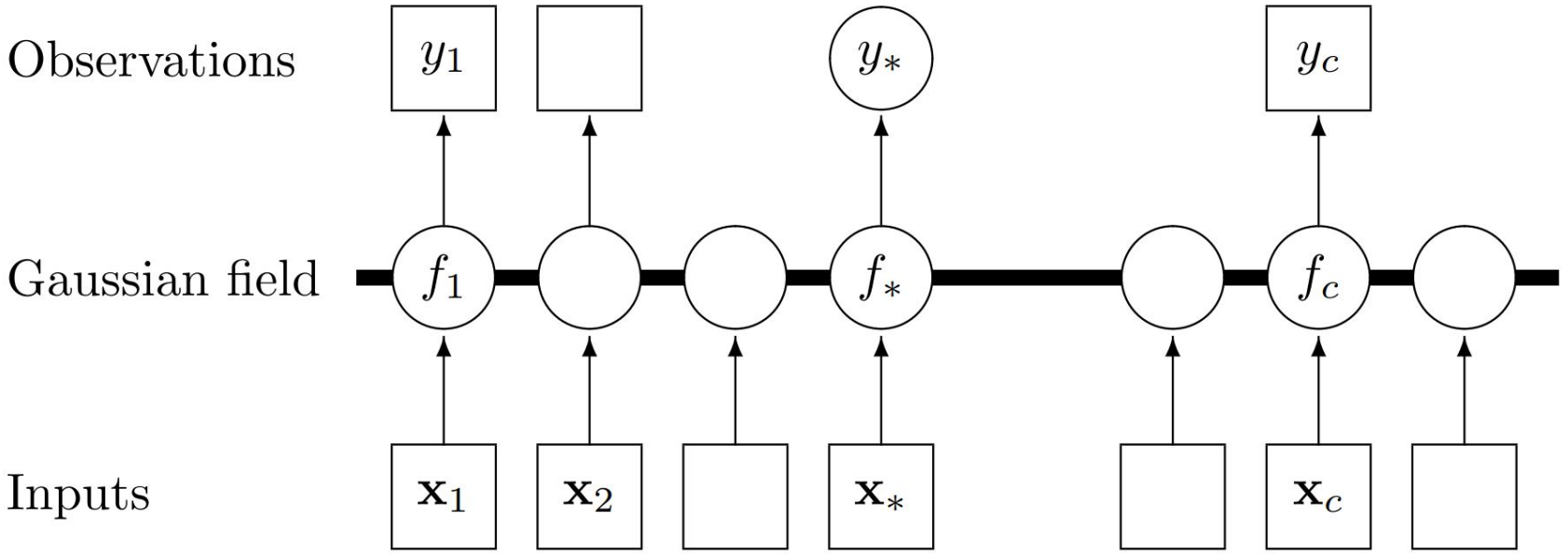
- Likelihood
  - Regression: $\mathbf{y} \sim \mathcal{N}(\mathbf{f}, \sigma^2\mathbf{I})$
  - Nonconjugate: classification, ordinal regression, multi-output, etc.

Gaussian Processes

Definition

$$f(\mathbf{x}) \sim \mathcal{GP}\left(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\right)$$

[GPML, Rasmussen & Williams]



Gaussian Processes

Weight-space view

weight space

$$f(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}), \quad \mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

function space

$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

Bayesian (generalized) linear regression

deep counterpart: BNN

Gaussian Processes

Exact inference

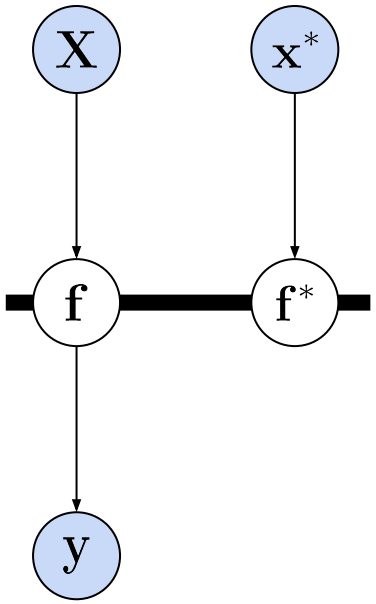
$$p(f(\mathbf{x}^*), \mathbf{f}|\mathbf{y}) \propto p(f(\mathbf{x}^*), \mathbf{f}) p(\mathbf{y}|\mathbf{f})$$

- For Gaussian likelihoods:

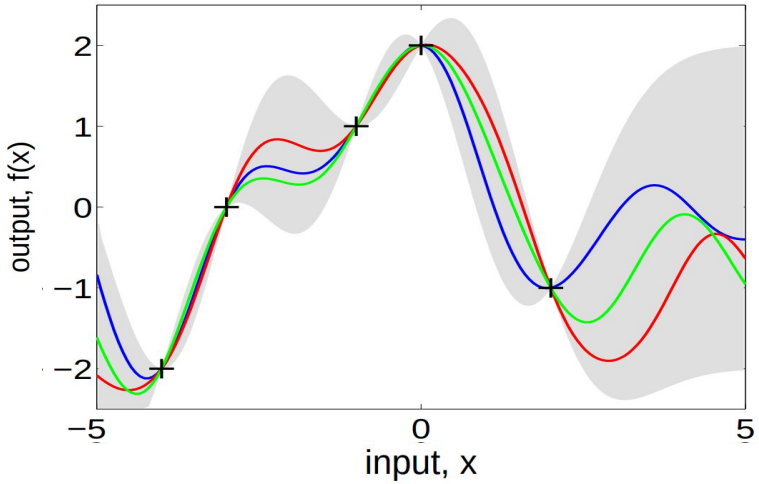
$$f(\mathbf{x}^*)|\mathbf{y} \sim \mathcal{GP}\left(m_{|\mathbf{y}}(\mathbf{x}^*), k_{|\mathbf{y}}(\mathbf{x}^*, \mathbf{x}^{*\prime})\right)$$

$$m_{|\mathbf{y}}(\mathbf{x}^*) = \mathbf{k}_{*,\mathcal{D}}^\top (\mathbf{K}_{\mathcal{D},\mathcal{D}} + \sigma^2 \mathbf{I})^{-1} (\mathbf{y} - \mathbf{m}_{\mathcal{D}})$$

$$k_{|\mathbf{y}}(\mathbf{x}^*, \mathbf{x}^*) = k_{**} - \mathbf{k}_{*,\mathcal{D}}^\top (\mathbf{K}_{\mathcal{D},\mathcal{D}} + \sigma^2 \mathbf{I})^{-1} \mathbf{k}_{\mathcal{D},*}$$



Gaussian field



- $O(N^3)$ complexity
- intractable with non-conjugate likelihoods

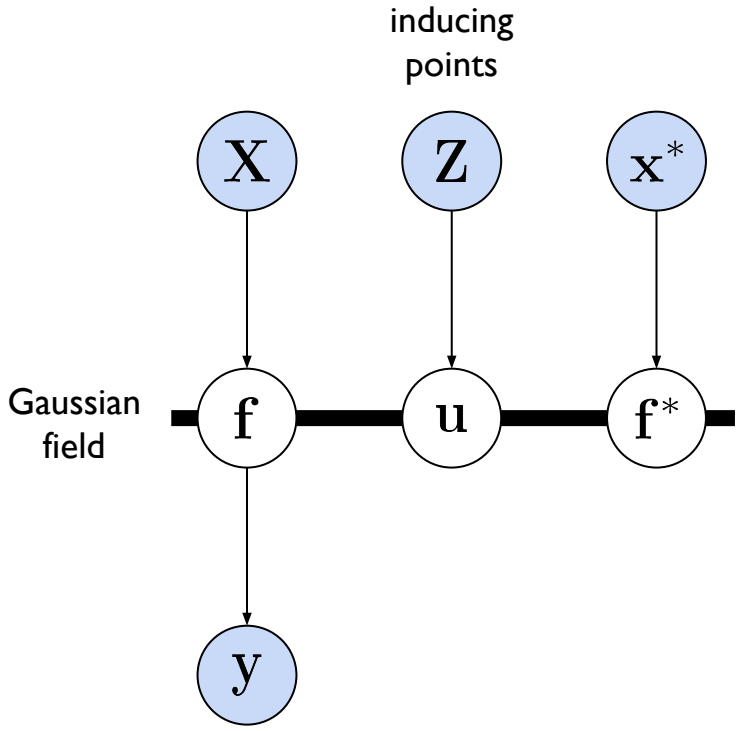
[GPML, Rasmussen & Williams]

Gaussian Processes

Sparse variational approximations

- Variational inference

$$\mathbb{E}_{q(\mathbf{f})} \log p(\mathbf{y}|\mathbf{f}) - \mathrm{KL}\left[q(\mathbf{f})||p(\mathbf{f})\right]$$



Sparse variational GP [Titsias, 09; Hensman et al., 13]

- non-conjugate likelihood
- O(M^2N) time, minibatch training
- joint hyperparameter learning

- Key idea: variational learning of inducing points

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

$$\mathcal{L}(q, \mathbf{Z}) := \mathbb{E}_{q(\mathbf{u})p(\mathbf{f}|\mathbf{u})}\left[\log p(\mathbf{y}|\mathbf{f})\right] - \mathrm{KL}[q(\mathbf{u})||p(\mathbf{u})]$$

- Tighten the lower bound by optimizing $\mathbf{Z}$, $q(\mathbf{u})$

Scalable Training of Inference Networks for Gaussian-Process Models

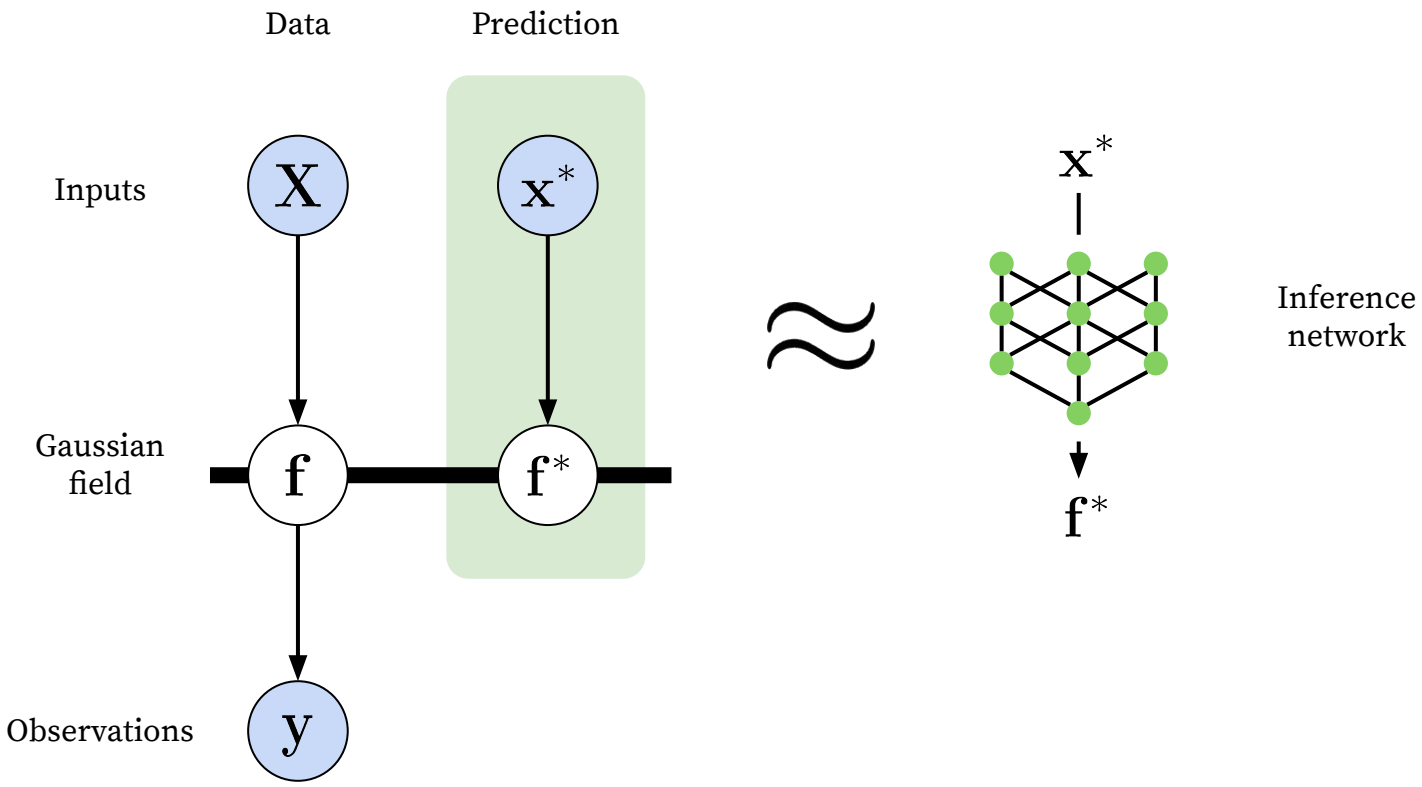
Joint work with

Emtiyaz Khan

Jun Zhu

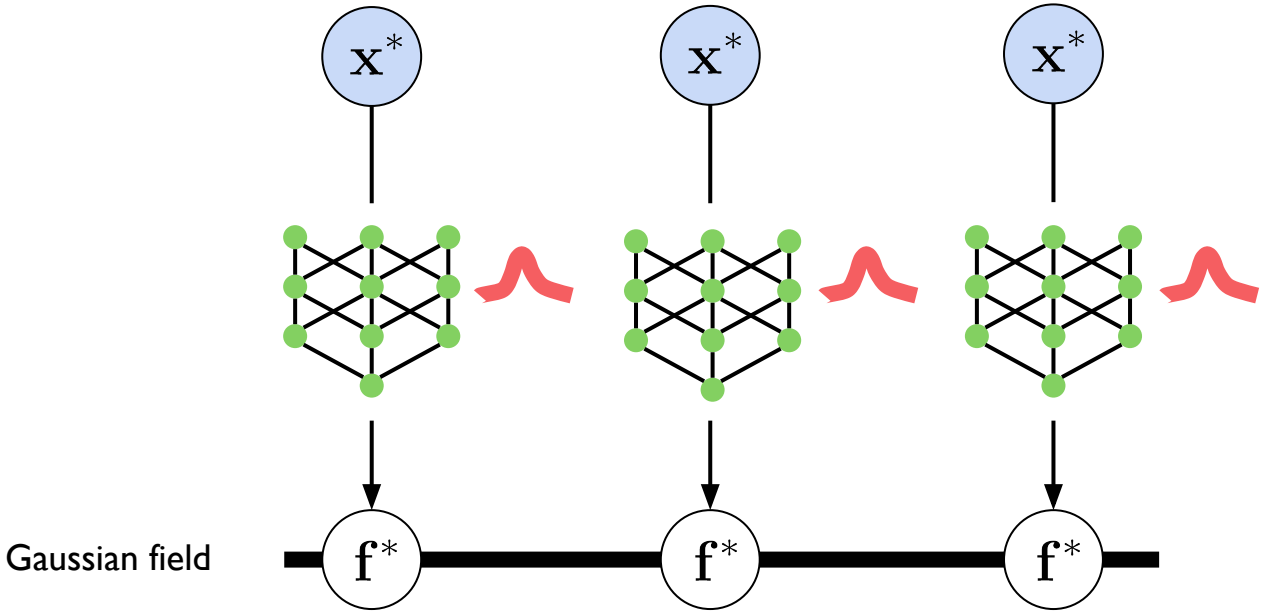
Scalable Training of Inference Networks for GP Models

Inference networks: remove sparse assumption



Scalable Training of Inference Networks for GP Models

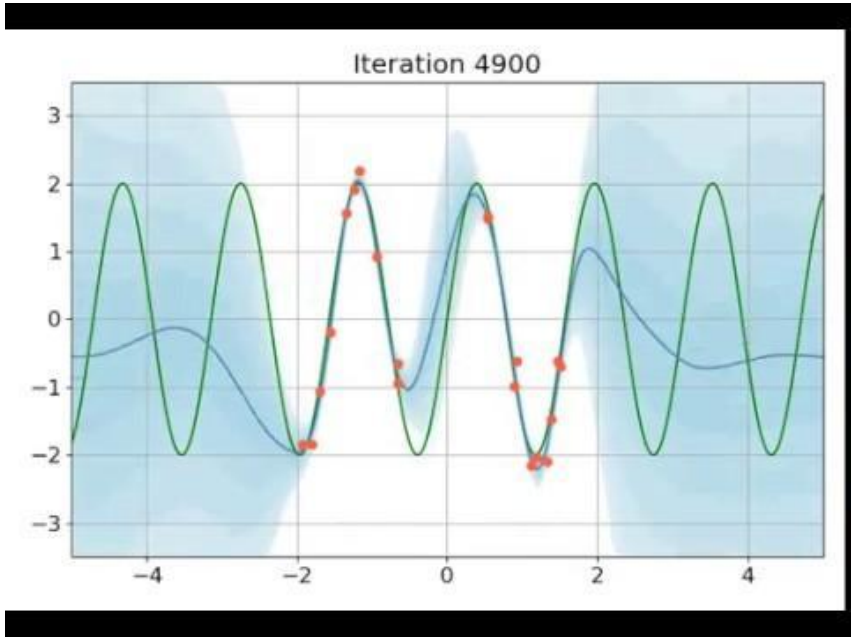
Inference networks

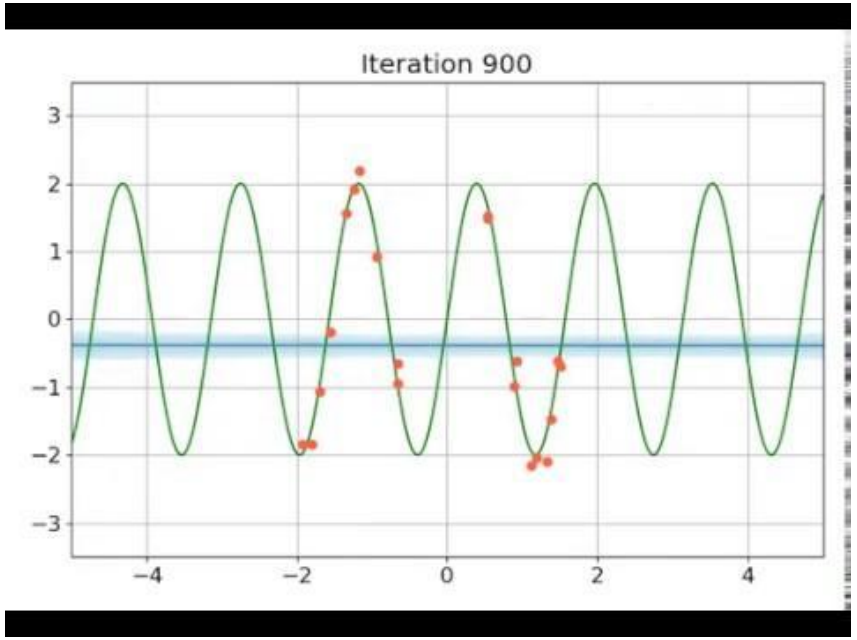


$$q(\mathbf{f}_{\mathcal{M}}) = \mathcal{N}(\mathbf{f}_{\mathcal{M}} | \mu_{\mathcal{M}}, \Sigma_{\mathcal{M}})$$

Bayesian Neural Nets as Inference Networks

Functional Variational Bayesian Neural Networks (Sun et al., ICLR 19)





Functional Variational Bayesian Neural Networks (Sun et al., 19)

Algorithm

- Consider matching variational and true posterior processes at arbitrary $\mathbf{X}^{\mathcal{M}}$

$$\mathrm{KL}[q(\mathbf{f}^{\mathcal{M}})\|p(\mathbf{f}^{\mathcal{M}}|\mathbf{y})] \leq \mathrm{KL}[q_\phi(\mathbf{f}^{\mathcal{M}}, \mathbf{f})\|p(\mathbf{f}^{\mathcal{M}}, \mathbf{f}|\mathbf{y})]$$

- Full batch fELBO

$$\mathcal{L}_{\mathbf{X}^{\mathcal{M}},\mathbf{X}}(q) = \log p(\mathcal{D}) - \mathrm{KL}[q(\mathbf{f}^{\mathcal{M}}, \mathbf{f})\|p(\mathbf{f}^{\mathcal{M}}, \mathbf{f}|\mathbf{y})].$$

$$= \sum_{(\mathbf{x},y)\in\mathcal{D}} \mathbb{E}_{q_\phi}\left[\log p(y|f(\mathbf{x}))\right] - \mathrm{KL}[q(\mathbf{f}^{\mathcal{M}}, \mathbf{f})\|p(\mathbf{f}^{\mathcal{M}}, \mathbf{f})]$$

- Practical fELBO

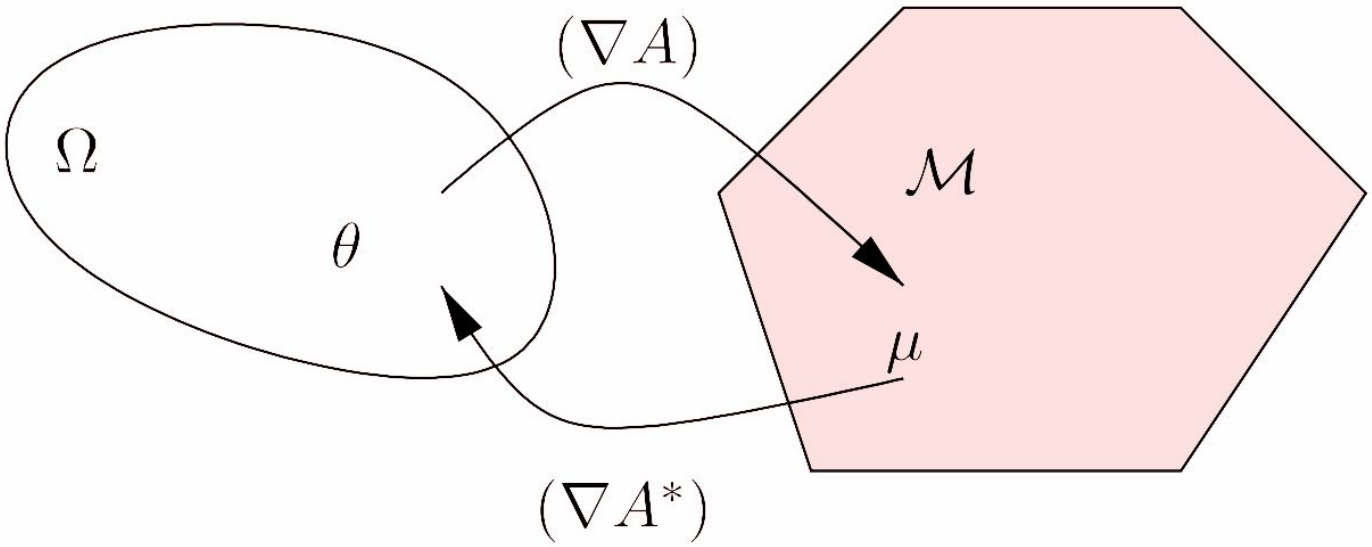
$$\frac{1}{|\mathcal{D}_s|} \sum_{(\mathbf{x},y)\in\mathcal{D}_s} \mathrm{E}_{q_\phi}\left[\log p(y|f(\mathbf{x}))\right] - \lambda\mathrm{KL}[q(\mathbf{f}^{\mathcal{D}_s}, \mathbf{f}^{M})\|p(\mathbf{f}^{\mathcal{D}_s}, \mathbf{f}^{M})].$$

- This objective is doing improper minibatch for the KL divergence term

Scalable Training of Inference Networks for GP Models

Stochastic mirror descent

$$\mu = \mathbb{E}_{p_\theta}\left[t(x)\right]$$



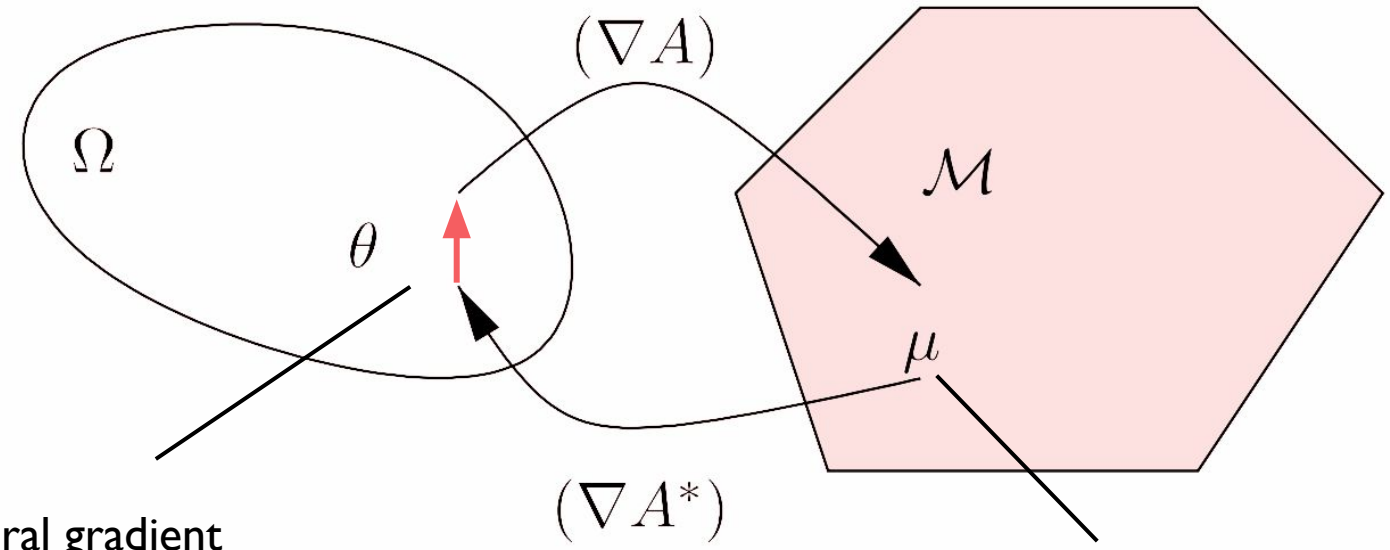
$$p(x) \propto \exp\{\theta^\top t(x) - A(\theta)\}$$

[Wainwright & Jordan, 08]

Scalable Training of Inference Networks for GP Models

Stochastic mirror descent

$$\mu = \mathbb{E}_{p_\theta}\left[t(x)\right]$$



natural gradient

mirror descent

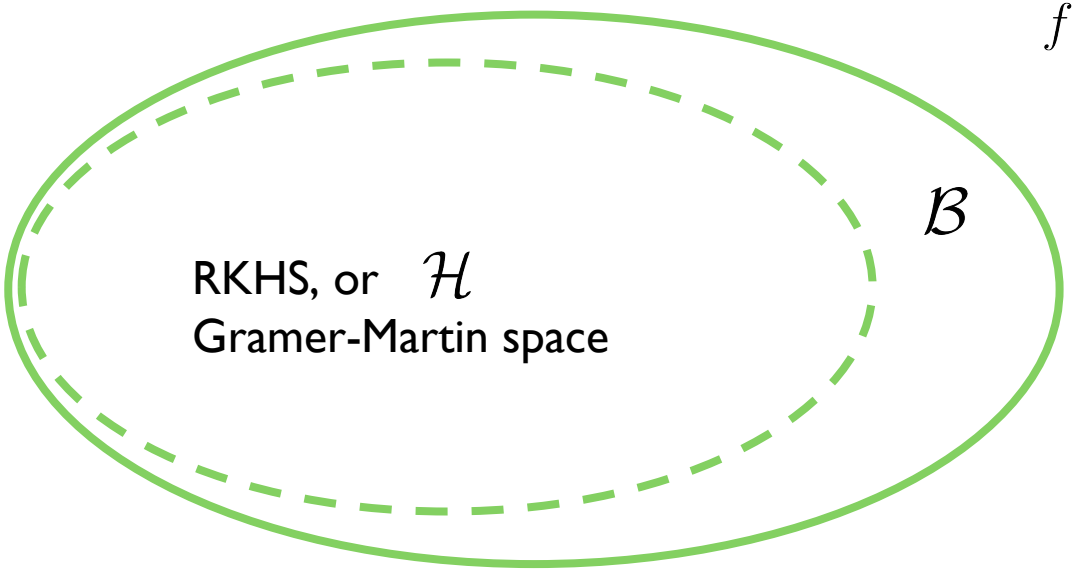
$$\mu_{t+1} = \underset{\mu}{\operatorname{argmax}}\ \mu^\top \nabla_\mu \mathcal{L}^*(\mu) - \frac{1}{\beta_t}\mathbb{B}_{A^*}\left[\mu \| \mu_t\right] \qquad \mathcal{L}(\theta) := \mathcal{L}^*(\mu)$$

$$\mu_{t+1} = \nabla A(\nabla A^*(\mu_t) - \beta_t \nabla_\mu \mathcal{L}^*(\mu_t))$$

[Raskutti & Mukherjee, 13; Amari, 16; Khan & Lin, 17]

Scalable Training of Inference Networks for GP Models

*Gaussian processes as Gaussian measures**



$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

$$||$$

$$f \sim \mathcal{N}(0, I)$$

$$\mathcal{L}(q) = \mathbb{E}_{q(f)} \log p(\mathbf{y}|f) - \mathrm{KL}[q(f)\|p(f)]$$

- Abstract Wiener space $(\mathcal{H}, \mathcal{B}, q)$: a way to define "decent" measure in function space.
- $\mathcal{H}$ is dense in $\mathcal{B}$
- Canonical Gaussian cylinder set measure on $\mathcal{H}$: not a proper measure, but useful.
  - identity map to $\mathcal{B}$ transforms it to a proper measure.

Scalable Training of Inference Networks for GP Models

Stochastic, functional mirror descent

- an equivalent, but simpler derivation [Dai et al., 16]
  - work with the functional density directly
  - minibatch approximation with stochastic functional gradient

$$q_{t+1} = \underset{q}{\operatorname{argmax}} \int \hat{\partial}\mathcal{L}(q_t) q(f) df - \frac{1}{\beta_t} \mathrm{KL}\left[q \| q_t\right]$$

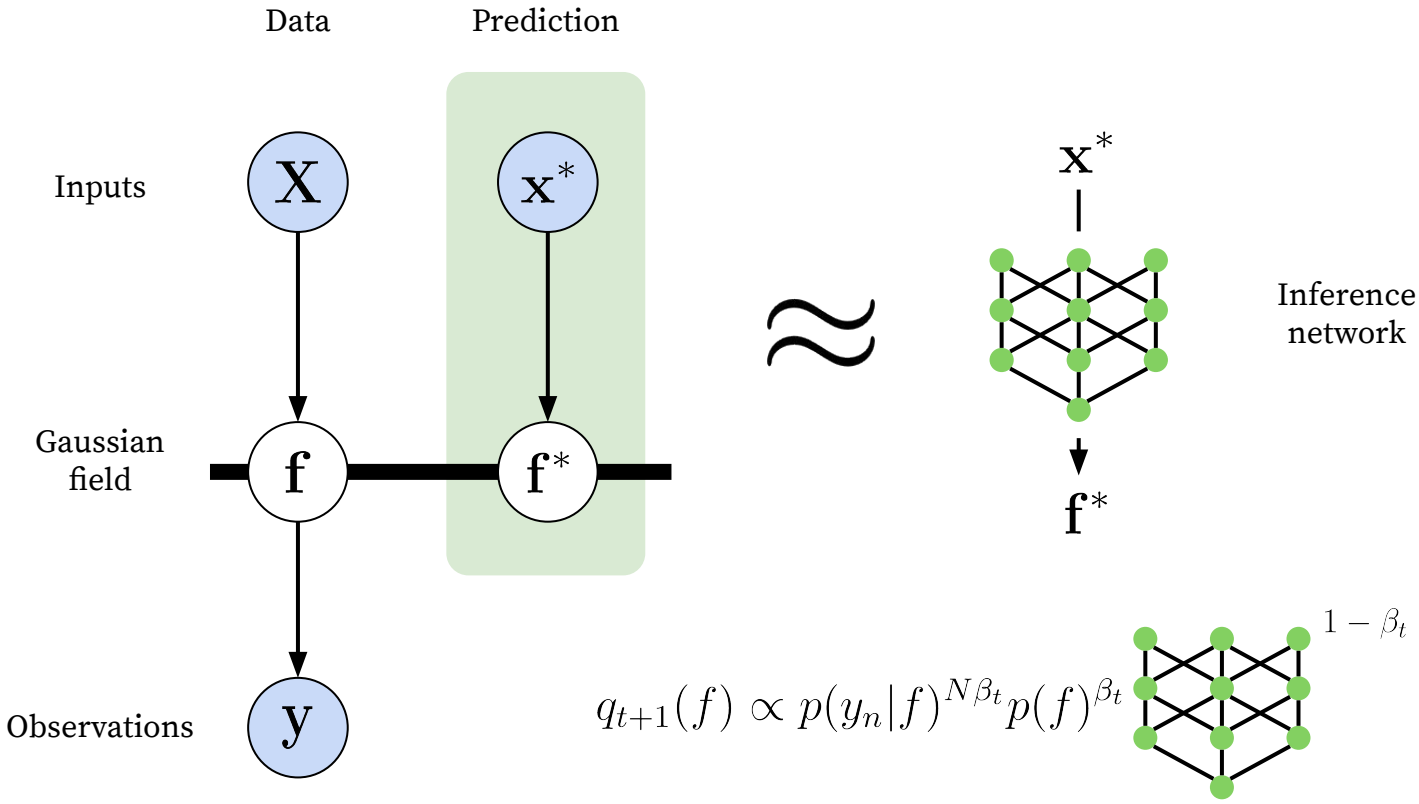
- closed-form solution as an adaptive Bayesian filter

$$q_{t+1}(f) \propto \underbrace{p(y_n|f)^{N\beta_t}}_{\text{seeing next data point}} \underbrace{p(f)^{\beta_t} q_t(f)^{1-\beta_t}}_{\text{adapted prior}}$$

- sequentially applying Bayes' rule is the most natural gradient

Scalable Training of Inference Networks for GP Models

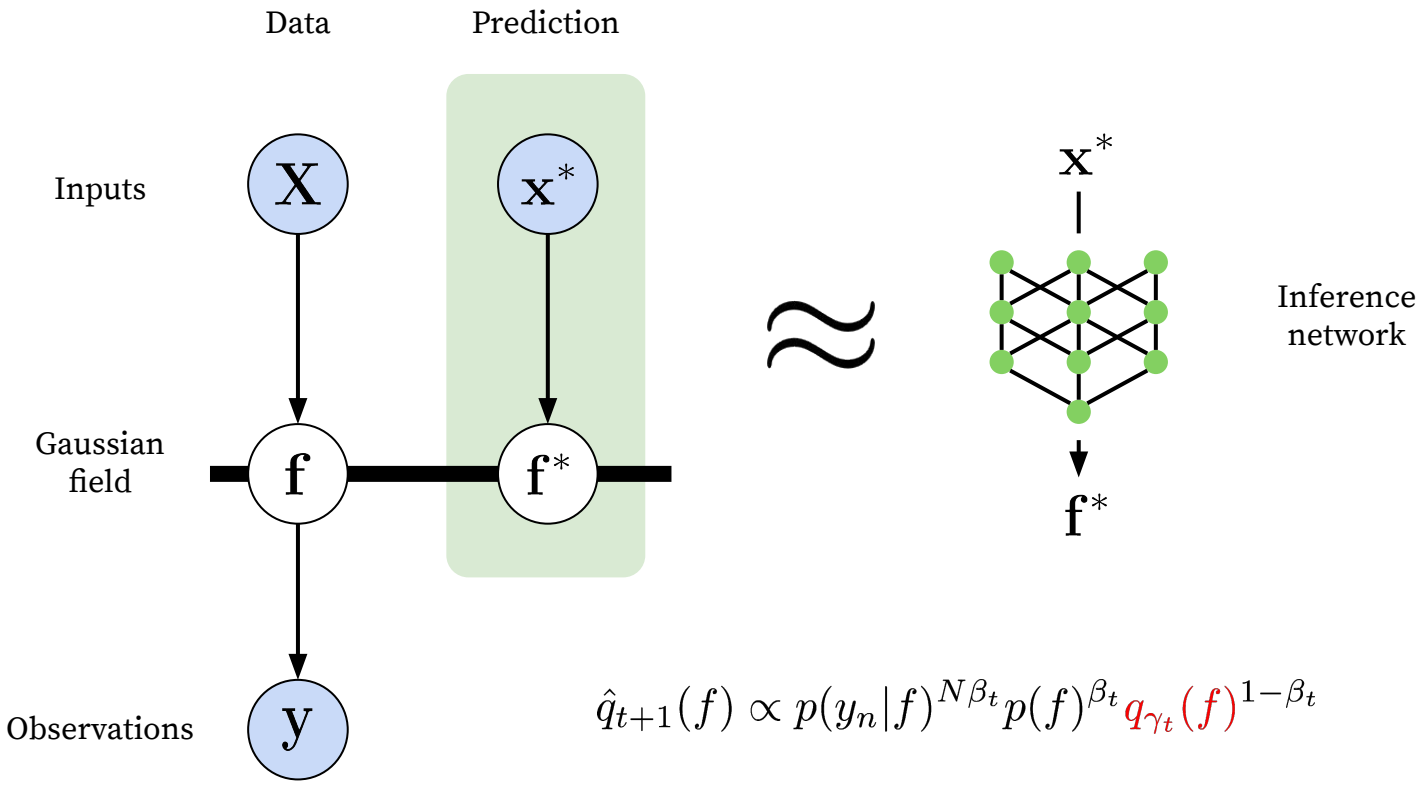
Minibatch training of inference networks



- the stochastic, functional mirror descent update is still intractable
- an idea from filtering: bootstrap
  - use a surrogate to pass on the information

Scalable Training of Inference Networks for GP Models

Minibatch training of inference networks



$$\hat{q}_{t+1}(f) \propto p(y_n|f)^{N\beta_t} p(f)^{\beta_t} q_{\gamma_t}(f)^{1-\beta_t}$$

- the stochastic, functional mirror descent update is still intractable
- an idea from filtering: bootstrap
  - similar idea: temporal difference (TD) learning with function approximation

Scalable Training of Inference Networks for GP Models

Minibatch training of inference networks

$\hat{q}_{t+1}(f) \propto p(y_n|f)^{N\beta_t} p(f)^{\beta_t} q_{\gamma_t}(f)^{1-\beta_t}$ is an attractive equation because

- $\hat{q}_{t+1}(f)$ is a GP
- if likelihood is Gaussian, all marginal distributions of $\hat{q}_{t+1}(f)$ in closed-form

- compute the marginals of $\hat{q}_{t+1}(f)$ at locations $\mathbf{X}_{\mathcal{M}}$
  - equivalent to GP regression

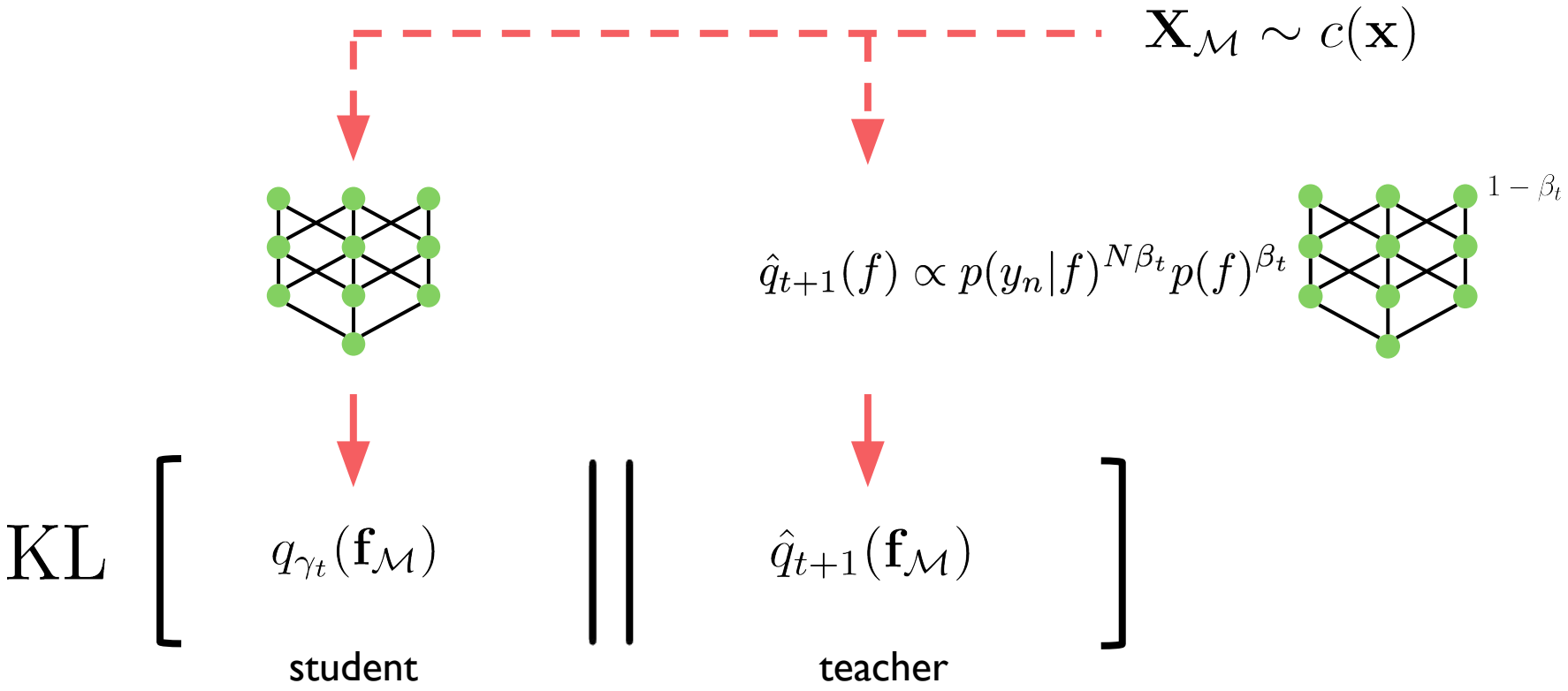
$$p(\mathbf{f}_{\mathcal{M}}, f_n)^{\beta_t} q_{\gamma_t}(\mathbf{f}_{\mathcal{M}}, f_n)^{1-\beta_t} := \mathcal{N}\left(\begin{bmatrix} \widetilde{\mathbf{m}}_{\mathcal{M}} \\ \widetilde{\mathbf{m}}_n \end{bmatrix}, \begin{bmatrix} \widetilde{\mathbf{K}}_{\mathcal{M},\mathcal{M}} & \widetilde{\mathbf{K}}_{\mathcal{M},n} \\ \widetilde{\mathbf{K}}_{n,\mathcal{M}} & \widetilde{\mathbf{K}}_{n,n} \end{bmatrix}\right)$$

$$\propto \quad \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \mathbf{K}_{\mathcal{M},\mathcal{M}} & \mathbf{K}_{\mathcal{M},n} \\ \mathbf{K}_{n,\mathcal{M}} & \mathbf{K}_{n,n} \end{bmatrix}\right)^{\beta_t} \times \mathcal{N}\left(\begin{bmatrix} \mu_{\mathcal{M}} \\ \mu_n \end{bmatrix}, \begin{bmatrix} \mathbf{\Sigma}_{\mathcal{M},\mathcal{M}} & \mathbf{\Sigma}_{\mathcal{M},n} \\ \mathbf{\Sigma}_{n,\mathcal{M}} & \mathbf{\Sigma}_{n,n} \end{bmatrix}\right)^{(1-\beta_t)}$$

$$\hat{q}_{t+1}(\mathbf{f}_{\mathcal{M}}, f_n) \propto \mathcal{N}(y_n|f_n, \sigma^2/(N\beta_t)) \times \mathcal{N}\left(\begin{bmatrix} \widetilde{\mathbf{m}}_{\mathcal{M}} \\ \widetilde{\mathbf{m}}_n \end{bmatrix}, \begin{bmatrix} \widetilde{\mathbf{K}}_{\mathcal{M},\mathcal{M}} & \widetilde{\mathbf{K}}_{\mathcal{M},n} \\ \widetilde{\mathbf{K}}_{n,\mathcal{M}} & \widetilde{\mathbf{K}}_{n,n} \end{bmatrix}\right)$$

Scalable Training of Inference Networks for GP Models

Minibatch training of inference networks



Scalable Training of Inference Networks for GP Models

Minibatch training of inference networks

Algorithm 1 GPNet for supervised learning

Input: $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$, $c(\mathbf{x})$, M, T, β, η.

Initialize the inference network q_γ.
for $t = 1, \ldots, T$ **do**
  Randomly sample a training data $(\mathbf{x}_n, y_n)$.
  Sample $\mathbf{X}_\mathcal{M} = (\mathbf{x}_1, \ldots, \mathbf{x}_M)$ from $c(\mathbf{x})$.
  if the likelihood is Gaussian **then**
    Compute $\hat{q}_{t+1}(\mathbf{f}_\mathcal{M})$ using (10).
    $\gamma_{t+1} \leftarrow \gamma_t - \eta \nabla_\gamma \mathrm{KL}\left[q_\gamma(\mathbf{f}_\mathcal{M}) \| \hat{q}_{t+1}(\mathbf{f}_\mathcal{M})\right]$.
  else
    $\gamma_{t+1} \leftarrow \gamma_t + \eta \nabla_\gamma \mathcal{L}_t(q_\gamma; q_{\gamma_t}, \mathbf{X}_\mathcal{M})$.
  end if
end for
return q_{γ_t}.

For non-conjugate likelihoods

$$\mathcal{L}_t(q_\gamma; q_{\gamma_t}, \mathbf{X}_\mathcal{M}) = \mathbb{E}_{q_\gamma(\mathbf{f}_\mathcal{M}, f_n)}\left[N\beta_t \log p(y_n|f_n) + \beta_t \log p(\mathbf{f}_\mathcal{M}, f_n) + (1-\beta_t)\log q_{\gamma_t}(\mathbf{f}_\mathcal{M}, f_n) - \log q_\gamma(\mathbf{f}_\mathcal{M}, f_n)\right]$$

Scalable Training of Inference Networks for GP Models

Examples of inference networks

weight space

function space

$$f(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}), \quad \mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \quad \Leftrightarrow \quad f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

Bayesian (generalized) linear regression

deep counterpart: BNN

- Inference network architecture can be derived from the weight-space posterior

$$q(f) : f(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}) + \xi_\theta(\mathbf{x}), \quad \mathbf{w} \sim \mathcal{N}(\mathbf{m}, \mathbf{V})$$

Scalable Training of Inference Networks for GP Models

Examples of inference networks

- Bayesian neural networks (BNN) [Sun et al., 19]
  - intractable output density
  - current solutions: costly, even infeasible with large models
- Random feature expansions (RFE) [Cutajar, et al., 18]
- Deep neural networks [Snoek et al., 15]

Scalable Training of Inference Networks for GP Models

Random feature expansion (RFE)

Random Fourier features [Rahimi & Recht,16]

- Bocher's theorem

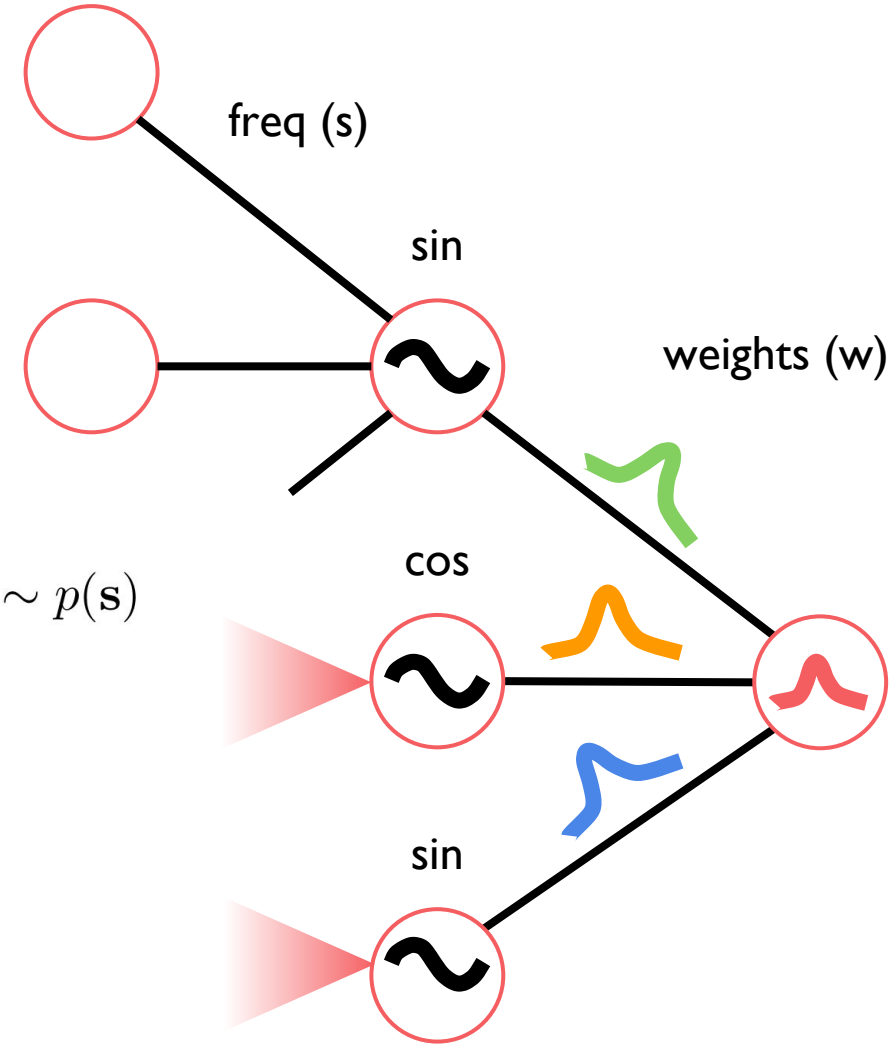
$$k(\mathbf{x}, \mathbf{x}') = \int e^{i\mathbf{s}^\top(\mathbf{x}-\mathbf{x}')} p(\mathbf{s}) d\mathbf{s}$$

- construct explicit (approximate) feature map for any translation-invariant kernel

$$k(\mathbf{x}, \mathbf{x}') \approx \frac{1}{M} \sum_{m=1}^{M} \cos(\mathbf{s}_m^\top(\mathbf{x} - \mathbf{x}')), \ \mathbf{s}_{1:M} \sim p(\mathbf{s})$$

$$\phi_r(\mathbf{x}) = \frac{1}{\sqrt{M}} [\cos(\mathbf{s}_1^\top \mathbf{x}), \ldots, \cos(\mathbf{s}_M^\top \mathbf{x}), \sin(\mathbf{s}_1^\top \mathbf{x}), \ldots, \sin(\mathbf{s}_M^\top \mathbf{x})]^\top$$

- BNNs as RF features of its infinite-width counterparts



Scalable Training of Inference Networks for GP Models

Deep neural networks

Two choices $q(f) : f(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}) + \xi_\theta(\mathbf{x}), \quad \mathbf{w} \sim \mathcal{N}(\mathbf{m}, \mathbf{V})$

- Directly parameterizing \phi(x) as neural networks with general nonlinearies such as tanh and ReLU.
- inject randomness into first-order expansion of neural networks

$$g(\mathbf{x}; \mathbf{\Omega}) = g(\mathbf{x}; \mathbf{\Omega}_0) + \nabla_{\mathbf{\Omega}} g(\mathbf{x}; \mathbf{\Omega}_0)(\mathbf{\Omega} - \mathbf{\Omega}_0) \qquad \mathbf{\Omega} \sim \mathcal{N}(\mathbf{\Omega}_0, \mathbf{V})$$

$$k_{\text{NTK}}(\mathbf{x}, \mathbf{x}') = \nabla_{\mathbf{\Omega}} g(\mathbf{x}; \mathbf{\Omega}_0)^\top \mathbf{V} \nabla_{\mathbf{\Omega}} g(\mathbf{x}; \mathbf{\Omega}_0)$$

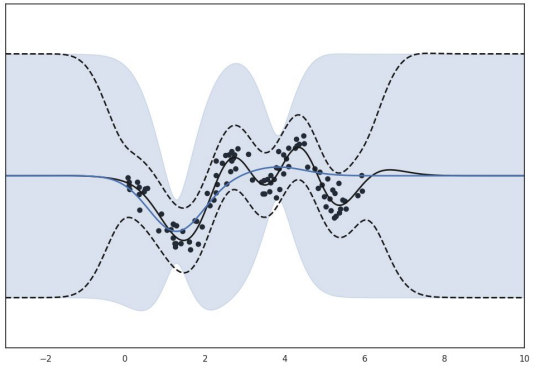
$$f \sim \mathcal{GP}(g(\cdot; \mathbf{\Omega}_0), k_{\text{NTK}})$$

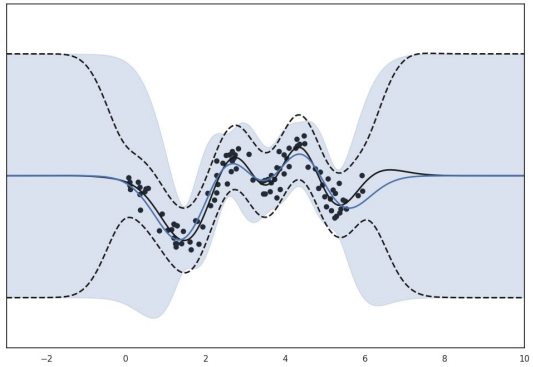
M=2

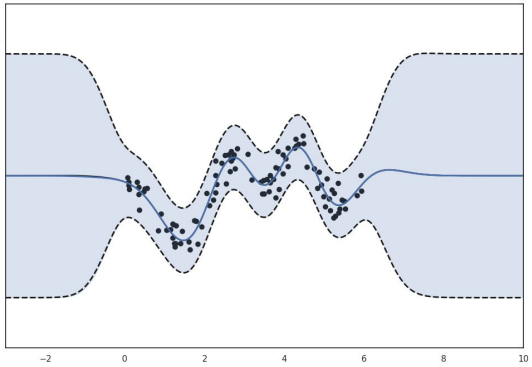
M=5

M=20

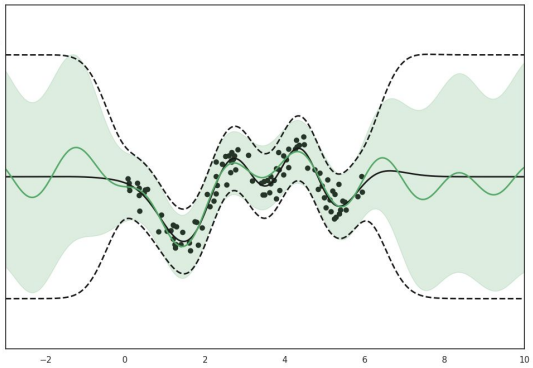
SVGP

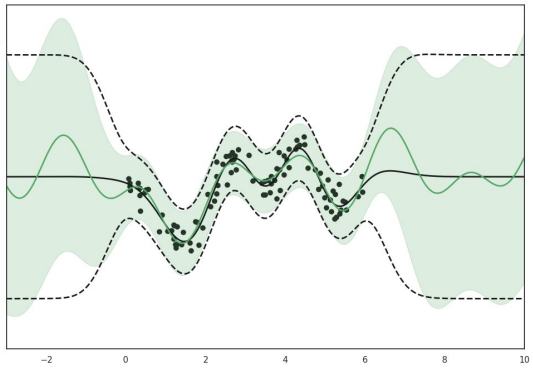


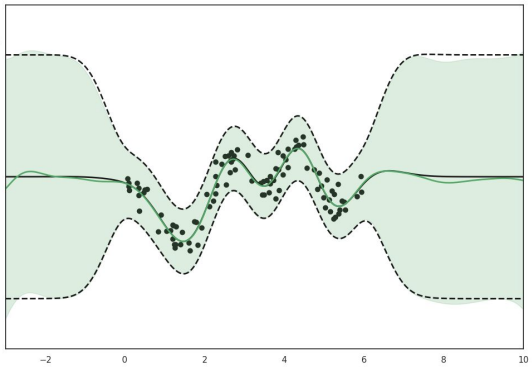




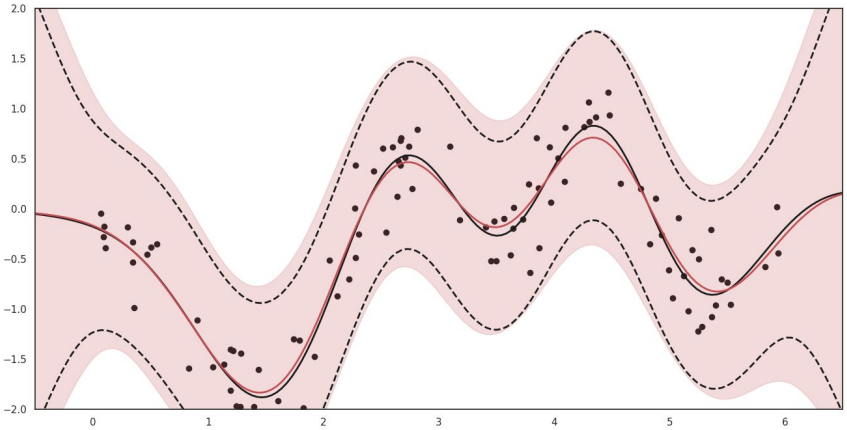
GPNet



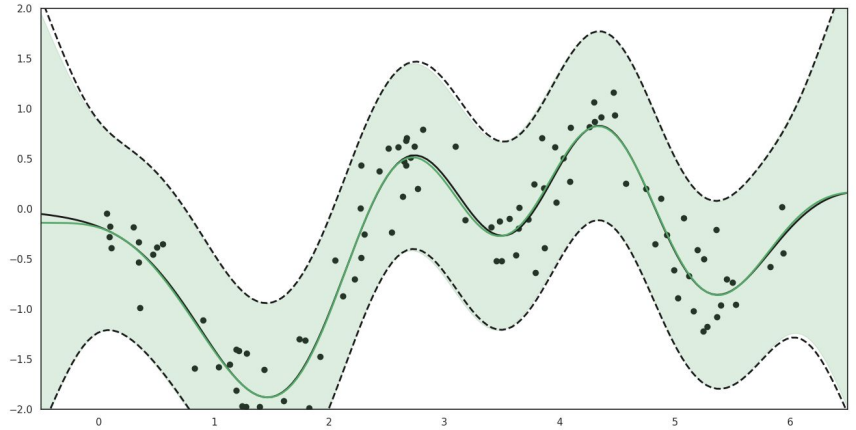




FBNN, M=20



GPNet, M=20



Scalable Training of Inference Networks for GP Models

Experiments: Regression

If $R(\hat{y}^*, y^* | \mathbf{x}^*) = \frac{1}{2} \| \hat{y}^* - y^* \|^2$

$$\min_{\hat{y}^*} \mathbb{E}_{\hat{y}^* | \mathbf{x}^*, \mathcal{D}} R(\hat{y}^*, y^* | \mathbf{x}^*) \implies \hat{y}^* = \mathbb{E}_{y^* | \mathbf{x}^*, \mathcal{D}}[y^*]$$

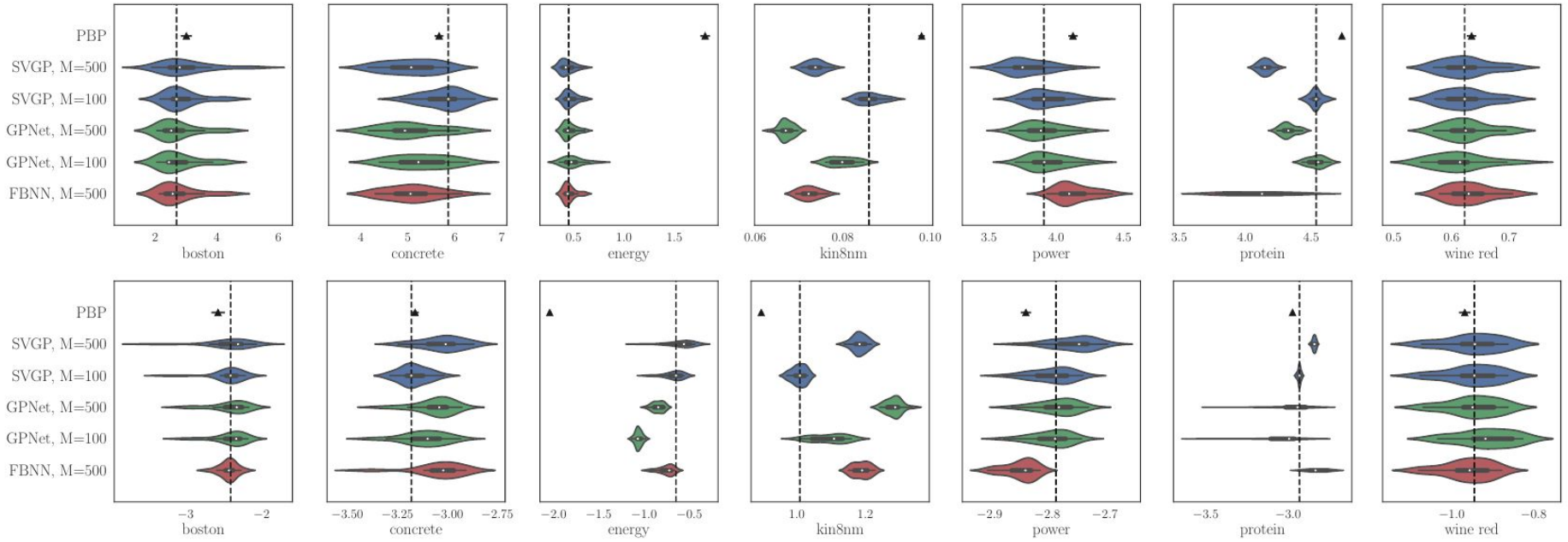
If $R(\hat{y}^*, y^* | \mathbf{x}^*) = | \hat{y}^* - y^* |$

$$\min_{\hat{y}^*} \mathbb{E}_{\hat{y}^* | \mathbf{x}^*, \mathcal{D}} R(\hat{y}^*, y^* | \mathbf{x}^*) \implies \hat{y}^* = \mathrm{MED}_{y^* | \mathbf{x}^*, \mathcal{D}}[y^*]$$

- The simplest case of Bayesian decision making
- The general principle behind still applies to diverse scenarios
  - Risk matters

Scalable Training of Inference Networks for GP Models

Experiments: Regression



Regression Benchmarks

Metric	M=100			M=500		
	SVGP	GPNet	FBNN	SVGP	GPNet	FBNN
RMSE	24.261	24.055	23.801	23.698	**23.675**	24.114
Test LL	-4.618	-4.616	-4.586	-4.594	-4.601	**-4.582**

Airline Delay (700K)

Conclusion & Future work

- function space / weight space
- natural gradient / mirror descent / Bayesian filter
- inference networks (tractable, flexible & scalable)
- what's next? multi-output GP, latent variable models (GP-LVM), deep GPs

Thanks

Code: https://github.com/thjashin/gp-infer-net