Part III: Deep Gaussian Processes

Daniel Hernández–Lobato
Computer Science Department
Universidad Autónoma de Madrid

http://dhnzl.org, daniel.hernandez@uam.es

Summary so Far about GPs

Summary so Far about GPs

Advantages of GPs:

Summary so Far about GPs

Advantages of GPs:

- Non-parametric models!

Summary so Far about GPs

Advantages of GPs:

- Non-parametric models!
- Exact Bayesian inference is tractable!

Summary so Far about GPs

Advantages of GPs:

- Non-parametric models!
- Exact Bayesian inference is tractable!
- They scale to very large datasets!

Summary so Far about GPs

Advantages of GPs:

- Non-parametric models!
- Exact Bayesian inference is tractable!
- They scale to very large datasets!
- Easy to introduce prior knowledge!

Summary so Far about GPs

Advantages of GPs:

- Non-parametric models!
- Exact Bayesian inference is tractable!
- They scale to very large datasets!
- Easy to introduce prior knowledge!

Disadvantages of GPs:

Summary so Far about GPs

Advantages of GPs:

- Non-parametric models!
- Exact Bayesian inference is tractable!
- They scale to very large datasets!
- Easy to introduce prior knowledge!

Disadvantages of GPs:

- Strong assumptions made about $f(\mathbf{x})$!

Summary so Far about GPs

Advantages of GPs:

- Non-parametric models!
- Exact Bayesian inference is tractable!
- They scale to very large datasets!
- Easy to introduce prior knowledge!

Disadvantages of GPs:

- Strong assumptions made about $f(\mathbf{x})$!
- The predictive distribution is always Gaussian!

Summary so Far about GPs

Advantages of GPs:

- Non-parametric models!
- Exact Bayesian inference is tractable!
- They scale to very large datasets!
- Easy to introduce prior knowledge!

Disadvantages of GPs:

- Strong assumptions made about $f(\mathbf{x})$!
- The predictive distribution is always Gaussian!
- Do not learn specific features to represent the observed data!

Summary so Far about GPs

Advantages of GPs:

- Non-parametric models!
- Exact Bayesian inference is tractable!
- They scale to very large datasets!
- Easy to introduce prior knowledge!

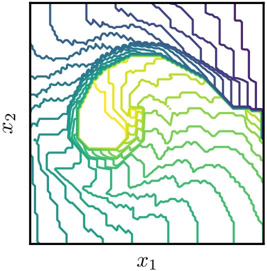
Disadvantages of GPs:

- Strong assumptions made about $f(\mathbf{x})$!
- The predictive distribution is always Gaussian!
- Do not learn specific features to represent the observed data!

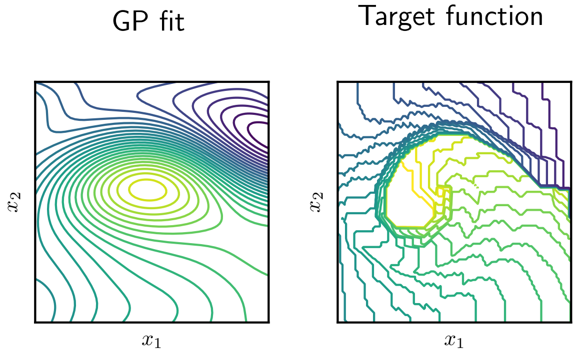
Deep GPs constitute a nice alternative to address these issues!

Motivation for Deep Gaussian Processes

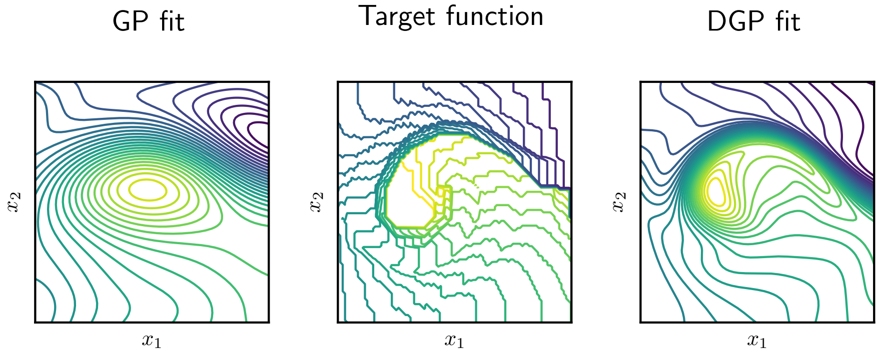
Target function



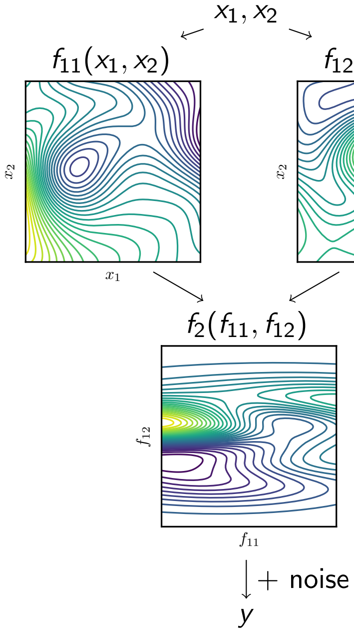
Motivation for Deep Gaussian Processes



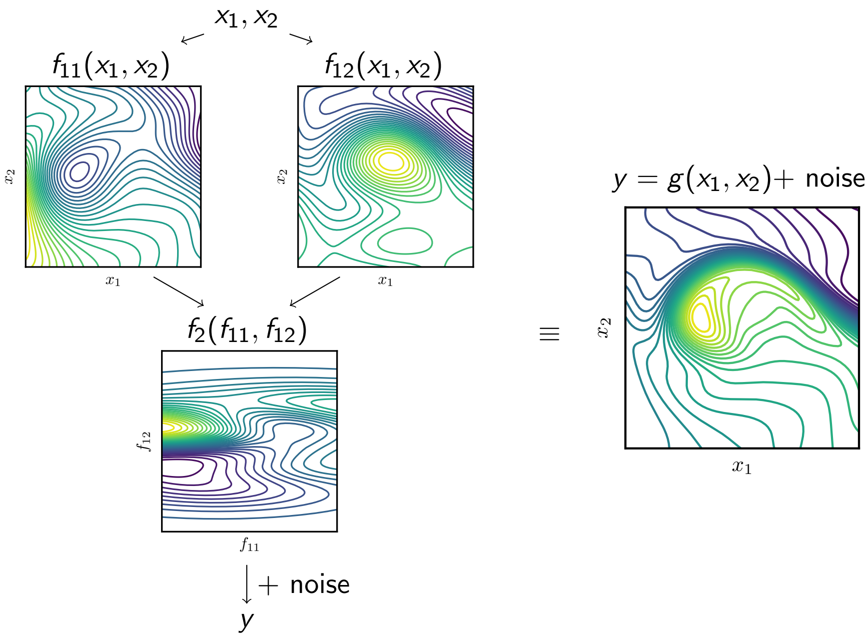
Motivation for Deep Gaussian Processes



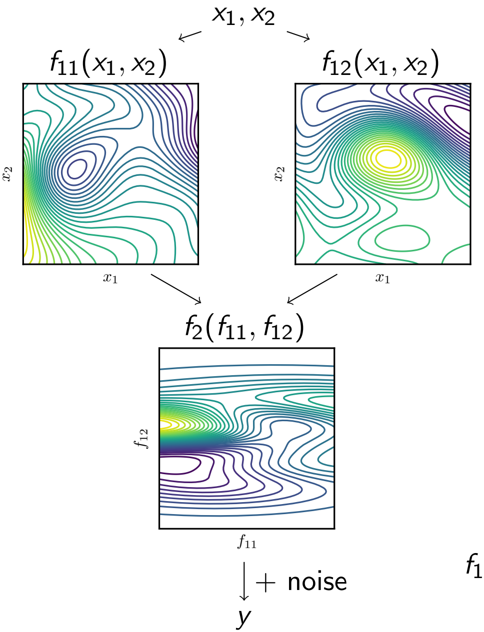
How do deep GPs work?

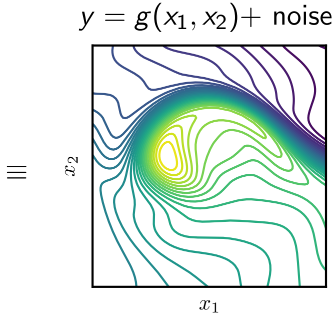


How do deep GPs work?



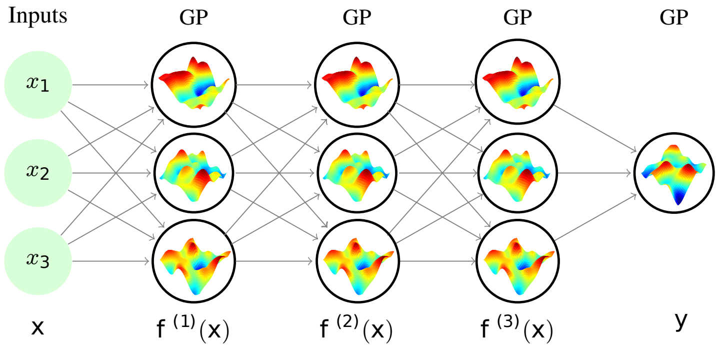
How do deep GPs work?





$f_{11}, f_{12}, f_2 \sim \mathcal{GP}(0, C(\cdot, \cdot))$

Deep GPs as Deep Neural Networks

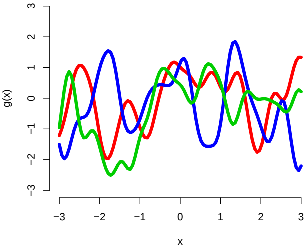


Deep GPs: Composition of Functions

$$y = f(g(\mathbf{x}))\,, \quad f(\mathbf{x}) \sim \mathcal{GP}(0, C_f(\mathbf{x}, \mathbf{x}')) \quad g(\mathbf{x}) \sim \mathcal{GP}(0, C_g(\mathbf{x}, \mathbf{x}'))$$

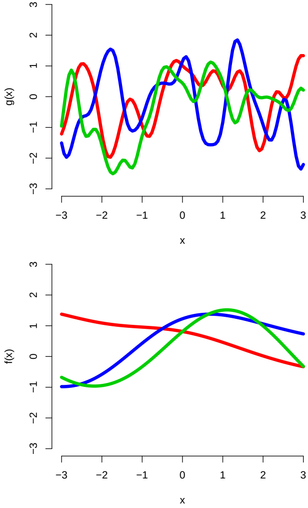
Deep GPs: Composition of Functions

$$y = f(g(\mathbf{x}))\,, \quad f(\mathbf{x}) \sim \mathcal{GP}(0, C_f(\mathbf{x}, \mathbf{x}')) \quad g(\mathbf{x}) \sim \mathcal{GP}(0, C_g(\mathbf{x}, \mathbf{x}'))$$



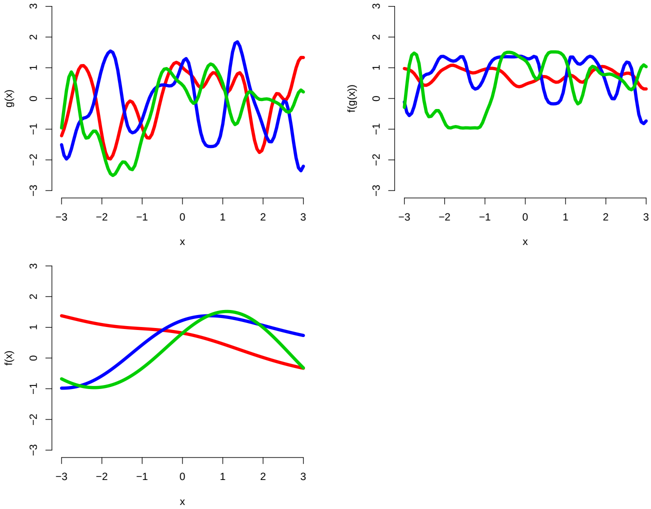
Deep GPs: Composition of Functions

$$y = f(g(\mathbf{x}))\,, \quad f(\mathbf{x}) \sim \mathcal{GP}(0, C_f(\mathbf{x}, \mathbf{x}')) \quad g(\mathbf{x}) \sim \mathcal{GP}(0, C_g(\mathbf{x}, \mathbf{x}'))$$



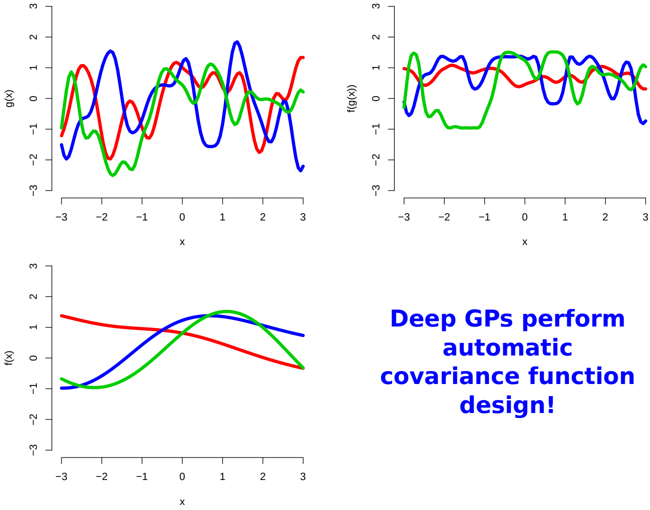
Deep GPs: Composition of Functions

$$y = f(g(\mathbf{x}))\,, \quad f(\mathbf{x}) \sim \mathcal{GP}(0, C_f(\mathbf{x}, \mathbf{x}')) \quad g(\mathbf{x}) \sim \mathcal{GP}(0, C_g(\mathbf{x}, \mathbf{x}'))$$



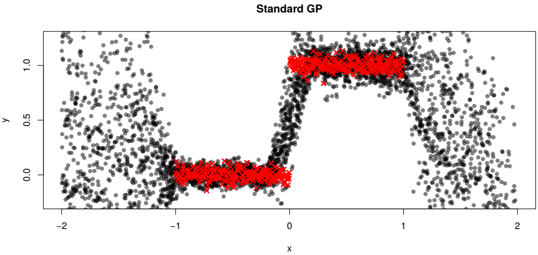
Deep GPs: Composition of Functions

$$y = f(g(\mathbf{x})), \quad f(\mathbf{x}) \sim \mathcal{GP}(0, C_f(\mathbf{x}, \mathbf{x}')) \quad g(\mathbf{x}) \sim \mathcal{GP}(0, C_g(\mathbf{x}, \mathbf{x}'))$$

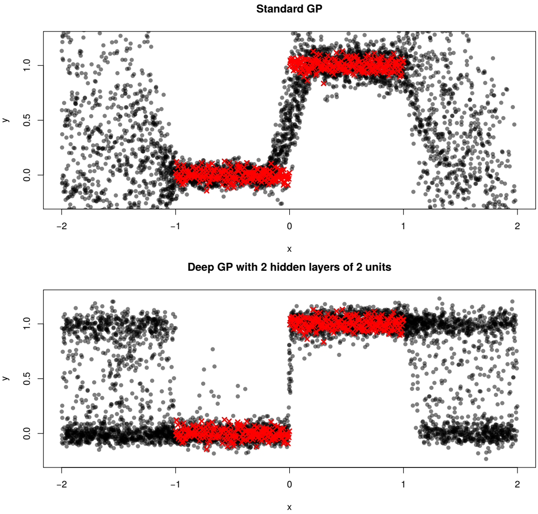


Deep GPs perform automatic covariance function design!

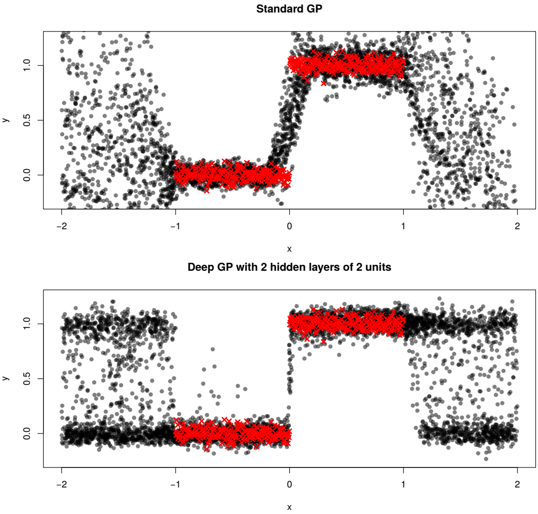
Deep GP Predictive Distribution



Deep GP Predictive Distribution



Deep GP Predictive Distribution



In a deep GP the predictive distribution needs not be Gaussian!

Why deep GPs?

Advantages:

Why deep GPs?

Advantages:

- Useful input warping: automatic, non-parametric kernel design.

Why deep GPs?

Advantages:

- Useful input warping: automatic, non-parametric kernel design.
- Repair damage done by sparse approximations to GPs.

Why deep GPs?

Advantages:

- Useful input warping: automatic, non-parametric kernel design.
- Repair damage done by sparse approximations to GPs.
- More accurate predictions and better uncertainty estimates.

Why deep GPs?

Advantages:

- Useful input warping: automatic, non-parametric kernel design.
- Repair damage done by sparse approximations to GPs.
- More accurate predictions and better uncertainty estimates.
- Better cost scaling w.r.t. depth L rather than inducing points M.

Why deep GPs?

Advantages:

- Useful input warping: automatic, non-parametric kernel design.
- Repair damage done by sparse approximations to GPs.
- More accurate predictions and better uncertainty estimates.
- Better cost scaling w.r.t. depth L rather than inducing points M.

Drawbacks:

Why deep GPs?

Advantages:

- Useful input warping: automatic, non-parametric kernel design.
- Repair damage done by sparse approximations to GPs.
- More accurate predictions and better uncertainty estimates.
- Better cost scaling w.r.t. depth L rather than inducing points M.

Drawbacks:

- Require complicated approximate inference methods.

Why deep GPs?

Advantages:

- Useful input warping: automatic, non-parametric kernel design.
- Repair damage done by sparse approximations to GPs.
- More accurate predictions and better uncertainty estimates.
- Better cost scaling w.r.t. depth L rather than inducing points M.

Drawbacks:

- Require complicated approximate inference methods.
- High computational cost of approximate inference.

Bayesian inference

Posterior over latent functions (typically at the observed data $\mathbf{X}$):

$$p(\mathbf{f}^1, \mathbf{f}^2, \mathbf{f}^3 | \mathbf{Y}) = \frac{p(\mathbf{f}^1)p(\mathbf{f}^2)p(\mathbf{f}^3)\; p(\mathbf{Y}|\mathbf{f}^1, \mathbf{f}^2, \mathbf{f}^3, \mathbf{X})}{p(\mathbf{Y})}$$

- GP priors
- Likelihood function
- Marginal likelihood

But the posterior $p(\mathbf{f}^1, \mathbf{f}^2, \mathbf{f}^3 | \mathbf{Y})$ is **intractable**.

Inducing Points Representation

Latent variables: from $\mathcal{O}(N)$ to $\mathcal{O}(M)$, with $M \ll N$.

Distribution on f given by GP with inducing inputs $\bar{\mathbf{X}}$ and outputs $\mathbf{u}$.

Inducing Points Representation

Latent variables: from $\mathcal{O}(N)$ to $\mathcal{O}(M)$, with $M \ll N$.

Distribution on f given by GP with inducing inputs $\bar{\mathbf{X}}$ and outputs $\mathbf{u}$.

If $\mathbf{u}$ is known, then $p(f(\mathbf{x}^\star)|\mathbf{u}) = \mathcal{N}(f(\mathbf{x}^\star)|m^\star, v^\star)$, where

$$
\begin{aligned}
m^\star &= \mathbf{\Sigma}_{f^\star,\mathbf{u}}\mathbf{\Sigma}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{u}\,,\\
v^\star &= \mathbf{\Sigma}_{f^\star,f^\star} - \mathbf{\Sigma}_{f^\star,\mathbf{u}}\mathbf{\Sigma}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{\Sigma}_{\mathbf{u},f^\star}\,.
\end{aligned}
$$

Inducing Points Representation

Latent variables: from $\mathcal{O}(N)$ to $\mathcal{O}(M)$, with $M \ll N$.

Distribution on f given by GP with inducing inputs $\bar{\mathbf{X}}$ and outputs $\mathbf{u}$.

If $\mathbf{u}$ is known, then $p(f(\mathbf{x}^\star)|\mathbf{u}) = \mathcal{N}(f(\mathbf{x}^\star)|m^\star, v^\star)$, where

$$m^\star = \boldsymbol{\Sigma}_{f^\star,\mathbf{u}}\boldsymbol{\Sigma}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{u}\,,$$
$$v^\star = \boldsymbol{\Sigma}_{f^\star,f^\star} - \boldsymbol{\Sigma}_{f^\star,\mathbf{u}}\boldsymbol{\Sigma}_{\mathbf{u},\mathbf{u}}^{-1}\boldsymbol{\Sigma}_{\mathbf{u},f^\star}\,.$$

If $p(\mathbf{u}) = \mathcal{N}(\mathbf{u}|\mathbf{m},\mathbf{S})$, then $p(f(\mathbf{x}^\star)) = \mathcal{N}(f(\mathbf{x}^\star)|m^\star, v^\star)$, where

$$m^\star = \boldsymbol{\Sigma}_{f^\star,\mathbf{u}}\boldsymbol{\Sigma}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{m}\,,$$
$$v^\star = \boldsymbol{\Sigma}_{f^\star,f^\star} - \boldsymbol{\Sigma}_{f^\star,\mathbf{u}}\boldsymbol{\Sigma}_{\mathbf{u},\mathbf{u}}^{-1}\boldsymbol{\Sigma}_{\mathbf{u},f^\star} + \boldsymbol{\Sigma}_{f^\star,\mathbf{u}}\boldsymbol{\Sigma}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{S}\boldsymbol{\Sigma}_{\mathbf{u},\mathbf{u}}^{-1}\boldsymbol{\Sigma}_{\mathbf{u},f^\star}$$

Inducing Points Representation

Latent variables: from $\mathcal{O}(N)$ to $\mathcal{O}(M)$, with $M \ll N$.

Distribution on f given by GP with inducing inputs $\bar{\mathbf{X}}$ and outputs $\mathbf{u}$.

If $\mathbf{u}$ is known, then $p(f(\mathbf{x}^\star)|\mathbf{u}) = \mathcal{N}(f(\mathbf{x}^\star)|m^\star, v^\star)$, where

$$
\begin{aligned}
m^\star &= \boldsymbol{\Sigma}_{f^\star,\mathbf{u}} \boldsymbol{\Sigma}_{\mathbf{u},\mathbf{u}}^{-1} \mathbf{u}\,, \\
v^\star &= \boldsymbol{\Sigma}_{f^\star,f^\star} - \boldsymbol{\Sigma}_{f^\star,\mathbf{u}} \boldsymbol{\Sigma}_{\mathbf{u},\mathbf{u}}^{-1} \boldsymbol{\Sigma}_{\mathbf{u},f^\star}\,.
\end{aligned}
$$

If $p(\mathbf{u}) = \mathcal{N}(\mathbf{u}|\mathbf{m}, \mathbf{S})$, then $p(f(\mathbf{x}^\star)) = \mathcal{N}(f(\mathbf{x}^\star)|m^\star, v^\star)$, where

$$
\begin{aligned}
m^\star &= \boldsymbol{\Sigma}_{f^\star,\mathbf{u}} \boldsymbol{\Sigma}_{\mathbf{u},\mathbf{u}}^{-1} \mathbf{m}\,, \\
v^\star &= \boldsymbol{\Sigma}_{f^\star,f^\star} - \boldsymbol{\Sigma}_{f^\star,\mathbf{u}} \boldsymbol{\Sigma}_{\mathbf{u},\mathbf{u}}^{-1} \boldsymbol{\Sigma}_{\mathbf{u},f^\star} + \boldsymbol{\Sigma}_{f^\star,\mathbf{u}} \boldsymbol{\Sigma}_{\mathbf{u},\mathbf{u}}^{-1} \mathbf{S} \boldsymbol{\Sigma}_{\mathbf{u},\mathbf{u}}^{-1} \boldsymbol{\Sigma}_{\mathbf{u},f^\star}
\end{aligned}
$$

Given u or a Gaussian for u, $f(\mathbf{x}^\star)$ is fully specified!

Deep GPs Joint Distribution

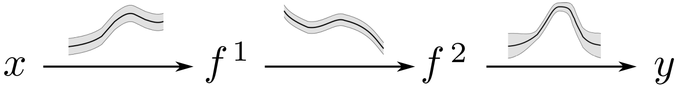
$$p(\mathbf{y}, \{\mathbf{u}^l, \mathbf{f}^l\}_{l=1}^L) = \overbrace{\prod_{i=1}^{N} p(y_i | f_i^L)}^{\text{Likelihood}} \times$$
$$\underbrace{\prod_{l=1}^{L} p(\mathbf{f}^l | \mathbf{u}^l, \overline{\mathbf{X}}^l) p(\mathbf{u}^l | \overline{\mathbf{X}}^l)}_{\text{Deep GP prior}}$$

Deep GPs Joint Distribution

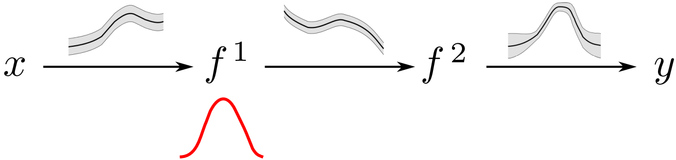
$$p(\mathbf{y}, \{\mathbf{u}^l, \mathbf{f}^l\}_{l=1}^{L}) = \overbrace{\prod_{i=1}^{N} p(y_i | f_i^L)}^{\text{Likelihood}} \times \underbrace{\prod_{l=1}^{L} p(\mathbf{f}^l | \mathbf{u}^l, \overline{\mathbf{X}}^l) p(\mathbf{u}^l | \overline{\mathbf{X}}^l)}_{\text{Deep GP prior}}$$

Ideally we would like to make inference about $\{\mathbf{u}^l, \mathbf{f}^l\}_{l=1}^{L}$!

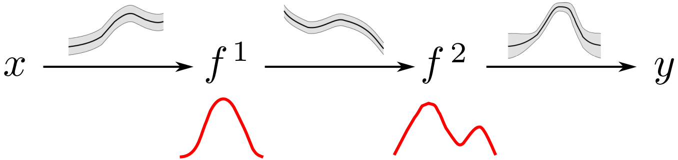
Challenges of Approximate Inference for DGPs



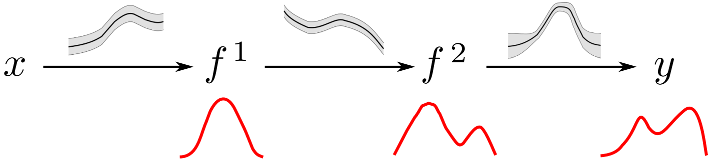
Challenges of Approximate Inference for DGPs



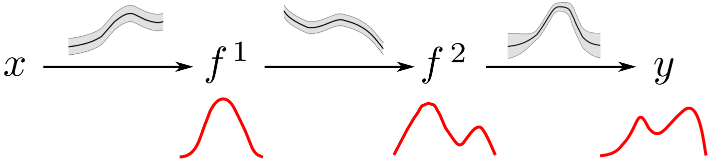
Challenges of Approximate Inference for DGPs



Challenges of Approximate Inference for DGPs



Challenges of Approximate Inference for DGPs



The predictive distribution after the first layer is non Gaussian!

Methods for Training DGPs

- Using VI and an analytic lower bound.

Methods for Training DGPs

- Using VI and an analytic lower bound.
- Using approximate expectation propagation.

Methods for Training DGPs

- Using VI and an analytic lower bound.
- Using approximate expectation propagation.
- Using stochastic variational inference.

Methods for Training DGPs

- Using VI and an analytic lower bound.
- Using approximate expectation propagation.
- Using stochastic variational inference.
- By minimizing alpha divergences.

Analytic ELBO via Variational Inference

The early attempts for approximate inference in DGPs considered fixed q that lead to an analytic ELBO!

Analytic ELBO via Variational Inference

The early attempts for approximate inference in DGPs considered fixed q that lead to an analytic ELBO!

For this, noisy versions of the variables at each layer **but last** are introduced:

$$\tilde{\mathbf{f}}^l = \mathbf{f}^l + \boldsymbol{\epsilon}\,, \qquad\qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda}_l)\,,$$

with $\boldsymbol{\Lambda}_l$ a diagonal matrix for $l = 1, \ldots, L-1$.

Analytic ELBO via Variational Inference

The early attempts for approximate inference in DGPs considered fixed q that lead to an analytic ELBO!

For this, noisy versions of the variables at each layer **but last** are introduced:

$$\tilde{\mathbf{f}}^l = \mathbf{f}^l + \boldsymbol{\epsilon}\,, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda}_l)\,,$$

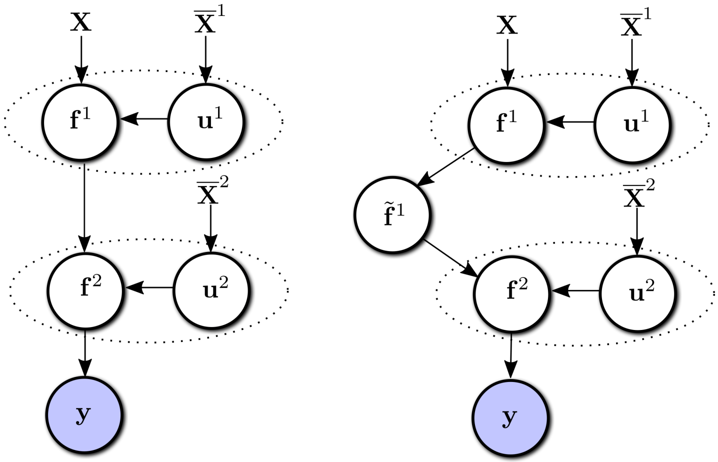
with $\boldsymbol{\Lambda}_l$ a diagonal matrix for $l = 1, \ldots, L-1$.

The joint distribution is now:

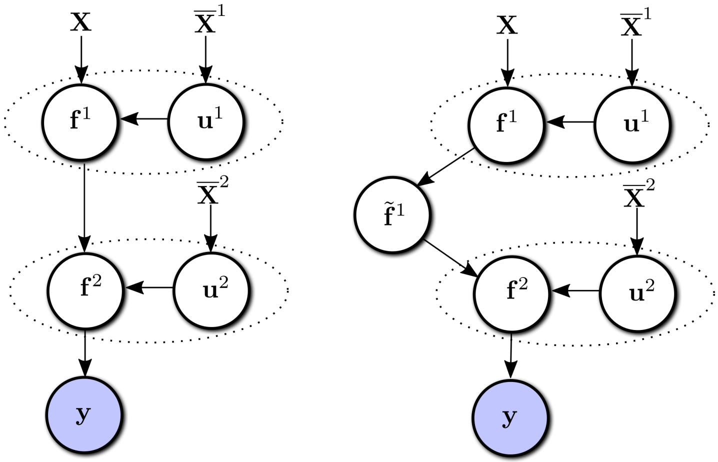
$$p(\mathbf{y}, \{\mathbf{u}^l, \mathbf{f}^l\}_{l=1}^{L}, \{\tilde{\mathbf{f}}^l\}_{l=1}^{L-1}) = \overbrace{\prod_{i=1}^{N} p(y_i|f_i^L)}^{\text{Likelihood}} \times$$

$$\underbrace{p(\mathbf{f}^L|\mathbf{u}^L, \overline{\mathbf{X}}^L) p(\mathbf{u}^L|\overline{\mathbf{X}}^L) \prod_{l=1}^{L-1} p(\tilde{\mathbf{f}}^l|\mathbf{f}^l) p(\mathbf{f}^l|\mathbf{u}^l, \overline{\mathbf{X}}^l) p(\mathbf{u}^l|\overline{\mathbf{X}}^l)}_{\text{Deep GP prior}}$$

Original Graphical Model and Extended



Original Graphical Model and Extended



Both models are equivalent, but this setting simplifies inference!

Analytic ELBO via Variational Inference

The posterior approximation q considered assumes independence among layers!

Analytic ELBO via Variational Inference

The posterior approximation q considered assumes independence among layers!

Posterior approximation:

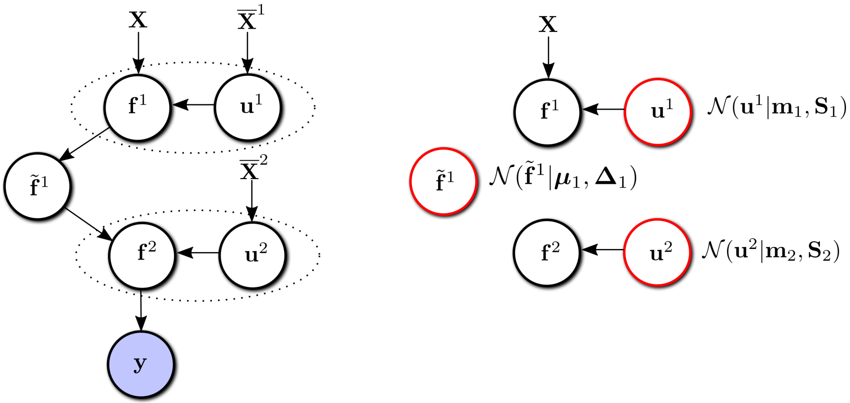
$$q(\{\mathbf{u}^l, \mathbf{f}^l, \tilde{\mathbf{f}}^l\}_{l=1}^{L}) = q(\mathbf{u}^L) p(\mathbf{f}^L | \mathbf{u}^L, \overline{\mathbf{X}}^l) \prod_{l=1}^{L-1} q(\mathbf{u}^l) q(\tilde{\mathbf{f}}^l) p(\mathbf{f}^l | \mathbf{u}^l, \overline{\mathbf{X}}^l)\,,$$

where the input to the layer $l+1$ is $\tilde{\mathbf{f}}^l$ and

$$q(\mathbf{u}^l) = \mathcal{N}(\mathbf{u}^l | \mathbf{m}_l, \mathbf{S}_l)\,, \qquad q(\tilde{\mathbf{f}}^l) = \mathcal{N}(\tilde{\mathbf{f}}^l | \boldsymbol{\mu}_l, \boldsymbol{\Delta}_l)\,,$$

with $\boldsymbol{\Delta}_l$ a diagonal matrix.

Graphical Model and Approximate Distribution



Analytic Variational Inference for DGPs

Minimizes KL$(q(\{\mathbf{u}^l, \mathbf{f}^l\}_{l=1}^L, \{\tilde{\mathbf{f}}^l\}_{l=1}^{L-1}) | p(\{\mathbf{u}^l, \mathbf{f}^l\}_{l=1}^L, \{\tilde{\mathbf{f}}^l\}_{l=1}^{L-1} | \mathbf{y}))$

(Damianou, 2013)

Analytic Variational Inference for DGPs

Minimizes $\mathbf{KL}(q(\{\mathbf{u}^l, \mathbf{f}^l\}_{l=1}^{L}, \{\tilde{\mathbf{f}}^l\}_{l=1}^{L-1}) | p(\{\mathbf{u}^l, \mathbf{f}^l\}_{l=1}^{L}, \{\tilde{\mathbf{f}}^l\}_{l=1}^{L-1} | \mathbf{y}))$

Equivalent to maximizing the lower bound on $\log p(\mathbf{y})$:

$$\mathcal{L} = \mathbb{E}_q \left[\log \frac{\prod_{i=1}^{N} p(y_i | f_i^L) \cancel{p(\mathbf{f}^L | \mathbf{u}^L)} p(\mathbf{u}^L) \prod_{l=1}^{L-1} p(\tilde{\mathbf{f}}^l | \mathbf{f}^l) \cancel{p(\mathbf{f}^l | \mathbf{u}^l)} p(\mathbf{u}^l)}{\cancel{p(\mathbf{f}^L | \mathbf{u}^L)} q(\mathbf{u}^L) \prod_{l=1}^{L-1} q(\tilde{\mathbf{f}}^l) \cancel{p(\mathbf{f}^l | \mathbf{u}^l)} q(\mathbf{u}^l)} \right] .$$

$$= \sum_{i=1}^{N} \mathbb{E}_q[\log p(y_i | f_i^L)] + \sum_{l=1}^{L-1} \left[\mathbb{E}_q[\log p(\tilde{\mathbf{f}}^l | \mathbf{f}^l)] + H[q(\tilde{\mathbf{f}}^l)] \right]$$

$$+ \sum_{l=1}^{L} \mathrm{KL}(q(\mathbf{u}^l) | p(\mathbf{u}^l)) .$$

(Damianou, 2013)

Analytic Variational Inference for DGPs

Minimizes KL$(q(\{\mathbf{u}^l, \mathbf{f}^l\}_{l=1}^L, \{\tilde{\mathbf{f}}^l\}_{l=1}^{L-1}) | p(\{\mathbf{u}^l, \mathbf{f}^l\}_{l=1}^L, \{\tilde{\mathbf{f}}^l\}_{l=1}^{L-1} | \mathbf{y}))$

Equivalent to maximizing the lower bound on $\log p(\mathbf{y})$:

$$\mathcal{L} = \mathbb{E}_q \left[\log \frac{\prod_{i=1}^N p(y_i | f_i^L) \cancel{p(\mathbf{f}^L | \mathbf{u}^L)} p(\mathbf{u}^L) \prod_{l=1}^{L-1} p(\tilde{\mathbf{f}}^l | \mathbf{f}^l) \cancel{p(\mathbf{f}^l | \mathbf{u}^l)} p(\mathbf{u}^l)}{\cancel{p(\mathbf{f}^L | \mathbf{u}^L)} q(\mathbf{u}^L) \prod_{l=1}^{L-1} q(\tilde{\mathbf{f}}^l) \cancel{p(\mathbf{f}^l | \mathbf{u}^l)} q(\mathbf{u}^l)} \right] .$$

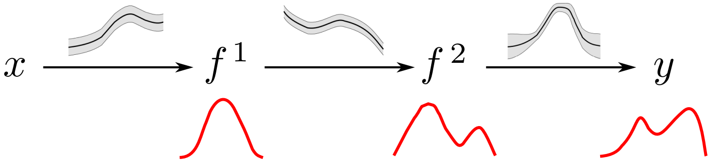
$$= \sum_{i=1}^N \mathbb{E}_q[\log p(y_i | f_i^L)] + \sum_{l=1}^{L-1} \left[\mathbb{E}_q[\log p(\tilde{\mathbf{f}}^l | \mathbf{f}^l)] + H[q(\tilde{\mathbf{f}}^l)] \right]$$

$$+ \sum_{l=1}^{L} \mathrm{KL}(q(\mathbf{u}^l) | p(\mathbf{u}^l)) .$$

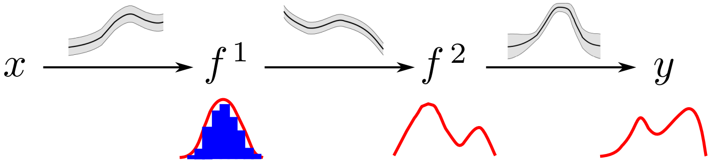
Which can be evaluated in closed-form (form some cov. functions) and maximized to find q and good model hyper-parameters!

(Damianou, 2013)

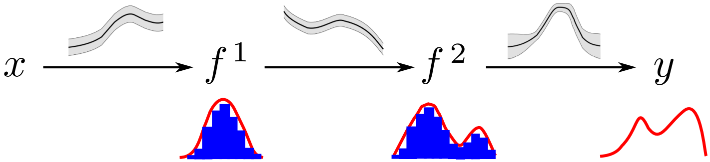
Predictive Distribution via Monte Carlo Sampling



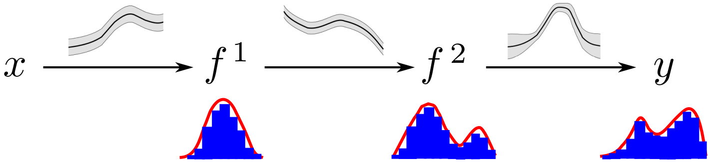
Predictive Distribution via Monte Carlo Sampling



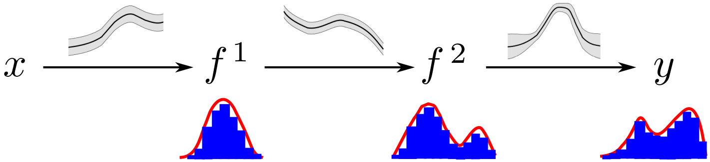
Predictive Distribution via Monte Carlo Sampling



Predictive Distribution via Monte Carlo Sampling

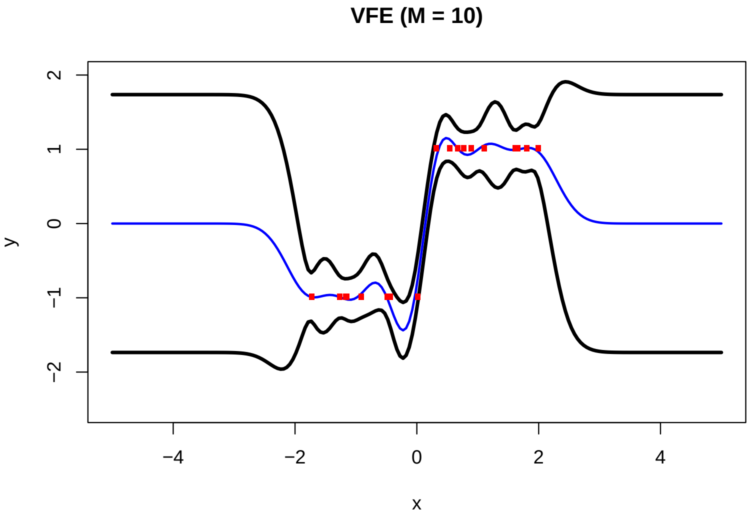


Predictive Distribution via Monte Carlo Sampling

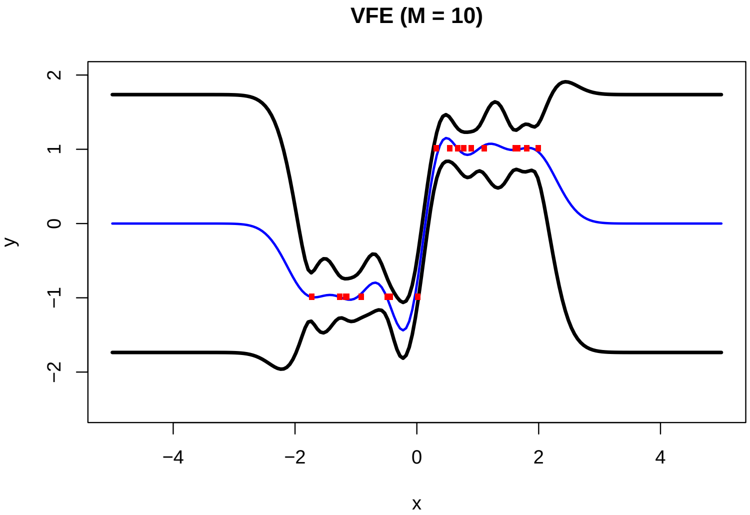


For a particular fixed input, the predictive distribution of each layer is Gaussian!

DGPs Tractable Bound: Illustrative Example

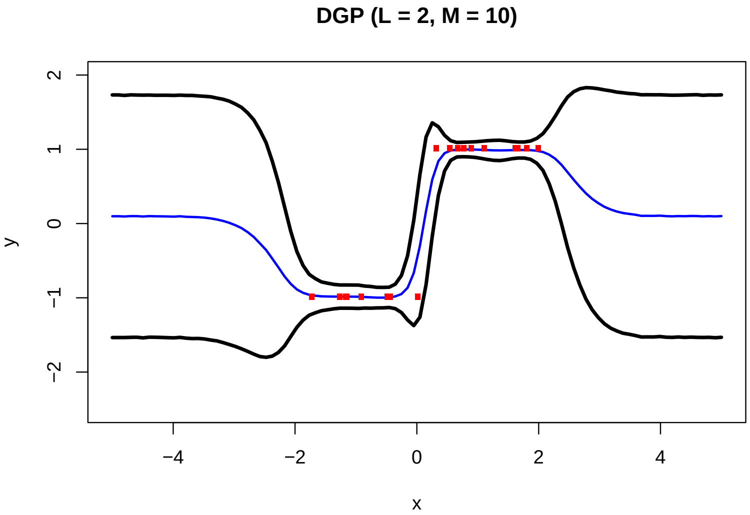


DGPs Tractable Bound: Illustrative Example

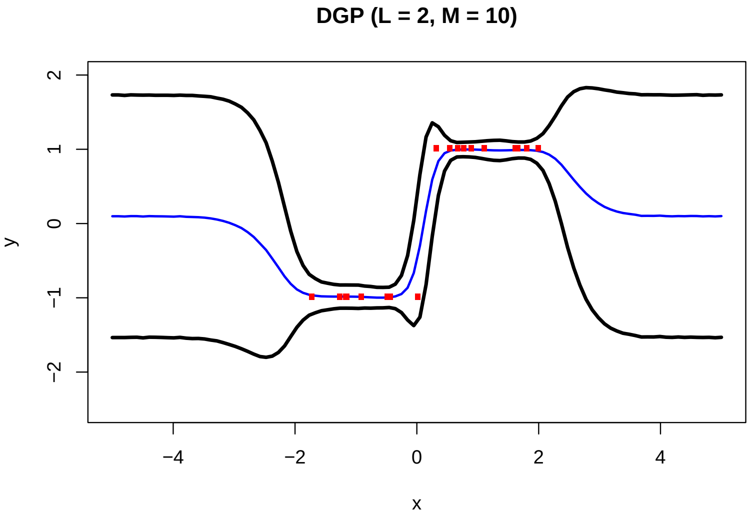


The VFE sparse GP reduces the length-scale to explain the data!

DGPs Tractable Bound: Illustrative Example



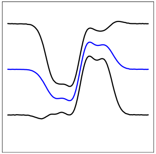
DGPs Tractable Bound: Illustrative Example



The DGP provides a more sensible predictive distribution!

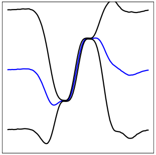
DGPs Tractable Bound: Illustrative Example

$$x$$
$$\downarrow$$
$$f^1(x)$$



$$\Big\downarrow + \text{noise}$$

$$f^2(\tilde{f}_1)$$



$$\Big\downarrow + \text{noise}$$

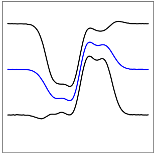
$$y$$

DGPs Tractable Bound: Illustrative Example

x

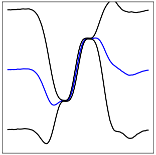
$\downarrow$

$f^1(x)$



$\downarrow$ + noise

$f^2(\tilde{f}_1)$

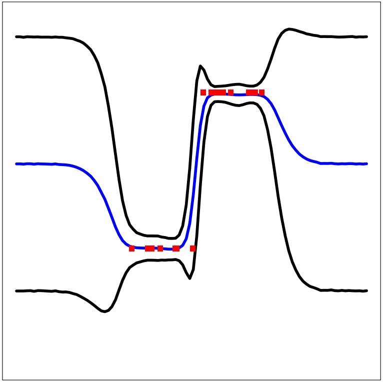


$\downarrow$ + noise

y

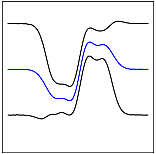
$\equiv$

$y = f^2(f^1(x) + \text{noise}) + \text{noise}$



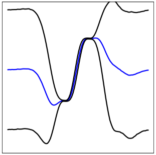
DGPs Tractable Bound: Illustrative Example

$$x \downarrow f^1(x)$$



$\downarrow$ + noise

$f^2(\tilde{f}_1)$

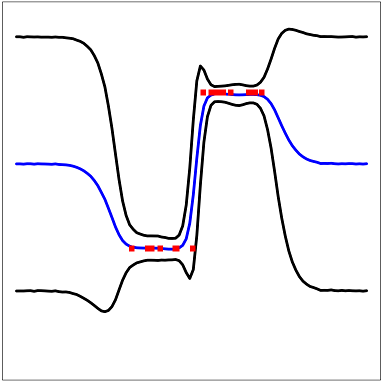


$\downarrow$ + noise

y

$\equiv$

$$y = f^2(f^1(x) + \text{noise}) + \text{ noise}$$



$$f^1, f^2 \sim \mathcal{GP}(0, C(\cdot,\cdot))$$

Limitations of DGPs via Tractable VI Bound

- The posterior approximation q assumes independence between layers inputs and outputs.

Limitations of DGPs via Tractable VI Bound

- The posterior approximation q assumes independence between layers inputs and outputs.

- The tractable VI bound is limited to certain covariance functions, *e.g.*, the squared exponential covariance function.

Limitations of DGPs via Tractable VI Bound

- The posterior approximation q assumes independence between layers inputs and outputs.
- The tractable VI bound is limited to certain covariance functions, *e.g.*, the squared exponential covariance function.
- The original method did not consider mini-batch training and scales linearly with N, which makes infeasible addressing large problems.

DGPs and Approximate Expectation Propagation

Features:

- Does not assume independence between inputs and outputs in each layer in the approximate distribution q.

DGPs and Approximate Expectation Propagation

Features:

- Does not assume independence between inputs and outputs in each layer in the approximate distribution q.
- Uses the FITC approximation for tractable scaling and allows for mini-batch training. Thus, the model is changed.

DGPs and Approximate Expectation Propagation

Features:

- Does not assume independence between inputs and outputs in each layer in the approximate distribution q.
- Uses the FITC approximation for tractable scaling and allows for mini-batch training. Thus, the model is changed.
- Relies on a modified version of EP to estimate the approximate distribution q using standard optimization techniques.

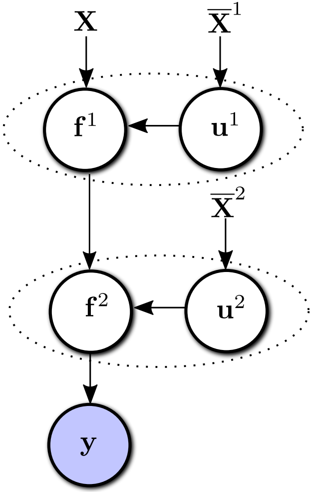
DGPs and Approximate Expectation Propagation

Features:

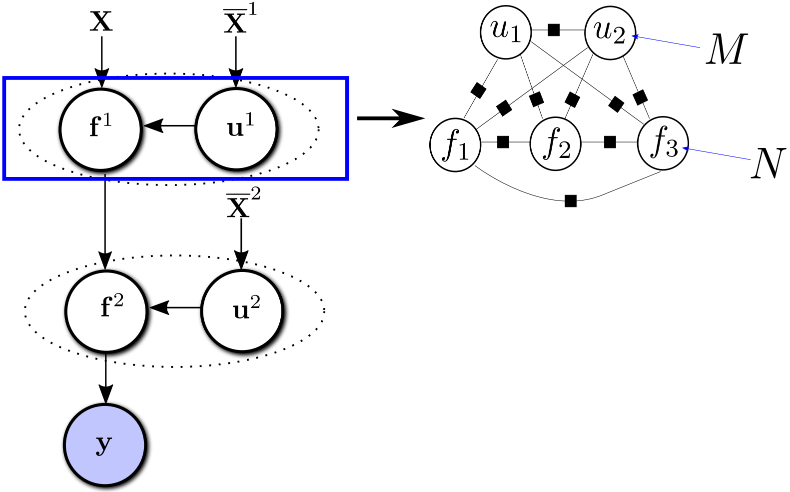
- Does not assume independence between inputs and outputs in each layer in the approximate distribution q.
- Uses the FITC approximation for tractable scaling and allows for mini-batch training. Thus, the model is changed.
- Relies on a modified version of EP to estimate the approximate distribution q using standard optimization techniques.
- The intractable predictive distribution at each layer is approximated by a Gaussian with the same moments.

(Bui et al., 2016)

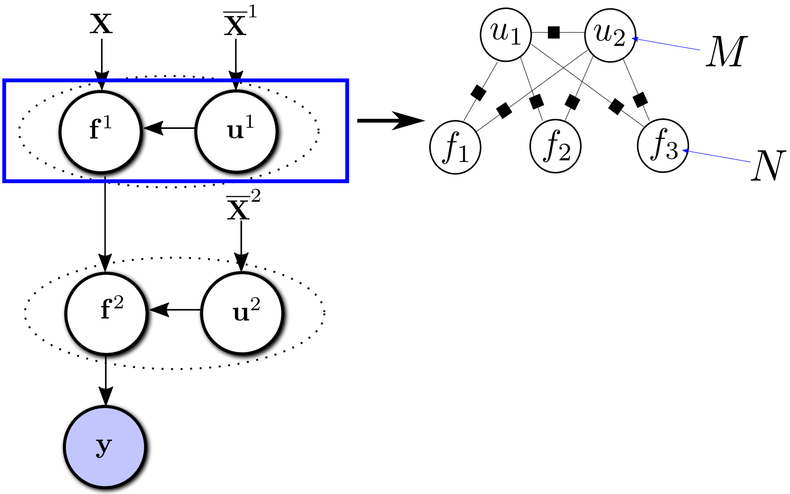
Alternative Graphical Model



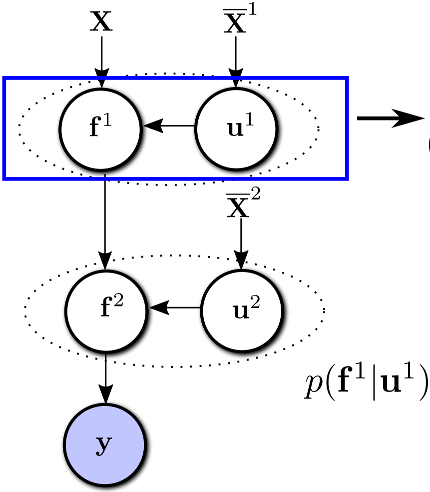
Alternative Graphical Model

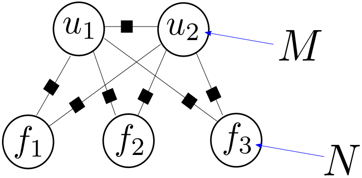


Alternative Graphical Model



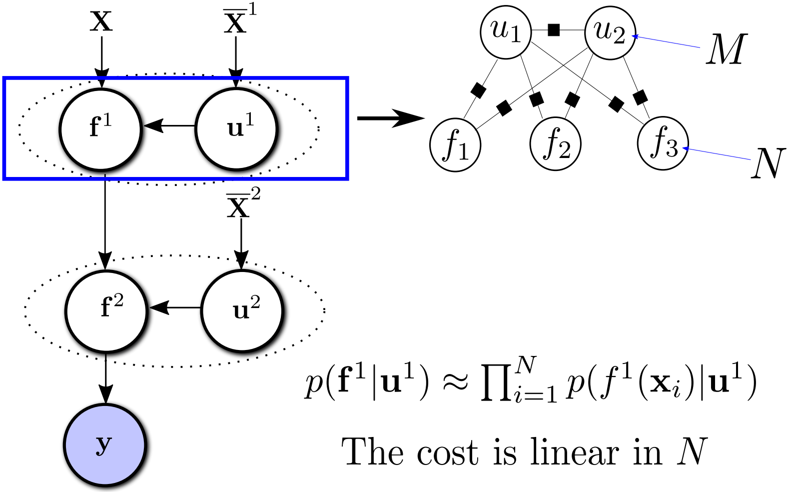
Alternative Graphical Model





$$p(\mathbf{f}^1|\mathbf{u}^1) \approx \prod_{i=1}^{N} p(f^1(\mathbf{x}_i)|\mathbf{u}^1)$$

Alternative Graphical Model



$$p(\mathbf{f}^1|\mathbf{u}^1) \approx \prod_{i=1}^{N} p(f^1(\mathbf{x}_i)|\mathbf{u}^1)$$

The cost is linear in N

Approximate Deep GP Joint Distribution

$$p(\mathbf{y}, \{\mathbf{u}^l, \mathbf{f}^l\}_{l=1}^L) = \overbrace{\prod_{i=1}^N p(y_i | f_i^L)}^{\text{Likelihood}} \times$$

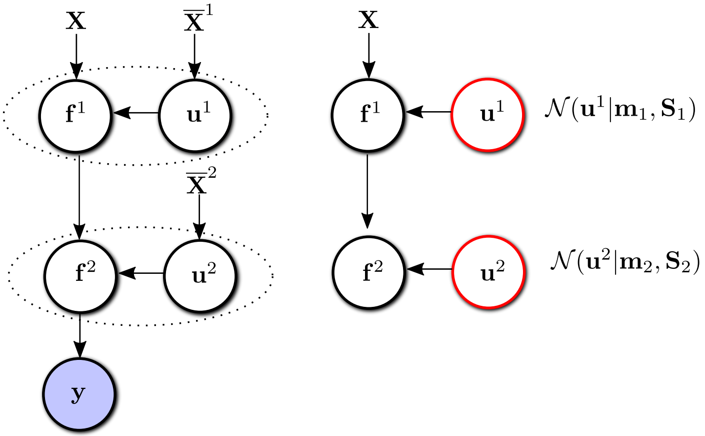
$$\underbrace{\prod_{l=1}^L \tilde{p}(\mathbf{f}^l | \mathbf{u}^l, \overline{\mathbf{X}}^l) p(\mathbf{u}^l | \overline{\mathbf{X}}^l)}_{\text{Approximate Deep GP prior } \tilde{p}(\{\mathbf{f}^l, \mathbf{u}^l\}_{l=1}^L)}$$

Approximate Deep GP Joint Distribution

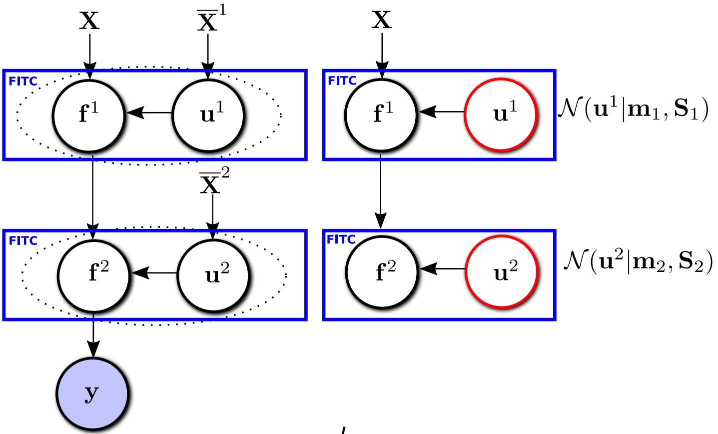
$$p(\mathbf{y}, \{\mathbf{u}^l, \mathbf{f}^l\}_{l=1}^L) = \overbrace{\prod_{i=1}^N p(y_i | f_i^L)}^{\text{Likelihood}} \times \underbrace{\prod_{l=1}^L \tilde{p}(\mathbf{f}^l | \mathbf{u}^l, \overline{\mathbf{X}}^l) p(\mathbf{u}^l | \overline{\mathbf{X}}^l)}_{\text{Approximate Deep GP prior } \tilde{p}(\{\mathbf{f}^l, \mathbf{u}^l\}_{l=1}^L)}$$

The FITC approximation enforces $\tilde{p}(\mathbf{f}^l | \mathbf{u}^l, \overline{\mathbf{X}}^l)$ to factorize across the N data instances!

Graphical Model and Approximate Distribution



Graphical Model and Approximate Distribution



$$q(\{\mathbf{f}^l, \mathbf{u}^l\}_{l=1}^L) = \prod_{l=1}^L \tilde{p}(\mathbf{f}^{l-1}|\mathbf{u}^l)\, q(\mathbf{u}^l)$$

- Fixed and factorizing across data
- Tunable Gaussian

Graphical Illustration of EP for DGPs

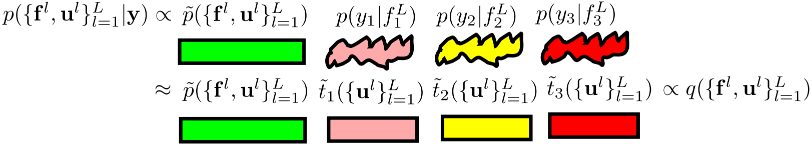
Approximates $\boxed{p(\{\mathbf{f}^l, \mathbf{u}^l\}_{l=1}^L | \mathbf{y}) \propto \tilde{p}(\{\mathbf{f}^l, \mathbf{u}^l\}_{l=1}^L) \prod_{i=1}^N p(y_i | f_i^L)}$ with

$\boxed{q(\{\mathbf{f}^l, \mathbf{u}^l\}_{l=1}^L) \propto \tilde{p}(\{\mathbf{f}^l, \mathbf{u}^l\}_{l=1}^L) \prod_{i=1}^N \tilde{t}_i(\{\mathbf{u}^l\}_{l=1}^L)}$

Graphical Illustration of EP for DGPs

Approximates $\boxed{p(\{\mathbf{f}^l,\mathbf{u}^l\}_{l=1}^L|\mathbf{y}) \propto \tilde{p}(\{\mathbf{f}^l,\mathbf{u}^l\}_{l=1}^L)\prod_{i=1}^N p(y_i|f_i^L)}$ with

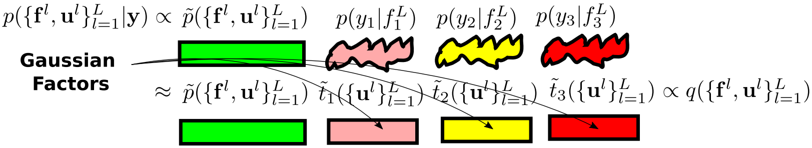
$$\boxed{q(\{\mathbf{f}^l,\mathbf{u}^l\}_{l=1}^L) \propto \tilde{p}(\{\mathbf{f}^l,\mathbf{u}^l\}_{l=1}^L)\prod_{i=1}^N \tilde{t}_i(\{\mathbf{u}^l\}_{l=1}^L)}$$



Graphical Illustration of EP for DGPs

Approximates $\boxed{p(\{\mathbf{f}^l,\mathbf{u}^l\}_{l=1}^L|\mathbf{y}) \propto \tilde{p}(\{\mathbf{f}^l,\mathbf{u}^l\}_{l=1}^L)\prod_{i=1}^N p(y_i|f_i^L)}$ with

$$\boxed{q(\{\mathbf{f}^l,\mathbf{u}^l\}_{l=1}^L) \propto \tilde{p}(\{\mathbf{f}^l,\mathbf{u}^l\}_{l=1}^L)\prod_{i=1}^N \tilde{t}_i(\{\mathbf{u}^l\}_{l=1}^L)}$$



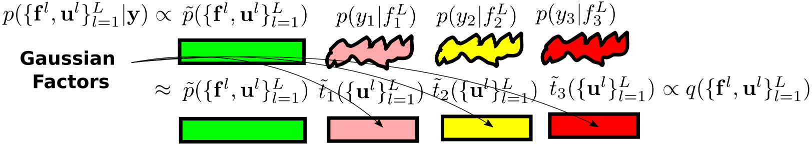
The $\tilde{t}_i$ are tuned by minimizing the KL-divergence $\mathrm{KL}[\hat{p}_i||q] \quad \forall i$,

$$\text{where} \quad \begin{array}{rcll} \hat{p}_i(\{\mathbf{f}^l,\mathbf{u}^l\}_{l=1}^L) & \propto & p(y_i|f_i^L) & \prod_{j\neq i}\tilde{t}_j(\{\mathbf{u}^l\}_{l=1}^L)\tilde{p}(\{\mathbf{f}^l,\mathbf{u}^l\}_{l=1}^L) \\ q(\{\mathbf{f}^l,\mathbf{u}^l\}_{l=1}^L) & \propto & \tilde{t}_i(\{\mathbf{u}\}_{l=1}^L) & \prod_{j\neq i}\tilde{t}_j(\{\mathbf{u}^l\}_{l=1}^L)\tilde{p}(\{\mathbf{f}^l,\mathbf{u}^l\}_{l=1}^L) \end{array}.$$

Graphical Illustration of EP for DGPs

Approximates $\boxed{p(\{\mathbf{f}^l, \mathbf{u}^l\}_{l=1}^L | \mathbf{y}) \propto \tilde{p}(\{\mathbf{f}^l, \mathbf{u}^l\}_{l=1}^L) \prod_{i=1}^N p(y_i | f_i^L)}$ with

$$\boxed{q(\{\mathbf{f}^l, \mathbf{u}^l\}_{l=1}^L) \propto \tilde{p}(\{\mathbf{f}^l, \mathbf{u}^l\}_{l=1}^L) \prod_{i=1}^N \tilde{t}_i(\{\mathbf{u}^l\}_{l=1}^L)}$$



The $\tilde{t}_i$ are tuned by minimizing the KL-divergence $\text{KL}[\hat{p}_i || q] \quad \forall i$,

$$\text{where} \quad \begin{array}{rcll} \hat{p}_i(\{\mathbf{f}^l, \mathbf{u}^l\}_{l=1}^L) & \propto & p(y_i | f_i^L) & \prod_{j \neq i} \tilde{t}_j(\{\mathbf{u}^l\}_{l=1}^L) \tilde{p}(\{\mathbf{f}^l, \mathbf{u}^l\}_{l=1}^L) \\ q(\{\mathbf{f}^l, \mathbf{u}^l\}_{l=1}^L) & \propto & \tilde{t}_i(\{\mathbf{u}\}_{l=1}^L) & \prod_{j \neq i} \tilde{t}_j(\{\mathbf{u}^l\}_{l=1}^L) \tilde{p}(\{\mathbf{f}^l, \mathbf{u}^l\}_{l=1}^L) \end{array}.$$

Since $\tilde{p}(\{\mathbf{f}^l, \mathbf{u}^l\}_{l=1}^L)$ is fixed, we only have to match the moments of $\hat{p}_j$ and q over $\{\mathbf{u}^l\}_{l=1}^L$!

EP as an Optimization Problem

The EP approximation to the **evidence** $p(\mathbf{y})$ is given by:

$$\log Z_{\text{EP}} = g(\boldsymbol{\eta}_q) - g(\boldsymbol{\eta}_{\text{prior}}) + \sum_{i=1}^{N} \log Z_i + g(\boldsymbol{\eta}_q) - g(\boldsymbol{\eta}_q^{\setminus i})$$

EP as an Optimization Problem

The EP approximation to the **evidence** $p(\mathbf{y})$ is given by:

$$\log Z_{\text{EP}} = g(\boldsymbol{\eta}_q) - g(\boldsymbol{\eta}_{\text{prior}}) + \sum_{i=1}^{N} \log Z_i + g(\boldsymbol{\eta}_q) - g(\boldsymbol{\eta}_q^{\setminus i})$$

Besides the EP updates, the EP solution for q is found by solving:

$$\max_{q} \min_{\tilde{t}_1,\ldots,\tilde{t}_N} \log Z_{\text{EP}} \quad \text{subject to} \quad q \propto \tilde{p} \prod_{i=1}^{N} \tilde{t}_i \,.$$

EP as an Optimization Problem

The EP approximation to the **evidence** $p(\mathbf{y})$ is given by:

$$\log Z_{\text{EP}} = g(\boldsymbol{\eta}_q) - g(\boldsymbol{\eta}_{\text{prior}}) + \sum_{i=1}^{N} \log Z_i + g(\boldsymbol{\eta}_q) - g(\boldsymbol{\eta}_q^{\setminus i})$$

Besides the EP updates, the EP solution for q is found by solving:

$$\max_q \min_{\tilde{t}_1,\ldots,\tilde{t}_N} \log Z_{\text{EP}} \quad \text{subject to} \quad q \propto \tilde{p} \prod_{i=1}^{N} \tilde{t}_i \,.$$

Can be solved with a **double-loop** algorithm.

EP as an Optimization Problem

The EP approximation to the **evidence** $p(\mathbf{y})$ is given by:

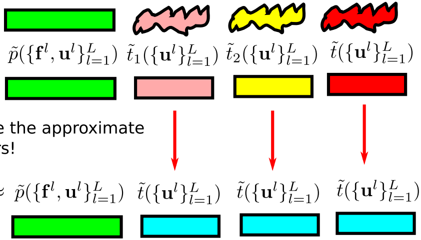
$$\log Z_{\text{EP}} = g(\boldsymbol{\eta}_q) - g(\boldsymbol{\eta}_{\text{prior}}) + \sum_{i=1}^{N} \log Z_i + g(\boldsymbol{\eta}_q) - g(\boldsymbol{\eta}_q^{\setminus i})$$

Besides the EP updates, the EP solution for q is found by solving:

$$\max_q \min_{\tilde{t}_1,\ldots,\tilde{t}_N} \log Z_{\text{EP}} \quad \text{subject to} \quad q \propto \tilde{p} \prod_{i=1}^{N} \tilde{t}_i \,.$$

Can be solved with a **double-loop** algorithm. **Too slow in practice!**

Approximate Expectation Propagation



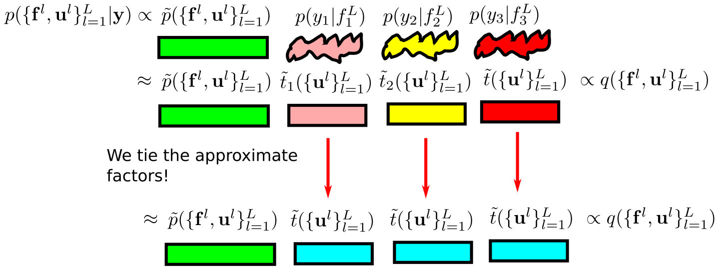
$$p(\{\mathbf{f}^l,\mathbf{u}^l\}_{l=1}^L|\mathbf{y}) \propto \tilde{p}(\{\mathbf{f}^l,\mathbf{u}^l\}_{l=1}^L) \quad p(y_1|f_1^L) \quad p(y_2|f_2^L) \quad p(y_3|f_3^L)$$

$$\approx \tilde{p}(\{\mathbf{f}^l,\mathbf{u}^l\}_{l=1}^L) \; \tilde{t}_1(\{\mathbf{u}^l\}_{l=1}^L) \; \tilde{t}_2(\{\mathbf{u}^l\}_{l=1}^L) \; \tilde{t}(\{\mathbf{u}^l\}_{l=1}^L) \; \propto q(\{\mathbf{f}^l,\mathbf{u}^l\}_{l=1}^L)$$

We tie the approximate factors!

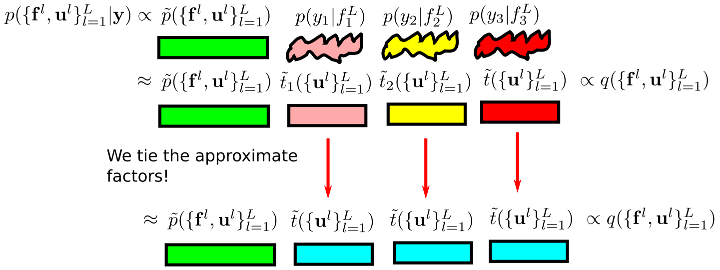
$$\approx \tilde{p}(\{\mathbf{f}^l,\mathbf{u}^l\}_{l=1}^L) \; \tilde{t}(\{\mathbf{u}^l\}_{l=1}^L) \; \tilde{t}(\{\mathbf{u}^l\}_{l=1}^L) \; \tilde{t}(\{\mathbf{u}^l\}_{l=1}^L) \; \propto q(\{\mathbf{f}^l,\mathbf{u}^l\}_{l=1}^L)$$

Approximate Expectation Propagation



- $\max\limits_{q} \min\limits_{\tilde{f}_1,\ldots,\tilde{f}_N}$ problem $\rightarrow$ $\max\limits_{q}$ problem, **no double-loop needed!**

Approximate Expectation Propagation



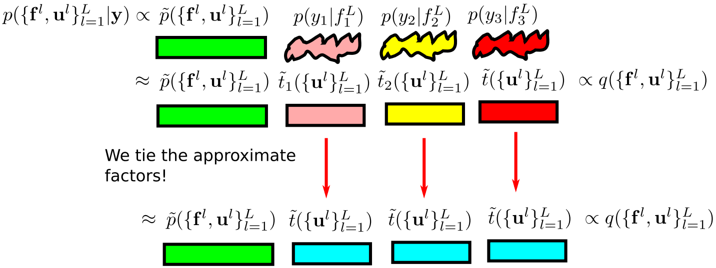
- $\max_{q} \min_{\tilde{f}_1,\ldots,\tilde{f}_N}$ problem $\rightarrow$ $\max_{q}$ problem, **no double-loop needed!**

The final objective is:

$$\log Z_{\text{EP}} = g(\boldsymbol{\eta}_q) - g(\boldsymbol{\eta}_{\text{prior}}) + \textstyle\sum_{i=1}^N \log Z_i + g(\boldsymbol{\eta}_q) - g(\boldsymbol{\eta}_q^{\text{cav}})$$

which is suitable for standard optimization and mini-batch training.

Approximate Expectation Propagation



- $\max_q \min_{\tilde{f}_1,\ldots,\tilde{f}_N}$ problem $\rightarrow$ $\max_q$ problem, **no double-loop needed!**

The final objective is:

$$\log Z_{\text{EP}} = g(\boldsymbol{\eta}_q) - g(\boldsymbol{\eta}_{\text{prior}}) + \sum_{i=1}^N \log \tilde{Z}_i + g(\boldsymbol{\eta}_q) - g(\boldsymbol{\eta}_q^{\text{cav}})$$

which is suitable for standard optimization and mini-batch training.

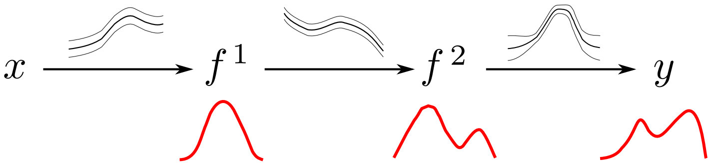
Approximating $\log Z_i$

Note that $\log Z_i = \log \int p(y_i|f_i^L) q^{\setminus i}(f_i^L) df_i^L$ is the log predictive likelihood of instance i when removed from the training set.

Approximating $\log Z_i$

Note that $\log Z_i = \log \int p(y_i|f_i^L)q^{\setminus i}(f_i^L)df_i^L$ is the log predictive likelihood of instance i when removed from the training set.

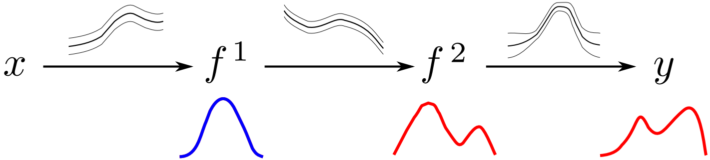
We can use an iterative Gaussian approximation:



Approximating $\log Z_i$

Note that $\log Z_i = \log \int p(y_i|f_i^L) q^{\setminus i}(f_i^L) df_i^L$ is the log predictive likelihood of instance i when removed from the training set.

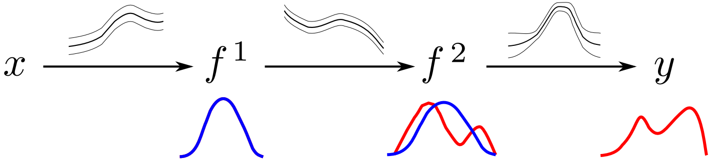
We can use an iterative Gaussian approximation:



Approximating $\log Z_i$

Note that $\log Z_i = \log \int p(y_i|f_i^L) q^{\setminus i}(f_i^L) df_i^L$ is the log predictive likelihood of instance i when removed from the training set.

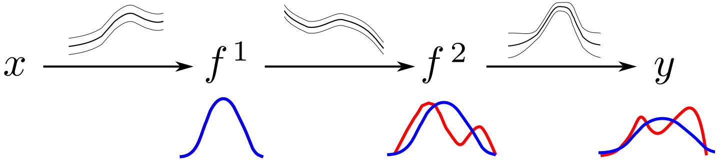
We can use an iterative Gaussian approximation:



Approximating $\log Z_i$

Note that $\log Z_i = \log \int p(y_i|f_i^L) q^{\setminus i}(f_i^L) df_i^L$ is the log predictive likelihood of instance i when removed from the training set.

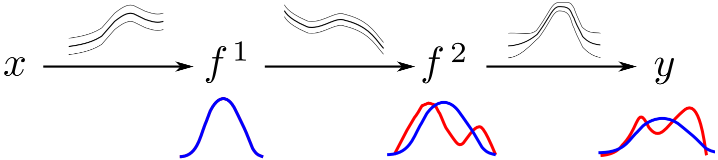
We can use an iterative Gaussian approximation:



Approximating $\log Z_i$

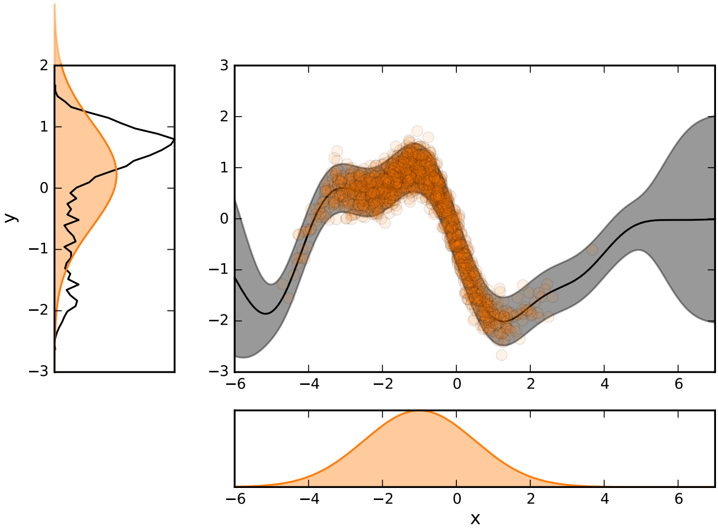
Note that $\log Z_i = \log \int p(y_i|f_i^L)q^{\setminus i}(f_i^L)df_i^L$ is the log predictive likelihood of instance i when removed from the training set.

We can use an iterative Gaussian approximation:

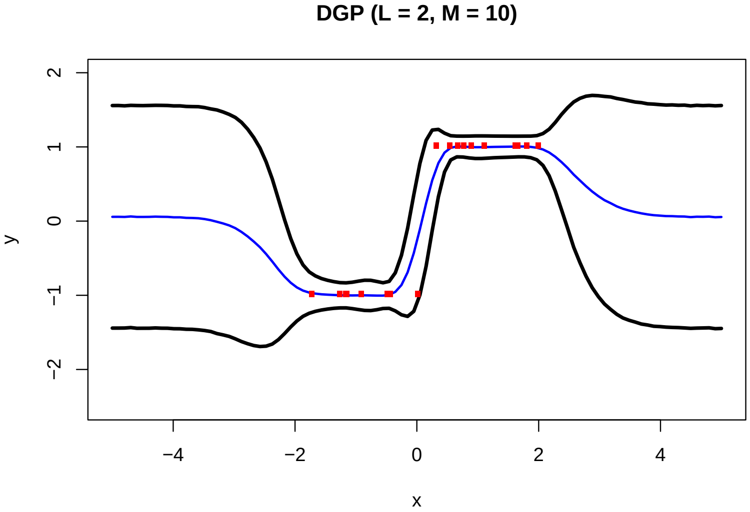


Doable for certain covariance functions, e.g., the squared exponential!

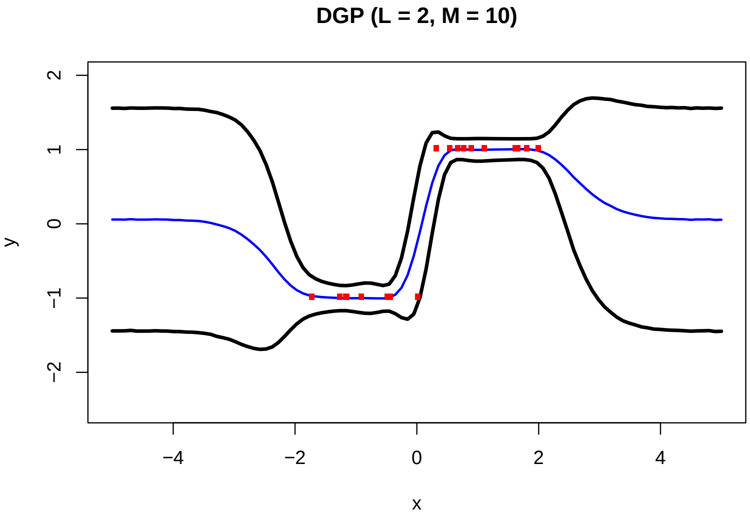
Gaussian Projection Example



Approx. EP for DGPs: Illustrative Example



Approx. EP for DGPs: Illustrative Example



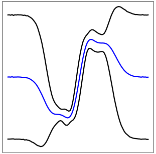
The AEP method provides a similar predictive distribution to the previous method!

Approx. EP for DGPs: Illustrative Example

x

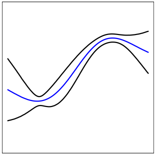
$\downarrow$

$f^1(x)$



$\downarrow$

$f^2(f_1)$



$\downarrow$

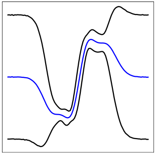
$y + \text{noise}$

Approx. EP for DGPs: Illustrative Example

x

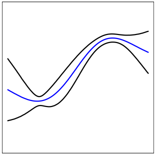
$\downarrow$

$f^1(x)$



$\downarrow$

$f^2(f_1)$

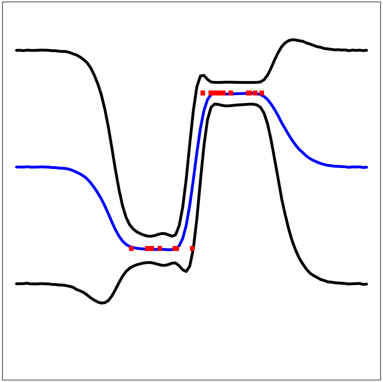


$\downarrow$

$y +$ noise

$\equiv$

$y = f^2(f^1(x)) +$ noise

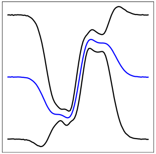


Approx. EP for DGPs: Illustrative Example

x

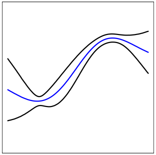
$\downarrow$

$f^1(x)$



$\downarrow$

$f^2(f_1)$

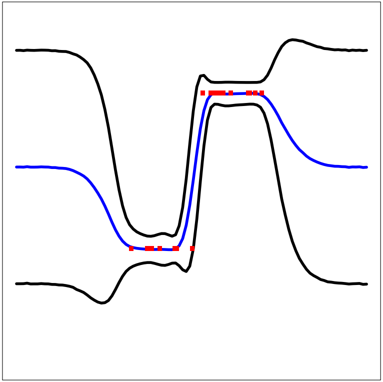


$\downarrow$

$y +$ noise

$\equiv$

$y = f^2(f^1(x)) +$ noise



$f^1, f^2 \sim \mathcal{GP}(0, C(\cdot, \cdot))$

Limitations of Approx. EP for DGPs

- The approximate predictive distribution of q at each layer is a Gaussian projection, which can be a crude approximation.

Limitations of Approx. EP for DGPs

- The approximate predictive distribution of q at each layer is a Gaussian projection, which can be a crude approximation.
- It is limited to certain covariance functions, *e.g.*, the squared exponential covariance function.

Limitations of Approx. EP for DGPs

- The approximate predictive distribution of q at each layer is a Gaussian projection, which can be a crude approximation.
- It is limited to certain covariance functions, *e.g.*, the squared exponential covariance function.
- It modifies the deep GP prior and hence the model, by introducing the FITC approximation.

Doubly Stochastic Variational Inference for DGPs

Features:

- Considers dependencies between inputs and outputs at each layer.

Doubly Stochastic Variational Inference for DGPs

Features:

- Considers dependencies between inputs and outputs at each layer.
- Does not change the DGP prior, which is kept intact.

Doubly Stochastic Variational Inference for DGPs

Features:

- Considers dependencies between inputs and outputs at each layer.
- Does not change the DGP prior, which is kept intact.
- Uses stochastic variational inference to approximate the posterior.

Doubly Stochastic Variational Inference for DGPs

Features:

- Considers dependencies between inputs and outputs at each layer.
- Does not change the DGP prior, which is kept intact.
- Uses stochastic variational inference to approximate the posterior.
- Each layer predictive distribution is approximated by Monte Carlo.

(Salimbeni, 2017)

Black-box Variational Inference

VI works when we can compute $\mathbb{E}_q[\log p(\mathbf{f}, \mathbf{y})]$ in closed form!

Black-box Variational Inference

VI works when we can compute $\mathbb{E}_q[\log p(\mathbf{f}, \mathbf{y})]$ in closed form!

In some situations that is not the case!

Black-box Variational Inference

VI works when we can compute $\mathbb{E}_q[\log p(\mathbf{f}, \mathbf{y})]$ in closed form!

In some situations that is not the case!

Black-box VI uses a Monte Carlo estimator of $\partial\mathcal{L}(q_\theta)/d\theta$ and stochastic optimization techniques to maximize $\mathcal{L}$!

Black-box Variational Inference

VI works when we can compute $\mathbb{E}_q[\log p(\mathbf{f}, \mathbf{y})]$ in closed form!

In some situations that is not the case!

Black-box VI uses a Monte Carlo estimator of $\partial \mathcal{L}(q_\theta)/d\theta$ and stochastic optimization techniques to maximize $\mathcal{L}$!

Black-box VI can be used with arbitrarily complicated models:

$$
\begin{aligned}
\frac{\partial \mathcal{L}(q_\theta)}{\partial \theta} &= \frac{\partial}{\partial \theta} \mathbb{E}_q[\log p(\mathbf{f}, \mathbf{y})] + \frac{\partial H_q}{\partial \theta} \\
&= \mathbb{E}_q \left[\log p(\mathbf{f}, \mathbf{y}) \frac{\partial}{\partial \theta} \log q_\theta(\mathbf{f}) \right] + \frac{\partial H_q}{\partial \theta} \\
&\approx \frac{1}{S} \sum_{s=1}^{S} \log p(\mathbf{f}_s, \mathbf{y}) \frac{\partial}{\partial \theta} \log q_\theta(\mathbf{f}_s) + \frac{\partial H_q}{\partial \theta}
\end{aligned}
$$

Black-box Variational Inference

VI works when we can compute $\mathbb{E}_q[\log p(\mathbf{f}, \mathbf{y})]$ in closed form!

In some situations that is not the case!

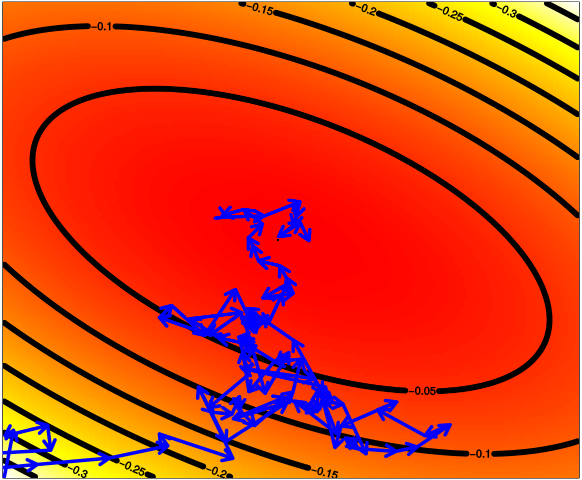
Black-box VI uses a Monte Carlo estimator of $\partial \mathcal{L}(q_\theta)/d\theta$ and stochastic optimization techniques to maximize $\mathcal{L}$!

Black-box VI can be used with arbitrarily complicated models:

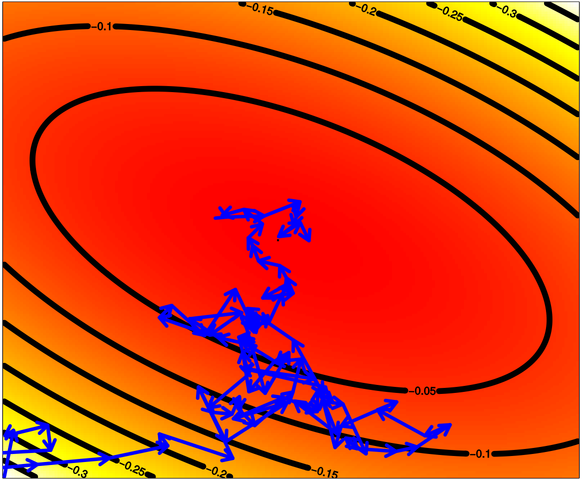
$$\begin{aligned}
\frac{\partial \mathcal{L}(q_\theta)}{\partial \theta} &= \frac{\partial}{\partial \theta} \mathbb{E}_q[\log p(\mathbf{f}, \mathbf{y})] + \frac{\partial H_q}{\partial \theta} \\
&= \mathbb{E}_q \left[\log p(\mathbf{f}, \mathbf{y}) \frac{\partial}{\partial \theta} \log q_\theta(\mathbf{f}) \right] + \frac{\partial H_q}{\partial \theta} \\
&\approx \frac{1}{S} \sum_{s=1}^{S} \log p(\mathbf{f}_s, \mathbf{y}) \frac{\partial}{\partial \theta} \log q_\theta(\mathbf{f}_s) + \frac{\partial H_q}{\partial \theta}
\end{aligned}$$

This is an unbiased estimate of the gradient and can be plugged in any stochastic optimization algorithm!

Stochastic Optimization



Stochastic Optimization



To converge to a local neighborhood of the optimum stochastic methods only require an unbiased estimate of the gradient!

Reparametrization Trick

The previous estimator of the gradient can have high variance and exhibit low convergence!

Reparametrization Trick

The previous estimator of the gradient can have high variance and exhibit low convergence!

Sometimes the randomness can be separated from the parameters:

$$f \sim \mathcal{N}(\mu, \sigma^2)\,, \qquad f = \mu + \sigma\epsilon\,, \qquad \epsilon \sim \mathcal{N}(0, 1)$$

Reparametrization Trick

The previous estimator of the gradient can have high variance and exhibit low convergence!

Sometimes the randomness can be separated from the parameters:

$$f \sim \mathcal{N}(\mu, \sigma^2)\,, \qquad f = \mu + \sigma\epsilon\,, \qquad \epsilon \sim \mathcal{N}(0,1)$$

This allows to obtain another estimator of the gradient:

$$\begin{aligned}\frac{\partial \mathcal{L}(q_\theta)}{\partial \theta} &= \frac{\partial}{\partial \theta}\mathbb{E}_q[\log p(\mathbf{f}, \mathbf{y})] + \frac{\partial H_q}{\partial \theta} \\ &\approx \frac{\partial}{\partial \theta}\frac{1}{S}\sum_{s=1}^{S} \log p(\phi(\boldsymbol{\epsilon}_s; \theta), \mathbf{y}) + \frac{\partial H_q}{\partial \theta}\end{aligned}$$

where $\mathbf{f}_s = \phi(\boldsymbol{\epsilon}_s; \theta)$ for some function $\phi(\cdot\,; \theta)$.

Reparametrization Trick

The previous estimator of the gradient can have high variance and exhibit low convergence!

Sometimes the randomness can be separated from the parameters:

$$f \sim \mathcal{N}(\mu, \sigma^2)\,, \qquad f = \mu + \sigma\epsilon\,, \qquad \epsilon \sim \mathcal{N}(0, 1)$$

This allows to obtain another estimator of the gradient:

$$\begin{aligned} \frac{\partial \mathcal{L}(q_\theta)}{\partial \theta} &= \frac{\partial}{\partial \theta}\mathbb{E}_q[\log p(\mathbf{f}, \mathbf{y})] + \frac{\partial H_q}{\partial \theta} \\ &\approx \frac{\partial}{\partial \theta}\frac{1}{S}\sum_{s=1}^{S} \log p(\phi(\boldsymbol{\epsilon}_s; \theta), \mathbf{y}) + \frac{\partial H_q}{\partial \theta} \end{aligned}$$

where $\mathbf{f}_s = \phi(\boldsymbol{\epsilon}_s; \theta)$ for some function $\phi(\cdot; \theta)$.

This other estimator has less variance and leads to better results!

Deep GPs Joint Distribution

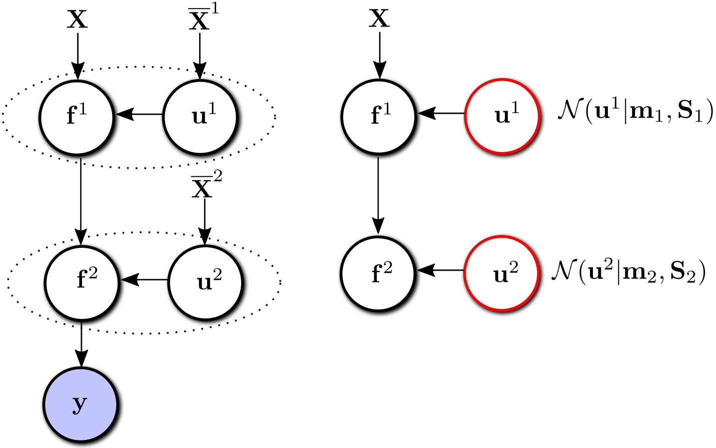
$$p(\mathbf{y}, \{\mathbf{u}^l, \mathbf{f}^l\}_{l=1}^L) = \overbrace{\prod_{i=1}^N p(y_i | f_i^L)}^{\text{Likelihood}} \times$$
$$\underbrace{\prod_{l=1}^L p(\mathbf{f}^l | \mathbf{u}^l, \overline{\mathbf{X}}^l) p(\mathbf{u}^l | \overline{\mathbf{X}}^l)}_{\text{Deep GP prior}}$$

Deep GPs Joint Distribution

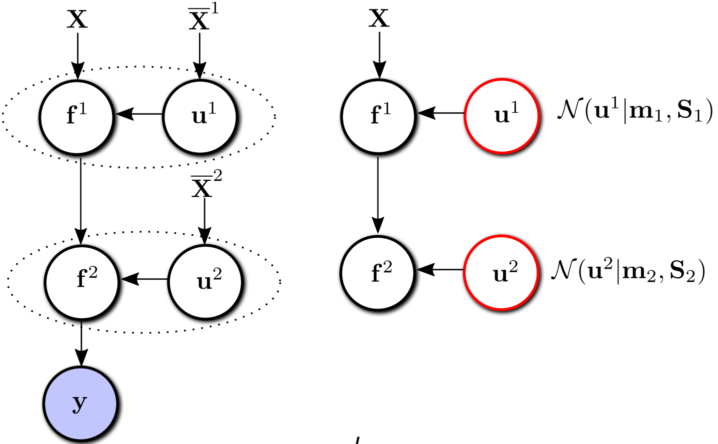
$$p(\mathbf{y}, \{\mathbf{u}^l, \mathbf{f}^l\}_{l=1}^{L}) = \overbrace{\prod_{i=1}^{N} p(y_i | f_i^L)}^{\text{Likelihood}} \times \underbrace{\prod_{l=1}^{L} p(\mathbf{f}^l | \mathbf{u}^l, \overline{\mathbf{X}}^l) p(\mathbf{u}^l | \overline{\mathbf{X}}^l)}_{\text{Deep GP prior}}$$

No change in the model is made at all!

Graphical Model and Posterior Approximation



Graphical Model and Posterior Approximation



$$q(\{\mathbf{f}^l, \mathbf{u}^l\}_{l=1}^L) = \prod_{l=1}^{L} p(\mathbf{f}^l|\mathbf{u}^l)\, q(\mathbf{u}^l)$$

- Fixed
- Tunable

Variational Inference for Deep GPs

Based on minimizing KL$(q(\{\mathbf{u}^l, \mathbf{f}^l\}_{l=1}^L)|p(\{\mathbf{u}^l, \mathbf{f}^l\}_{l=1}^L|\mathbf{y}))$

(Salimbeni, 2017)

Variational Inference for Deep GPs

Based on minimizing $\mathbf{KL}(q(\{\mathbf{u}^l, \mathbf{f}^l\}_{l=1}^L)|p(\{\mathbf{u}^l, \mathbf{f}^l\}_{l=1}^L|\mathbf{y}))$

Equivalent to maximizing the lower bound on $\log p(\mathbf{y})$:

$$\mathcal{L} = \mathbb{E}_q \left[\log \frac{\prod_{i=1}^N p(y_i|f_i^L) \prod_{l=1}^L \cancel{p(\mathbf{f}^l|\mathbf{u}^l)} p(\mathbf{u}^l)}{\prod_{l=1}^L \cancel{p(\mathbf{f}^l|\mathbf{u}^l)} q(\mathbf{u}^l)} \right] .$$

$$= \sum_{i=1}^N \mathbb{E}_q[\log p(y_i|f_i^L)] - \sum_{l=1}^L \mathrm{KL}(q(\mathbf{u}^l)|p(\mathbf{u}^l)) .$$

(Salimbeni, 2017)

Variational Inference for Deep GPs

Based on minimizing $\mathbf{KL}(q(\{\mathbf{u}^l, \mathbf{f}^l\}_{l=1}^L)|p(\{\mathbf{u}^l, \mathbf{f}^l\}_{l=1}^L|\mathbf{y}))$

Equivalent to maximizing the lower bound on $\log p(\mathbf{y})$:

$$\mathcal{L} = \mathbb{E}_q\left[\log \frac{\prod_{i=1}^N p(y_i|f_i^L)\prod_{l=1}^L \cancel{p(\mathbf{f}^l|\mathbf{u}^l)}\, p(\mathbf{u}^l)}{\prod_{l=1}^L \cancel{p(\mathbf{f}^l|\mathbf{u}^l)}\, q(\mathbf{u}^l)}\right].$$

$$= \sum_{i=1}^N \mathbb{E}_q[\log p(y_i|f_i^L)] - \sum_{l=1}^L \mathrm{KL}(q(\mathbf{u}^l)|p(\mathbf{u}^l)).$$

- The expectations can be approximated by Monte Carlo.

(Salimbeni, 2017)

Variational Inference for Deep GPs

Based on minimizing $\mathbf{KL}(q(\{\mathbf{u}^l, \mathbf{f}^l\}_{l=1}^L)|p(\{\mathbf{u}^l, \mathbf{f}^l\}_{l=1}^L|\mathbf{y}))$

Equivalent to maximizing the lower bound on $\log p(\mathbf{y})$:

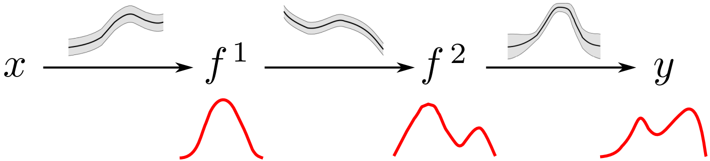
$$\mathcal{L} = \mathbb{E}_q \left[\log \frac{\prod_{i=1}^N p(y_i|f_i^L) \prod_{l=1}^L \cancel{p(\mathbf{f}^l|\mathbf{u}^l)} p(\mathbf{u}^l)}{\prod_{l=1}^L \cancel{p(\mathbf{f}^l|\mathbf{u}^l)} q(\mathbf{u}^l)} \right] .$$

$$= \sum_{i=1}^N \mathbb{E}_q[\log p(y_i|f_i^L)] - \sum_{l=1}^L \mathrm{KL}(q(\mathbf{u}^l)|p(\mathbf{u}^l)) .$$

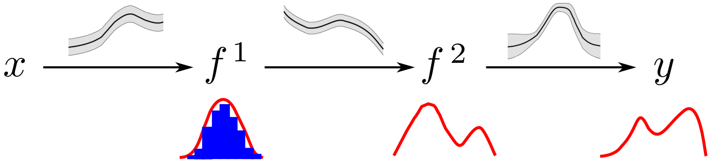
- The expectations can be approximated by Monte Carlo.
- Suitable for mini-batch training by subsampling the data.

(Salimbeni, 2017)

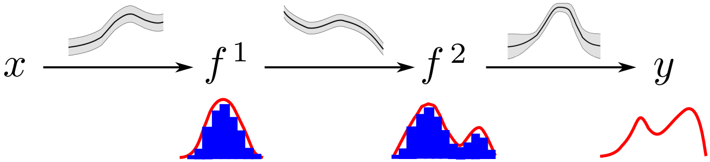
Predictive Distribution via Monte Carlo Sampling



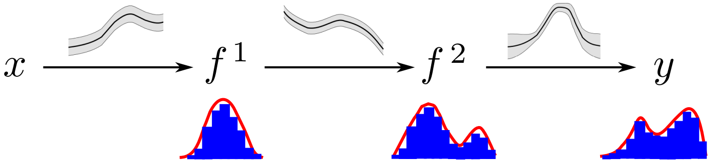
Predictive Distribution via Monte Carlo Sampling



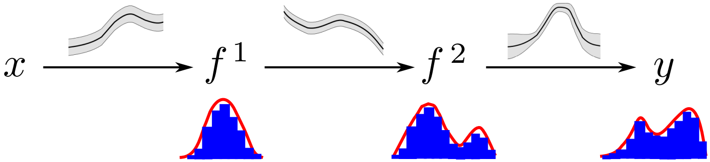
Predictive Distribution via Monte Carlo Sampling



Predictive Distribution via Monte Carlo Sampling

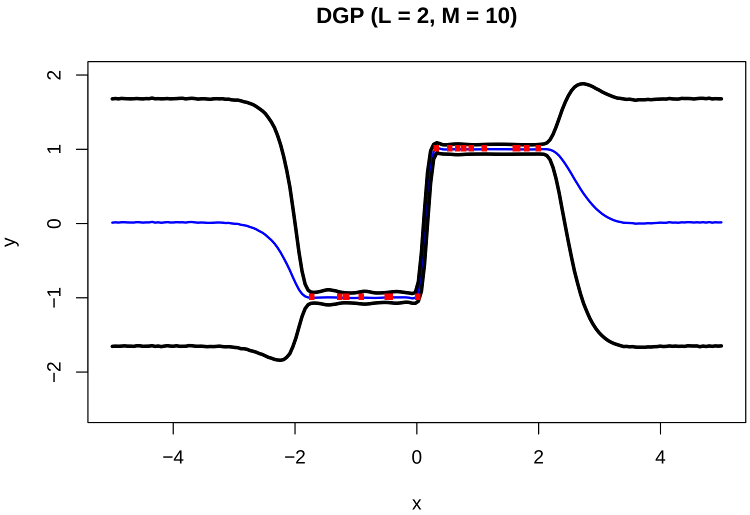


Predictive Distribution via Monte Carlo Sampling

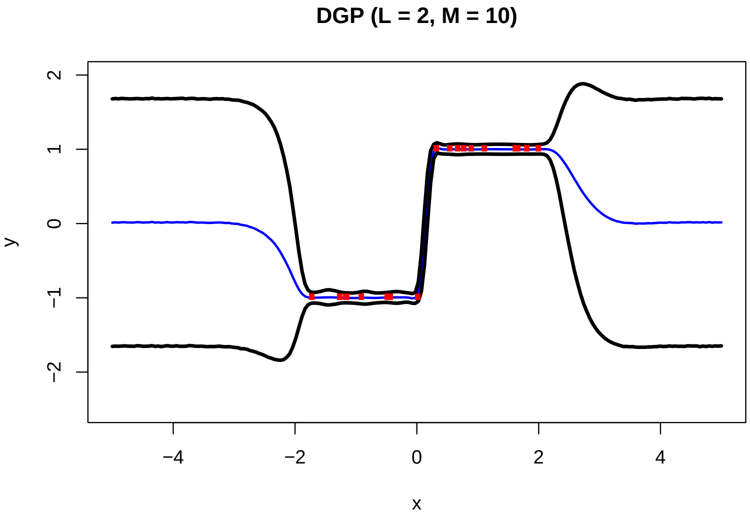


Used not only for testing, but also during training, unlike the previous methods!

DSVI for DGPs: Illustrative Example



DSVI for DGPs: Illustrative Example



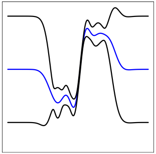
DSVI provides better results than the previous methods!

DSVI for DGPs: Illustrative Example

x

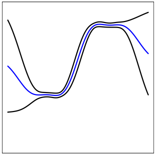
$\downarrow$

$f^1(x)$



$\downarrow$

$f^2(f_1)$

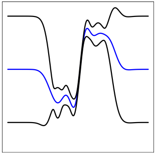


$\downarrow$

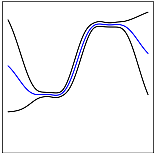
$y + \text{noise}$

DSVI for DGPs: Illustrative Example

$$x$$
$$\downarrow$$
$$f^1(x)$$



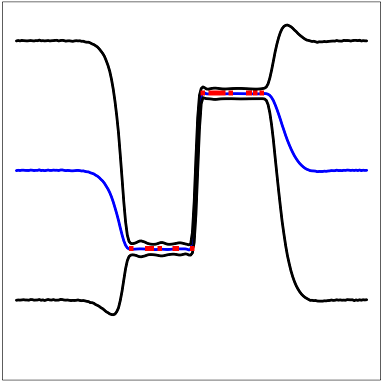
$$\downarrow$$
$$f^2(f_1)$$



$$\downarrow$$
$$y + \text{noise}$$

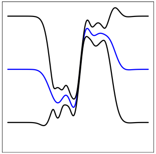
$$\equiv$$

$$y = f^2(f^1(x)) + \text{noise}$$

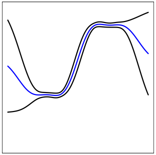


DSVI for DGPs: Illustrative Example

$$x \downarrow f^1(x)$$



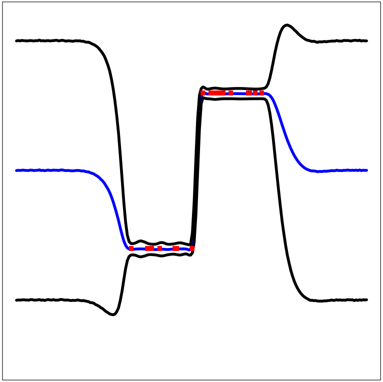
$$\downarrow f^2(f_1)$$



$$\downarrow y + \text{noise}$$

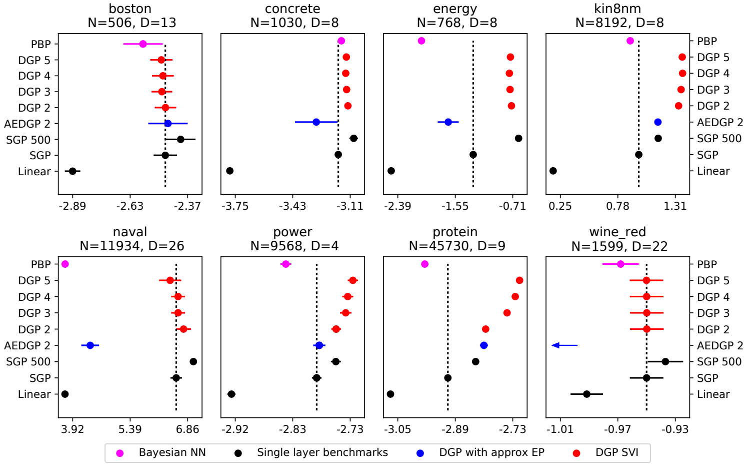
$$\equiv$$

$$y = f^2(f^1(x)) + \text{noise}$$

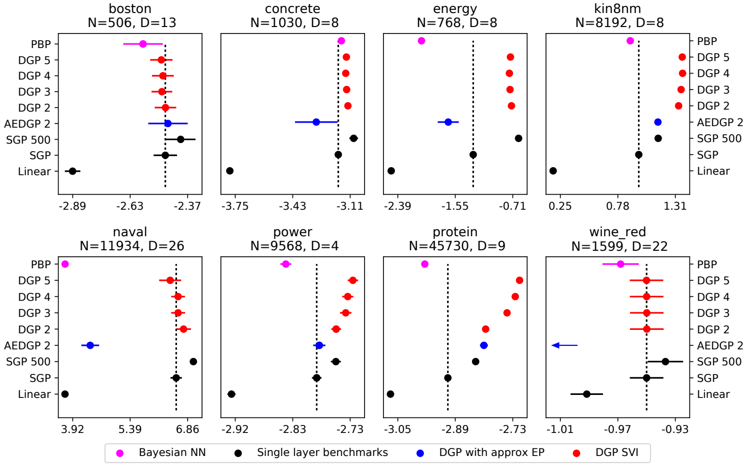


$$f^1, f^2 \sim \mathcal{GP}(0, C(\cdot, \cdot))$$

DGPs via DSVI: LL Experimental Results



DGPs via DSVI: LL Experimental Results



DGPs perform similar or better than the sparse GP and adding more layers does not seem to overfit!

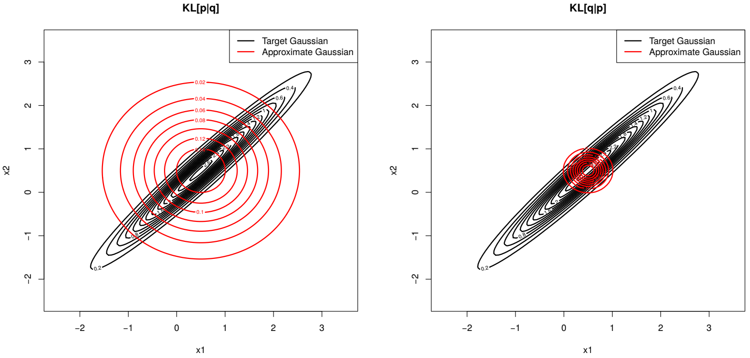
(Salimbeni, 2017)

Limitations of DSVI for DGPs

DSVI and the approximate EP method for training DGPs target different divergences: KL$[q|p]$ and KL$[p|q]$!

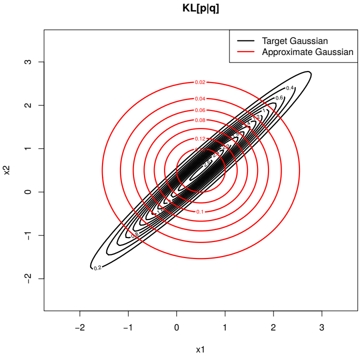
Limitations of DSVI for DGPs

DSVI and the approximate EP method for training DGPs target different divergences: KL$[q|p]$ and KL$[p|q]$!



Limitations of DSVI for DGPs

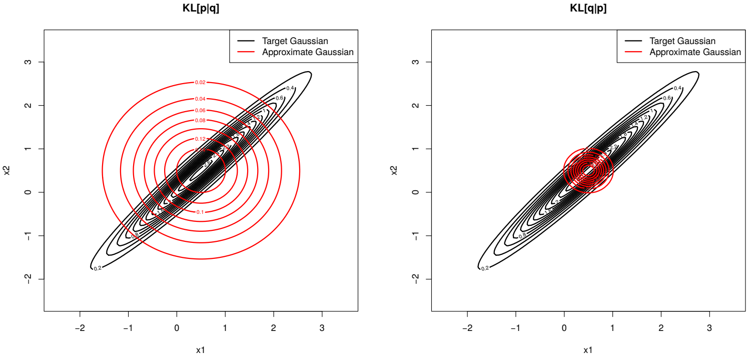
DSVI and the approximate EP method for training DGPs target different divergences: KL$[q|p]$ and KL$[p|q]$!



KL$[q|p]$ may result in too compact approximations while KL$[p|q]$ may put mass in regions with no posterior density.

Limitations of DSVI for DGPs

DSVI and the approximate EP method for training DGPs target different divergences: KL$[q|p]$ and KL$[p|q]$!



KL$[q|p]$ may result in too compact approximations while KL$[p|q]$ may put mass in regions with no posterior density. Can we have something in between?

Alpha Divergence

$$D_\alpha(p||q) = \frac{\int_{\boldsymbol{\theta}} \left(\alpha p(\boldsymbol{\theta}) + (1-\alpha)q(\boldsymbol{\theta}) - p(\boldsymbol{\theta})^{\alpha} q(\boldsymbol{\theta})^{1-\alpha}\right) \, d\boldsymbol{\theta}}{\alpha(1-\alpha)} .$$

(Amari, 1985).

Alpha Divergence

$$D_\alpha(p||q) = \frac{\int_{\boldsymbol{\theta}} \left(\alpha p(\boldsymbol{\theta}) + (1-\alpha) q(\boldsymbol{\theta}) - p(\boldsymbol{\theta})^\alpha q(\boldsymbol{\theta})^{1-\alpha}\right) \, d\boldsymbol{\theta}}{\alpha(1-\alpha)} \, .$$

(Amari, 1985).

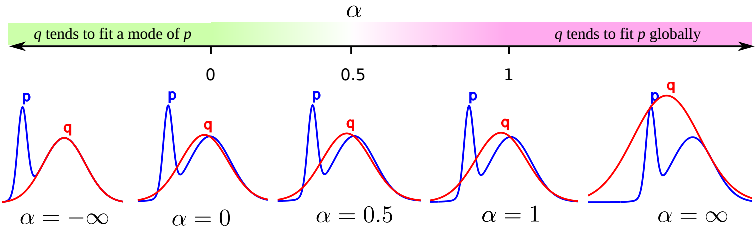


Figure source: (Minka, 2005).

Alpha Divergence

$$D_\alpha(p||q) = \frac{\int_{\boldsymbol{\theta}} \left(\alpha p(\boldsymbol{\theta}) + (1-\alpha) q(\boldsymbol{\theta}) - p(\boldsymbol{\theta})^\alpha q(\boldsymbol{\theta})^{1-\alpha} \right) \, d\boldsymbol{\theta}}{\alpha(1-\alpha)} .$$

(Amari, 1985).

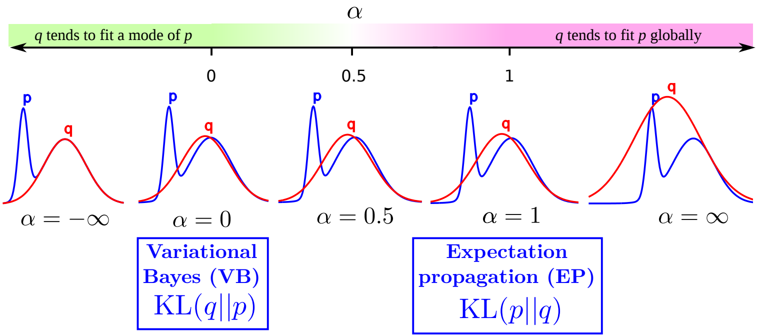


Figure source: (Minka, 2005).

Alpha Divergence

$$D_\alpha(p||q) = \frac{\int_{\boldsymbol{\theta}} \left(\alpha p(\boldsymbol{\theta}) + (1-\alpha) q(\boldsymbol{\theta}) - p(\boldsymbol{\theta})^\alpha q(\boldsymbol{\theta})^{1-\alpha}\right) \, d\boldsymbol{\theta}}{\alpha(1-\alpha)} .$$

(Amari, 1985).

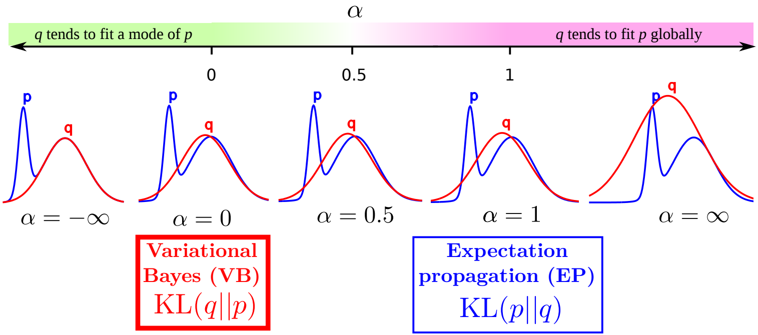


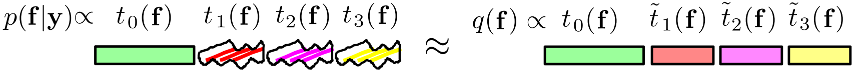
Figure source: (Minka, 2005).

Local α-divergence minimization (Power EP)

Approximates $\boxed{p(\mathbf{f}|\mathbf{y}) \propto t_0(\mathbf{f}) \prod_{j=1}^{N} t_j(\mathbf{f})}$ with $\boxed{q(\mathbf{f}) \propto t_0(\mathbf{f}) \prod_{j=1}^{N} \tilde{t}_j(\mathbf{t})}$

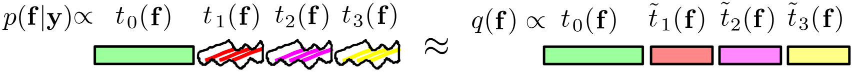
Local α-divergence minimization (Power EP)

Approximates $\boxed{p(\mathbf{f}|\mathbf{y}) \propto t_0(\mathbf{f}) \prod_{j=1}^{N} t_j(\mathbf{f})}$ with $\boxed{q(\mathbf{f}) \propto t_0(\mathbf{f}) \prod_{j=1}^{N} \tilde{t}_j(\mathbf{t})}$



Local α-divergence minimization (Power EP)

Approximates $\boxed{p(\mathbf{f}|\mathbf{y}) \propto t_0(\mathbf{f}) \prod_{j=1}^{N} t_j(\mathbf{f})}$ with $\boxed{q(\mathbf{f}) \propto t_0(\mathbf{f}) \prod_{j=1}^{N} \tilde{t}_j(\mathbf{t})}$

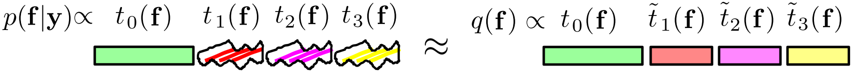


The $\tilde{t}_j$ are tuned by minimizing local α-divergences

$$
\mathrm{D}_\alpha[\hat{p}_j || q] \quad \text{for } j = 1, \ldots, N\,, \quad \text{where} \quad \begin{array}{rcl} \hat{p}_j(\mathbf{f}) & \propto & t_j(\mathbf{f}) \prod_{i \neq j} \tilde{t}_i(\mathbf{f}) \\ q(\mathbf{f}) & \propto & \tilde{t}_j(\mathbf{f}) \prod_{i \neq j} \tilde{t}_i(\mathbf{f}) \end{array}.
$$

Local α-divergence minimization (Power EP)

Approximates $\boxed{p(\mathbf{f}|\mathbf{y}) \propto t_0(\mathbf{f}) \prod_{j=1}^N t_j(\mathbf{f})}$ with $\boxed{q(\mathbf{f}) \propto t_0(\mathbf{f}) \prod_{j=1}^N \tilde{t}_j(\mathbf{t})}$



The $\tilde{t}_j$ are tuned by minimizing local α-divergences

$$\mathrm{D}_\alpha[\hat{p}_j || q] \quad \text{for } j = 1, \ldots, N\,, \quad \text{where} \quad \begin{array}{ccc} \hat{p}_j(\mathbf{f}) & \propto & t_j(\mathbf{f}) \prod_{i \neq j} \tilde{t}_i(\mathbf{f}) \\ q(\mathbf{f}) & \propto & \tilde{t}_j(\mathbf{f}) \prod_{i \neq j} \tilde{t}_i(\mathbf{f}) \end{array}.$$

It turns out that the α-divergence can be minimized in terms of the KL-divergence!

α-divergence minimization via KL minimization

Power EP steps to refine $\tilde{t}_i$:

α-divergence minimization via KL minimization

Power EP steps to refine $\tilde{t}_i$:

1. Compute cavity: $q^{\setminus \alpha i} \propto q / \tilde{t}_i^{\alpha}$.

α-divergence minimization via KL minimization

Power EP steps to refine $\tilde{t}_i$:

1. Compute cavity: $q^{\setminus \alpha i} \propto q / \tilde{t}_i^{\alpha}$.
2. Minimize $\mathrm{KL}(Z_i^{-1} t_i^{\alpha} q^{\setminus \alpha i} || q)$ to find q^{new}.

α-divergence minimization via KL minimization

Power EP steps to refine $\tilde{t}_i$:

1. Compute cavity: $q^{\setminus \alpha i} \propto q / \tilde{t}_i^{\alpha}$.
2. Minimize $\mathrm{KL}(Z_i^{-1} t_i^{\alpha} q^{\setminus \alpha i} || q)$ to find q^{new}.
3. Update factor: $\tilde{t}_i^{\mathrm{new}} = (Z_i q^{\mathrm{new}} / q^{\setminus \alpha i})^{\frac{1}{\alpha}}$.

α-divergence minimization via KL minimization

Power EP steps to refine $\tilde{t}_i$:

1. Compute cavity: $q^{\backslash \alpha i} \propto q / \tilde{t}_i^{\alpha}$.
2. Minimize $\mathrm{KL}(Z_i^{-1} t_i^{\alpha} q^{\backslash \alpha i} || q)$ to find q^{new}.
3. Update factor: $\tilde{t}_i^{\mathrm{new}} = (Z_i q^{\mathrm{new}} / q^{\backslash \alpha i})^{\frac{1}{\alpha}}$.

At convergence the moments of $\tilde{p} = Z_i^{-1} f_i^{\alpha} q^{\backslash \alpha i}$ and q match!

α-divergence minimization via KL minimization

Power EP steps to refine $\tilde{t}_i$:

1. Compute cavity: $q^{\backslash \alpha i} \propto q / \tilde{t}_i^{\alpha}$.
2. Minimize $\mathrm{KL}(Z_i^{-1} t_i^{\alpha} q^{\backslash \alpha i} || q)$ to find q^{new}.
3. Update factor: $\tilde{t}_i^{\mathrm{new}} = (Z_i q^{\mathrm{new}} / q^{\backslash \alpha i})^{\frac{1}{\alpha}}$.

At convergence the moments of $\tilde{p} = Z_i^{-1} f_i^{\alpha} q^{\backslash \alpha i}$ and q match!

$$\nabla_{\eta_q} D_{\alpha}[p_i || q] = \frac{Z_{\tilde{p}}}{\alpha} \left(\mathbb{E}_q[s(\boldsymbol{\theta})] - \mathbb{E}_{\tilde{p}}[s(\boldsymbol{\theta})] \right) \propto \nabla_{\eta_q} \mathrm{KL}[\tilde{p} || q]$$

where $\tilde{p} \propto (t_i q^{\backslash i})^{\alpha} q^{1-\alpha} = t_i^{\alpha} q^{\backslash \alpha i}$.

α-divergence minimization via KL minimization

Power EP steps to refine $\tilde{t}_i$:

1. Compute cavity: $q^{\backslash \alpha i} \propto q / \tilde{t}_i^{\alpha}$.
2. Minimize $\mathrm{KL}(Z_i^{-1} t_i^{\alpha} q^{\backslash \alpha i} || q)$ to find q^{new}.
3. Update factor: $\tilde{t}_i^{\mathrm{new}} = (Z_i q^{\mathrm{new}} / q^{\backslash \alpha i})^{\frac{1}{\alpha}}$.

At convergence the moments of $\tilde{p} = Z_i^{-1} f_i^{\alpha} q^{\backslash \alpha i}$ and q match!

$$\nabla_{\eta_q} D_\alpha[p_i || q] = \frac{Z_{\tilde{p}}}{\alpha} \left(\mathbb{E}_q[s(\boldsymbol{\theta})] - \mathbb{E}_{\tilde{p}}[s(\boldsymbol{\theta})] \right) \propto \nabla_{\eta_q} \mathrm{KL}[\tilde{p} || q]$$

where $\tilde{p} \propto (t_i q^{\backslash i})^{\alpha} q^{1-\alpha} = t_i^{\alpha} q^{\backslash \alpha i}$.

At convergence $\nabla_{\eta_q} D_\alpha[p_n || q]$ equals zero!

PEP as an Optimization Problem

The PEP approximation to the **evidence** $p(\mathbf{y})$ is given by:

$$\log Z_{\mathrm{PEP}} = g(\boldsymbol{\eta}_q) - g(\boldsymbol{\eta}_{\mathrm{prior}}) + \sum_{i=1}^{N} \frac{1}{\alpha} \left(\log Z_i + g(\boldsymbol{\eta}_q) - g(\boldsymbol{\eta}_q^{\setminus \alpha i}) \right)$$

PEP as an Optimization Problem

The PEP approximation to the **evidence** $p(\mathbf{y})$ is given by:

$$\log Z_{\mathrm{PEP}} = g(\boldsymbol{\eta}_q) - g(\boldsymbol{\eta}_{\mathrm{prior}}) + \sum_{i=1}^{N} \frac{1}{\alpha} \left(\log Z_i + g(\boldsymbol{\eta}_q) - g(\boldsymbol{\eta}_q^{\setminus \alpha i}) \right)$$

Besides the PEP updates, the PEP solution for q is found by solving:

$$\max_{q} \min_{\tilde{t}_1,\ldots,\tilde{t}_N} \log Z_{\mathrm{PEP}} \quad \text{subject to} \quad q \propto \tilde{p} \prod_{i=1}^{N} \tilde{t}_i \,.$$

PEP as an Optimization Problem

The PEP approximation to the **evidence** $p(\mathbf{y})$ is given by:

$$\log Z_{\text{PEP}} = g(\boldsymbol{\eta}_q) - g(\boldsymbol{\eta}_{\text{prior}}) + \sum_{i=1}^{N} \frac{1}{\alpha} \left(\log Z_i + g(\boldsymbol{\eta}_q) - g(\boldsymbol{\eta}_q^{\setminus \alpha i}) \right)$$

Besides the PEP updates, the PEP solution for q is found by solving:

$$\max_{q} \min_{\tilde{t}_1,\ldots,\tilde{t}_N} \log Z_{\text{PEP}} \quad \text{subject to} \quad q \propto \tilde{p} \prod_{i=1}^{N} \tilde{t}_i \,.$$

Can be solved with a **double-loop** algorithm.

PEP as an Optimization Problem

The PEP approximation to the **evidence** $p(\mathbf{y})$ is given by:

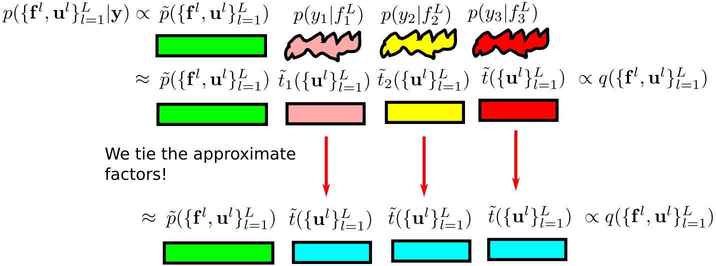
$$\log Z_{\text{PEP}} = g(\boldsymbol{\eta}_q) - g(\boldsymbol{\eta}_{\text{prior}}) + \sum_{i=1}^{N} \frac{1}{\alpha} \left(\log Z_i + g(\boldsymbol{\eta}_q) - g(\boldsymbol{\eta}_q^{\backslash \alpha i}) \right)$$

Besides the PEP updates, the PEP solution for q is found by solving:

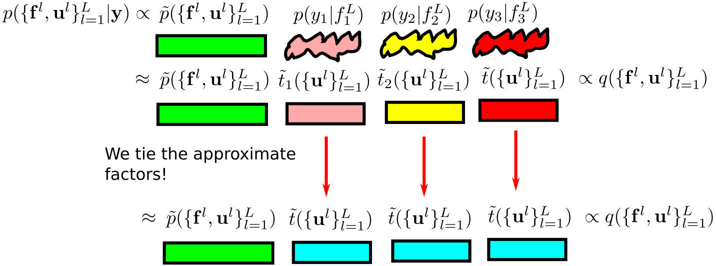
$$\max_q \min_{\tilde{t}_1,\ldots,\tilde{t}_N} \log Z_{\text{PEP}} \quad \text{subject to} \quad q \propto \tilde{p} \prod_{i=1}^{N} \tilde{t}_i \,.$$

Can be solved with a **double-loop** algorithm. **Too slow in practice!**

Approximate Power Expectation Propagation

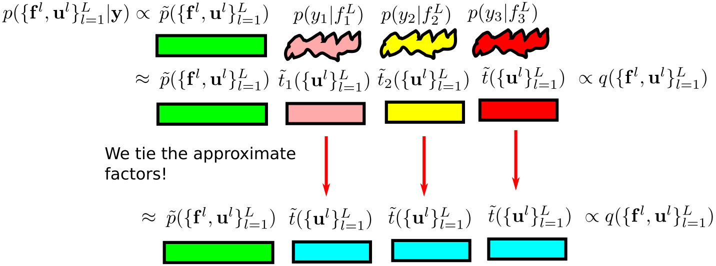


Approximate Power Expectation Propagation



- $\max_{q} \min_{\tilde{f}_1,\ldots,\tilde{f}_N}$ problem $\rightarrow$ $\max_{q}$ problem, **no double-loop needed!**

Approximate Power Expectation Propagation



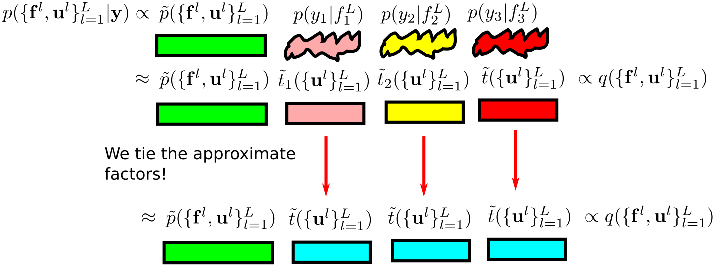
- $\max_{q} \min_{\tilde{f}_1,\ldots,\tilde{f}_N}$ problem $\rightarrow$ $\max_{q}$ problem, **no double-loop needed!**

The final objective is:

$$\log Z_{\text{PEP}} = g(\boldsymbol{\eta}_q) - g(\boldsymbol{\eta}_{\text{prior}}) + \sum_{i=1}^{N} \frac{1}{\alpha}\Big(\log Z_i + g(\boldsymbol{\eta}_q) - g(\boldsymbol{\eta}_{q_\alpha^{\text{cav}}})\Big)$$

which is suitable for standard optimization and mini-batch training.
(Villacampa, 2022)(Li, 2017)

Approximate Power Expectation Propagation



- $\max_{q} \min_{\tilde{f}_1,\ldots,\tilde{f}_N}$ problem $\rightarrow$ $\max_{q}$ problem, **no double-loop needed!**

The final objective is:

$$\log Z_{\text{PEP}} = g(\boldsymbol{\eta}_q) - g(\boldsymbol{\eta}_{\text{prior}}) + \sum_{i=1}^{N} \frac{1}{\alpha}\left(\log \tilde{Z}_i + g(\boldsymbol{\eta}_q) - g(\boldsymbol{\eta}_{q_\alpha^{\text{cav}}})\right)$$

which is suitable for standard optimization and mini-batch training.
(Villacampa, 2022)(Li, 2017)

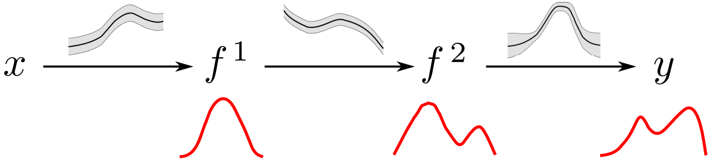
Approximating $\log Z_i$

Note that $\log Z_i = \log \int p(y_i|f_i^L)^\alpha q^{\setminus \alpha i}(f_i^L) df_i^L$ is the log predictive likelihood of instance i when removed from the training set.

Approximating $\log Z_i$

Note that $\log Z_i = \log \int p(y_i|f_i^L)^\alpha q^{\backslash \alpha i}(f_i^L) df_i^L$ is the log predictive likelihood of instance i when removed from the training set.

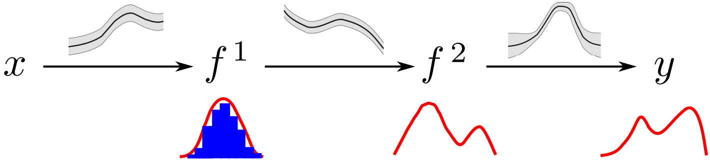
We can use a Monte Carlo approximation:



Approximating $\log Z_i$

Note that $\log Z_i = \log \int p(y_i|f_i^L)^\alpha q^{\backslash \alpha i}(f_i^L) df_i^L$ is the log predictive likelihood of instance i when removed from the training set.

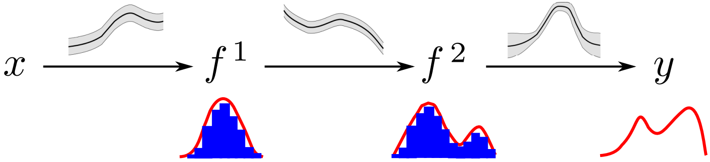
We can use a Monte Carlo approximation:



Approximating $\log Z_i$

Note that $\log Z_i = \log \int p(y_i|f_i^L)^\alpha q^{\setminus \alpha i}(f_i^L) df_i^L$ is the log predictive likelihood of instance i when removed from the training set.

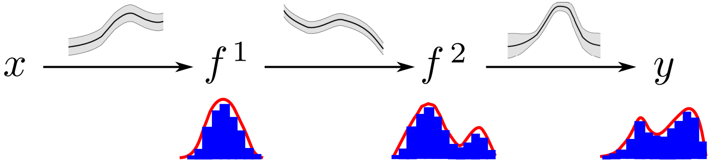
We can use a Monte Carlo approximation:



Approximating $\log Z_i$

Note that $\log Z_i = \log \int p(y_i|f_i^L)^\alpha q^{\setminus \alpha i}(f_i^L) df_i^L$ is the log predictive likelihood of instance i when removed from the training set.

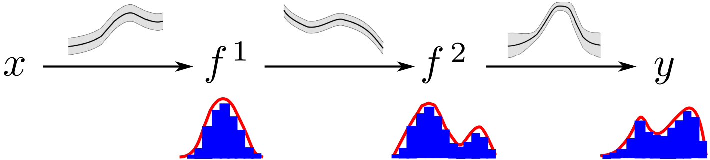
We can use a Monte Carlo approximation:



Approximating $\log Z_i$

Note that $\log Z_i = \log \int p(y_i|f_i^L)^\alpha q^{\setminus \alpha i}(f_i^L) df_i^L$ is the log predictive likelihood of instance i when removed from the training set.

We can use a Monte Carlo approximation:



Expected to be more accurate than the Gaussian projection method used by AEP!

Further Approximations

Consider $\alpha \approx 0$ or $N \to \infty$ (*i.e.*, the cavity becomes q):

$$\log Z_{\text{PEP}} \approx g(\boldsymbol{\eta}_q) - g(\boldsymbol{\eta}_{\text{prior}}) + \sum_{i=1}^{N} \frac{1}{\alpha} \big(\log \tilde{Z}_i + g(\boldsymbol{\eta}_q) - g(\boldsymbol{\eta}_{q_\alpha^{\text{cav}}}) \big)$$

$$= \sum_{i=1}^{N} \frac{1}{\alpha} \log \tilde{Z}_i - \mathrm{R}_\beta[q_{\text{cav}} | \text{prior}] \,,$$

with $\mathrm{R}_\beta[q_{\text{cav}} | \text{prior}]$ a Rényi divergence, becomes similar to

$$\log Z_{\text{PEP}} \approx \sum_{i=1}^{N} \frac{1}{\alpha} \log \tilde{Z}_i - \mathrm{KL}[q | \text{prior}] \,,$$

Further Approximations

Consider $\alpha \approx 0$ or $N \to \infty$ (*i.e.*, the cavity becomes q):

$$\log Z_{\mathrm{PEP}} \approx g(\boldsymbol{\eta}_q) - g(\boldsymbol{\eta}_{\mathrm{prior}}) + \sum_{i=1}^{N} \frac{1}{\alpha}\left(\log \tilde{Z}_i + g(\boldsymbol{\eta}_q) - g(\boldsymbol{\eta}_{q_\alpha^{\mathrm{cav}}})\right)$$

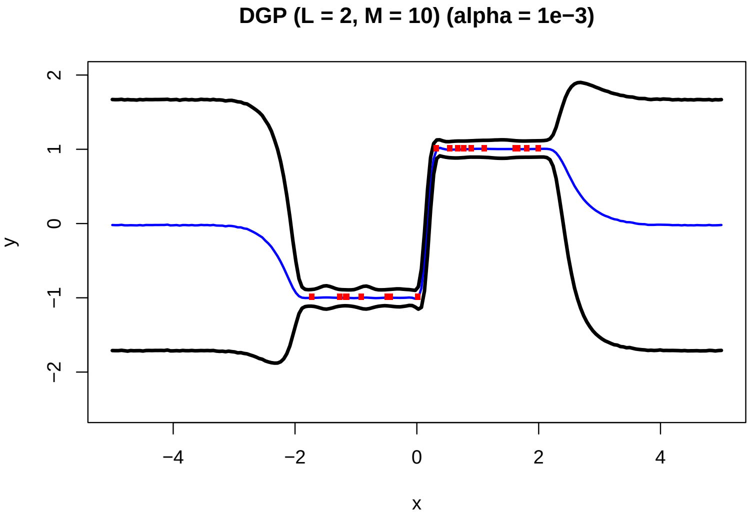
$$= \sum_{i=1}^{N} \frac{1}{\alpha} \log \tilde{Z}_i - \mathrm{R}_\beta[q_{\mathrm{cav}}|\mathrm{prior}]\,,$$

with $\mathrm{R}_\beta[q_{\mathrm{cav}}|\mathrm{prior}]$ a Rényi divergence, becomes similar to

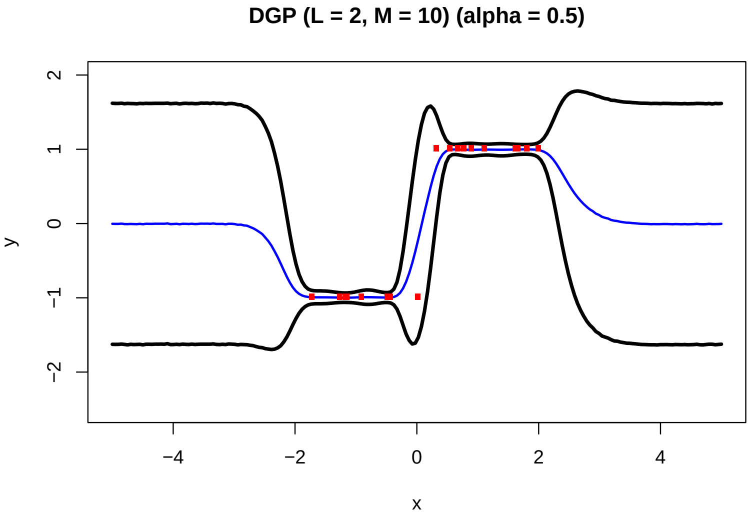
$$\log Z_{\mathrm{PEP}} \approx \sum_{i=1}^{N} \frac{1}{\alpha} \log \tilde{Z}_i - \mathrm{KL}[q|\mathrm{prior}]\,,$$

Which for $\alpha \to 0$ gives the DVSVI objective and for $\alpha = 1$ is expected to give similar results to AEP (better estimating $\log Z_i$)!

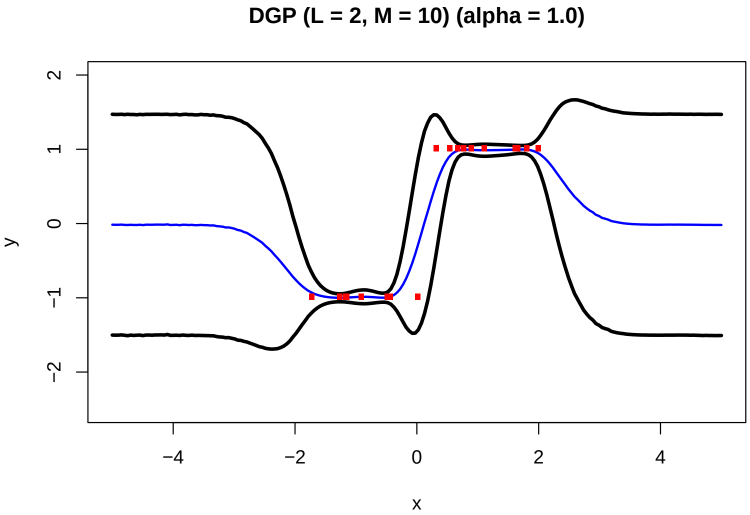
α-Divergence Minimization: Illustrative Example



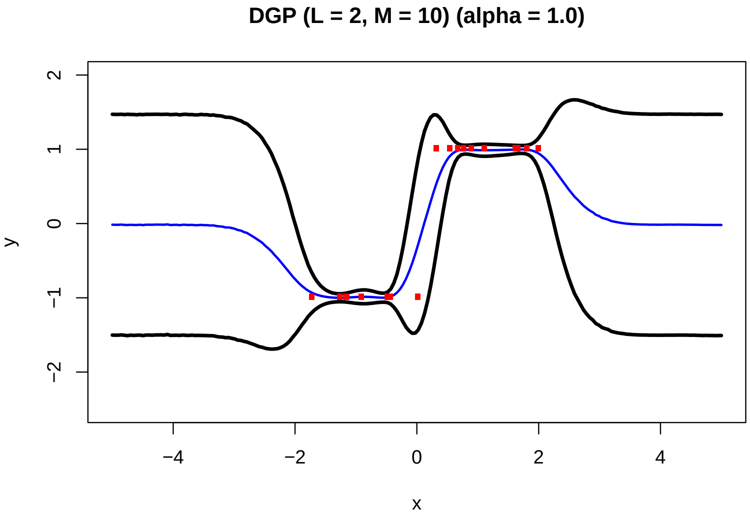
α-Divergence Minimization: Illustrative Example



α-Divergence Minimization: Illustrative Example

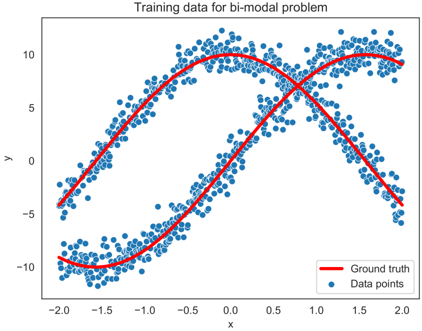


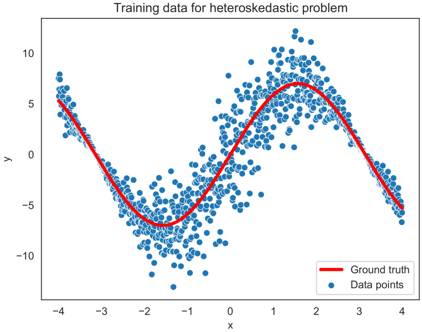
α-Divergence Minimization: Illustrative Example



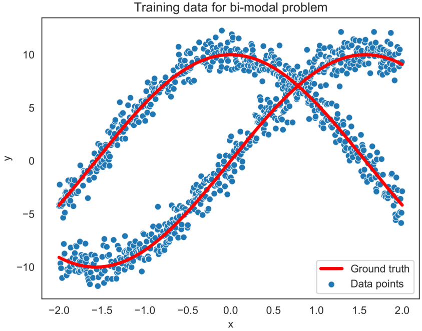
The value of α has an impact on the final predictive distribution!

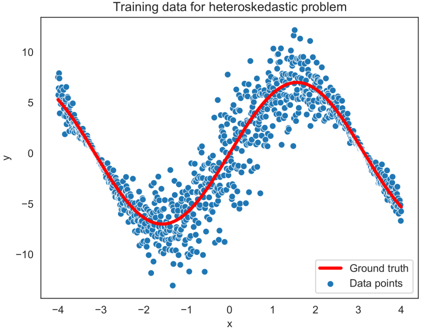
α-Divergence Minimization: Toy Problems





α-Divergence Minimization: Toy Problems

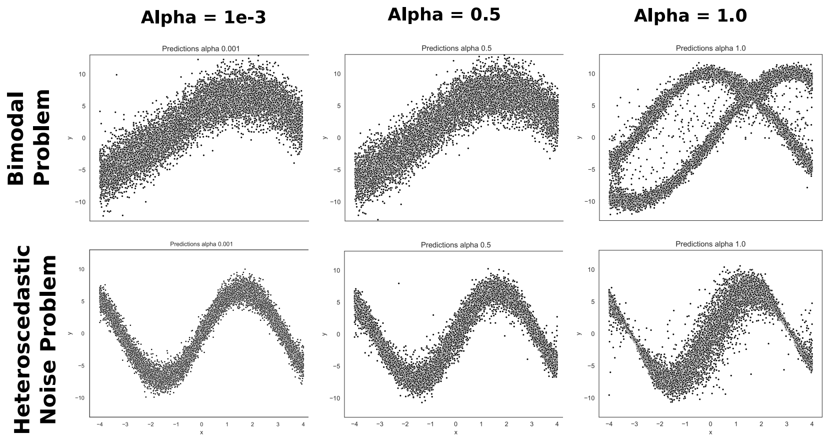




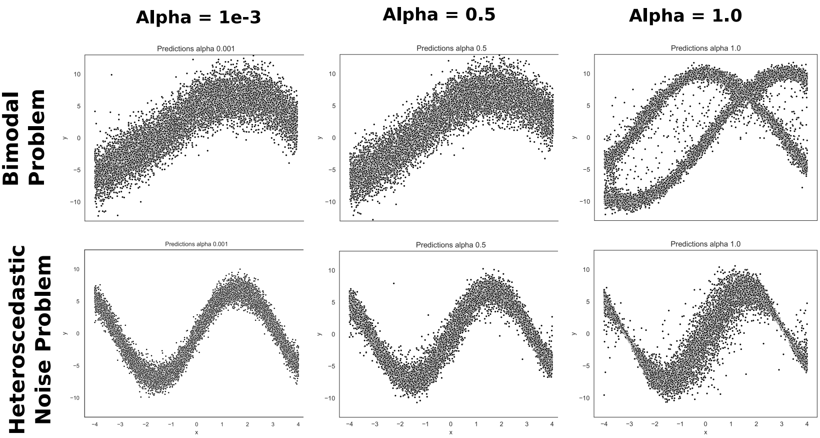
The first problem has heteroscedastic noise. The second, a bimodal predictive distribution!

(Depeweng, 2016)

α-Divergence Minimization: Toy Problems



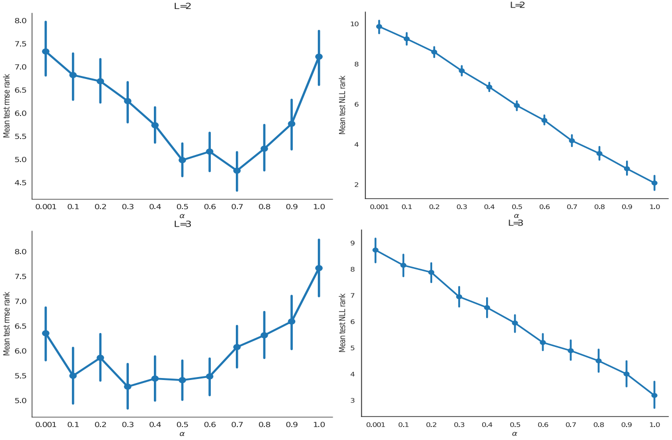
α-Divergence Minimization: Toy Problems



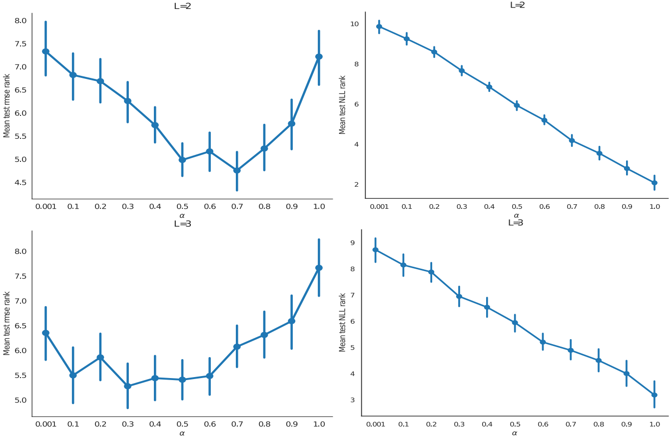
The value $\alpha = 1.0$ provides more sensible predictive distributions!

(Villacampa, 2022)

α-Divergence Minimization: Average Ranks



α-Divergence Minimization: Average Ranks



The value $\alpha = 1.0$ provides better results in terms of the NLL and intermediate values of α give better RMSE!

(Villacampa, 2022)

Summary about DGPs

Summary about DGPs

- Useful for input warping: automatic, non-parametric kernel design.

Summary about DGPs

- Useful for input warping: automatic, non-parametric kernel design.
- Repair damage done by sparse approximations to GPs.

Summary about DGPs

- Useful for input warping: automatic, non-parametric kernel design.
- Repair damage done by sparse approximations to GPs.
- More accurate predictions and better uncertainty estimates.

Summary about DGPs

- Useful for input warping: automatic, non-parametric kernel design.
- Repair damage done by sparse approximations to GPs.
- More accurate predictions and better uncertainty estimates.
- Better cost scaling w.r.t. depth L rather than inducing points M.

Summary about DGPs

- Useful for input warping: automatic, non-parametric kernel design.
- Repair damage done by sparse approximations to GPs.
- More accurate predictions and better uncertainty estimates.
- Better cost scaling w.r.t. depth L rather than inducing points M.
- More complex inference: DSVI, AEP, α-divergence minimization.

Summary about DGPs

- Useful for input warping: automatic, non-parametric kernel design.
- Repair damage done by sparse approximations to GPs.
- More accurate predictions and better uncertainty estimates.
- Better cost scaling w.r.t. depth L rather than inducing points M.
- More complex inference: DSVI, AEP, α-divergence minimization.
- α-divergence minimization generalizes the other methods.

References

- Damianou, A., & Lawrence, N. D. (2013, April). Deep Gaussian processes. In Artificial intelligence and statistics (pp. 207-215).
- Bui, T., Hernández-Lobato, D., Hernandez-Lobato, J., Li, Y., & Turner, R. (2016, June). Deep Gaussian processes for regression using approximate expectation propagation. In International conference on machine learning (pp. 1472-1481).
- Salimbeni, H., & Deisenroth, M. (2017). Doubly stochastic variational inference for deep Gaussian processes. Advances in neural information processing systems, 30.
- Li, Y., & Gal, Y. (2017, July). Dropout inference in Bayesian neural networks with alpha-divergences. In International conference on machine learning (pp. 2052-2061).
- Villacampa-Calvo, C., & Hernandez-Lobato, D. (2020). Alpha divergence minimization in multi-class Gaussian process classification. Neurocomputing, 378, 210-227.