Approximate Inference in Multi-class and Deep Gaussian Processes by Minimizing Alpha Divergences

Daniel Hernández–Lobato
Computer Science Department
Universidad Autónoma de Madrid

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

Joint work with
Carlos Villacampa-Calvo and
Gonzalo Hernández-Muñoz

Outline

- Introduction to Multi-class GPs
  1. Multi-class GPs using Variational Inference
  2. Multi-class GPs using Expectation Propagation
  3. Multi-class GPs using Alpha Divergence Minimization

- Introduction to Deep-GPs
  1. Deep-GPs using Variational Inference
  2. Deep-GPs using Approximate Expectation Propagation
  3. Deep-GPs using Alpha Divergence Minimization

Introduction to Multi-class Classification with GPs

Given $\mathbf{x}_i$ we want to make **predictions** about $y_i \in \{1, \ldots, C\}$, $C > 2$.

One can **assume** that (Kim & Ghahramani, 2006):

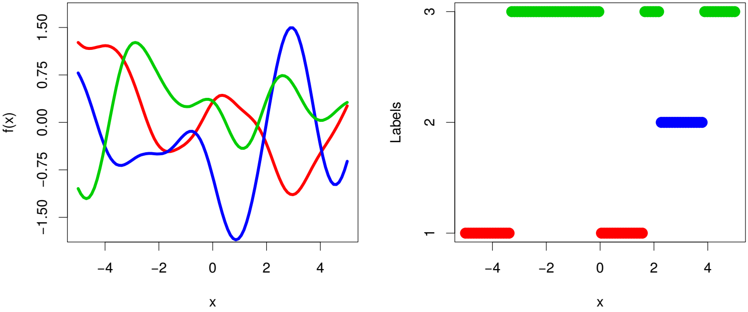
$$y_i = \arg\max_k \quad f^k(\mathbf{x}_i) \quad \text{for} \quad k \in \{1, \ldots, C\}$$

Introduction to Multi-class Classification with GPs

Given $\mathbf{x}_i$ we want to make **predictions** about $y_i \in \{1, \ldots, C\}$, $C > 2$.

One can **assume** that (Kim & Ghahramani, 2006):

$$y_i = \arg\max_k \quad f^k(\mathbf{x}_i) \quad \text{for} \quad k \in \{1, \ldots, C\}$$

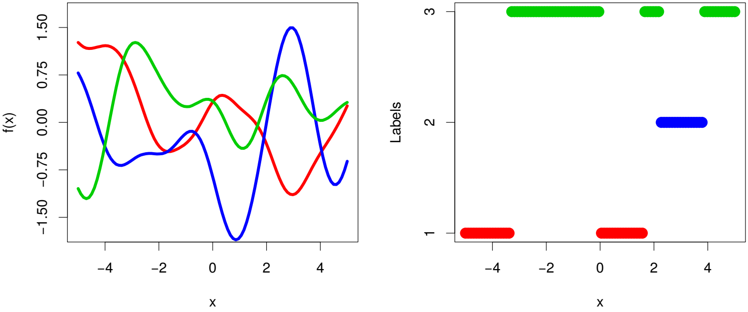


Introduction to Multi-class Classification with GPs

Given $\mathbf{x}_i$ we want to make **predictions** about $y_i \in \{1, \ldots, C\}$, $C > 2$.

One can **assume** that (Kim & Ghahramani, 2006):

$$y_i = \arg\max_k \quad f^k(\mathbf{x}_i) \quad \text{for} \quad k \in \{1, \ldots, C\}$$



Find $p(\mathbf{f}|\mathbf{y}) = p(\mathbf{y}|\mathbf{f})p(\mathbf{f})/p(\mathbf{y})$ **under** $p(\mathbf{f}^k) \sim \mathcal{GP}(0, k(\cdot, \cdot))$.

Efficient Methods for Multi-Class GPs

Introduce M inducing points $\{\overline{\mathbf{X}}^k\}_{k=1}^C$ per each class label k.

Efficient Methods for Multi-Class GPs

Introduce M inducing points $\{\overline{\mathbf{X}}^k\}_{k=1}^C$ per each class label k.

The **posterior approximation** is $q(\mathbf{f}) = \int p(\mathbf{f}|\overline{\mathbf{f}})q(\overline{\mathbf{f}})d\overline{\mathbf{f}}$

$$q(\overline{\mathbf{f}}) = \prod_{k=1}^C \mathcal{N}(\overline{\mathbf{f}}^k|\boldsymbol{\mu}^k, \boldsymbol{\Sigma}^k)$$

$$\overline{\mathbf{f}}^k = (f^k(\overline{\mathbf{x}}_1^k), \ldots, f^k(\overline{\mathbf{x}}_M^k))^\text{T} \qquad \overline{\mathbf{X}}^k = (\overline{\mathbf{x}}_1^k, \ldots, \overline{\mathbf{x}}_M^k)^\text{T}$$

where $q(\overline{\mathbf{f}})$ intuitively approximates $p(\overline{\mathbf{f}}|\mathbf{y})$ and $p(\mathbf{f}|\overline{\mathbf{f}}) = \prod_{k=1}^C p(\mathbf{f}^k|\overline{\mathbf{f}}^k)$.

Efficient Methods for Multi-Class GPs

Introduce M inducing points $\{\overline{\mathbf{X}}^k\}_{k=1}^C$ per each class label k.

The **posterior approximation** is $q(\mathbf{f}) = \int p(\mathbf{f}|\bar{\mathbf{f}})q(\bar{\mathbf{f}})d\bar{\mathbf{f}}$

$$q(\bar{\mathbf{f}}) = \prod_{k=1}^C \mathcal{N}(\bar{\mathbf{f}}^k|\boldsymbol{\mu}^k, \boldsymbol{\Sigma}^k)$$

$$\bar{\mathbf{f}}^k = (f^k(\bar{\mathbf{x}}_1^k), \ldots, f^k(\bar{\mathbf{x}}_M^k))^\text{T} \qquad \overline{\mathbf{X}}^k = (\bar{\mathbf{x}}_1^k, \ldots, \bar{\mathbf{x}}_M^k)^\text{T}$$

where $q(\bar{\mathbf{f}})$ intuitively approximates $p(\bar{\mathbf{f}}|\mathbf{y})$ and $p(\mathbf{f}|\bar{\mathbf{f}}) = \prod_{k=1}^C p(\mathbf{f}^k|\bar{\mathbf{f}}^k)$.

The number of **latent variables** goes from CN to CM, with $M \ll N$.

Efficient Methods for Multi-Class GPs

Introduce M inducing points $\{\overline{\mathbf{X}}^k\}_{k=1}^C$ per each class label k.

The **posterior approximation** is $q(\mathbf{f}) = \int p(\mathbf{f}|\overline{\mathbf{f}})q(\overline{\mathbf{f}})d\overline{\mathbf{f}}$

$$q(\overline{\mathbf{f}}) = \prod_{k=1}^C \mathcal{N}(\overline{\mathbf{f}}^k|\boldsymbol{\mu}^k, \boldsymbol{\Sigma}^k)$$

$$\overline{\mathbf{f}}^k = (f^k(\overline{\mathbf{x}}_1^k), \ldots, f^k(\overline{\mathbf{x}}_M^k))^\mathsf{T} \qquad \overline{\mathbf{X}}^k = (\overline{\mathbf{x}}_1^k, \ldots, \overline{\mathbf{x}}_M^k)^\mathsf{T}$$

where $q(\overline{\mathbf{f}})$ intuitively approximates $p(\overline{\mathbf{f}}|\mathbf{y})$ and $p(\mathbf{f}|\overline{\mathbf{f}}) = \prod_{k=1}^C p(\mathbf{f}^k|\overline{\mathbf{f}}^k)$.

The number of **latent variables** goes from CN to CM, with $M \ll N$.

Minibatches and stochastic gradients reduce the cost to $\mathcal{O}(CM)$.

Stochastic Variational Inference for Multi-class GPs

Hensman *et al.*, 2015, use a **robust likelihood** function:

$$p(y_i|\mathbf{f}_i) = (1-\epsilon)p_i + \frac{\epsilon}{C-1}(1-p_i) \quad \text{with} \quad p_i = \begin{cases} 1 & \text{if} \quad y_i = arg\,\max_k \quad f^k(\mathbf{x}_i) \\ 0 & \text{otherwise} \end{cases}$$

Stochastic Variational Inference for Multi-class GPs

Hensman *et al.*, 2015, use a **robust likelihood** function:

$$p(y_i|\mathbf{f}_i) = (1-\epsilon)p_i + \frac{\epsilon}{C-1}(1-p_i) \quad \text{with} \quad p_i = \begin{cases} 1 & \text{if} \quad y_i = \underset{k}{arg\,max} \quad f^k(\mathbf{x}_i) \\ 0 & \text{otherwise} \end{cases}$$

Based on minimizing $\text{KL}(p(\mathbf{f}|\bar{\mathbf{f}})q(\bar{\mathbf{f}})|p(\bar{\mathbf{f}},\mathbf{f}|\mathbf{y}))$:

Stochastic Variational Inference for Multi-class GPs

Hensman *et al.*, 2015, use a **robust likelihood** function:

$$p(y_i|\mathbf{f}_i) = (1-\epsilon)p_i + \frac{\epsilon}{C-1}(1-p_i) \quad \text{with} \quad p_i = \begin{cases} 1 & \text{if} \quad y_i = arg\,\max_k \quad f^k(\mathbf{x}_i) \\ 0 & \text{otherwise} \end{cases}$$

Based on minimizing $\mathrm{KL}(p(\mathbf{f}|\bar{\mathbf{f}})q(\bar{\mathbf{f}})|p(\bar{\mathbf{f}},\mathbf{f}|\mathbf{y}))$:

$$\mathcal{L}(q) = \sum_{i=1}^{N} \mathbb{E}_q\left[\log p(y_i|\mathbf{f}_i)\right] - \mathrm{KL}(q(\bar{\mathbf{f}})|p(\bar{\mathbf{f}}))$$

Stochastic Variational Inference for Multi-class GPs

Hensman *et al.*, 2015, use a **robust likelihood** function:

$$p(y_i|\mathbf{f}_i) = (1-\epsilon)p_i + \frac{\epsilon}{C-1}(1-p_i) \quad \text{with} \quad p_i = \begin{cases} 1 & \text{if} \quad y_i = \arg\max_k \quad f^k(\mathbf{x}_i) \\ 0 & \text{otherwise} \end{cases}$$

Based on minimizing $\mathrm{KL}(p(\mathbf{f}|\bar{\mathbf{f}})q(\bar{\mathbf{f}})|p(\bar{\mathbf{f}},\mathbf{f}|\mathbf{y}))$:

$$\mathcal{L}(q) = \sum_{i=1}^{N} \mathbb{E}_q\left[\log p(y_i|\mathbf{f}_i)\right] - \mathrm{KL}(q(\bar{\mathbf{f}})|p(\bar{\mathbf{f}}))$$

- Stochastic optimization of $q(\bar{\mathbf{f}})$ and the hyper-parameters!

Stochastic Variational Inference for Multi-class GPs

Hensman *et al.*, 2015, use a **robust likelihood** function:

$$p(y_i|\mathbf{f}_i) = (1-\epsilon)p_i + \frac{\epsilon}{C-1}(1-p_i) \quad \text{with} \quad p_i = \begin{cases} 1 & \text{if} \quad y_i = arg\max_k \quad f^k(\mathbf{x}_i) \\ 0 & \text{otherwise} \end{cases}$$

Based on minimizing $\text{KL}(p(\mathbf{f}|\bar{\mathbf{f}})q(\bar{\mathbf{f}})|p(\bar{\mathbf{f}},\mathbf{f}|\mathbf{y}))$:

$$\mathcal{L}(q) = \sum_{i=1}^{N} \mathbb{E}_q \left[\log p(y_i|\mathbf{f}_i)\right] - \text{KL}(q(\bar{\mathbf{f}})|p(\bar{\mathbf{f}}))$$

- Stochastic optimization of $q(\bar{\mathbf{f}})$ and the hyper-parameters!
- The cost is $\mathcal{O}(CM^3)$ (uses **quadratures**)!

Expectation Propagation (EP)

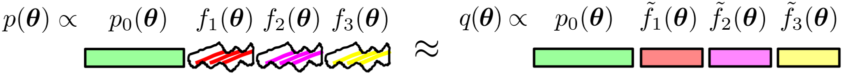
Let $\boldsymbol{\theta}$ summarize the latent variables of the model.

Approximates $\boxed{p(\boldsymbol{\theta}) \propto p_0(\boldsymbol{\theta}) \prod_{n=1}^{N} f_n(\boldsymbol{\theta})}$ with $\boxed{q(\boldsymbol{\theta}) \propto p_0(\boldsymbol{\theta}) \prod_{n=1}^{N} \tilde{f}_n(\boldsymbol{\theta})}$

Expectation Propagation (EP)

Let $\boldsymbol{\theta}$ summarize the latent variables of the model.

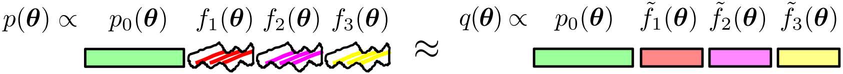
Approximates $p(\boldsymbol{\theta}) \propto p_0(\boldsymbol{\theta}) \prod_{n=1}^{N} f_n(\boldsymbol{\theta})$ with $q(\boldsymbol{\theta}) \propto p_0(\boldsymbol{\theta}) \prod_{n=1}^{N} \tilde{f}_n(\boldsymbol{\theta})$



Expectation Propagation (EP)

Let $\boldsymbol{\theta}$ summarize the latent variables of the model.

Approximates $p(\boldsymbol{\theta}) \propto p_0(\boldsymbol{\theta}) \prod_{n=1}^{N} f_n(\boldsymbol{\theta})$ with $q(\boldsymbol{\theta}) \propto p_0(\boldsymbol{\theta}) \prod_{n=1}^{N} \tilde{f}_n(\boldsymbol{\theta})$



The $\tilde{f}_n$ are tuned by minimizing the KL divergence

$$D_{\text{KL}}[p_n || q] \quad \text{for } n = 1, \ldots, N\,, \quad \text{where} \quad \begin{array}{rcl} p_n(\boldsymbol{\theta}) & \propto & f_n(\boldsymbol{\theta}) \prod_{j \neq n} \tilde{f}_j(\boldsymbol{\theta}) \\ q(\boldsymbol{\theta}) & \propto & \tilde{f}_n(\boldsymbol{\theta}) \prod_{j \neq n} \tilde{f}_j(\boldsymbol{\theta}) \end{array}.$$

Model Specification (Villacampa-Calvo and Hernández-Lobato, 2017)

Consider that $y_i = \arg\max_k \; f^k(\mathbf{x}_i)$, which gives the **likelihood**:

$$p(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^{N} p(y_i|\mathbf{f}_i) = \prod_{i=1}^{N} \prod_{k \neq y_i} \Theta(f^{y_i}(\mathbf{x}_i) - f^k(\mathbf{x}_i))$$

Model Specification (Villacampa-Calvo and Hernández-Lobato, 2017)

Consider that $y_i = \arg\max_k \quad f^k(\mathbf{x}_i)$, which gives the **likelihood**:

$$p(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^{N} p(y_i|\mathbf{f}_i) = \prod_{i=1}^{N} \prod_{k \neq y_i} \Theta(f^{y_i}(\mathbf{x}_i) - f^k(\mathbf{x}_i))$$

The **posterior approximation** is also set to be $q(\mathbf{f}) = \int p(\mathbf{f}|\overline{\mathbf{f}})q(\overline{\mathbf{f}})d\overline{\mathbf{f}}$.

Model Specification (Villacampa-Calvo and Hernández-Lobato, 2017)

Consider that $y_i = \arg\max_k \; f^k(\mathbf{x}_i)$, which gives the **likelihood**:

$$p(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^N p(y_i|\mathbf{f}_i) = \prod_{i=1}^N \prod_{k \neq y_i} \Theta(f^{y_i}(\mathbf{x}_i) - f^k(\mathbf{x}_i))$$

The **posterior approximation** is also set to be $q(\mathbf{f}) = \int p(\mathbf{f}|\bar{\mathbf{f}})q(\bar{\mathbf{f}})d\bar{\mathbf{f}}$.

We enforce that $q(\bar{\mathbf{f}}) \approx p(\bar{\mathbf{f}}|\mathbf{y})$. The **posterior** over $\bar{\mathbf{f}}$ is:

$$p(\bar{\mathbf{f}}|\mathbf{y}) = \frac{\int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\bar{\mathbf{f}})d\mathbf{f}\, p(\bar{\mathbf{f}})}{p(\mathbf{y})} \approx \frac{[\prod_{i=1}^N \int p(y_i|\mathbf{f}_i)p(\mathbf{f}_i|\bar{\mathbf{f}})d\mathbf{f}_i]p(\bar{\mathbf{f}})}{p(\mathbf{y})}$$

where we have used the FITC approximation $p(\mathbf{f}|\bar{\mathbf{f}}) \approx \prod_{i=1}^N p(\mathbf{f}_i|\bar{\mathbf{f}})$.

Model Specification (Villacampa-Calvo and Hernández-Lobato, 2017)

Consider that $y_i = \arg\max_k \; f^k(\mathbf{x}_i)$, which gives the **likelihood**:

$$p(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^N p(y_i|\mathbf{f}_i) = \prod_{i=1}^N \prod_{k \neq y_i} \Theta(f^{y_i}(\mathbf{x}_i) - f^k(\mathbf{x}_i))$$

The **posterior approximation** is also set to be $q(\mathbf{f}) = \int p(\mathbf{f}|\overline{\mathbf{f}})q(\overline{\mathbf{f}})d\overline{\mathbf{f}}$.

We enforce that $q(\overline{\mathbf{f}}) \approx p(\overline{\mathbf{f}}|\mathbf{y})$. The **posterior** over $\overline{\mathbf{f}}$ is:

$$p(\overline{\mathbf{f}}|\mathbf{y}) = \frac{\int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\overline{\mathbf{f}})d\mathbf{f}\, p(\overline{\mathbf{f}})}{p(\mathbf{y})} \approx \frac{[\prod_{i=1}^N \int p(y_i|\mathbf{f}_i)p(\mathbf{f}_i|\overline{\mathbf{f}})d\mathbf{f}_i]p(\overline{\mathbf{f}})}{p(\mathbf{y})}$$

where we have used the FITC approximation $p(\mathbf{f}|\overline{\mathbf{f}}) \approx \prod_{i=1}^N p(\mathbf{f}_i|\overline{\mathbf{f}})$.

The corresponding **likelihood factors** are:

$$\phi_i(\overline{\mathbf{f}}) = \int \left[\prod_{k \neq y_i} \Theta\left(f_i^{y_i} - f_i^k\right)\right] \prod_{k=1}^C p(f_i^k|\overline{\mathbf{f}}^k)d\mathbf{f}_i$$
$$\approx \prod_{k \neq y_i} p(f_i^{y_i} > f_i^k) = \prod_{k \neq y_i} \Phi(\alpha_i^k)$$

Efficient EP using Mini-batches

Consider a **minibatch** of data $\mathcal{M}_b$:

Efficient EP using Mini-batches

Consider a **minibatch** of data $\mathcal{M}_b$:

1. Refine in parallel all approximate factors $\tilde{\phi}_{i,k}$ corresponding to $\mathcal{M}_b$.

Efficient EP using Mini-batches

Consider a **minibatch** of data $\mathcal{M}_b$:

1. Refine in parallel all approximate factors $\tilde{\phi}_{i,k}$ corresponding to $\mathcal{M}_b$.
2. Reconstruct the posterior approximation q.

Efficient EP using Mini-batches

Consider a **minibatch** of data $\mathcal{M}_b$:

1. Refine in parallel all approximate factors $\tilde{\phi}_{i,k}$ corresponding to $\mathcal{M}_b$.
2. Reconstruct the posterior approximation q.
3. Get a noisy estimate of the grad of $\log Z_q$ w.r.t to each ξ_j^k and $\overline{x}_{i,d}^k$.

Efficient EP using Mini-batches

Consider a **minibatch** of data $\mathcal{M}_b$:

1. Refine in parallel all approximate factors $\tilde{\phi}_{i,k}$ corresponding to $\mathcal{M}_b$.
2. Reconstruct the posterior approximation q.
3. Get a noisy estimate of the grad of $\log Z_q$ w.r.t to each ξ_j^k and $\overline{x}_{i,d}^k$.
4. Update all model hyper-parameters.

Efficient EP using Mini-batches

Consider a **minibatch** of data $\mathcal{M}_b$:

1. Refine in parallel all approximate factors $\tilde{\phi}_{i,k}$ corresponding to $\mathcal{M}_b$.
2. Reconstruct the posterior approximation q.
3. Get a noisy estimate of the grad of $\log Z_q$ w.r.t to each ξ_j^k and $\overline{x}_{i,d}^k$.
4. Update all model hyper-parameters.
5. Reconstruct the posterior approximation q.

Efficient EP using Mini-batches

Consider a **minibatch** of data $\mathcal{M}_b$:

1. Refine in parallel all approximate factors $\tilde{\phi}_{i,k}$ corresponding to $\mathcal{M}_b$.
2. Reconstruct the posterior approximation q.
3. Get a noisy estimate of the grad of $\log Z_q$ w.r.t to each ξ_j^k and $\overline{x}_{i,d}^k$.
4. Update all model hyper-parameters.
5. Reconstruct the posterior approximation q.

If $|\mathcal{M}_b| < M$ the **cost** is $\mathcal{O}(CM^3)$.

α-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).

α-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).

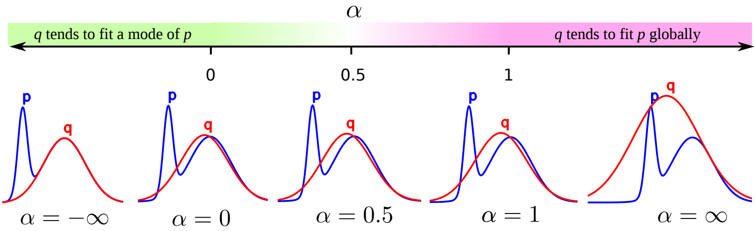


Figure source: (Minka, 2005).

α-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).

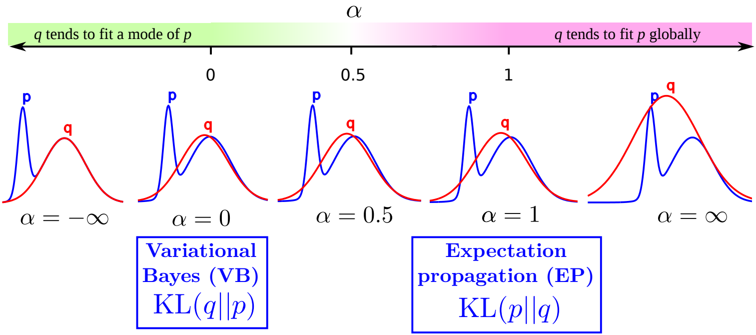


Figure source: (Minka, 2005).

Local α-divergence minimization (Power EP)

Approximates $p(\boldsymbol{\theta}) \propto p_0(\boldsymbol{\theta}) \prod_{n=1}^{N} f_n(\boldsymbol{\theta})$ with $q(\boldsymbol{\theta}) \propto p_0(\boldsymbol{\theta}) \prod_{n=1}^{N} \tilde{f}_n(\boldsymbol{\theta})$

(Minka, 2004)

Local α-divergence minimization (Power EP)

Approximates $p(\boldsymbol{\theta}) \propto p_0(\boldsymbol{\theta}) \prod_{n=1}^{N} f_n(\boldsymbol{\theta})$ with $q(\boldsymbol{\theta}) \propto p_0(\boldsymbol{\theta}) \prod_{n=1}^{N} \tilde{f}_n(\boldsymbol{\theta})$

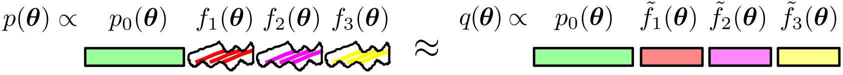
(Minka, 2004)

$$p(\boldsymbol{\theta}) \propto \; p_0(\boldsymbol{\theta}) \; f_1(\boldsymbol{\theta}) \; f_2(\boldsymbol{\theta}) \; f_3(\boldsymbol{\theta}) \quad \approx \quad q(\boldsymbol{\theta}) \propto \; p_0(\boldsymbol{\theta}) \; \tilde{f}_1(\boldsymbol{\theta}) \; \tilde{f}_2(\boldsymbol{\theta}) \; \tilde{f}_3(\boldsymbol{\theta})$$

Local α-divergence minimization (Power EP)

Approximates $p(\boldsymbol{\theta}) \propto p_0(\boldsymbol{\theta}) \prod_{n=1}^{N} f_n(\boldsymbol{\theta})$ with $q(\boldsymbol{\theta}) \propto p_0(\boldsymbol{\theta}) \prod_{n=1}^{N} \tilde{f}_n(\boldsymbol{\theta})$

(Minka, 2004)



The $\tilde{f}_n$ are tuned by minimizing the local α-divergences

$$D_\alpha[p_n || q] \quad \text{for } n = 1, \ldots, N, \quad \text{where} \quad \begin{array}{rcl} p_n(\boldsymbol{\theta}) & \propto & f_n(\boldsymbol{\theta}) \prod_{j \neq n} \tilde{f}_j(\boldsymbol{\theta}) \\ q(\boldsymbol{\theta}) & \propto & \tilde{f}_n(\boldsymbol{\theta}) \prod_{j \neq n} \tilde{f}_j(\boldsymbol{\theta}) \end{array}.$$

α-divergence minimization via KL minimization

Power EP steps to refine $\tilde{f}_n$:

α-divergence minimization via KL minimization

Power EP steps to refine $\tilde{f}_n$:

1. Compute cavity: $q^{\setminus \alpha n} \propto q / \tilde{f}_n^{\alpha}$.

α-divergence minimization via KL minimization

Power EP steps to refine $\tilde{f}_n$:

1. Compute cavity: $q^{\setminus \alpha n} \propto q / \tilde{f}_n^{\alpha}$.
2. Minimize $\mathrm{KL}(Z_n^{-1} f_n^{\alpha} q^{\setminus \alpha n} || q)$ to find q^{new}.

α-divergence minimization via KL minimization

Power EP steps to refine $\tilde{f}_n$:

1. Compute cavity: $q^{\backslash \alpha n} \propto q / \tilde{f}_n^{\alpha}$.
2. Minimize $\mathrm{KL}(Z_n^{-1} f_n^{\alpha} q^{\backslash \alpha n} || q)$ to find q^{new}.
3. Update factor: $\tilde{f}_n^{\text{new}} = (Z_n q^{\text{new}} / q^{\backslash \alpha n})^{\frac{1}{\alpha}}$.

α-divergence minimization via KL minimization

Power EP steps to refine $\tilde{f}_n$:

1. Compute cavity: $q^{\backslash \alpha n} \propto q / \tilde{f}_n^{\alpha}$.
2. Minimize $\mathrm{KL}(Z_n^{-1} f_n^{\alpha} q^{\backslash \alpha n} || q)$ to find q^{new}.
3. Update factor: $\tilde{f}_n^{\mathrm{new}} = (Z_n q^{\mathrm{new}} / q^{\backslash \alpha n})^{\frac{1}{\alpha}}$.

At convergence the moments of $\tilde{p} = Z_n^{-1} f_n^{\alpha} q^{\backslash \alpha n}$ and q match!

α-divergence minimization via KL minimization

Power EP steps to refine $\tilde{f}_n$:

1. Compute cavity: $q^{\backslash \alpha n} \propto q / \tilde{f}_n^{\alpha}$.
2. Minimize $\mathrm{KL}(Z_n^{-1} f_n^{\alpha} q^{\backslash \alpha n} || q)$ to find q^{new}.
3. Update factor: $\tilde{f}_n^{\mathrm{new}} = (Z_n q^{\mathrm{new}} / q^{\backslash \alpha n})^{\frac{1}{\alpha}}$.

At convergence the moments of $\tilde{p} = Z_n^{-1} f_n^{\alpha} q^{\backslash \alpha n}$ and q match!

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

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

α-divergence minimization via KL minimization

Power EP steps to refine $\tilde{f}_n$:

1. Compute cavity: $q^{\setminus \alpha n} \propto q/\tilde{f}_n^\alpha$.
2. Minimize $\mathrm{KL}(Z_n^{-1} f_n^\alpha q^{\setminus \alpha n} || q)$ to find q^{new}.
3. Update factor: $\tilde{f}_n^{\text{new}} = (Z_n q^{\text{new}} / q^{\setminus \alpha n})^{\frac{1}{\alpha}}$.

At convergence the moments of $\tilde{p} = Z_n^{-1} f_n^\alpha q^{\setminus \alpha n}$ and q match!

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

where $\tilde{p} \propto (f_n q^{\setminus n})^\alpha q^{1-\alpha} = f_n^\alpha q^{\setminus \alpha n}$.

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

Alternative Algorithms for PEP

The Power-EP approximation to the **evidence** is given by

$$\log Z_{\text{PEP}} = \log Z_q - \log Z_{\text{prior}} + \sum_{n=1}^{N} \frac{1}{\alpha} \log \mathbf{E}_q \left[\left(\frac{f_n(\boldsymbol{\theta})}{\tilde{f}_n(\boldsymbol{\theta})} \right)^{\alpha} \right] \,,$$

Alternative Algorithms for PEP

The Power-EP approximation to the **evidence** is given by

$$\log Z_{\text{PEP}} = \log Z_q - \log Z_{\text{prior}} + \sum_{n=1}^{N} \frac{1}{\alpha} \log \mathbf{E}_q \left[\left(\frac{f_n(\boldsymbol{\theta})}{\tilde{f}_n(\boldsymbol{\theta})} \right)^{\alpha} \right],$$

The power-EP solution for q can be obtained by solving

$$\max_{q} \min_{\tilde{f}_1,\ldots,\tilde{f}_N} \log Z_{\text{PEP}} \quad \text{subject to} \quad q(\boldsymbol{\theta}) = p_0(\boldsymbol{\theta}) \prod_{n=1}^{N} \tilde{f}_n(\boldsymbol{\theta}).$$

Alternative Algorithms for PEP

The Power-EP approximation to the **evidence** is given by

$$\log Z_{\text{PEP}} = \log Z_q - \log Z_{\text{prior}} + \sum_{n=1}^{N} \frac{1}{\alpha} \log \mathbf{E}_q \left[\left(\frac{f_n(\boldsymbol{\theta})}{\tilde{f}_n(\boldsymbol{\theta})} \right)^{\alpha} \right] ,$$

The power-EP solution for q can be obtained by solving

$$\max_{q} \min_{\tilde{f}_1,\ldots,\tilde{f}_N} \log Z_{\text{PEP}} \quad \text{subject to} \quad q(\boldsymbol{\theta}) = p_0(\boldsymbol{\theta}) \prod_{n=1}^{N} \tilde{f}_n(\boldsymbol{\theta}) .$$

Solved with **double-loop** algorithm (Heskes, 2002).

Alternative Algorithms for PEP

The Power-EP approximation to the **evidence** is given by

$$\log Z_{\text{PEP}} = \log Z_q - \log Z_{\text{prior}} + \sum_{n=1}^{N} \frac{1}{\alpha} \log \mathbf{E}_q \left[\left(\frac{f_n(\boldsymbol{\theta})}{\tilde{f}_n(\boldsymbol{\theta})} \right)^{\alpha} \right],$$

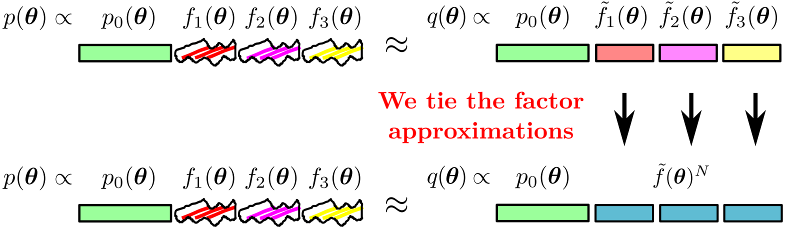
The power-EP solution for q can be obtained by solving

$$\max_{q} \min_{\tilde{f}_1,\ldots,\tilde{f}_N} \log Z_{\text{PEP}} \quad \text{subject to} \quad q(\boldsymbol{\theta}) = p_0(\boldsymbol{\theta}) \prod_{n=1}^{N} \tilde{f}_n(\boldsymbol{\theta}).$$

Solved with **double-loop** algorithm (Heskes, 2002). **Too slow in practice!**

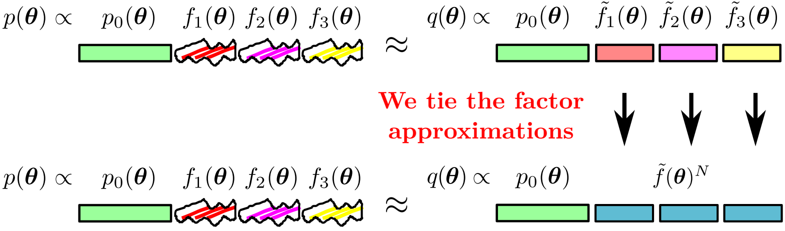
Approximate Power EP (APEP)

By following (Li et al., 2015) (Bui et al., 2016):



Approximate Power EP (APEP)

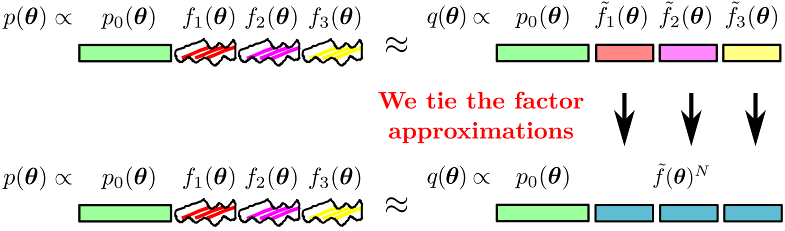
By following (Li et al., 2015) (Bui et al., 2016):



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

Approximate Power EP (APEP)

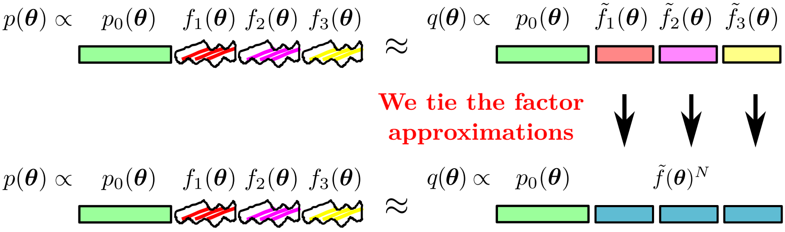
By following (Li et al., 2015) (Bui et al., 2016):



- $\max_{q} \min_{\tilde{f}_1,\ldots,\tilde{f}_N}$ problem $\rightarrow$ $\max_{q}$ problem, **no double-loop needed!**
- **Memory saving** scales as $\mathcal{O}(N)$.

Approximate Power EP (APEP)

By following (Li et al., 2015) (Bui et al., 2016):



- $\max_{q} \min_{\tilde{f}_1,\ldots,\tilde{f}_N}$ problem $\rightarrow$ $\max_{q}$ problem, **no double-loop needed!**
- **Memory saving** scales as $\mathcal{O}(N)$.
- **Standard optimization tools** can be used (stochastic gradients).

Refined Prior Approximate Power EP (RPAPEP)

As $\alpha \to 0$ the PEP and APEP solution converges to a VI solution.

Refined Prior Approximate Power EP (RPAPEP)

As $\alpha \to 0$ the PEP and APEP solution converges to a VI solution.

Can all VI solutions be reached by minimizing the APEP objective?

Refined Prior Approximate Power EP (RPAPEP)

As $\alpha \to 0$ the PEP and APEP solution converges to a VI solution.

Can all VI solutions be reached by minimizing the APEP objective?

No since they use different parameterizations of q:

APEP or PEP	VI
$q \propto p_0 \tilde{f}^N$	$q \equiv$ Gaussian distribution

Refined Prior Approximate Power EP (RPAPEP)

As $\alpha \to 0$ the PEP and APEP solution converges to a VI solution.

Can all VI solutions be reached by minimizing the APEP objective?

No since they use different parameterizations of q:

APEP or PEP	**VI**
$q \propto p_0 \tilde{f}^N$	$q \equiv$ Gaussian distribution

To avoid this we let $q \propto \tilde{f}^N$ and process the prior too!

Refined Prior Approximate Power EP (RPAPEP)

As $\alpha \to 0$ the PEP and APEP solution converges to a VI solution.

Can all VI solutions be reached by minimizing the APEP objective?

No since they use different parameterizations of q:

APEP or PEP	**VI**
$q \propto p_0 \tilde{f}^N$	$q \equiv$ Gaussian distribution

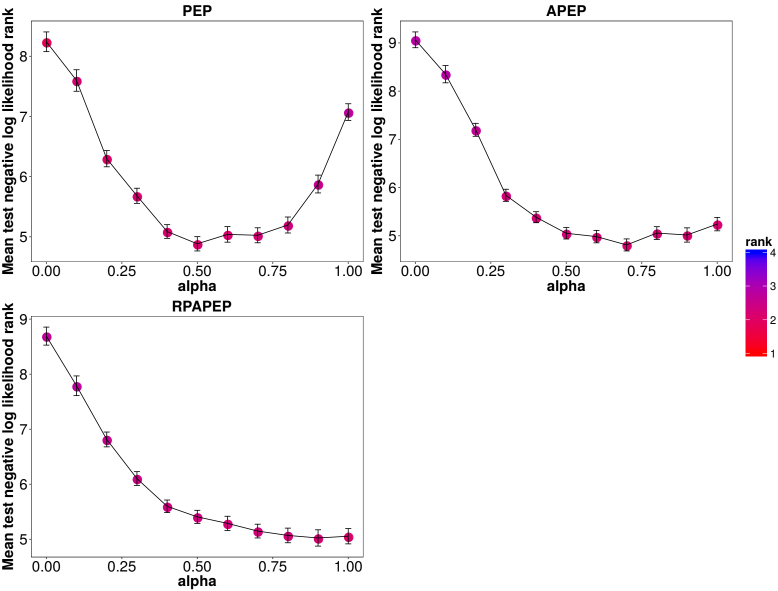
To avoid this we let $q \propto \tilde{f}^N$ and process the prior too!

$$\log Z_{\text{PEP}} = \log Z_q + \sum_{n=0}^{N} \frac{1}{\alpha} \log \mathbf{E}_q \left[\left(\frac{f_n(\boldsymbol{\theta})}{\tilde{f}(\boldsymbol{\theta})} \right)^{\alpha} \right] ,$$

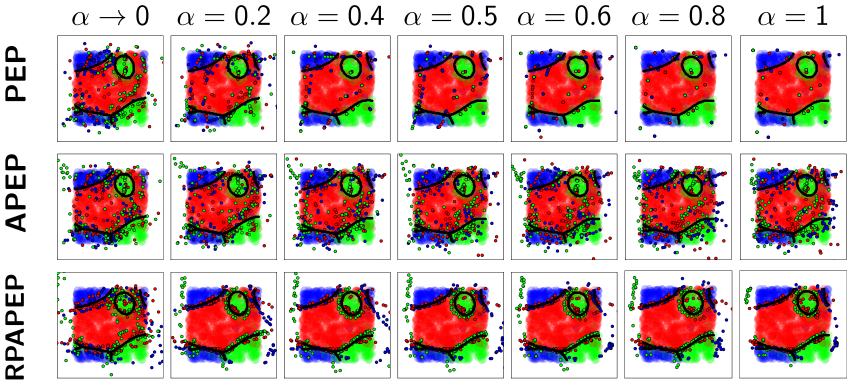
Experiments: UCI Datasets

Dataset	#Instances	#Attributes	#Classes
Glass	214	9	6
New-thyroid	215	5	3
Satellite	6435	36	6
Svmguide2	391	20	3
Vehicle	846	18	4
Vowel	540	10	6
Waveform	1000	21	3
Wine	178	13	3

Experiments: UCI Datasets

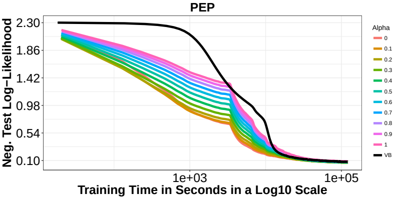


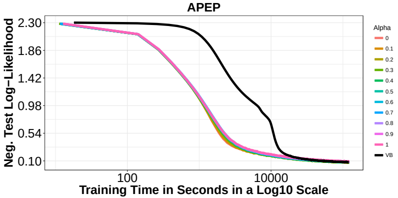
Toy Problem: Inducing Point Locations

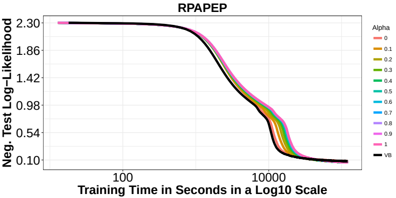


MNIST Dataset

10 classes, 60,000 training instances.

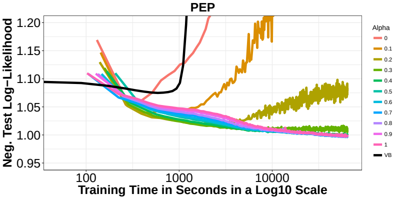


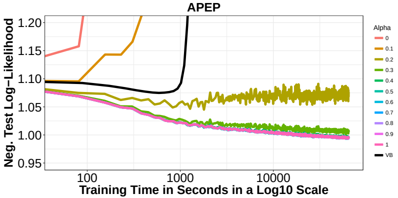


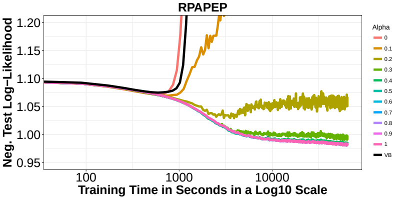


Airline Delays

3 classes, 2 million training instances.







Conclusions so far...

- We have described a **collection of methods** to approximately minimize α-divergences in MGPC.

Conclusions so far...

- We have described a **collection of methods** to approximately minimize α-divergences in MGPC.
- **Efficient** training and memory usage with cost $\mathcal{O}(CM^3)$.

Conclusions so far...

- We have described a **collection of methods** to approximately minimize α-divergences in MGPC.
- **Efficient** training and memory usage with cost $\mathcal{O}(CM^3)$.
- Extensive **experimental comparisons**.

Conclusions so far...

- We have described a **collection of methods** to approximately minimize α-divergences in MGPC.
- **Efficient** training and memory usage with cost $\mathcal{O}(CM^3)$.
- Extensive **experimental comparisons**.
- $\alpha = 0.5$ gives over-all **good results** in the experiments.

Conclusions so far...

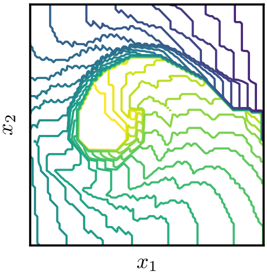
- We have described a **collection of methods** to approximately minimize α-divergences in MGPC.
- **Efficient** training and memory usage with cost $\mathcal{O}(CM^3)$.
- Extensive **experimental comparisons**.
- $\alpha = 0.5$ gives over-all **good results** in the experiments.
- $\alpha = 0.5$ sometimes outperforms **VB or EP methods** for MGPC.

Conclusions so far...

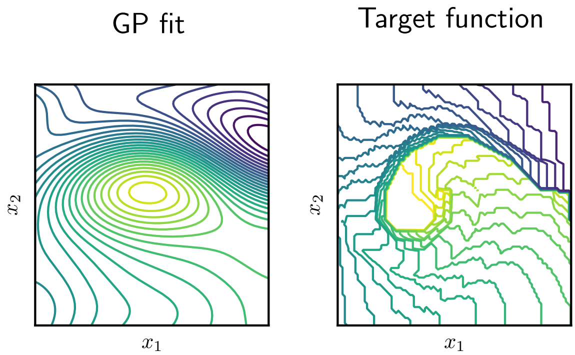
- We have described a **collection of methods** to approximately minimize α-divergences in MGPC.
- **Efficient** training and memory usage with cost $\mathcal{O}(CM^3)$.
- Extensive **experimental comparisons**.
- $\alpha = 0.5$ gives over-all **good results** in the experiments.
- $\alpha = 0.5$ sometimes outperforms **VB or EP methods** for MGPC.
- VB sometimes gives **bad test log-likelihoods**.

Motivation for Deep Gaussian Processes

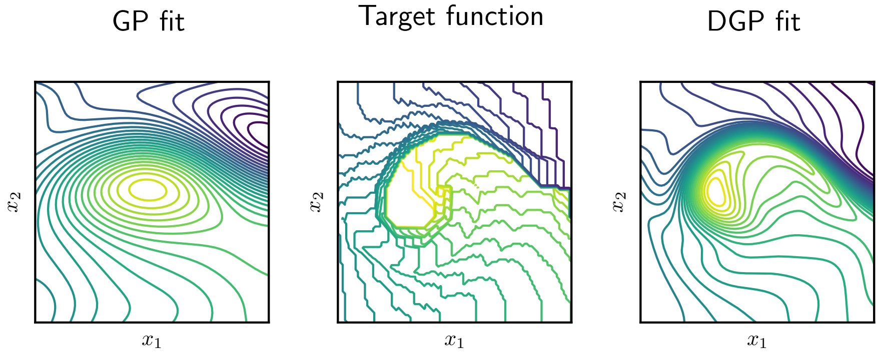
Target function



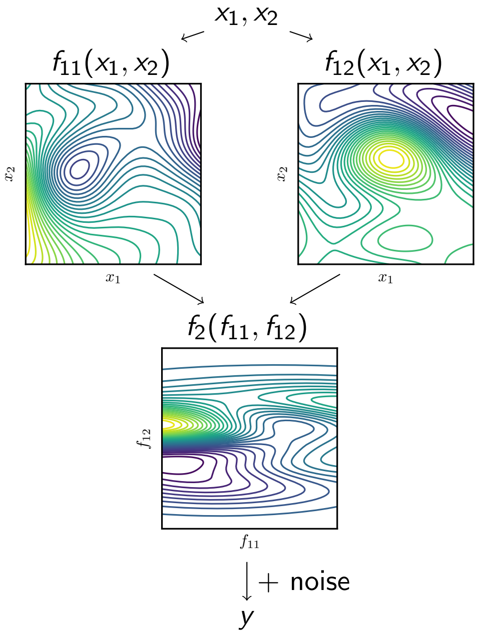
Motivation for Deep Gaussian Processes



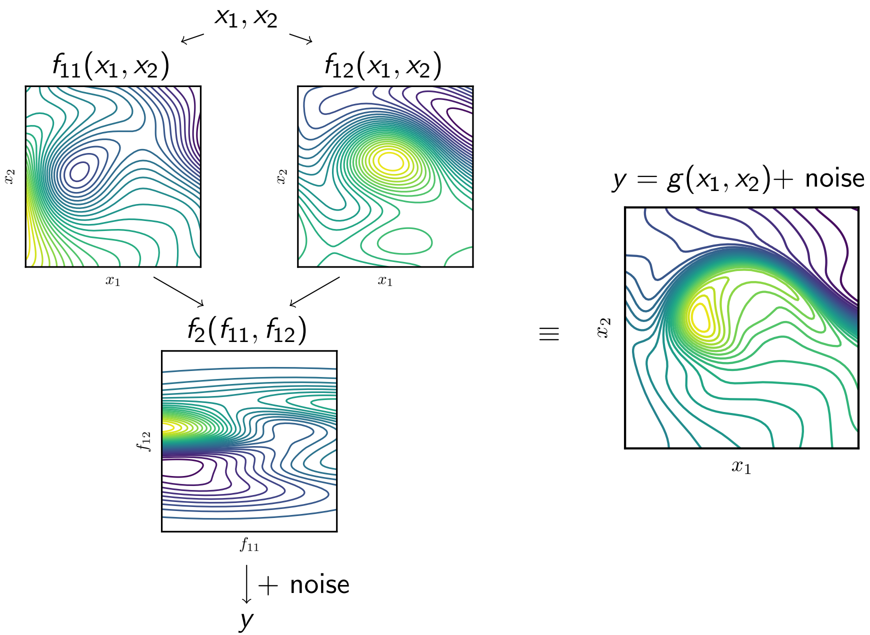
Motivation for Deep Gaussian Processes



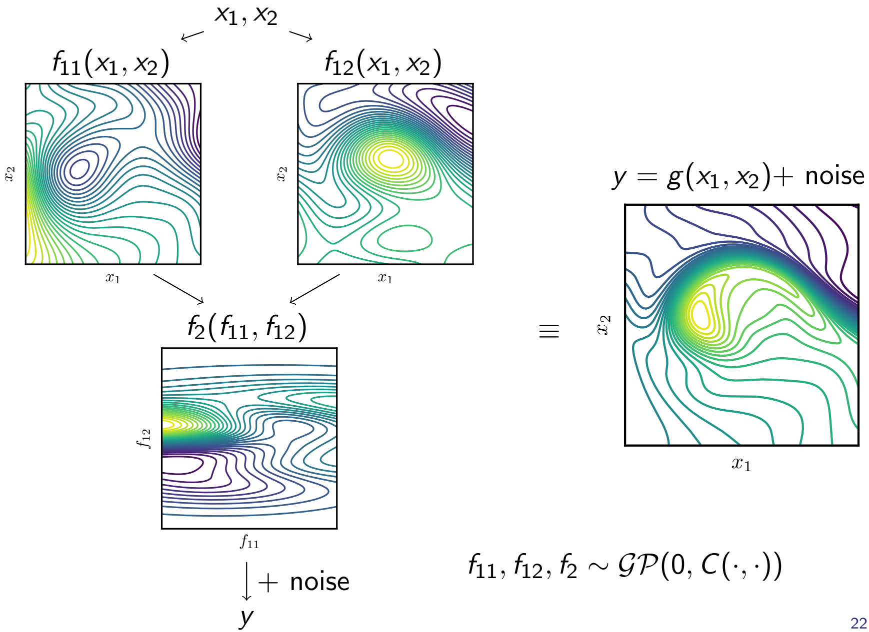
How do deep GPs work?



How do deep GPs work?

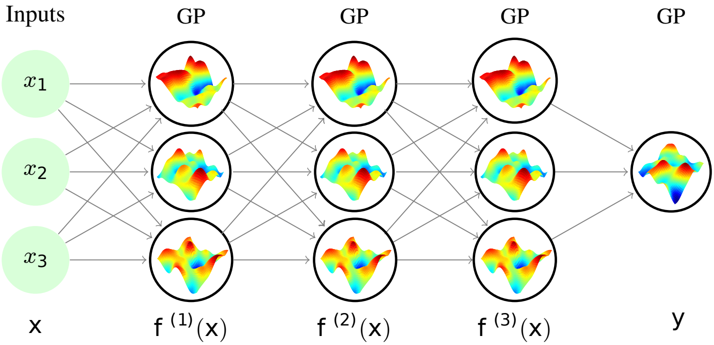


How do deep GPs work?



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

Deep GPs as Deep Neural Networks



Why deep GPs?

Advantages:

- useful input warping: automatic, nonparametric 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, nonparametric kernel design
- repair damage done by sparse approximations to GPs
- more accurate predictions and better uncertainty estimates

Drawbacks:

- require complicated approximate inference methods
- high computational cost

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 $p(f(\mathbf{x})|\mathbf{u}) = \mathcal{N}(f(\mathbf{x})|m_{\mathbf{x}}, v_{\mathbf{x}})$, where

$$m_{\mathbf{x}} = \mathbf{k}_{\mathbf{x},\bar{\mathbf{X}}}\mathbf{K}_{\bar{\mathbf{X}},\bar{\mathbf{X}}}^{-1}\mathbf{u}\,,$$
$$v_{\mathbf{x}} = \mathbf{k}_{\mathbf{x},\mathbf{x}} - \mathbf{k}_{\mathbf{x},\bar{\mathbf{X}}}\mathbf{K}_{\bar{\mathbf{X}},\bar{\mathbf{X}}}^{-1}\mathbf{k}_{\bar{\mathbf{X}},\mathbf{x}}\,.$$

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 $p(f(\mathbf{x})|\mathbf{u}) = \mathcal{N}(f(\mathbf{x})|m_{\mathbf{x}}, v_{\mathbf{x}})$, where

$$m_{\mathbf{x}} = \mathbf{k}_{\mathbf{x},\bar{\mathbf{X}}} \mathbf{K}_{\bar{\mathbf{X}},\bar{\mathbf{X}}}^{-1} \mathbf{u}\,,$$
$$v_{\mathbf{x}} = \mathbf{k}_{\mathbf{x},\mathbf{x}} - \mathbf{k}_{\mathbf{x},\bar{\mathbf{X}}} \mathbf{K}_{\bar{\mathbf{X}},\bar{\mathbf{X}}}^{-1} \mathbf{k}_{\bar{\mathbf{X}},\mathbf{x}}\,.$$

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

$$m_{\mathbf{x}} = \mathbf{k}_{\mathbf{x},\bar{\mathbf{X}}} \mathbf{K}_{\bar{\mathbf{X}},\bar{\mathbf{X}}}^{-1} \mathbf{m}\,,$$
$$v_{\mathbf{x}} = \mathbf{k}_{\mathbf{x},\mathbf{x}} - \mathbf{k}_{\mathbf{x},\bar{\mathbf{X}}} \mathbf{K}_{\bar{\mathbf{X}},\bar{\mathbf{X}}}^{-1} \mathbf{k}_{\bar{\mathbf{X}},\mathbf{x}} + \mathbf{k}_{\mathbf{x},\bar{\mathbf{X}}} \mathbf{K}_{\bar{\mathbf{X}},\bar{\mathbf{X}}}^{-1} \mathbf{S} \mathbf{K}_{\bar{\mathbf{X}},\bar{\mathbf{X}}}^{-1} \mathbf{k}_{\bar{\mathbf{X}},\mathbf{x}}\,.$$

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 $p(f(\mathbf{x})|\mathbf{u}) = \mathcal{N}(f(\mathbf{x})|m_{\mathbf{x}}, v_{\mathbf{x}})$, where

$$m_{\mathbf{x}} = \mathbf{k}_{\mathbf{x},\bar{\mathbf{X}}}\mathbf{K}_{\bar{\mathbf{X}},\bar{\mathbf{X}}}^{-1}\mathbf{u}\,,$$
$$v_{\mathbf{x}} = \mathbf{k}_{\mathbf{x},\mathbf{x}} - \mathbf{k}_{\mathbf{x},\bar{\mathbf{X}}}\mathbf{K}_{\bar{\mathbf{X}},\bar{\mathbf{X}}}^{-1}\mathbf{k}_{\bar{\mathbf{X}},\mathbf{x}}\,.$$

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

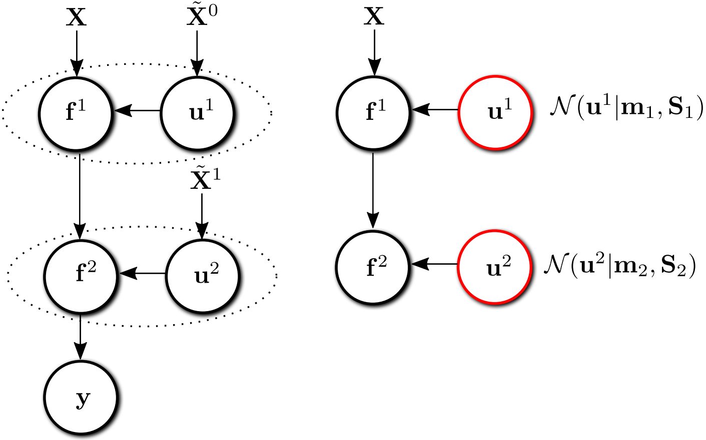
$$m_{\mathbf{x}} = \mathbf{k}_{\mathbf{x},\bar{\mathbf{X}}}\mathbf{K}_{\bar{\mathbf{X}},\bar{\mathbf{X}}}^{-1}\mathbf{m}\,,$$
$$v_{\mathbf{x}} = \mathbf{k}_{\mathbf{x},\mathbf{x}} - \mathbf{k}_{\mathbf{x},\bar{\mathbf{X}}}\mathbf{K}_{\bar{\mathbf{X}},\bar{\mathbf{X}}}^{-1}\mathbf{k}_{\bar{\mathbf{X}},\mathbf{x}} + \mathbf{k}_{\mathbf{x},\bar{\mathbf{X}}}\mathbf{K}_{\bar{\mathbf{X}},\bar{\mathbf{X}}}^{-1}\mathbf{S}\mathbf{K}_{\bar{\mathbf{X}},\bar{\mathbf{X}}}^{-1}\mathbf{k}_{\bar{\mathbf{X}},\mathbf{x}}\,.$$

Given u or a Gaussian for u, f is fully specified!

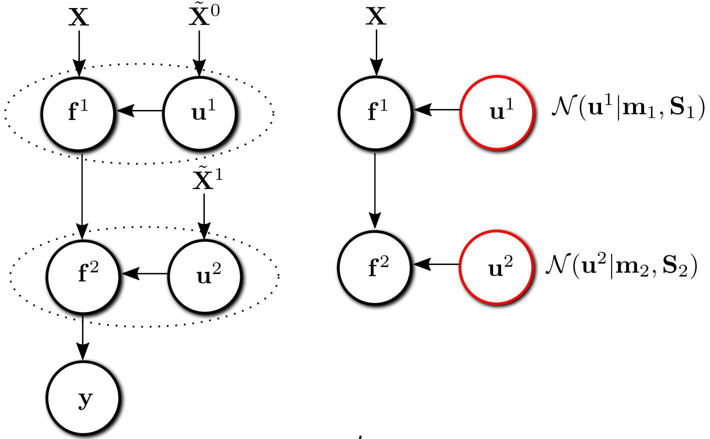
Deep Gaussian Process Joint Distribution.

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

Prob. Graphical Model and Posterior Approx.



Prob. Graphical Model and Posterior Approx.



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

- Fixed
- Tuneable

Variational Inference for Deep GPs

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

(Salimbeni, 2017)

Variational Inference for Deep GPs

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

Equivalent to maximizing:

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

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

(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:

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

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

- Suitable for stochastic optimization.

(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:

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

- Suitable for stochastic optimization.
- The expectations can be approximated by Monte Carlo.

(Salimbeni, 2017)

Approximate Expectation Propagation

The likelihood factors to be refined by EP are $p(y_i|f_i^L)$.

(Bui, 2016)

Approximate Expectation Propagation

The likelihood factors to be refined by EP are $p(y_i|f_i^L)$.

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

$$\log Z_{\mathrm{EP}} = \log Z_q - \log Z_{\mathrm{prior}} + \sum_{n=1}^{N} \log \mathbf{E}_q \left[\left(\frac{f_n(\boldsymbol{\theta})}{\tilde{f}_n(\boldsymbol{\theta})} \right) \right] ,$$

(Bui, 2016)

Approximate Expectation Propagation

The likelihood factors to be refined by EP are $p(y_i|f_i^L)$.

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

$$\log Z_{\text{EP}} = \log Z_q - \log Z_{\text{prior}} + \sum_{n=1}^{N} \log \mathbf{E}_q \left[\left(\frac{f_n(\boldsymbol{\theta})}{\tilde{f}_n(\boldsymbol{\theta})} \right) \right] ,$$

The EP solution for q can be obtained by solving

$$\max_{q} \min_{\tilde{f}_1,\ldots,\tilde{f}_N} \log Z_{\text{EP}} \quad \text{subject to} \quad q(\boldsymbol{\theta}) = p_0(\boldsymbol{\theta}) \prod_{n=1}^{N} \tilde{f}_n(\boldsymbol{\theta}) \,.$$

(Bui, 2016)

Approximate Expectation Propagation

The likelihood factors to be refined by EP are $p(y_i|f_i^L)$.

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

$$\log Z_{\text{EP}} = \log Z_q - \log Z_{\text{prior}} + \sum_{n=1}^{N} \log \mathbf{E}_q \left[\left(\frac{f_n(\boldsymbol{\theta})}{\tilde{f}_n(\boldsymbol{\theta})} \right) \right] ,$$

The EP solution for q can be obtained by solving

$$\max_q \min_{\tilde{f}_1,\ldots,\tilde{f}_N} \log Z_{\text{EP}} \quad \text{subject to} \quad q(\boldsymbol{\theta}) = p_0(\boldsymbol{\theta}) \prod_{n=1}^{N} \tilde{f}_n(\boldsymbol{\theta}) \,.$$

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

(Bui, 2016)

Approximate Expectation Propagation

The likelihood factors to be refined by EP are $p(y_i|f_i^L)$.

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

$$\log Z_{\text{EP}} = \log Z_q - \log Z_{\text{prior}} + \sum_{n=1}^{N} \log \mathbf{E}_q \left[\left(\frac{f_n(\boldsymbol{\theta})}{\tilde{f}_n(\boldsymbol{\theta})} \right) \right] ,$$

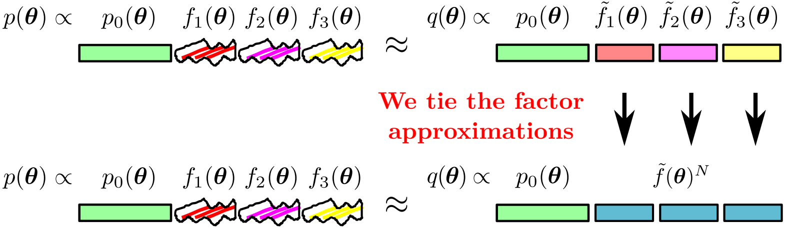
The EP solution for q can be obtained by solving

$$\max_{q} \min_{\tilde{f}_1,\ldots,\tilde{f}_N} \log Z_{\text{EP}} \quad \text{subject to} \quad q(\boldsymbol{\theta}) = p_0(\boldsymbol{\theta}) \prod_{n=1}^{N} \tilde{f}_n(\boldsymbol{\theta}) \, .$$

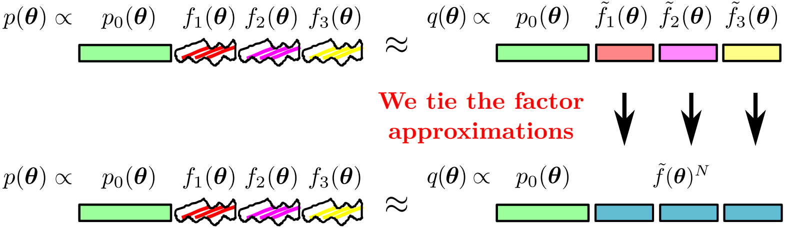
Can be solved with a **double-loop** algorithm. **Too slow in practice!**

(Bui, 2016)

Approximate Expectation Propagation

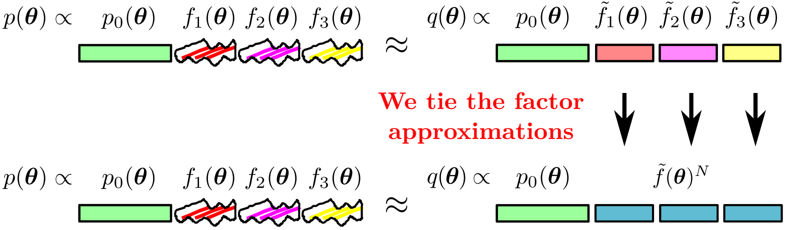


Approximate Expectation Propagation



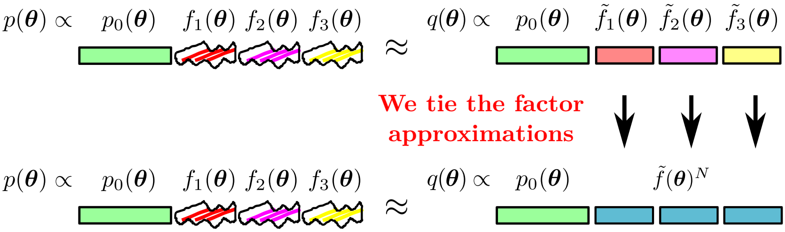
- $\max_q \min_{\tilde{f}_1,\ldots,\tilde{f}_N}$ problem $\to \max_q$ problem, **no double-loop needed!**

Approximate Expectation Propagation



- $\max_{q} \min_{\tilde{f}_1,\ldots,\tilde{f}_N}$ problem $\rightarrow$ $\max_{q}$ problem, **no double-loop needed!**
- **Memory saving** scales as $\mathcal{O}(N)$.

Approximate Expectation Propagation



- $\max_q \min_{\tilde{f}_1,\dots,\tilde{f}_N}$ problem $\rightarrow$ $\max_q$ problem, **no double-loop needed!**
- **Memory saving** scales as $\mathcal{O}(N)$.
- **Standard optimization tools** can be used (stochastic gradients).

Approximate EP

One only needs to optimize

$$\log Z_{\mathrm{EP}} = \log Z_q - \log Z_{\mathrm{prior}} + \sum_{n=1}^{N} \log \mathbf{E}_q \left[\left(\frac{f_n(\boldsymbol{\theta})}{\tilde{f}(\boldsymbol{\theta})} \right) \right] .$$

Approximate EP

One only needs to optimize

$$\log Z_{\text{EP}} = \log Z_q - \log Z_{\text{prior}} + \sum_{n=1}^{N} \log \mathbf{E}_q \left[\left(\frac{f_n(\boldsymbol{\theta})}{\tilde{f}(\boldsymbol{\theta})} \right) \right] .$$

But this requires integrating the exact likelihood factors (**intractable**).

Approximate EP

One only needs to optimize

$$\log Z_{\text{EP}} = \log Z_q - \log Z_{\text{prior}} + \sum_{n=1}^{N} \log \mathbf{E}_q \left[\left(\frac{f_n(\boldsymbol{\theta})}{\tilde{f}(\boldsymbol{\theta})} \right) \right] .$$

But this requires integrating the exact likelihood factors (**intractable**).

The **output distribution** after the second and next layers is too complex!

Approximate EP

One only needs to optimize

$$\log Z_{\text{EP}} = \log Z_q - \log Z_{\text{prior}} + \sum_{n=1}^{N} \log \mathbf{E}_q \left[\left(\frac{f_n(\boldsymbol{\theta})}{\tilde{f}(\boldsymbol{\theta})} \right) \right] .$$

But this requires integrating the exact likelihood factors (**intractable**).

The **output distribution** after the second and next layers is too complex!

Solution: **moment match** each GP output to a Gaussian at each layer.

Approximate EP

One only needs to optimize

$$\log Z_{\text{EP}} = \log Z_q - \log Z_{\text{prior}} + \sum_{n=1}^{N} \log \mathbf{E}_q \left[\left(\frac{f_n(\boldsymbol{\theta})}{\tilde{f}(\boldsymbol{\theta})} \right) \right] .$$

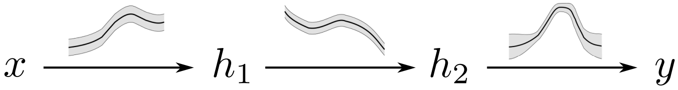
But this requires integrating the exact likelihood factors (**intractable**).

The **output distribution** after the second and next layers is too complex!

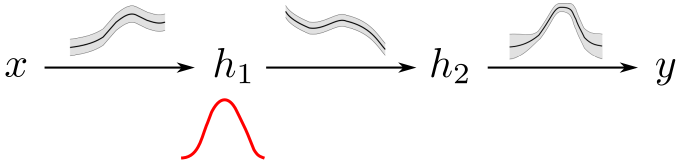
Solution: **moment match** each GP output to a Gaussian at each layer.

For some kernels it is possible to compute the moments of the GP predictive distribution with random Gaussian inputs!

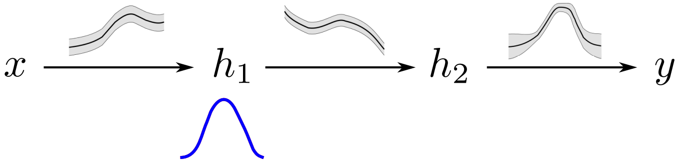
Iterative Gaussian Approximations



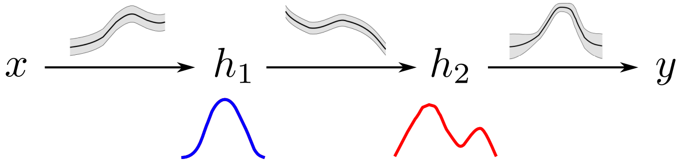
Iterative Gaussian Approximations



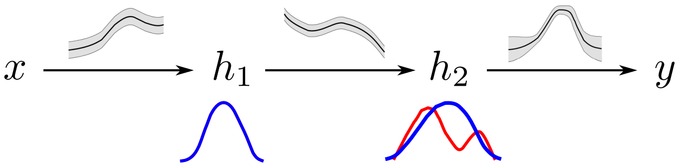
Iterative Gaussian Approximations



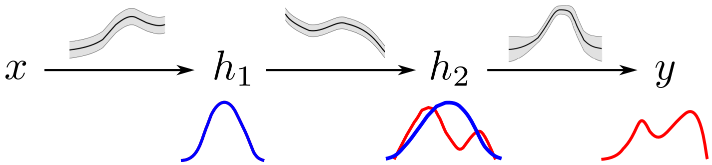
Iterative Gaussian Approximations



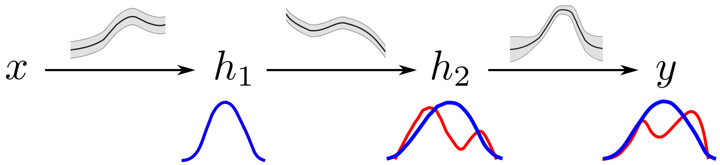
Iterative Gaussian Approximations



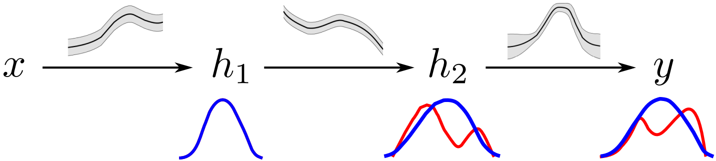
Iterative Gaussian Approximations



Iterative Gaussian Approximations



Iterative Gaussian Approximations



This approach allows to approximate the required expectations!

α-divergence Minimization for Deep GPs

One only needs to optimize the approximate Power EP objective:

$$\log Z_{\mathsf{EP}} = \log Z_q - \log Z_{\mathsf{prior}} + \frac{1}{\alpha} \sum_{n=1}^{N} \log \mathbf{E}_q \left[\left(\frac{f_n(\boldsymbol{\theta})}{\tilde{f}(\boldsymbol{\theta})} \right)^{\alpha} \right] .$$

α-divergence Minimization for Deep GPs

One only needs to optimize the approximate Power EP objective:

$$\log Z_{\mathrm{EP}} = \log Z_q - \log Z_{\mathrm{prior}} + \frac{1}{\alpha} \sum_{n=1}^{N} \log \mathbf{E}_q \left[\left(\frac{f_n(\boldsymbol{\theta})}{\tilde{f}(\boldsymbol{\theta})} \right)^{\alpha} \right] .$$

But this requires integrating the exact likelihood factors (**intractable**).

α-divergence Minimization for Deep GPs

One only needs to optimize the approximate Power EP objective:

$$\log Z_{\text{EP}} = \log Z_q - \log Z_{\text{prior}} + \frac{1}{\alpha}\sum_{n=1}^{N} \log \mathbf{E}_q \left[\left(\frac{f_n(\boldsymbol{\theta})}{\tilde{f}(\boldsymbol{\theta})} \right)^{\alpha} \right] .$$

But this requires integrating the exact likelihood factors (**intractable**).

We suggest to use a Monte Carlo approach similar to that of VI.

α-divergence Minimization for Deep GPs

One only needs to optimize the approximate Power EP objective:

$$\log Z_{\mathrm{EP}} = \log Z_q - \log Z_{\mathrm{prior}} + \frac{1}{\alpha} \sum_{n=1}^{N} \log \mathbf{E}_q \left[\left(\frac{f_n(\boldsymbol{\theta})}{\tilde{f}(\boldsymbol{\theta})} \right)^{\alpha} \right] .$$

But this requires integrating the exact likelihood factors (**intractable**).

We suggest to use a Monte Carlo approach similar to that of VI.

Expected to give better results than the Gaussian approximation!

Monte Carlo Approximation

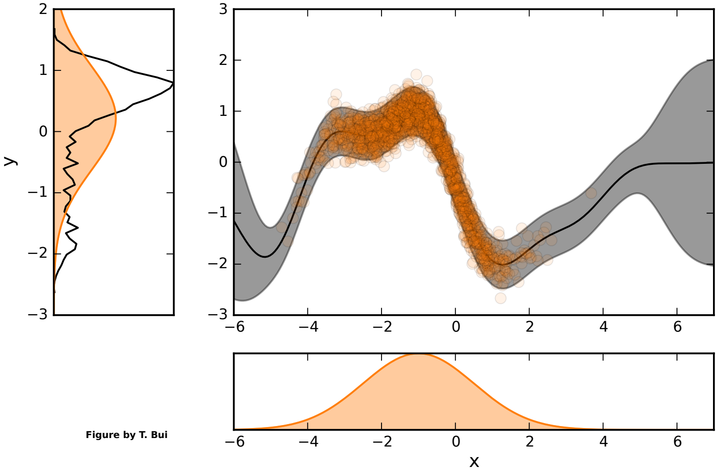
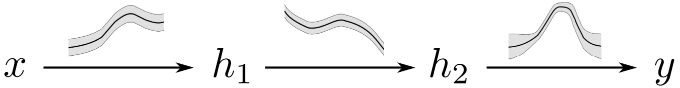


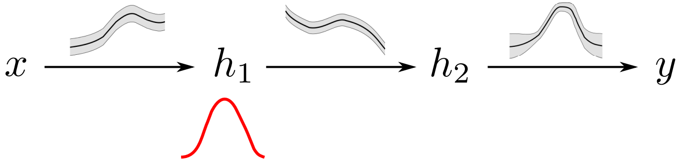
Figure by T. Bui

The predictive distribution with random Gaussian inputs may be very different from Gaussian!

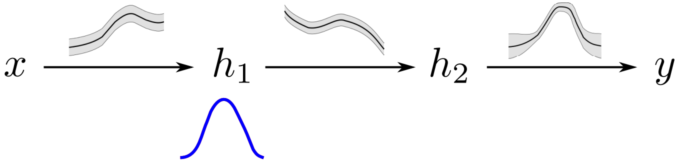
Monte Carlo Approximation



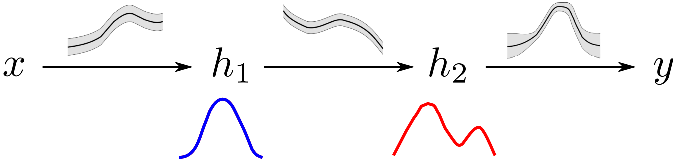
Monte Carlo Approximation



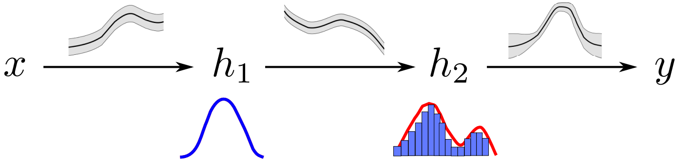
Monte Carlo Approximation



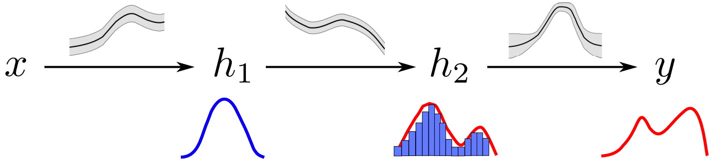
Monte Carlo Approximation



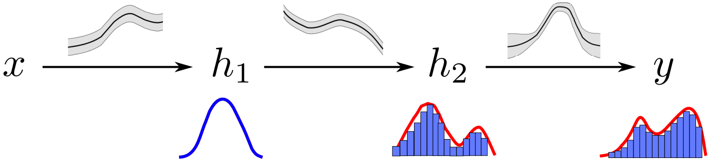
Monte Carlo Approximation



Monte Carlo Approximation



Monte Carlo Approximation



Monte Carlo Approximation

The required expectation is approximated as:

$$\frac{1}{\alpha}\log \mathbb{E}_q\left[\left(\frac{f_n(\boldsymbol{\theta})}{\tilde{f}(\boldsymbol{\theta})}\right)^{\alpha}\right] \approx \frac{1}{\alpha}\log\left(\frac{1}{S}\sum_{s=1}^{S} p(y_i|f_{i,s}^{L})\right) - \frac{g_q}{\alpha} + \frac{g_{q_{\text{cav}}^{\alpha}}}{\alpha}$$

Monte Carlo Approximation

The required expectation is approximated as:

$$\frac{1}{\alpha}\log \mathbb{E}_q\left[\left(\frac{f_n(\boldsymbol{\theta})}{\tilde{f}(\boldsymbol{\theta})}\right)^{\alpha}\right] \approx \frac{1}{\alpha}\log\left(\frac{1}{S}\sum_{s=1}^{S} p(y_i|f_{i,s}^L)\right) - \frac{g_q}{\alpha} + \frac{g_{q_{\text{cav}}^{\alpha}}}{\alpha}$$

$g_q \equiv$ Log. Normalizer of q.

$g_{q_{\text{cav}}^{\alpha}} \equiv$ Log. Normalizer of the approximate PEP cavity.

Monte Carlo Approximation

The required expectation is approximated as:

$$\frac{1}{\alpha} \log \mathbb{E}_q \left[\left(\frac{f_n(\boldsymbol{\theta})}{\tilde{f}(\boldsymbol{\theta})} \right)^\alpha \right] \approx \frac{1}{\alpha} \log \left(\frac{1}{S} \sum_{s=1}^{S} p(y_i | f_{i,s}^L) \right) - \frac{g_q}{\alpha} + \frac{g_{q_{\text{cav}}^\alpha}}{\alpha}$$

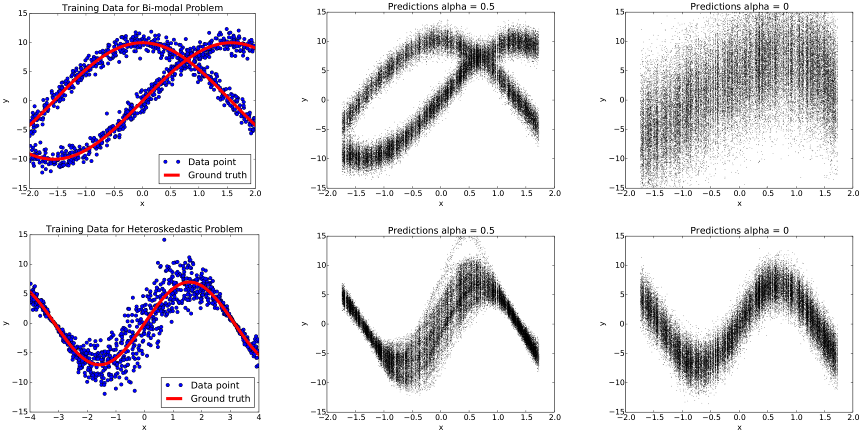
$g_q \equiv$ Log. Normalizer of q.

$g_{q_{\text{cav}}^\alpha} \equiv$ Log. Normalizer of the approximate PEP cavity.

This is a biased estimate, but the bias goes to zero as the number of samples S increases.

Expected Benefits of α-divergence Minimization

Similar to those of Bayesian neural networks...



(Depeweg et al., 2016)

Conclusions and Future Work

- Deep GP are flexible models for machine learning.

Conclusions and Future Work

- Deep GP are flexible models for machine learning.
- Can alleviate some of the limitations of standard GPs.

Conclusions and Future Work

- Deep GP are flexible models for machine learning.
- Can alleviate some of the limitations of standard GPs.
- Several ways of training them, including VI or AEP.

Conclusions and Future Work

- Deep GP are flexible models for machine learning.
- Can alleviate some of the limitations of standard GPs.
- Several ways of training them, including VI or AEP.
- DGPs can be trained by approximately minimizing α-divergences.

Conclusions and Future Work

- Deep GP are flexible models for machine learning.
- Can alleviate some of the limitations of standard GPs.
- Several ways of training them, including VI or AEP.
- DGPs can be trained by approximately minimizing α-divergences.
- α-divergence minimization may outperform VI or AEP methods.

Conclusions and Future Work

- Deep GP are flexible models for machine learning.
- Can alleviate some of the limitations of standard GPs.
- Several ways of training them, including VI or AEP.
- DGPs can be trained by approximately minimizing α-divergences.
- α-divergence minimization may outperform VI or AEP methods.

Future Work:

- Carry out experiments to assess the benefits of alpha divergence minimization for Deep GPs.

Thank you for your attention!

References I

- Bauer, M., van der Wilk, M., and Rasmussen, C. E. Understanding probabilistic sparse Gaussian process approximations. NIPS 29, pp. 1533-1541. 2016.
- Chai, K. M. A. Variational multinomial logit Gaussian process. JMLR, 13:1745-1808, 2012.
- Girolami, M. and Rogers, S. Variational Bayesian multinomial probit regression with Gaussian process priors. Neural Computation, 18:1790-1817, 2006.
- Hensman, J., Matthews, A. G., Filippone, M., and Ghahramani, Z. MCMC for variationally sparse Gaussian processes. NIPS 28, pp. 1648-1656. 2015.
- Hernández-Lobato, D. and Hernández-Lobato, J. M. Scalable Gaussian process classification via expectation propagation. AISTATS, pp. 168-176, 2016.
- Kim, H.-C. and Ghahramani, Z. Bayesian Gaussian process classification with the EM-EP algorithm. IEEE PAMI, 28, 1948-1959, 2006.
- Li, Y., Hernandez-Lobato, J. M., and Turner, R. E. Stochastic expectation propagation. NIPS 28, pp. 2323-2331. 2015.
- Naish-Guzman, A. and Holden, S. The generalized FITC approximation. NIPS 20, pp. 1057-1064. 2008.
- Riihimäki, J., Jylänki, P., and Vehtari, A. Nested expectation propagation for Gaussian process classification with a multinomial probit likelihood. JMLR, 14, 75-109, 2013.
- Snelson, E. and Ghahramani, Z. Sparse Gaussian processes using pseudo-inputs. NIPS 18, pp. 1257-1264, 2006.
- Williams, C. K. I. and Barber, D. Bayesian classification with Gaussian processes. IEEE PAMI, 20,1342-1351, 1998.

References II

- Damianou, A., and Lawrence, N. Deep gaussian processes. In Artificial Intelligence and Statistics (pp. 207-215), 2013.
- Bui, Thang, et al. Deep gaussian processes for regression using approximate expectation propagation. En International Conference on Machine Learning. 2016. p. 1472-1481.
- Salimbeni, H., and Deisenroth, M. (2017). Doubly stochastic variational inference for deep gaussian processes. In Advances in Neural Information Processing Systems (pp. 4588-4599).
- Hernandez-Lobato, J., Li, Y., Rowland, M., Bui, T., Hernandez-Lobato, D. and Turner, R.. (2016). Black-Box Alpha Divergence Minimization. Proceedings of The 33rd International Conference on Machine Learning, in PMLR 48:1511-1520
- Depeweg, S., Hernndez-Lobato, J. M., Doshi-Velez, F., and Udluft, S. (2016). Learning and policy search in stochastic dynamical systems with bayesian neural networks. arXiv preprint arXiv:1605.07127.
- T. Bui. Efficient Deterministic Approximate Bayesian Inference for Gaussian Process Models. PhD thesis, 2017.
- Duvenaud, D., Rippel, O., Adams, R., and Ghahramani, Z. (2014, April). Avoiding pathologies in very deep networks. In Artificial Intelligence and Statistics (pp. 202-210).

Specific Application of PEP to Multi-class GPC

The likelihood factors are the same as those of the VI approach:

$$p(y_i|\mathbf{f}_i) = (1-\epsilon)p_i + \frac{\epsilon}{C-1}(1-p_i) \quad \text{with} \quad p_i = \begin{cases} 1 & \text{if} \quad y_i = arg\,\max\limits_k \quad f^k(\mathbf{x}_i) \\ 0 & \text{otherwise} \end{cases}$$

The posterior approximation is:

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

At each step of PEP we have to update $\tilde{\phi}_i$ to minimize:

$$\text{KL}\left[p(y_i|\mathbf{f}_i)^\alpha p(\mathbf{f}|\bar{\mathbf{f}}) \frac{q(\bar{\mathbf{f}})}{\tilde{\phi}_i^\alpha} \,||\, p(\mathbf{f}|\bar{\mathbf{f}})q(\bar{\mathbf{f}}) \right]$$

Done by matching the moments of $\bar{\mathbf{f}}$! Requires quadratures!