Imperial College London

Distributed Gaussian Processes

Marc Deisenroth

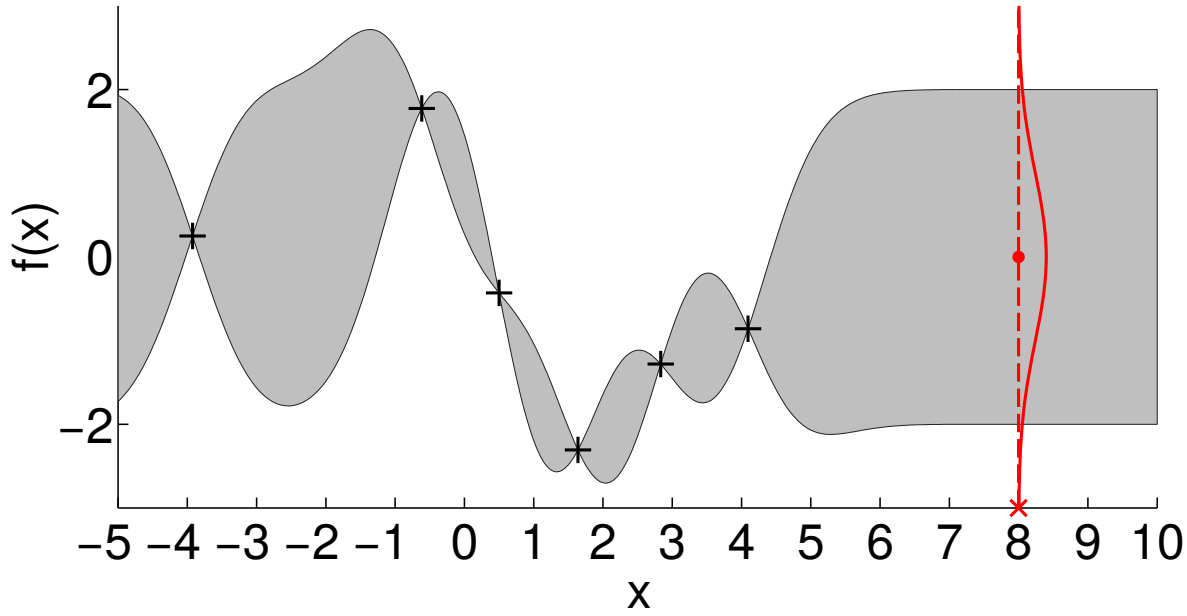
Department of Computing
Imperial College London

http://wp.doc.ic.ac.uk/sml/marc-deisenroth

Gaussian Process Summer School, University of Sheffield
15th September 2015

Table of Contents

Problem Setting



Objective

For a set of N observations $y_i = f(\boldsymbol{x}_i) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2)$, find a distribution over functions $p(f|\boldsymbol{X}, \boldsymbol{y})$ that explains the data

- GP is a good solution to this probabilistic regression problem

GP Training via Marginal Likelihood Maximization

GP Training

Maximize the evidence/marginal likelihood $p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta})$ with respect to the hyper-parameters $\boldsymbol{\theta}$: $\boldsymbol{\theta}^* \in \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta})$

$$\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) = -\frac{1}{2}\boldsymbol{y}^\top \boldsymbol{K}^{-1}\boldsymbol{y} - \frac{1}{2}\log|\boldsymbol{K}| + \text{const}$$

GP Training via Marginal Likelihood Maximization

GP Training

Maximize the evidence/marginal likelihood $p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta})$ with respect to the hyper-parameters $\boldsymbol{\theta}$: $\boldsymbol{\theta}^* \in \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta})$

$$\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) = -\frac{1}{2}\boldsymbol{y}^\top \boldsymbol{K}^{-1}\boldsymbol{y} - \frac{1}{2}\log|\boldsymbol{K}| + \text{const}$$

- Automatic trade-off between data fit and model complexity

GP Training via Marginal Likelihood Maximization

GP Training

Maximize the evidence/marginal likelihood $p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta})$ with respect to the hyper-parameters $\boldsymbol{\theta}$: $\boldsymbol{\theta}^* \in \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta})$

$$\log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta}) = -\tfrac{1}{2}\boldsymbol{y}^\top \boldsymbol{K}^{-1}\boldsymbol{y} - \tfrac{1}{2}\log|\boldsymbol{K}| + \text{const}$$

- Automatic trade-off between data fit and model complexity
- Gradient-based optimization possible:

$$\frac{\partial \log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \tfrac{1}{2}\boldsymbol{y}^\top \boldsymbol{K}^{-1}\frac{\partial \boldsymbol{K}}{\partial \boldsymbol{\theta}}\boldsymbol{K}^{-1}\boldsymbol{y} - \tfrac{1}{2}\mathrm{tr}\left(\boldsymbol{K}^{-1}\frac{\partial \boldsymbol{K}}{\partial \boldsymbol{\theta}}\right)$$

- Computational complexity: $\mathcal{O}(N^3)$ for $|\boldsymbol{K}|$ and $\boldsymbol{K}^{-1}$

GP Predictions

At a test point $\boldsymbol{x}_*$ the predictive (posterior) distribution is Gaussian:

$$
\begin{aligned}
p(f(\boldsymbol{x}_*)|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{\theta}) &= \mathcal{N}\left(f_* \,|\, m_*,\, \sigma_*^2\right) \\
m_* &= k(\boldsymbol{X}, \boldsymbol{x}_*)^\top \boldsymbol{K}^{-1} \boldsymbol{y} \\
\sigma_*^2 &= k(\boldsymbol{x}_*, \boldsymbol{x}_*) - k(\boldsymbol{X}, \boldsymbol{x}_*)^\top \boldsymbol{K}^{-1} k(\boldsymbol{X}, \boldsymbol{x}_*)
\end{aligned}
$$

GP Predictions

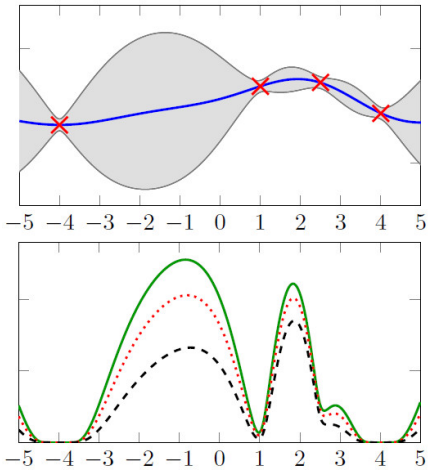
At a test point $\boldsymbol{x}_*$ the predictive (posterior) distribution is Gaussian:

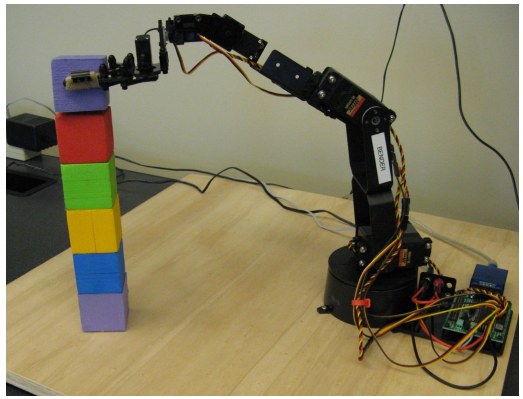
$$
\begin{aligned}
p(f(\boldsymbol{x}_*)|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{\theta}) &= \mathcal{N}\left(f_* \,|\, m_*,\, \sigma_*^2\right) \\
m_* &= k(\boldsymbol{X}, \boldsymbol{x}_*)^\top \boldsymbol{K}^{-1} \boldsymbol{y} \\
\sigma_*^2 &= k(\boldsymbol{x}_*, \boldsymbol{x}_*) - k(\boldsymbol{X}, \boldsymbol{x}_*)^\top \boldsymbol{K}^{-1} k(\boldsymbol{X}, \boldsymbol{x}_*)
\end{aligned}
$$

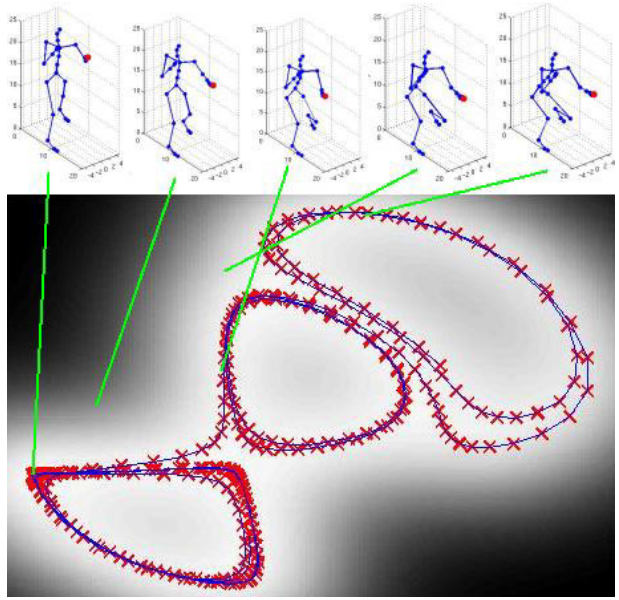
When you cache $\boldsymbol{K}^{-1}$ and $\boldsymbol{K}^{-1}\boldsymbol{y}$ after training, then

- The mean prediction can be computed in $\mathcal{O}(N)$
- The variance prediction can be computed in $\mathcal{O}(N^2)$

Application Areas







- Bayesian Optimization (Experimental Design)
  - Model unknown utility functions with GPs
- Reinforcement Learning and Robotics
  - Model value functions and/or dynamics with GPs
- Data visualization
  - Nonlinear dimensionality reduction (GP-LVM)

Limitations of Gaussian Processes

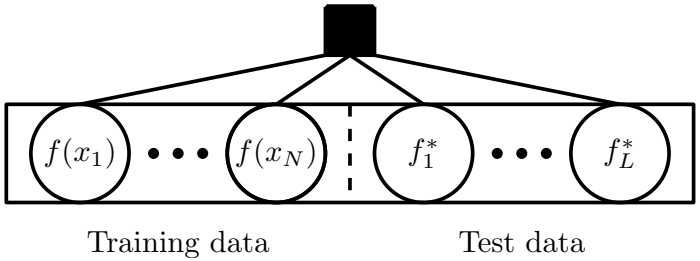
Computational and memory complexity

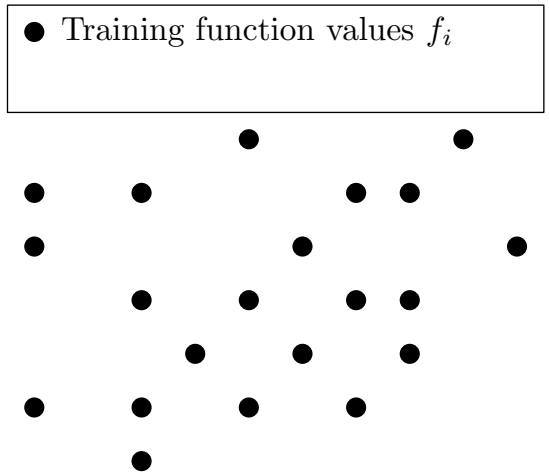
- Training scales in $\mathcal{O}(N^3)$
- Prediction (variances) scales in $\mathcal{O}(N^2)$
- Memory requirement: $\mathcal{O}(ND + N^2)$

⏩ **Practical limit** $N \approx 10,000$

Table of Contents

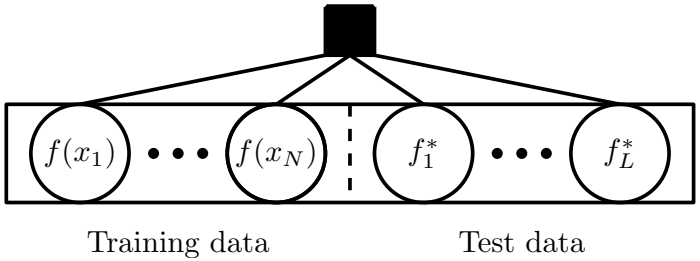
GP Factor Graph

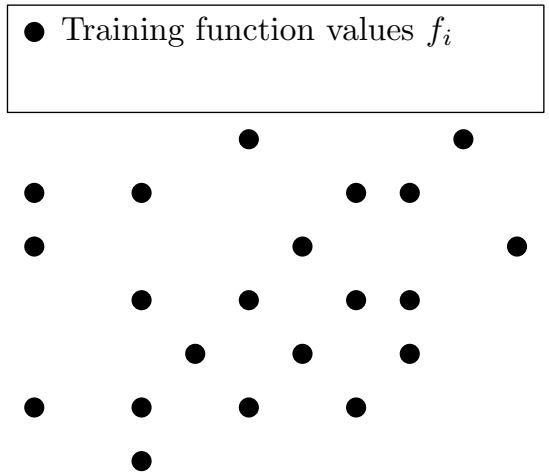




- Probabilistic graphical model (factor graph) of a GP
- All function values are jointly Gaussian distributed (e.g., training and test function values)

GP Factor Graph

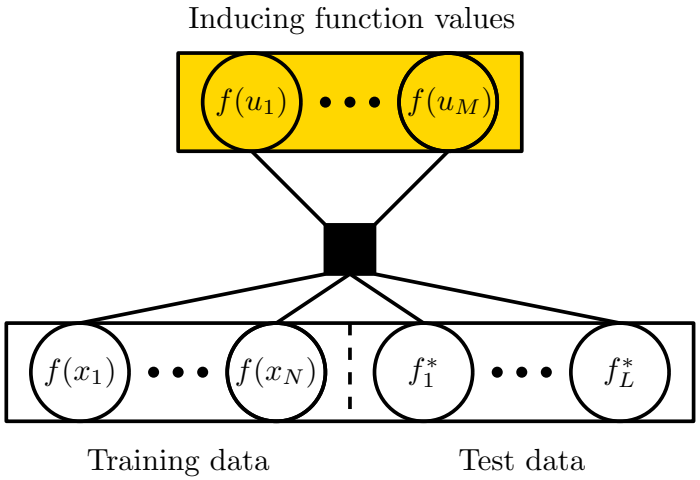


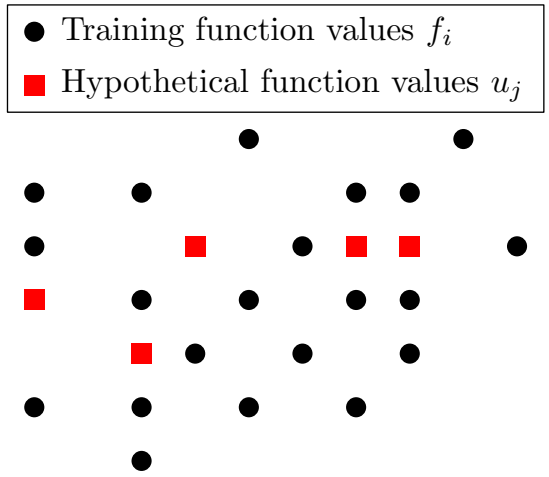


- Probabilistic graphical model (factor graph) of a GP
- All function values are jointly Gaussian distributed (e.g., training and test function values)
- GP prior

$$p(f, f_*) = \mathcal{N}\left(\begin{bmatrix}\mathbf{0}\\ \mathbf{0}\end{bmatrix}, \begin{bmatrix}\boldsymbol{K}_{ff} & \boldsymbol{K}_{f*}\\ \boldsymbol{K}_{*f} & \boldsymbol{K}_{**}\end{bmatrix}\right)$$

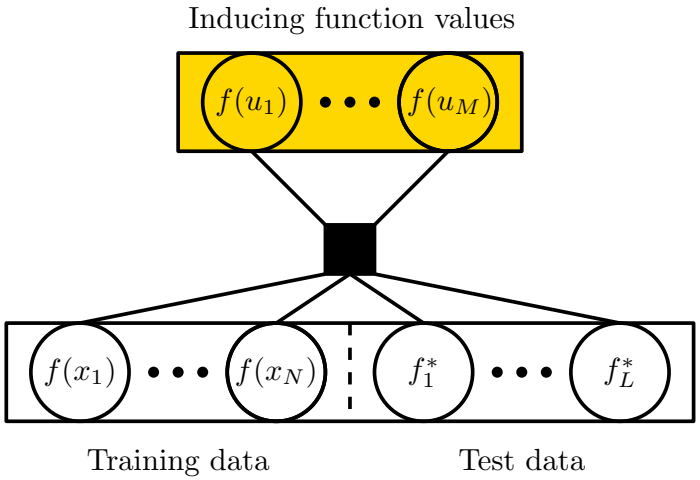
Inducing Variables

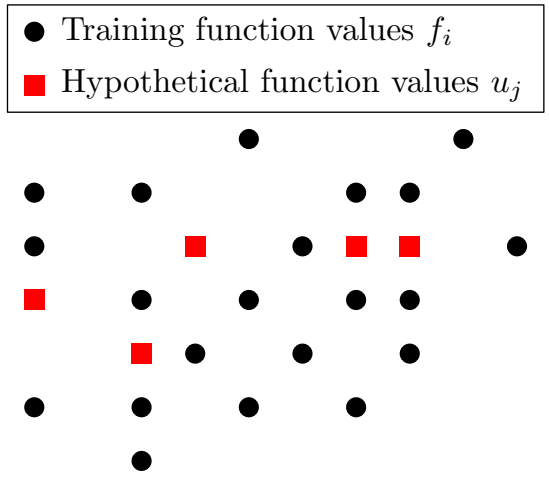




- Introduce inducing function values f_u
  - "Hypothetical" function values

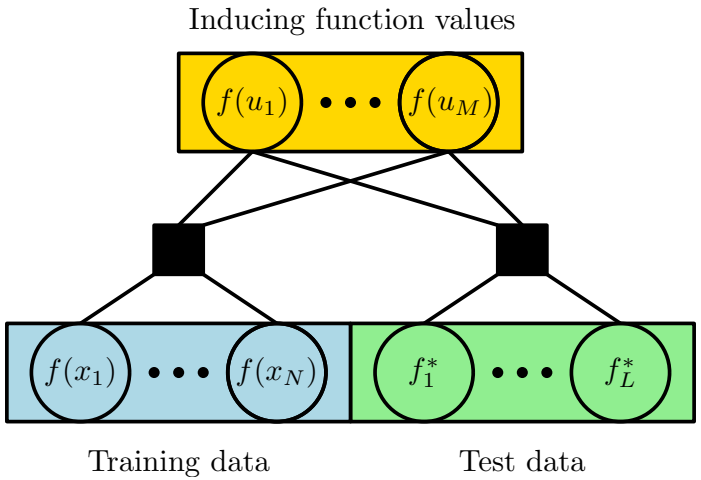
Inducing Variables





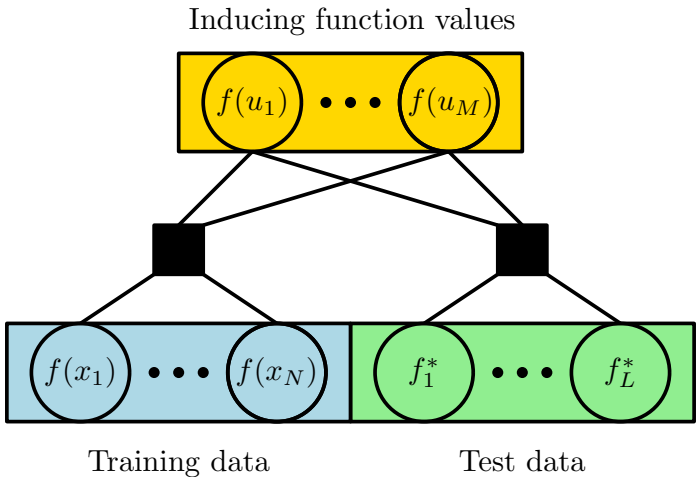
- Introduce inducing function values f_u
  - ⏩ "Hypothetical" function values
- All function values are still jointly Gaussian distributed (e.g., training, test and inducing function values)
- Approach: "Compress" real function values into inducing function values

Central Approximation Scheme



- Approximation: Training and test set are conditionally independent given the inducing function values: $f \perp\!\!\!\perp f_* | f_u$

Central Approximation Scheme

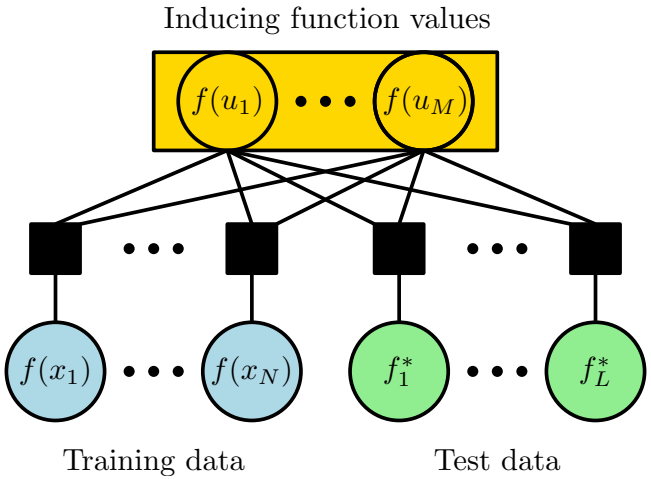


- Approximation: Training and test set are conditionally independent given the inducing function values: $f \perp\!\!\!\perp f_* | f_u$
- Then, the effective GP prior is

$$q(f, f_*) = \int p(f|f_u)p(f_*|f_u)p(f_u)df_u = \mathcal{N}\left(\begin{bmatrix}\mathbf{0}\\ \mathbf{0}\end{bmatrix}, \begin{bmatrix}\boldsymbol{K}_{ff} & \boldsymbol{Q}_{f*}\\ \boldsymbol{Q}_{*f} & \boldsymbol{K}_{**}\end{bmatrix}\right),$$

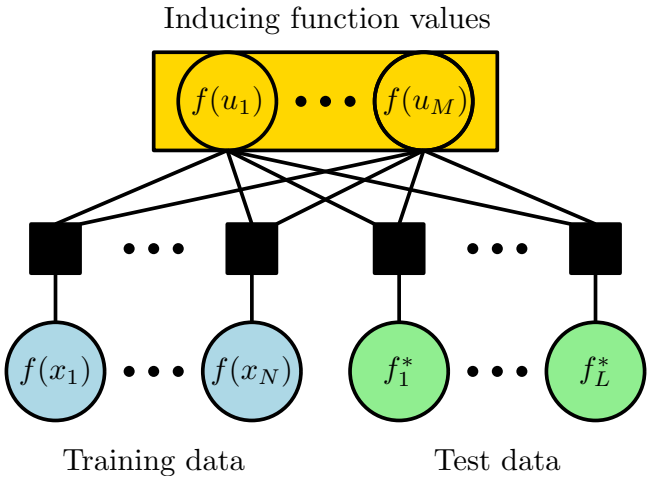
$\boldsymbol{Q}_{*f} := \boldsymbol{K}_{*f_u}\boldsymbol{K}_{f_u f_u}^{-1}\boldsymbol{K}_{f_u f}$ ⏩ Nyström approximation

FI(T)C Sparse Approximation



- Assume that training (and test sets) are fully independent given the inducing variables (Snelson & Ghahramani, 2006)

FI(T)C Sparse Approximation



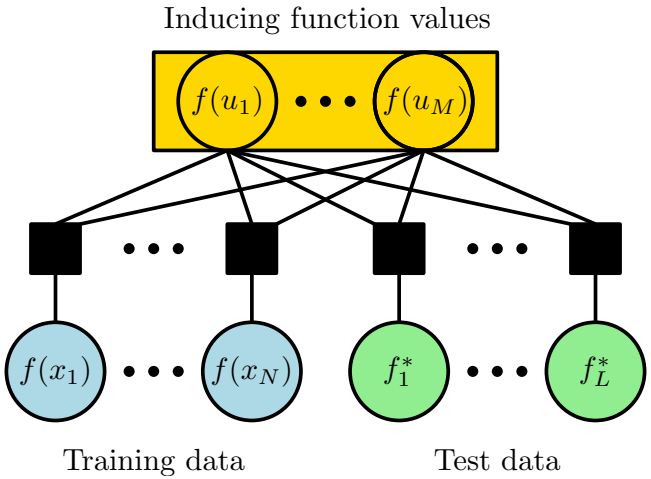
- Assume that training (and test sets) are fully independent given the inducing variables (Snelson & Ghahramani, 2006)

- Effective GP prior with this approximation

$$q(f, f_*) = \mathcal{N}\left(\begin{bmatrix}\mathbf{0}\\ \mathbf{0}\end{bmatrix}, \begin{bmatrix} \boldsymbol{Q}_{ff} - \mathrm{diag}(\boldsymbol{Q}_{ff} - \boldsymbol{K}_{ff}) & \boldsymbol{Q}_{f*} \\ \boldsymbol{Q}_{*f} & \boldsymbol{K}_{**} \end{bmatrix}\right)$$

- $\boldsymbol{Q}_{**} - \mathrm{diag}(\boldsymbol{Q}_{**} - \boldsymbol{K}_{**})$ can be used instead of $\boldsymbol{K}_{**}$ ⏩ FIC

FI(T)C Sparse Approximation



- Assume that training (and test sets) are fully independent given the inducing variables (Snelson & Ghahramani, 2006)

- Effective GP prior with this approximation

$$q(f, f_*) = \mathcal{N}\left(\begin{bmatrix}\mathbf{0}\\ \mathbf{0}\end{bmatrix}, \begin{bmatrix} \boldsymbol{Q}_{ff} - \mathrm{diag}(\boldsymbol{Q}_{ff} - \boldsymbol{K}_{ff}) & \boldsymbol{Q}_{f*} \\ \boldsymbol{Q}_{*f} & \boldsymbol{K}_{**} \end{bmatrix}\right)$$

- $\boldsymbol{Q}_{**} - \mathrm{diag}(\boldsymbol{Q}_{**} - \boldsymbol{K}_{**})$ can be used instead of $\boldsymbol{K}_{**}$ ⏩ FIC
- Training: $\mathcal{O}(NM^2)$, Prediction: $\mathcal{O}(M^2)$

Inducing Inputs

- FI(T)C sparse approximation exploits inducing function values $f(\boldsymbol{u}_i)$, where $\boldsymbol{u}_i$ are the corresponding inputs

Inducing Inputs

- FI(T)C sparse approximation exploits inducing function values $f(\boldsymbol{u}_i)$, where $\boldsymbol{u}_i$ are the corresponding inputs
- These **inputs are unknown** a priori ⏩ Find "optimal" ones

Inducing Inputs

- FI(T)C sparse approximation exploits inducing function values $f(\boldsymbol{u}_i)$, where $\boldsymbol{u}_i$ are the corresponding inputs
- These **inputs are unknown** a priori ⏩ Find "optimal" ones
- Find them by maximizing the FI(T)C marginal likelihood with respect to the inducing inputs (and the standard hyper-parameters):

$$\boldsymbol{u}_{1:M}^* \in \arg\max_{\boldsymbol{u}_{1:M}} q_{\text{FITC}}(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{u}_{1:M}, \boldsymbol{\theta})$$

Inducing Inputs

- FI(T)C sparse approximation exploits inducing function values $f(\boldsymbol{u}_i)$, where $\boldsymbol{u}_i$ are the corresponding inputs
- These **inputs are unknown** a priori ⏩ Find "optimal" ones
- Find them by maximizing the FI(T)C marginal likelihood with respect to the inducing inputs (and the standard hyper-parameters):

$$\boldsymbol{u}_{1:M}^* \in \arg\max_{\boldsymbol{u}_{1:M}} q_{\text{FITC}}(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{u}_{1:M}, \boldsymbol{\theta})$$

- Intuitively: The marginal likelihood is not only parameterized by the hyper-parameters $\boldsymbol{\theta}$, but also by the inducing inputs $\boldsymbol{u}_{1:M}$.

Inducing Inputs

- FI(T)C sparse approximation exploits inducing function values $f(\boldsymbol{u}_i)$, where $\boldsymbol{u}_i$ are the corresponding inputs
- These **inputs are unknown** a priori ⏩ Find "optimal" ones
- Find them by maximizing the FI(T)C marginal likelihood with respect to the inducing inputs (and the standard hyper-parameters):

$$\boldsymbol{u}_{1:M}^* \in \arg\max_{\boldsymbol{u}_{1:M}} q_{\text{FITC}}(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{u}_{1:M}, \boldsymbol{\theta})$$

- Intuitively: The marginal likelihood is not only parameterized by the hyper-parameters $\boldsymbol{\theta}$, but also by the inducing inputs $\boldsymbol{u}_{1:M}$.
- End up with a high-dimensional non-convex optimization problem with MD additional parameters

FITC Example

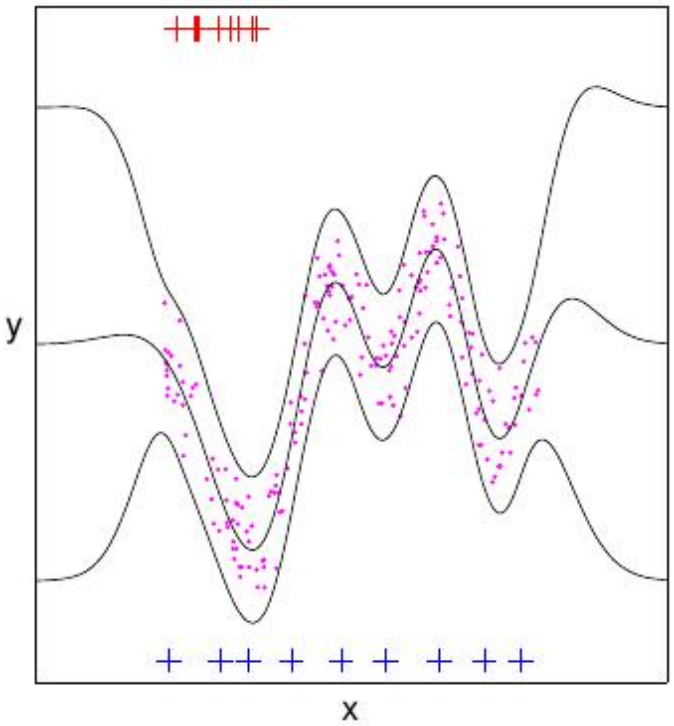


Figure from Ed Snelson

- Pink: Original data
- Red crosses: Initialization of inducing inputs
- Blue crosses: Location of inducing inputs after optimization

- Efficient compression of the original data set

Summary Sparse Gaussian Processes

- Sparse approximations typically approximate a GP with N data points by a model with $M \ll N$ data points

Summary Sparse Gaussian Processes

- Sparse approximations typically approximate a GP with N data points by a model with $M \ll N$ data points
- Selection of these M data points can be tricky and may involve non-trivial computations (e.g., optimizing inducing inputs)

Summary Sparse Gaussian Processes

- Sparse approximations typically approximate a GP with N data points by a model with $M \ll N$ data points
- Selection of these M data points can be tricky and may involve non-trivial computations (e.g., optimizing inducing inputs)
- Simple (random) subset selection is fast and generally robust (Chalupka et al., 2013)

Summary Sparse Gaussian Processes

- Sparse approximations typically approximate a GP with N data points by a model with $M \ll N$ data points
- Selection of these M data points can be tricky and may involve non-trivial computations (e.g., optimizing inducing inputs)
- Simple (random) subset selection is fast and generally robust (Chalupka et al., 2013)
- Computational complexity: $\mathcal{O}(M^3)$ or $O(NM^2)$ for training

Summary Sparse Gaussian Processes

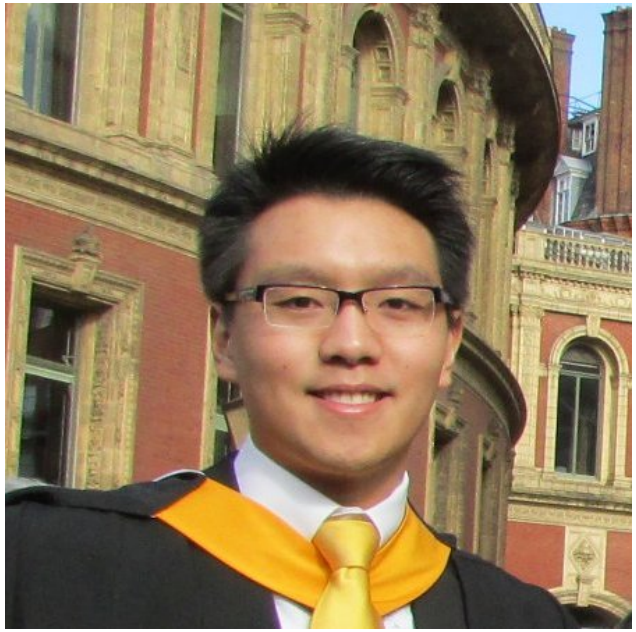
- Sparse approximations typically approximate a GP with N data points by a model with $M \ll N$ data points
- Selection of these M data points can be tricky and may involve non-trivial computations (e.g., optimizing inducing inputs)
- Simple (random) subset selection is fast and generally robust (Chalupka et al., 2013)
- Computational complexity: $\mathcal{O}(M^3)$ or $O(NM^2)$ for training
- Practical limit $M \leqslant 10^4$. Often: $M \in \mathcal{O}(10^2)$ in the case of inducing variables

Summary Sparse Gaussian Processes

- Sparse approximations typically approximate a GP with N data points by a model with $M \ll N$ data points
- Selection of these M data points can be tricky and may involve non-trivial computations (e.g., optimizing inducing inputs)
- Simple (random) subset selection is fast and generally robust (Chalupka et al., 2013)
- Computational complexity: $\mathcal{O}(M^3)$ or $O(NM^2)$ for training
- Practical limit $M \leqslant 10^4$. Often: $M \in \mathcal{O}(10^2)$ in the case of inducing variables
- If we set $M = N/100$, i.e., each inducing function value summarizes 100 real function values, our practical limit is $N \in \mathcal{O}(10^6)$

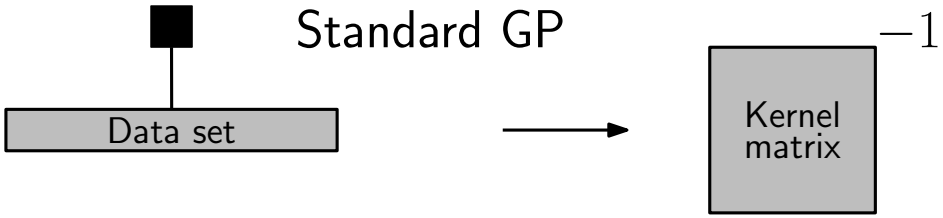
Table of Contents

Distributed Gaussian Processes



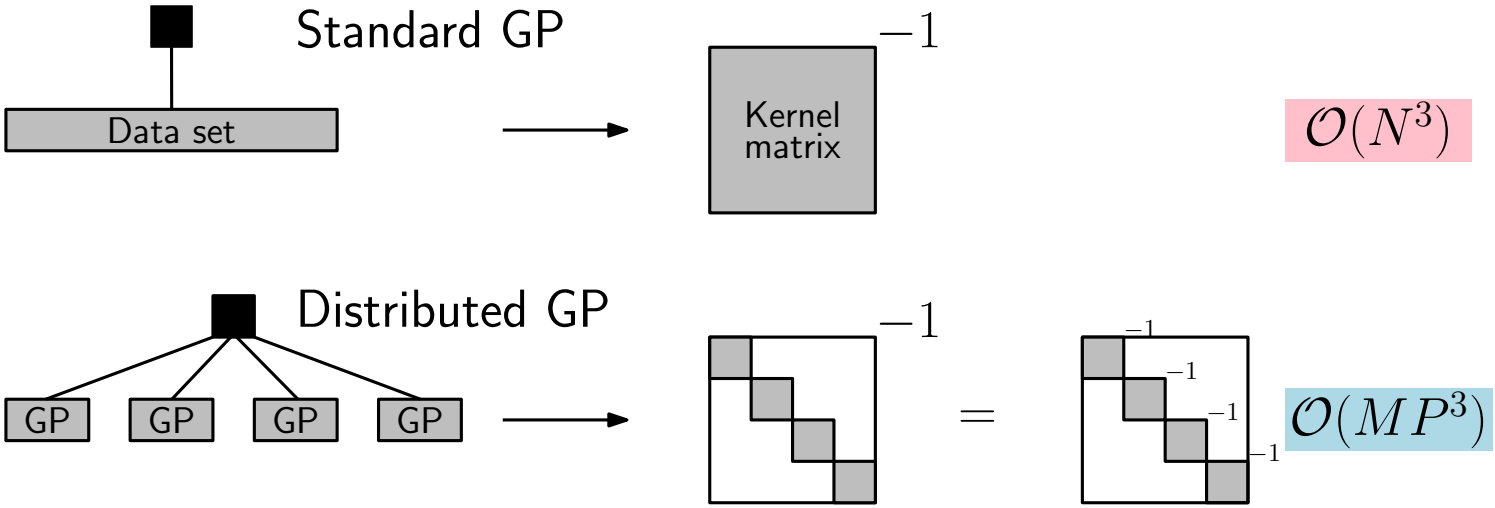
Joint work with Jun Wei Ng

An Orthogonal Approximation: Distributed GPs



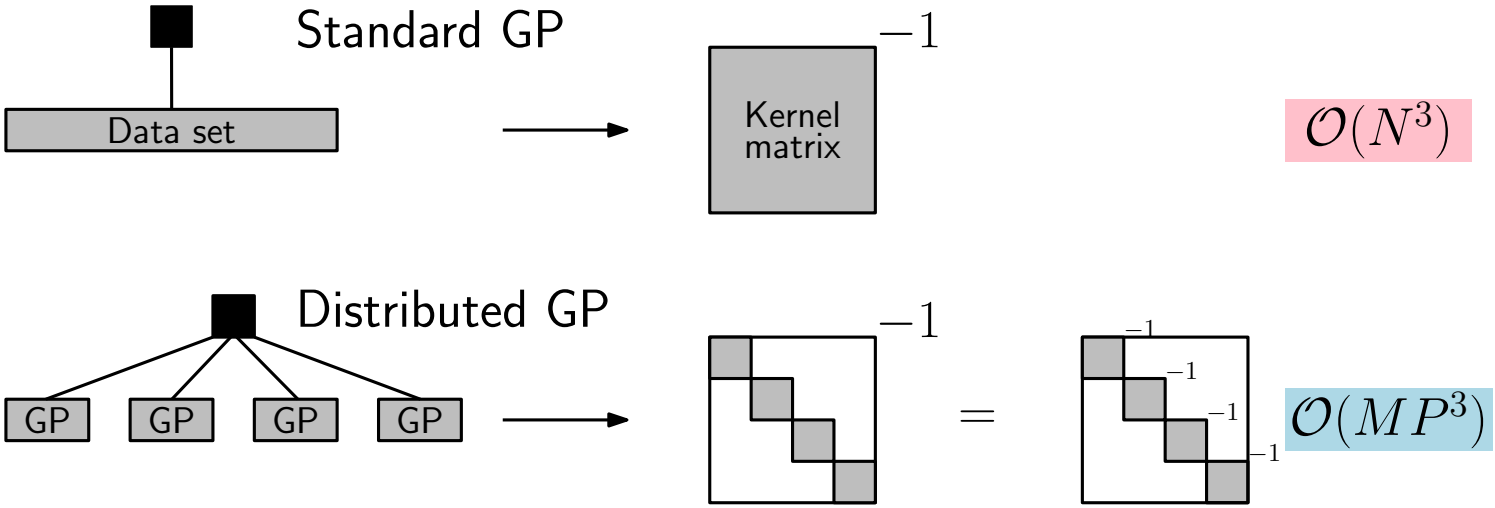
$\mathcal{O}(N^3)$

An Orthogonal Approximation: Distributed GPs



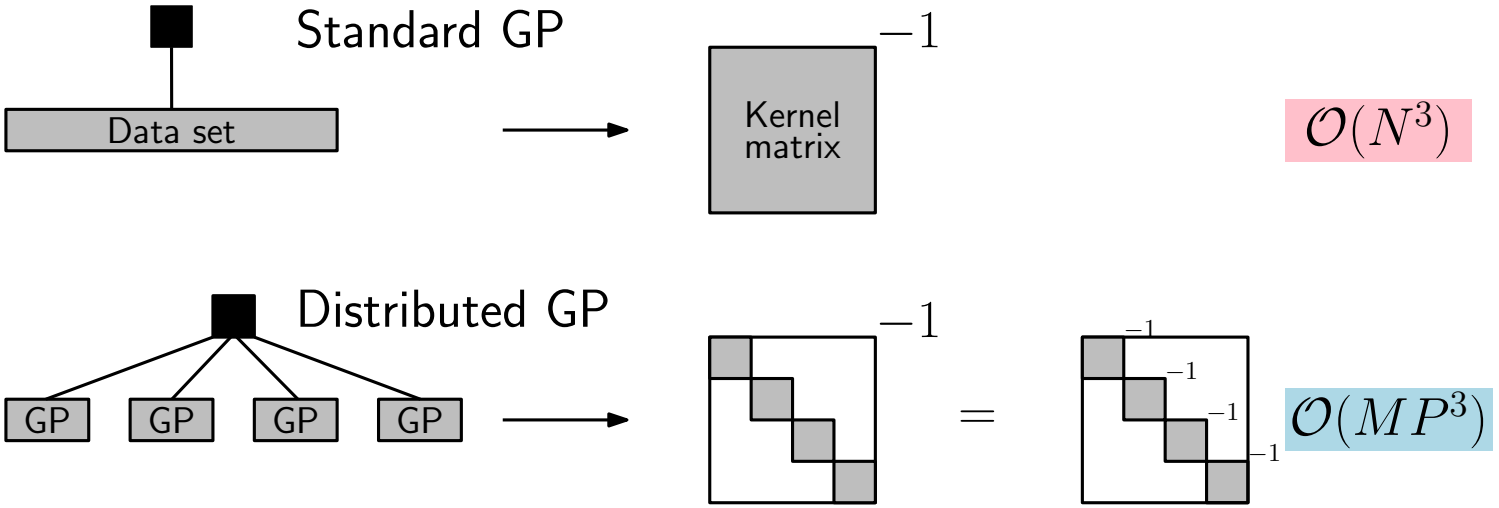
- Randomly split the full data set into M chunks

An Orthogonal Approximation: Distributed GPs



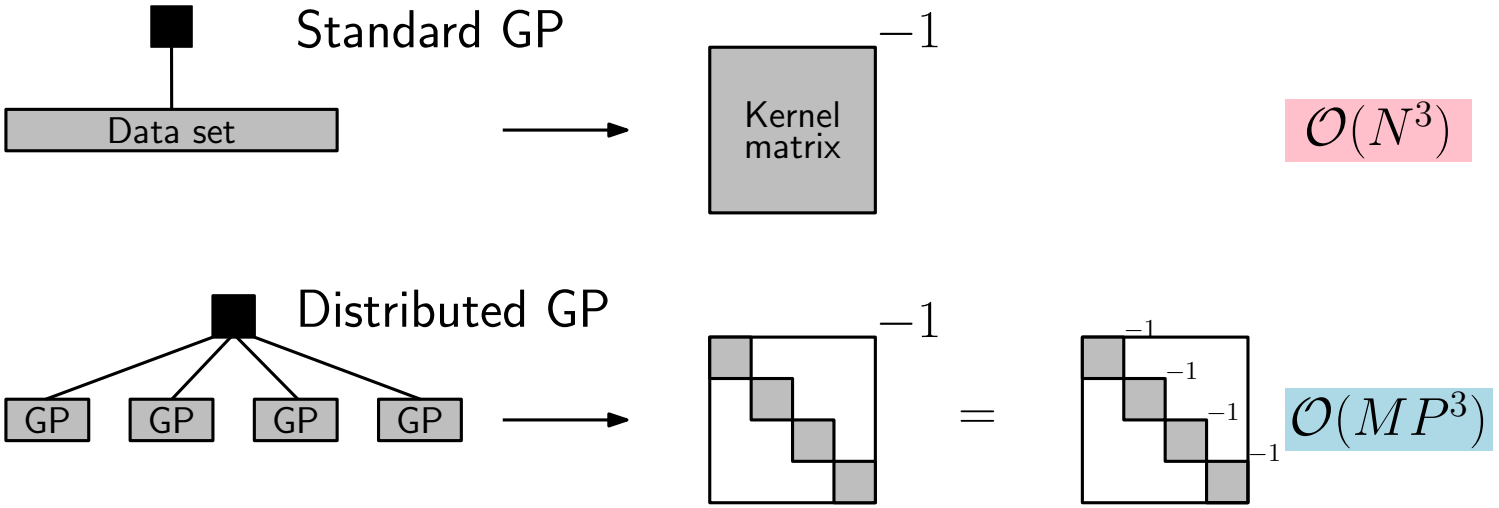
- Randomly split the full data set into M chunks
- Place M independent GP models (experts) on these small chunks

An Orthogonal Approximation: Distributed GPs



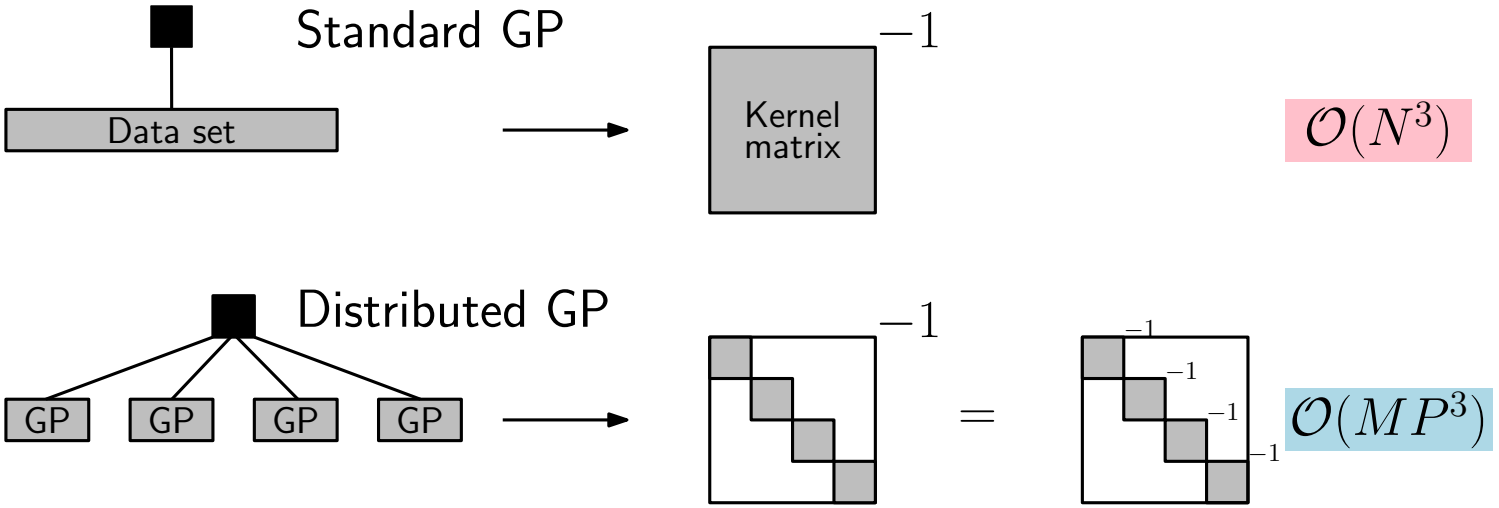
- Randomly split the full data set into M chunks
- Place M independent GP models (experts) on these small chunks
- Independent computations can be distributed

An Orthogonal Approximation: Distributed GPs



- Randomly split the full data set into M chunks
- Place M independent GP models (experts) on these small chunks
- Independent computations can be distributed
- Block-diagonal approximation of kernel matrix $\boldsymbol{K}$

An Orthogonal Approximation: Distributed GPs



- Randomly split the full data set into M chunks
- Place M independent GP models (experts) on these small chunks
- Independent computations can be distributed
- Block-diagonal approximation of kernel matrix $\boldsymbol{K}$
- Combine independent computations to an overall result

Training the Distributed GP

- Split data set of size N into M chunks of size P
- Independence of experts ⏩ Factorization of marginal likelihood:

$$\log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta}) \approx \sum_{k=1}^{M} \log p_k(\boldsymbol{y}^{(k)}|\boldsymbol{X}^{(k)},\boldsymbol{\theta})$$

Training the Distributed GP

- Split data set of size N into M chunks of size P
- Independence of experts ⏩ Factorization of marginal likelihood:

$$\log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta}) \approx \sum_{k=1}^{M} \log p_k(\boldsymbol{y}^{(k)}|\boldsymbol{X}^{(k)},\boldsymbol{\theta})$$

- Distributed optimization and training straightforward

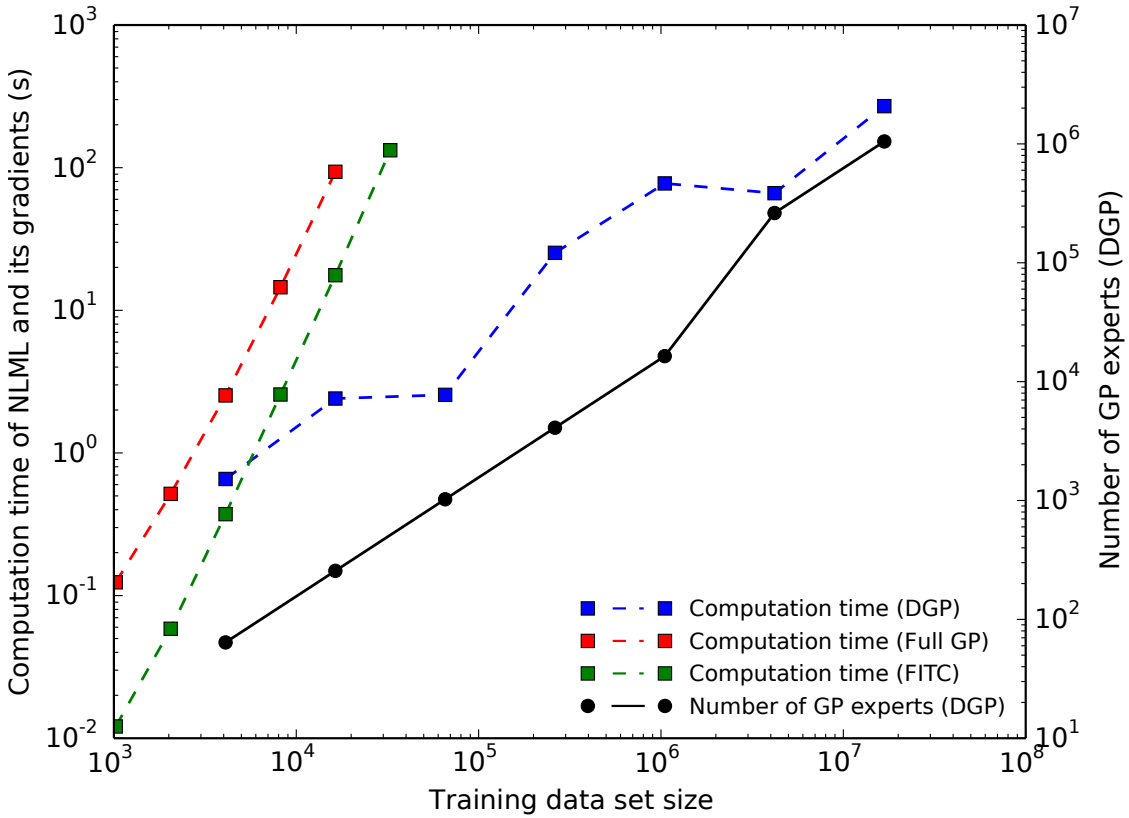
Training the Distributed GP

- Split data set of size N into M chunks of size P
- Independence of experts ⏩ Factorization of marginal likelihood:

$$\log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta}) \approx \sum_{k=1}^{M} \log p_k(\boldsymbol{y}^{(k)}|\boldsymbol{X}^{(k)},\boldsymbol{\theta})$$

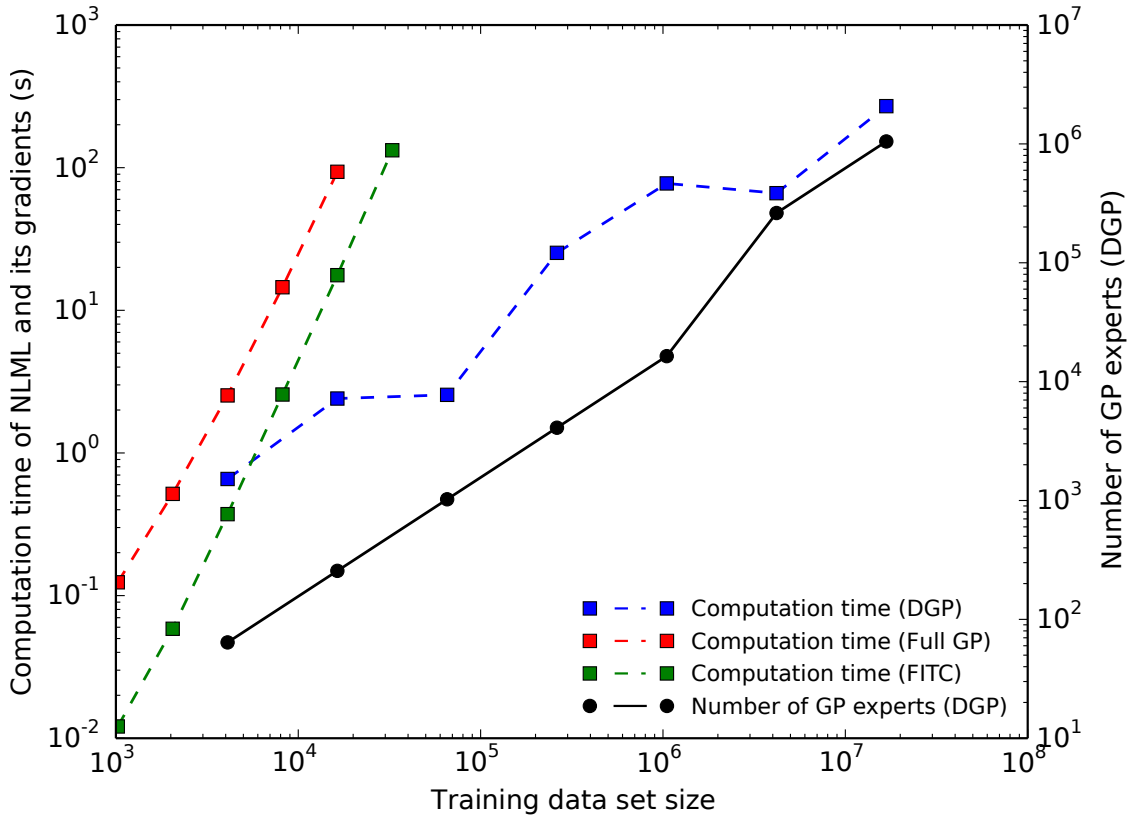
- Distributed optimization and training straightforward
- Computational complexity: $\mathcal{O}(MP^3)$ [instead of $\mathcal{O}(N^3)$]
  But distributed over many machines
- Memory footprint: $\mathcal{O}(MP^2 + ND)$ [instead of $\mathcal{O}(N^2 + ND)$]

Empirical Training Time



- NLML is proportional to training time

Empirical Training Time



- NLML is proportional to training time
- Full GP (16K training points) ≈ sparse GP (50K training points) ≈ distributed GP (16M training points)

⏩ Push practical limit by order(s) of magnitude

Practical Training Times

- Training* with $N = 10^6, D = 1$ on a laptop: $\approx$ 10–30 min
- Training* with $N = 5 \times 10^6, D = 8$ on a workstation: $\approx$ 4 hours

Practical Training Times

- Training* with $N = 10^6, D = 1$ on a laptop: $\approx$ 10–30 min
- Training* with $N = 5 \times 10^6, D = 8$ on a workstation: $\approx$ 4 hours

*: Maximize the marginal likelihood, stop when converged**

Practical Training Times

- Training* with $N = 10^6, D = 1$ on a laptop: $\approx$ 10–30 min
- Training* with $N = 5 \times 10^6, D = 8$ on a workstation: $\approx$ 4 hours

*: Maximize the marginal likelihood, stop when converged**

: Convergence often after 30–80 line searches*

Practical Training Times

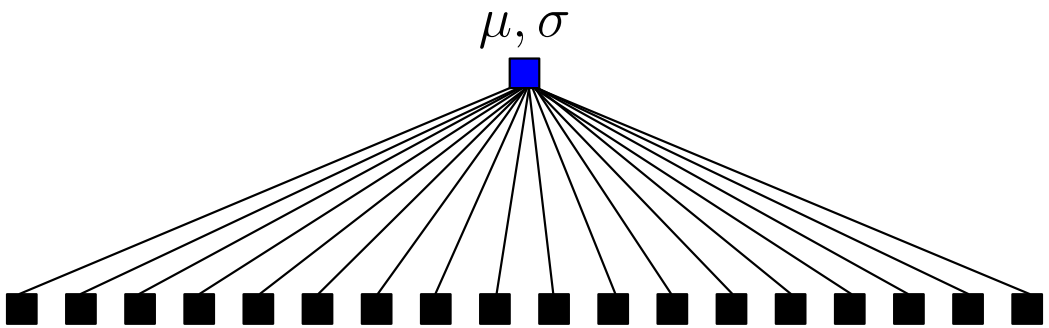
- Training* with $N = 10^6, D = 1$ on a laptop: $\approx$ 10–30 min
- Training* with $N = 5 \times 10^6, D = 8$ on a workstation: $\approx$ 4 hours

*: Maximize the marginal likelihood, stop when converged**

: Convergence often after 30–80 line searches*

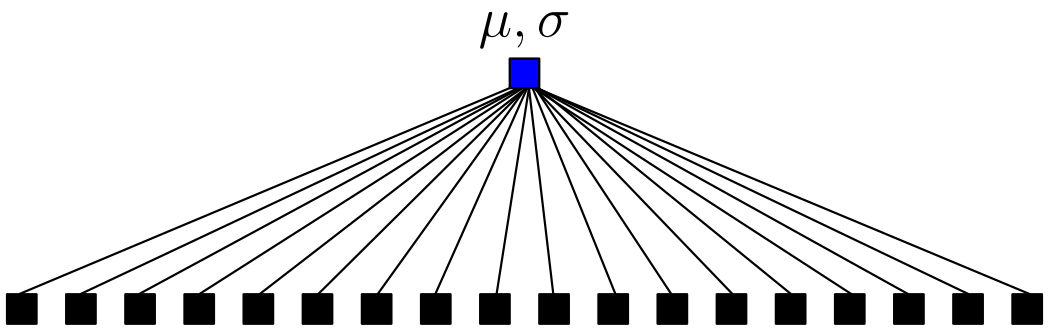
***: Line search $\approx$ 2–3 evaluations of marginal likelihood and its gradient (usually $\mathcal{O}(N^3)$)

Predictions with the Distributed GP



- Prediction of each GP expert is Gaussian $\mathcal{N}\left(\mu_i, \sigma_i^2\right)$
- How to combine them to an overall prediction $\mathcal{N}\left(\mu, \sigma^2\right)$?

Predictions with the Distributed GP



- Prediction of each GP expert is Gaussian $\mathcal{N}\left(\mu_i, \sigma_i^2\right)$
- How to combine them to an overall prediction $\mathcal{N}\left(\mu, \sigma^2\right)$?

⏩ Product-of-GP-experts

- PoE (product of experts; Ng & Deisenroth, 2014)
- gPoE (generalized product of experts; Cao & Fleet, 2014)
- BCM (Bayesian Committee Machine; Tresp, 2000)
- rBCM (robust BCM; Deisenroth & Ng, 2015)

Objectives

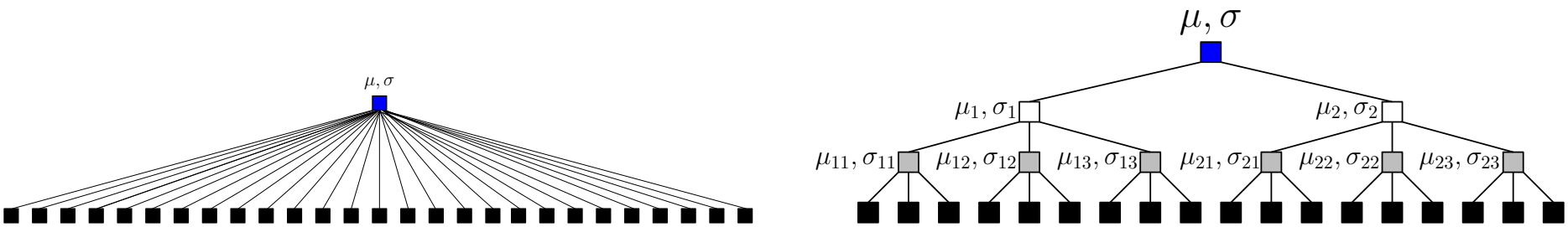


Figure: Two computational graphs

- Scale to large data sets ✓

Objectives

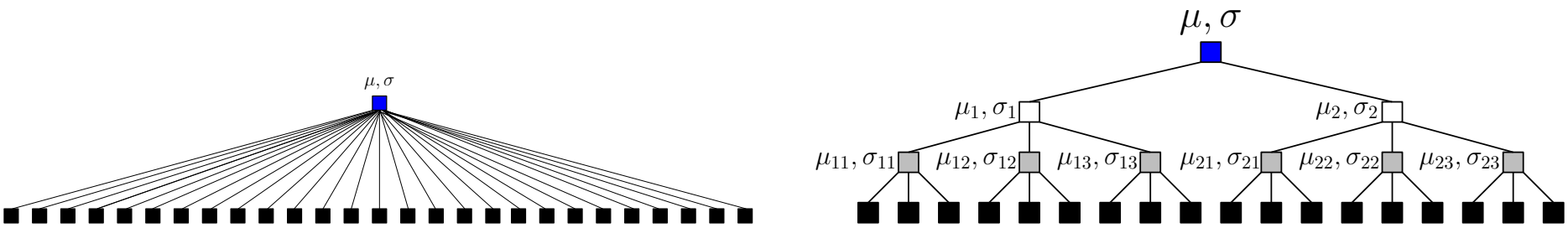


Figure: Two computational graphs

- Scale to large data sets ✓
- Good approximation of full GP ("ground truth")

Objectives

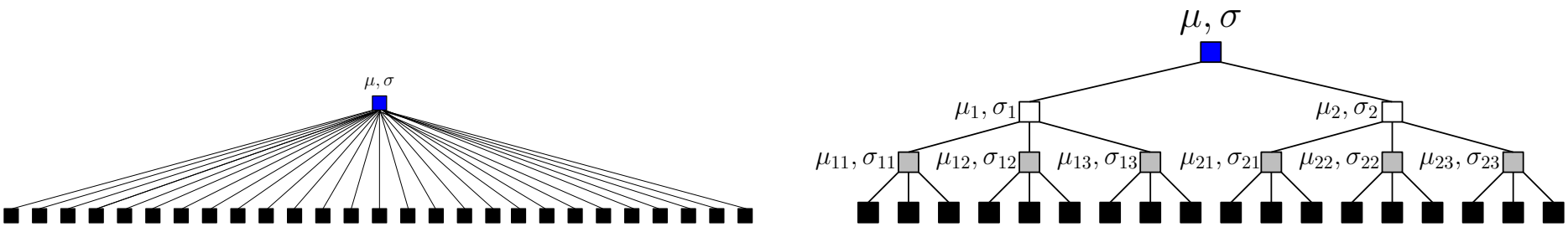


Figure: Two computational graphs

- Scale to large data sets ✓
- Good approximation of full GP ("ground truth")
- Predictions independent of computational graph

  ⏩ Runs on heterogeneous computing infrastructures (laptop, cluster, ...)

Objectives

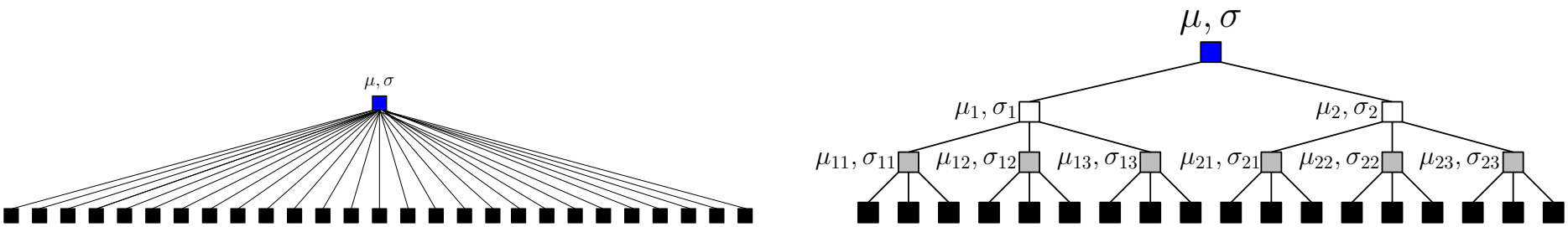
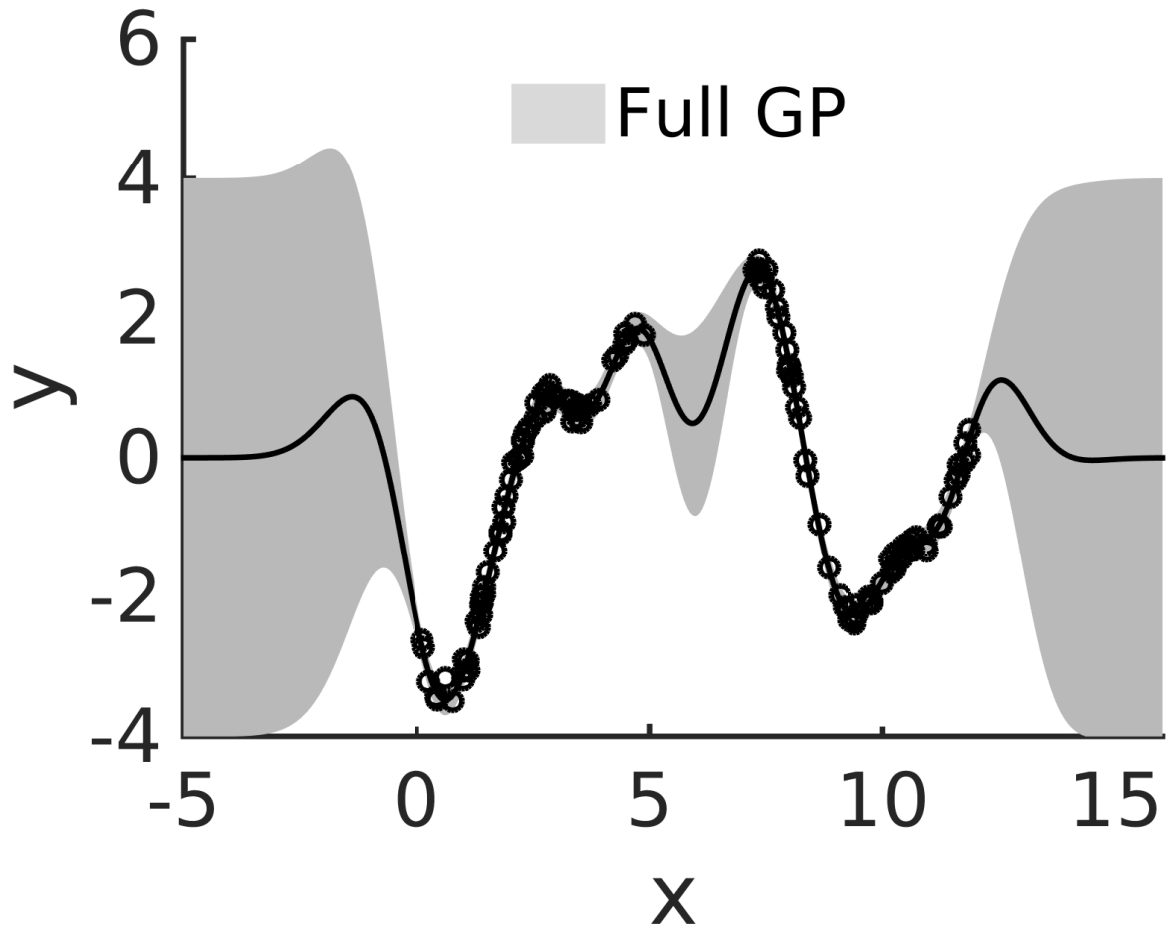


Figure: Two computational graphs

- Scale to large data sets ✓
- Good approximation of full GP ("ground truth")
- Predictions independent of computational graph

  ⏩ Runs on heterogeneous computing infrastructures (laptop, cluster, ...)
- Reasonable predictive variances

Running Example



- Investigate various product-of-experts models
  Same training procedure, but different mechanisms for predictions

Product of GP Experts

- Prediction model (independent predictors):

$$p(f_*|\boldsymbol{x}_*, \mathcal{D}) \propto \prod_{k=1}^{M} p_k(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)}),$$

$$p_k(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)}) = \mathcal{N}\left(f_* \,|\, \mu_k(\boldsymbol{x}_*),\, \sigma_k^2(\boldsymbol{x}_*)\right)$$

Product of GP Experts

- Prediction model (independent predictors):

$$p(f_*|\boldsymbol{x}_*, \mathcal{D}) \propto \prod_{k=1}^{M} p_k(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)}),$$

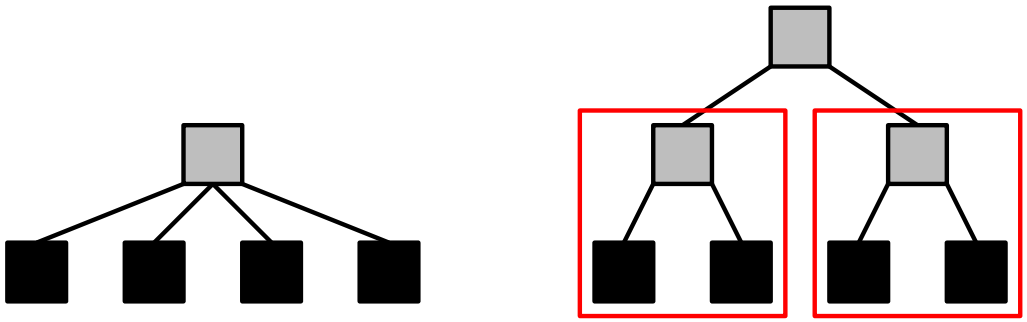
$$p_k(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)}) = \mathcal{N}\left(f_* \,|\, \mu_k(\boldsymbol{x}_*),\, \sigma_k^2(\boldsymbol{x}_*)\right)$$

- Predictive precision (inverse variance) and mean:

$$(\sigma_*^{\text{poe}})^{-2} = \sum\nolimits_k \sigma_k^{-2}(\boldsymbol{x}_*)$$

$$\mu_*^{\text{poe}} = (\sigma_*^{\text{poe}})^2 \sum\nolimits_k \sigma_k^{-2}(\boldsymbol{x}_*)\mu_k(\boldsymbol{x}_*)$$

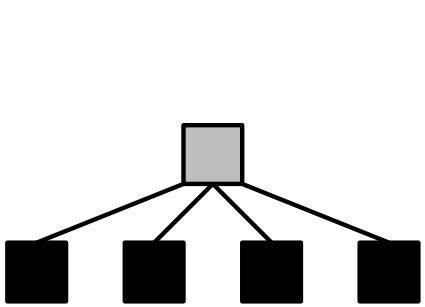
Computational Graph

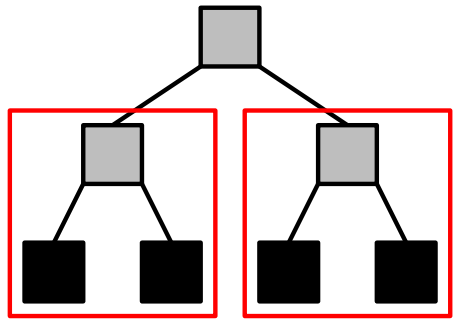


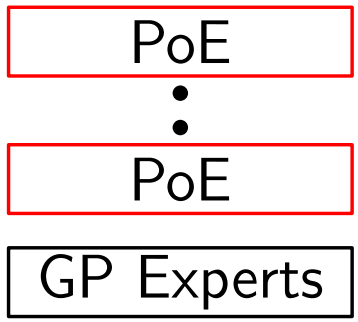
Prediction:

$$p(f_*|\boldsymbol{x}_*, \mathcal{D}) \propto \prod_{k=1}^{M} p_k(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)})$$

Computational Graph







Prediction:

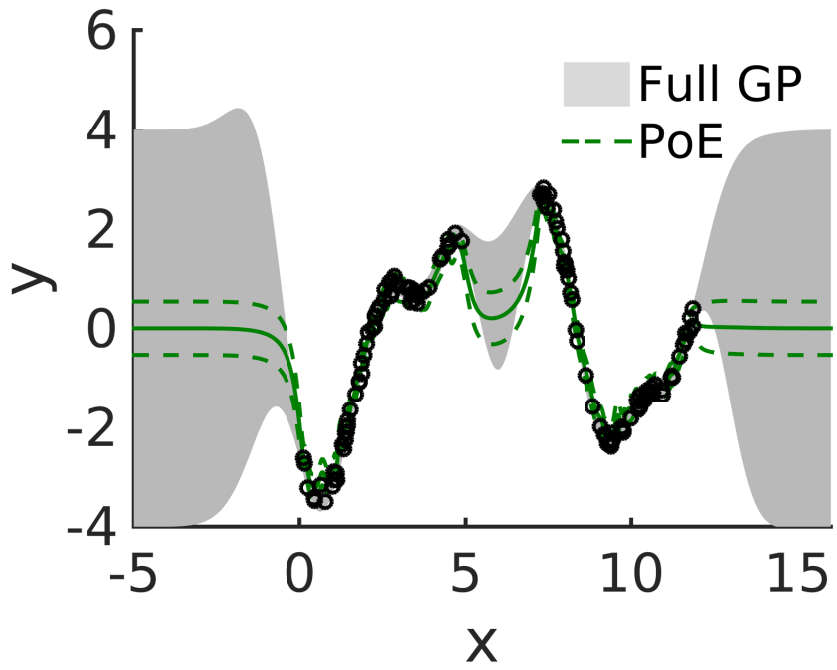
$$p(f_*|\boldsymbol{x}_*,\mathcal{D}) \propto \prod_{k=1}^{M} p_k(f_*|\boldsymbol{x}_*,\mathcal{D}^{(k)})$$

Multiplication is associative: $a * b * c * d = (a * b) * (c * d)$

$$\prod_{k=1}^{M} p_k(f_*|\mathcal{D}^{(k)}) = \prod_{k=1}^{L} \prod_{i=1}^{L_k} p_{k_i}(f_*|\mathcal{D}^{(k_i)}), \quad \sum_k L_k = M$$

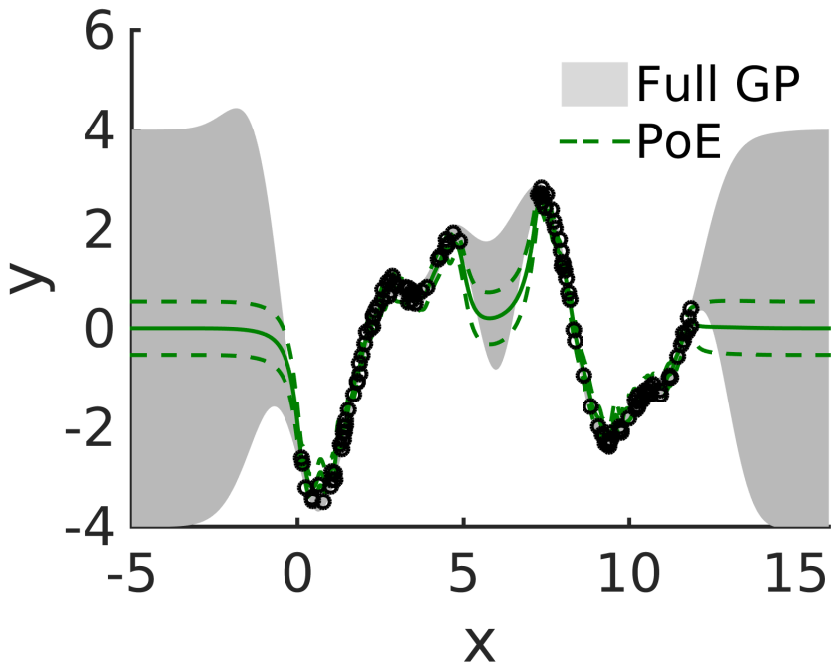
⏩ Independent of computational graph ✓

Product of GP Experts



- Unreasonable variances for $M > 1$:

Product of GP Experts



- Unreasonable variances for $M > 1$:

$$(\sigma_*^{\text{poe}})^{-2} = \sum_k \sigma_k^{-2}(\boldsymbol{x}_*)$$

- The more experts the more certain the prediction, even if every expert itself is very uncertain ✗ ⏩ Cannot fall back to the prior

Generalized Product of GP Experts

- Weight the responsiblity of each expert in PoE with β_k

Generalized Product of GP Experts

- Weight the responsiblity of each expert in PoE with β_k
- Prediction model (independent predictors):

$$p(f_*|\boldsymbol{x}_*, \mathcal{D}) \propto \prod_{k=1}^{M} p_k^{\beta_k}(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)})$$

$$p_k(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)}) = \mathcal{N}\left(f_* \,|\, \mu_k(\boldsymbol{x}_*),\, \sigma_k^2(\boldsymbol{x}_*)\right)$$

Generalized Product of GP Experts

- Weight the responsiblity of each expert in PoE with β_k
- Prediction model (independent predictors):

$$p(f_*|\boldsymbol{x}_*, \mathcal{D}) \propto \prod_{k=1}^{M} p_k^{\beta_k}(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)})$$

$$p_k(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)}) = \mathcal{N}\left(f_* \,|\, \mu_k(\boldsymbol{x}_*),\, \sigma_k^2(\boldsymbol{x}_*)\right)$$

- Predictive precision and mean:

$$(\sigma_*^{\text{gpoe}})^{-2} = \sum\nolimits_k \beta_k \sigma_k^{-2}(\boldsymbol{x}_*)$$

$$\mu_*^{\text{gpoe}} = (\sigma_*^{\text{gpoe}})^2 \sum\nolimits_k \beta_k \sigma_k^{-2}(\boldsymbol{x}_*)\, \mu_k(\boldsymbol{x}_*)$$

Generalized Product of GP Experts

- Weight the responsiblity of each expert in PoE with β_k
- Prediction model (independent predictors):

$$p(f_*|\boldsymbol{x}_*, \mathcal{D}) \propto \prod_{k=1}^{M} p_k^{\beta_k}(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)})$$

$$p_k(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)}) = \mathcal{N}\left(f_* \,|\, \mu_k(\boldsymbol{x}_*),\, \sigma_k^2(\boldsymbol{x}_*)\right)$$

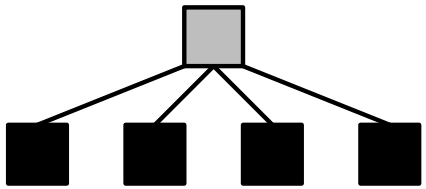
- Predictive precision and mean:

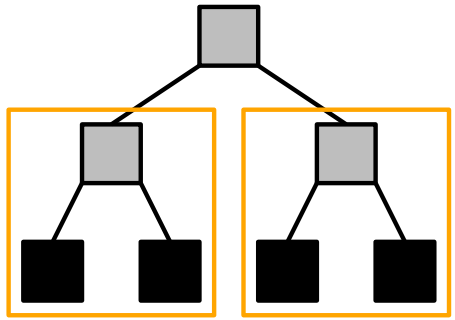
$$(\sigma_*^{\text{gpoe}})^{-2} = \sum\nolimits_k \beta_k \sigma_k^{-2}(\boldsymbol{x}_*)$$

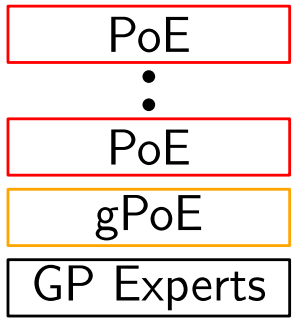
$$\mu_*^{\text{gpoe}} = (\sigma_*^{\text{gpoe}})^2 \sum\nolimits_k \beta_k \sigma_k^{-2}(\boldsymbol{x}_*)\, \mu_k(\boldsymbol{x}_*)$$

- With $\sum_k \beta_k = 1$, the model can fall back to the prior ✓
  "Log-opinion pool" model (Heskes, 1998)

Computational Graph



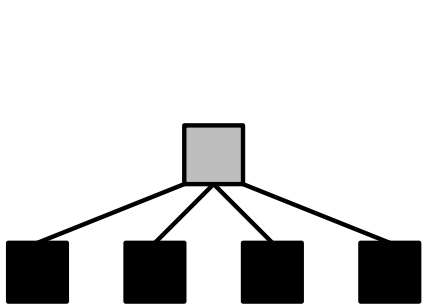


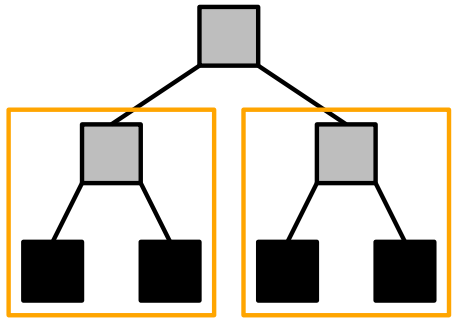


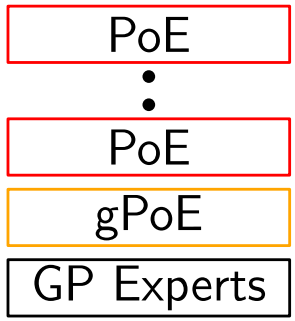
Prediction:

$$p(f_*|\boldsymbol{x}_*,\mathcal{D}) \propto \prod_{k=1}^{M} p_k^{\beta_k}(f_*|\boldsymbol{x}_*,\mathcal{D}^{(k)}) = \prod_{k=1}^{L}\prod_{i=1}^{L_k} p_{k_i}^{\beta_{k_i}}(f_*|\mathcal{D}^{(k_i)}), \quad \sum_{k,i}\beta_{k_i} = 1$$

Computational Graph





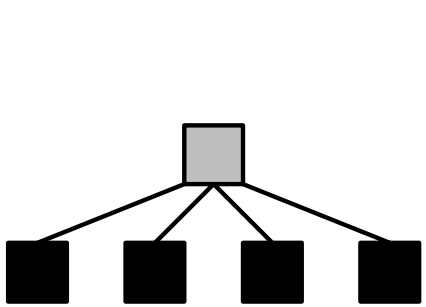


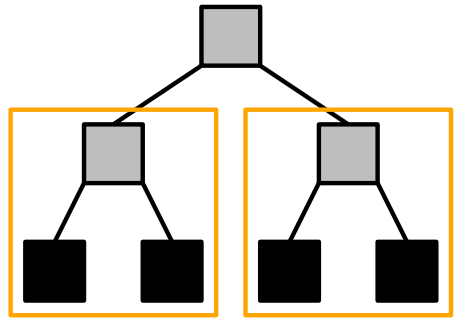
Prediction:

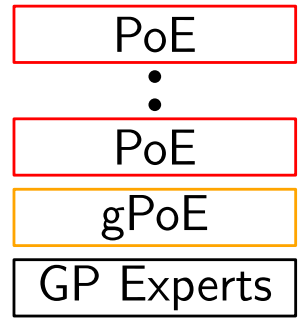
$$p(f_*|\boldsymbol{x}_*,\mathcal{D}) \propto \prod_{k=1}^{M} p_k^{\beta_k}(f_*|\boldsymbol{x}_*,\mathcal{D}^{(k)}) = \prod_{k=1}^{L}\prod_{i=1}^{L_k} p_{k_i}^{\beta_{k_i}}(f_*|\mathcal{D}^{(k_i)}), \quad \sum_{k,i}\beta_{k_i} = 1$$

- Independent of computational graph if $\sum_{k,i}\beta_{k_i} = 1$ ✓

Computational Graph





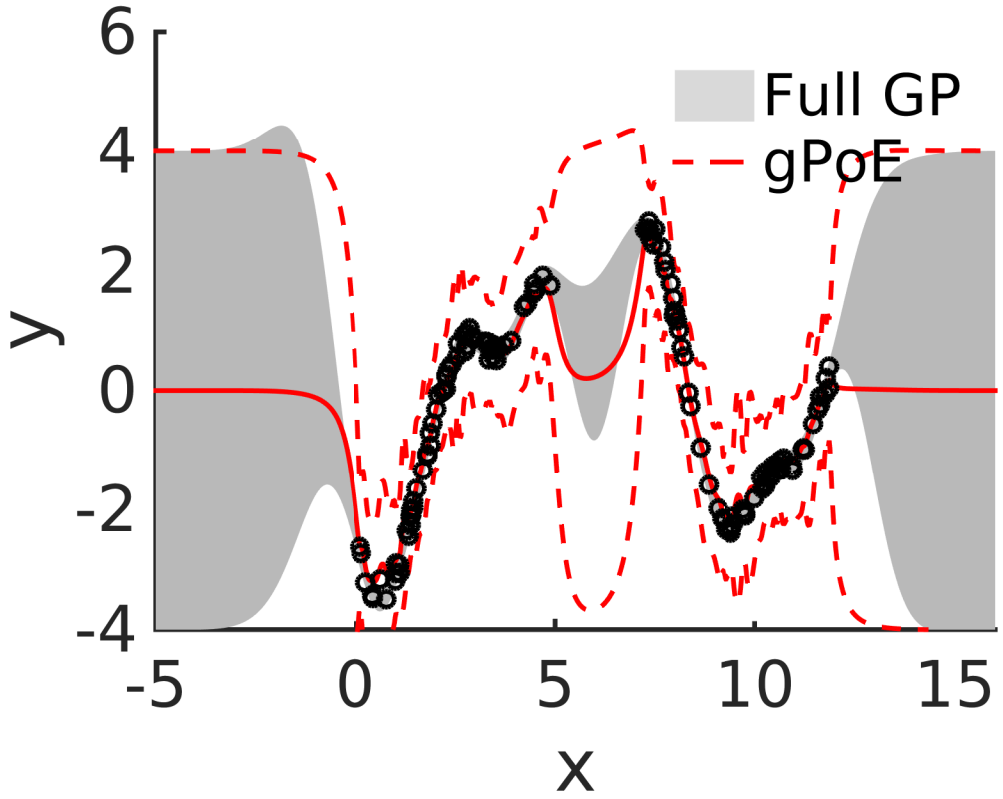


Prediction:

$$p(f_*|\boldsymbol{x}_*,\mathcal{D}) \propto \prod_{k=1}^{M} p_k^{\beta_k}(f_*|\boldsymbol{x}_*,\mathcal{D}^{(k)}) = \prod_{k=1}^{L}\prod_{i=1}^{L_k} p_{k_i}^{\beta_{k_i}}(f_*|\mathcal{D}^{(k_i)}), \quad \sum_{k,i}\beta_{k_i} = 1$$

- Independent of computational graph if $\sum_{k,i}\beta_{k_i} = 1$ ✓
- A priori setting of β_{k_i} required ✗
  - ⏩ $\beta_{k_i} = 1/M$ a priori (✓)

Generalized Product of GP Experts



- Same mean as PoE
- Model no longer overconfident and falls back to prior ✓
- Very conservative variances ✗

Bayesian Committee Machine

- Apply Bayes' theorem when combining predictions (and not only for computing predictions)

Bayesian Committee Machine

- Apply Bayes' theorem when combining predictions (and not only for computing predictions)
- Prediction model ($\mathcal{D}^{(j)} \perp\!\!\!\perp \mathcal{D}^{(k)} | f_*$):

$$p(f_*|\boldsymbol{x}_*, \mathcal{D}) \propto \frac{\prod_{k=1}^{M} p_k(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)})}{p^{M-1}(f_*)}$$

Bayesian Committee Machine

- Apply Bayes' theorem when combining predictions (and not only for computing predictions)
- Prediction model ($\mathcal{D}^{(j)} \perp\!\!\!\perp \mathcal{D}^{(k)} | f_*$):

$$p(f_*|\boldsymbol{x}_*, \mathcal{D}) \propto \frac{\prod_{k=1}^{M} p_k(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)})}{p^{M-1}(f_*)}$$

- Predictive precision and mean:

$$(\sigma_*^{\text{bcm}})^{-2} = \sum_{k=1}^{M} \sigma_k^{-2}(\boldsymbol{x}_*) - (M-1)\sigma_{**}^{-2}$$
$$\mu_*^{\text{bcm}} = (\sigma_*^{\text{bcm}})^2 \sum_{k=1}^{M} \sigma_k^{-2}(\boldsymbol{x}_*)\mu_k(\boldsymbol{x}_*)$$

Bayesian Committee Machine

- Apply Bayes' theorem when combining predictions (and not only for computing predictions)
- Prediction model ($\mathcal{D}^{(j)} \perp\!\!\!\perp \mathcal{D}^{(k)} | f_*$):

$$p(f_*|\boldsymbol{x}_*, \mathcal{D}) \propto \frac{\prod_{k=1}^{M} p_k(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)})}{p^{M-1}(f_*)}$$

- Predictive precision and mean:

$$(\sigma_*^{\text{bcm}})^{-2} = \sum_{k=1}^{M} \sigma_k^{-2}(\boldsymbol{x}_*) - (M-1)\sigma_{**}^{-2}$$
$$\mu_*^{\text{bcm}} = (\sigma_*^{\text{bcm}})^2 \sum_{k=1}^{M} \sigma_k^{-2}(\boldsymbol{x}_*)\mu_k(\boldsymbol{x}_*)$$

- Product of GP experts, divided by $M-1$ times the prior

Bayesian Committee Machine

- Apply Bayes' theorem when combining predictions (and not only for computing predictions)
- Prediction model ($\mathcal{D}^{(j)} \perp\!\!\!\perp \mathcal{D}^{(k)} | f_*$):

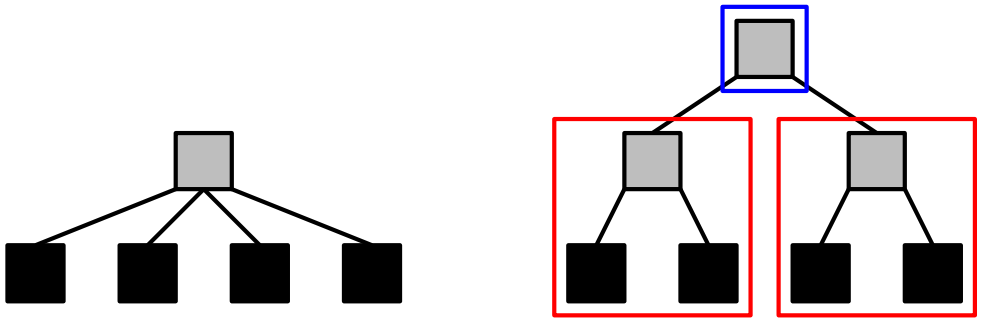
$$p(f_*|\boldsymbol{x}_*, \mathcal{D}) \propto \frac{\prod_{k=1}^{M} p_k(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)})}{p^{M-1}(f_*)}$$

- Predictive precision and mean:

$$(\sigma_*^{\text{bcm}})^{-2} = \sum_{k=1}^{M} \sigma_k^{-2}(\boldsymbol{x}_*) - (M-1)\sigma_{**}^{-2}$$
$$\mu_*^{\text{bcm}} = (\sigma_*^{\text{bcm}})^2 \sum_{k=1}^{M} \sigma_k^{-2}(\boldsymbol{x}_*)\mu_k(\boldsymbol{x}_*)$$

- Product of GP experts, divided by $M-1$ times the prior
- Guaranteed to fall back to the prior outside data regime ✓

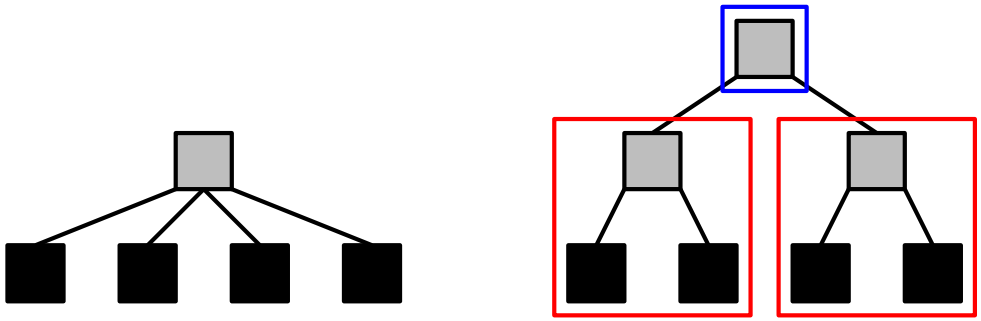
Computational Graph



Prediction:

$$p(f_*|\boldsymbol{x}_*,\mathcal{D}) \propto \frac{\prod_{k=1}^{M} p_k(f_*|\boldsymbol{x}_*,\mathcal{D}^{(k)})}{p^{M-1}(f_*)}$$

Computational Graph

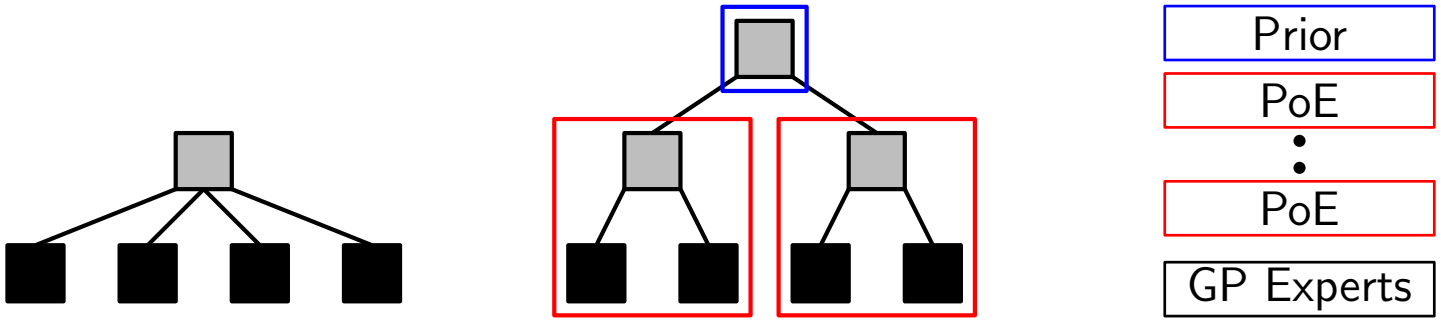


Prediction:

$$p(f_*|\boldsymbol{x}_*,\mathcal{D}) \propto \frac{\prod_{k=1}^{M} p_k(f_*|\boldsymbol{x}_*,\mathcal{D}^{(k)})}{p^{M-1}(f_*)}$$

$$\frac{\prod_{k=1}^{M} p_k(f_*|\mathcal{D}^{(k)})}{p^{M-1}(f_*)} = \frac{\prod_{k=1}^{L}\prod_{i=1}^{L_k} p_{k_i}(f_*|\mathcal{D}^{(k_i)})}{p^{M-1}(f_*)}$$

Computational Graph



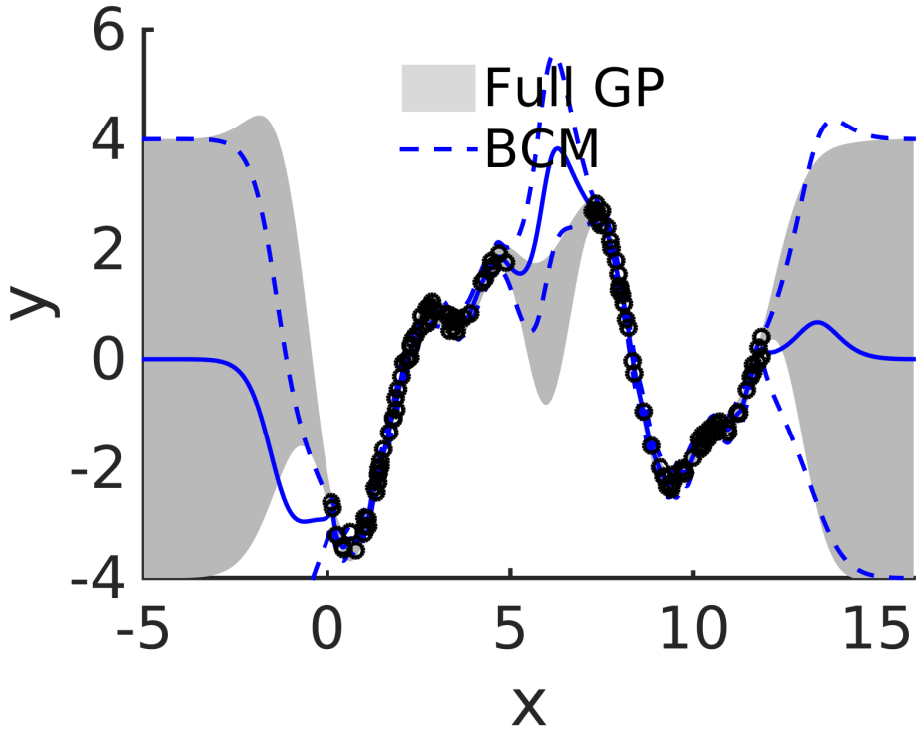
Prediction:

$$p(f_*|\boldsymbol{x}_*,\mathcal{D}) \propto \frac{\prod_{k=1}^{M} p_k(f_*|\boldsymbol{x}_*,\mathcal{D}^{(k)})}{p^{M-1}(f_*)}$$

$$\frac{\prod_{k=1}^{M} p_k(f_*|\mathcal{D}^{(k)})}{p^{M-1}(f_*)} = \frac{\prod_{k=1}^{L}\prod_{i=1}^{L_k} p_{k_i}(f_*|\mathcal{D}^{(k_i)})}{p^{M-1}(f_*)}$$

⏩ Independent of computational graph ✓

Bayesian Committee Machine



- Variance estimates are about right ✓
- When leaving the data regime, the BCM can produce junk ✗

  ⏩ **Robustify**

Robust Bayesian Committee Machine

- Merge gPoE (weighting of experts) with the BCM (Bayes' theorem when combining predictions)

Robust Bayesian Committee Machine

- Merge gPoE (weighting of experts) with the BCM (Bayes' theorem when combining predictions)
- Prediction model (conditional independence $\mathcal{D}^{(j)} \perp\!\!\!\perp \mathcal{D}^{(k)} | f_*$):

$$p(f_*|\boldsymbol{x}_*, \mathcal{D}) \propto \frac{\prod_{k=1}^{M} p_k^{\beta_k}(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)})}{p^{\sum_k \beta_k - 1}(f_*)}$$

Robust Bayesian Committee Machine

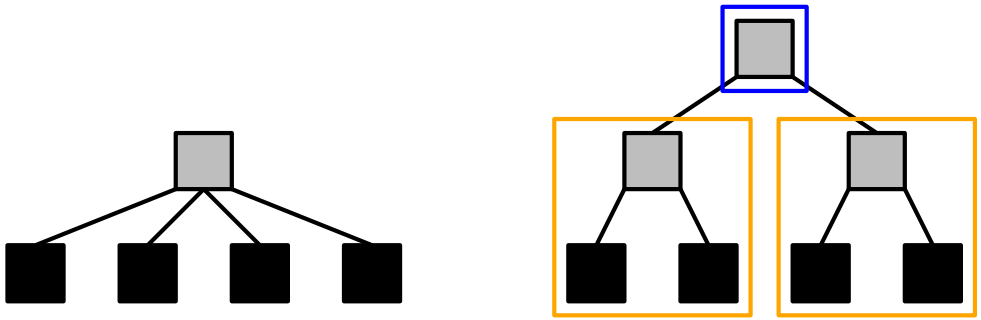
- Merge gPoE (weighting of experts) with the BCM (Bayes' theorem when combining predictions)
- Prediction model (conditional independence $\mathcal{D}^{(j)} \perp\!\!\!\perp \mathcal{D}^{(k)} | f_*$):

$$p(f_*|\boldsymbol{x}_*, \mathcal{D}) \propto \frac{\prod_{k=1}^M p_k^{\beta_k}(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)})}{p^{\sum_k \beta_k - 1}(f_*)}$$

- Predictive precision and mean:

$$(\sigma_*^{\text{rbcm}})^{-2} = \sum_{k=1}^M \beta_k \sigma_k^{-2}(\boldsymbol{x}_*) + (1 - \sum_{k=1}^M \beta_k)\sigma_{**}^{-2},$$
$$\mu_*^{\text{rbcm}} = (\sigma_*^{\text{rbcm}})^2 \sum_k \beta_k \sigma_k^{-2}(\boldsymbol{x}_*)\mu_k(\boldsymbol{x}_*)$$

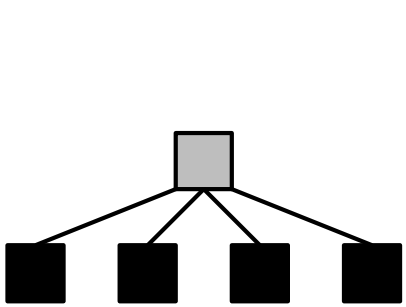
Computational Graph

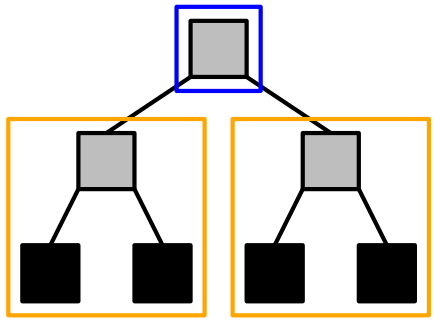


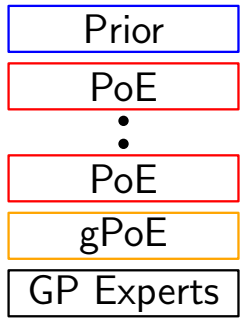
Prediction:

$$p(f_*|\boldsymbol{x}_*,\mathcal{D}) \propto \frac{\prod_{k=1}^{M} p_k^{\beta_k}(f_*|\boldsymbol{x}_*,\mathcal{D}^{(k)})}{p^{\sum_k \beta_k - 1}(f_*)} = \frac{\prod_{k=1}^{L}\prod_{i=1}^{L_k} p_{k_i}^{\beta_{k_i}}(f_*|\mathcal{D}^{(k_i)})}{p^{\sum_{k_i}\beta_{k_i}-1}(f_*)}$$

Computational Graph





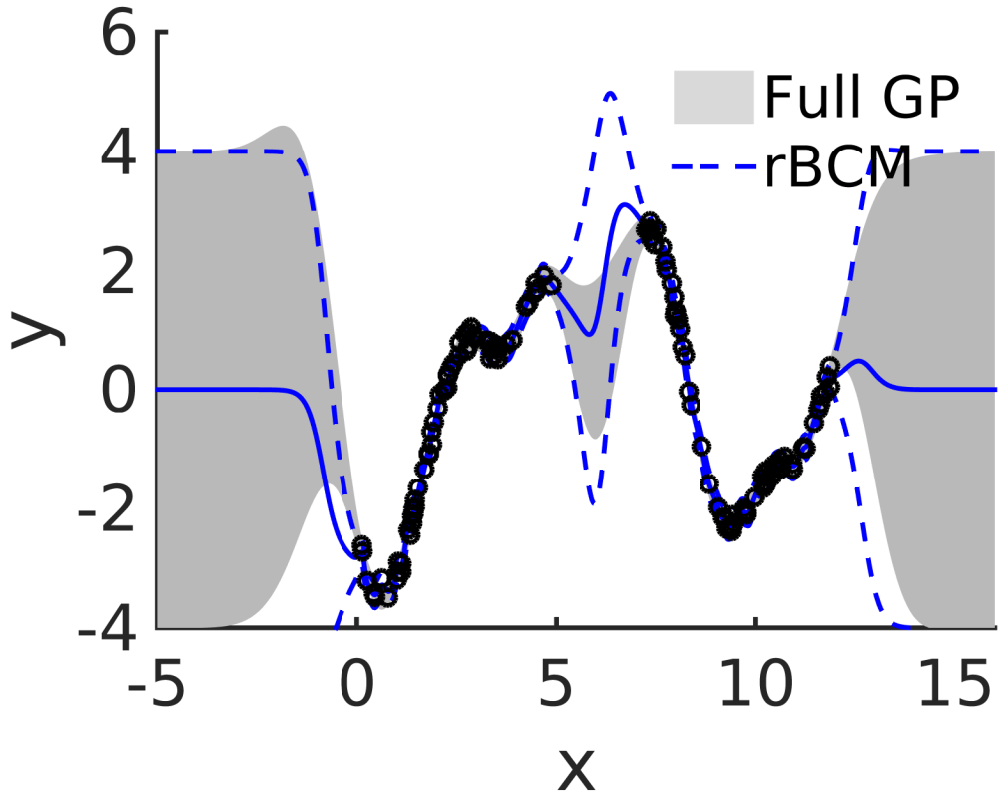


Prediction:

$$p(f_*|\boldsymbol{x}_*,\mathcal{D}) \propto \frac{\prod_{k=1}^{M} p_k^{\beta_k}(f_*|\boldsymbol{x}_*,\mathcal{D}^{(k)})}{p^{\sum_k \beta_k - 1}(f_*)} = \frac{\prod_{k=1}^{L}\prod_{i=1}^{L_k} p_{k_i}^{\beta_{k_i}}(f_*|\mathcal{D}^{(k_i)})}{p^{\sum_{k_i}\beta_{k_i}-1}(f_*)}$$

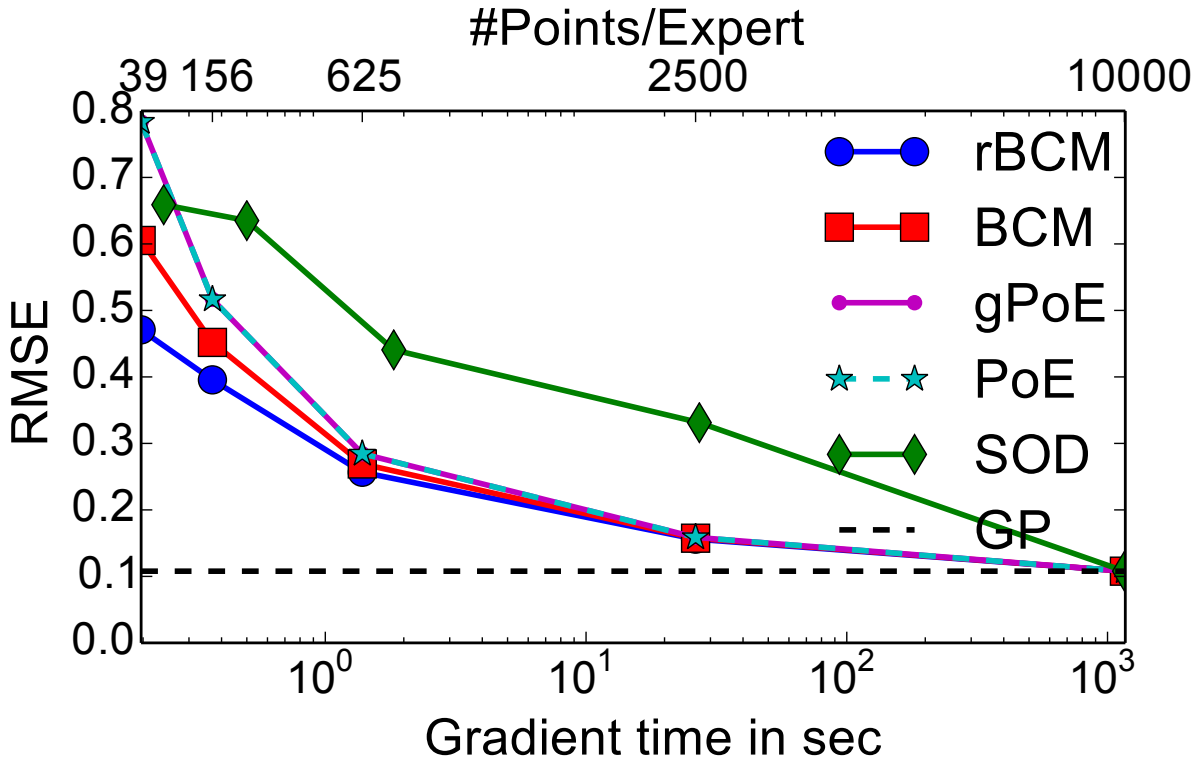
⏩ Independent of computational graph, even with arbitrary β_{k_i} ✓

Robust Bayesian Committee Machine



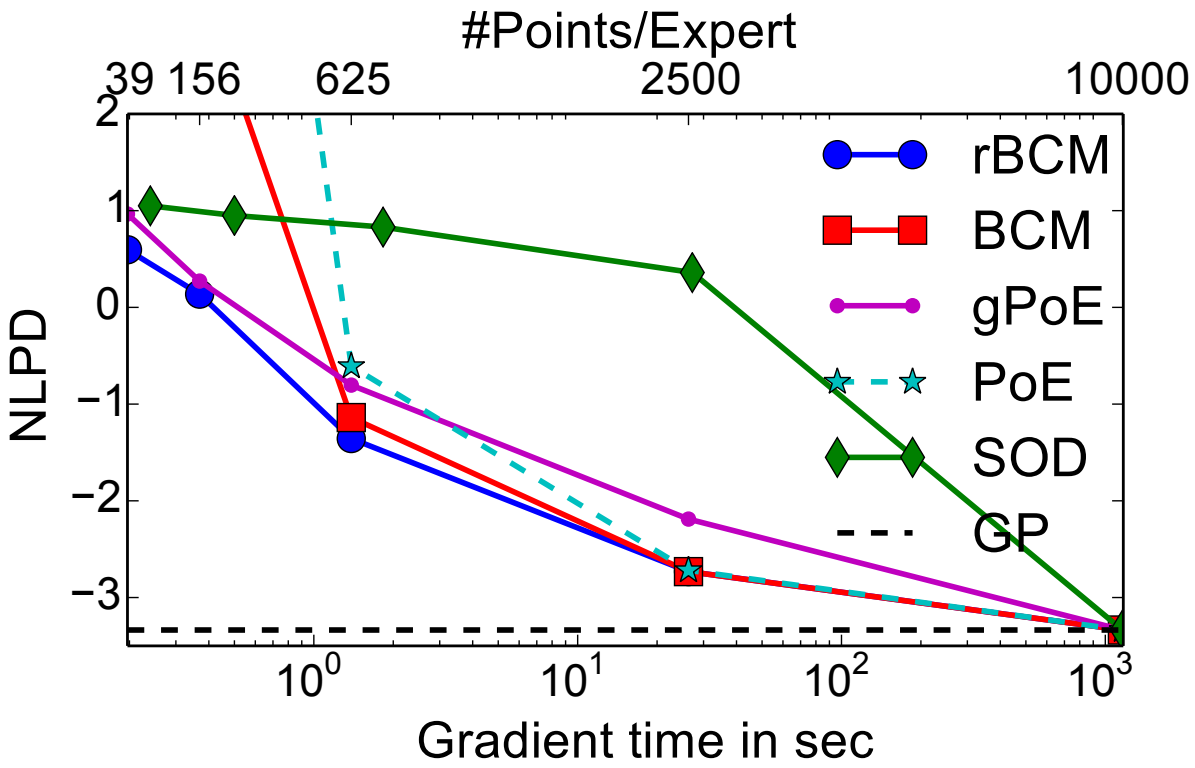
- Does not break down in case of weak experts ⏩ Robustified ✓
- Robust version of BCM ⏩ Reasonable predictions ✓
- Independent of computational graph (for all choices of β_k) ✓

Empirical Approximation Error



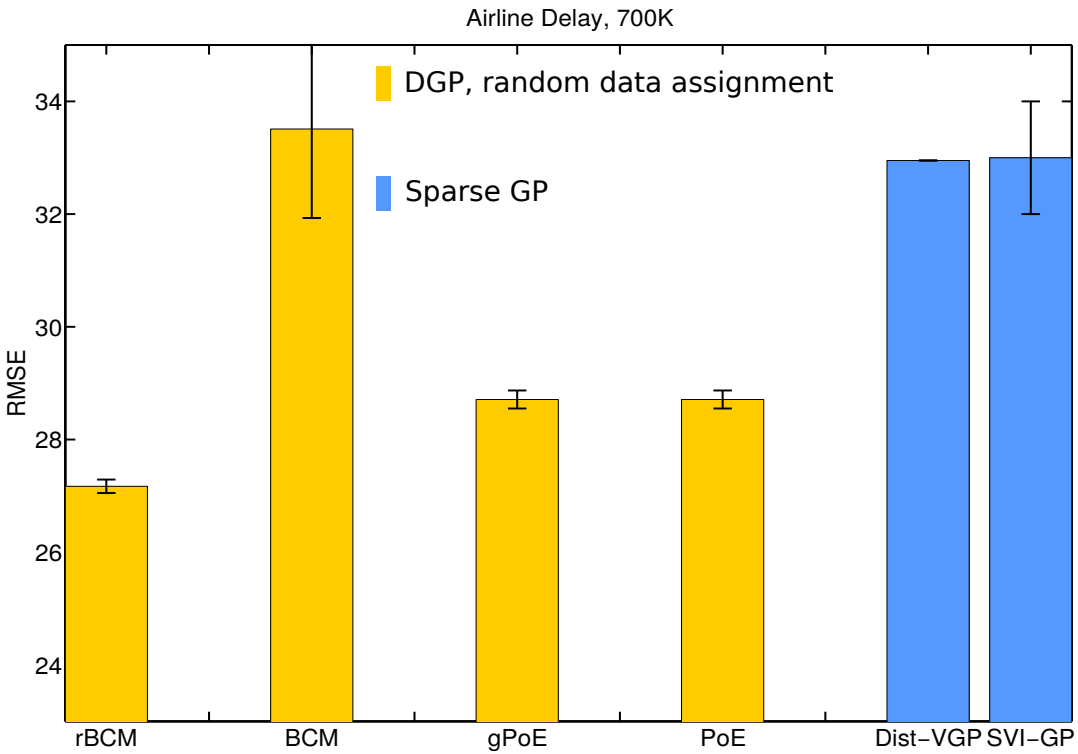
- Simulated robot arm data (10K training, 10K test)
- Hyper-parameters of ground-truth full GP
- RMSE as a function of the training time
- Sparse GP (SOR) performs worse than any distributed GP
- rBCM performs best with "weak" GP experts

Empirical Approximation Error (2)



- NLPD as a function of the training time ⏩ Mean and variance
- BCM and PoE are not robust for weak experts
- gPoE suffers from too conservative variances
- rBCM consistently outperforms other methods

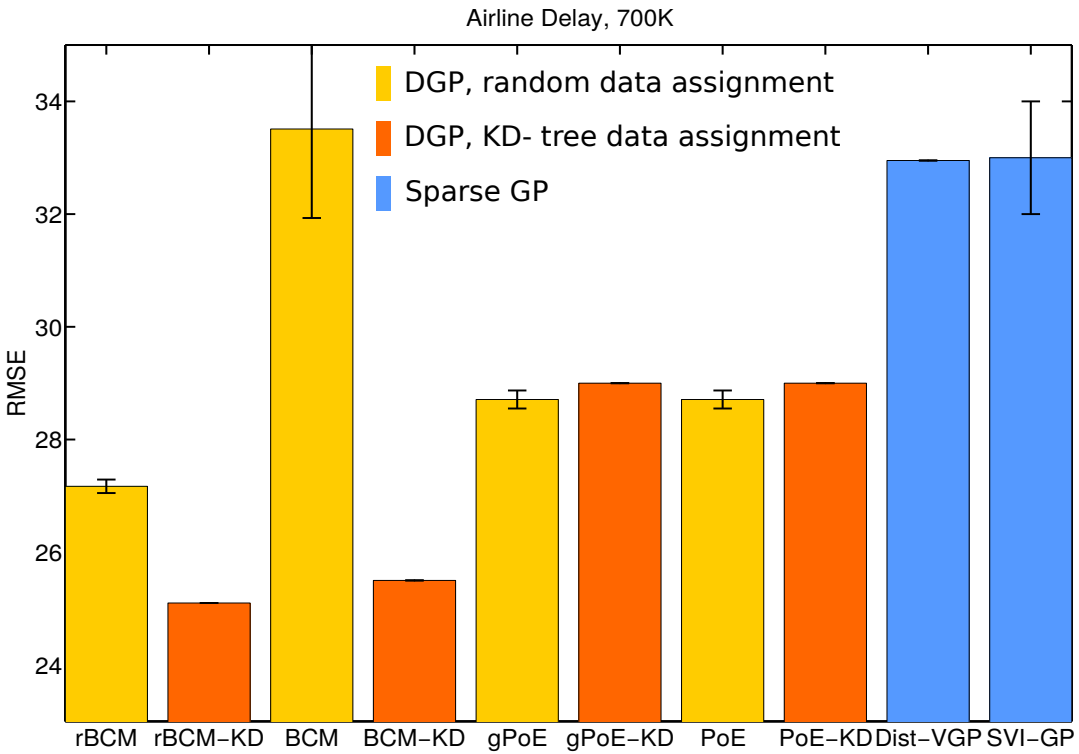
Large Data Sets: Airline Data (700K)



- (r)BCM and (g)PoE with 4096 GP experts
- Gradient time: 13 seconds (12 cores)
- Inducing inputs: Dist-VGP (Gal et al., 2014), SVI-GP (Hensman et al., 2013)

- rBCM performs best
- (g)PoE and BCM perform not worse than sparse GPs

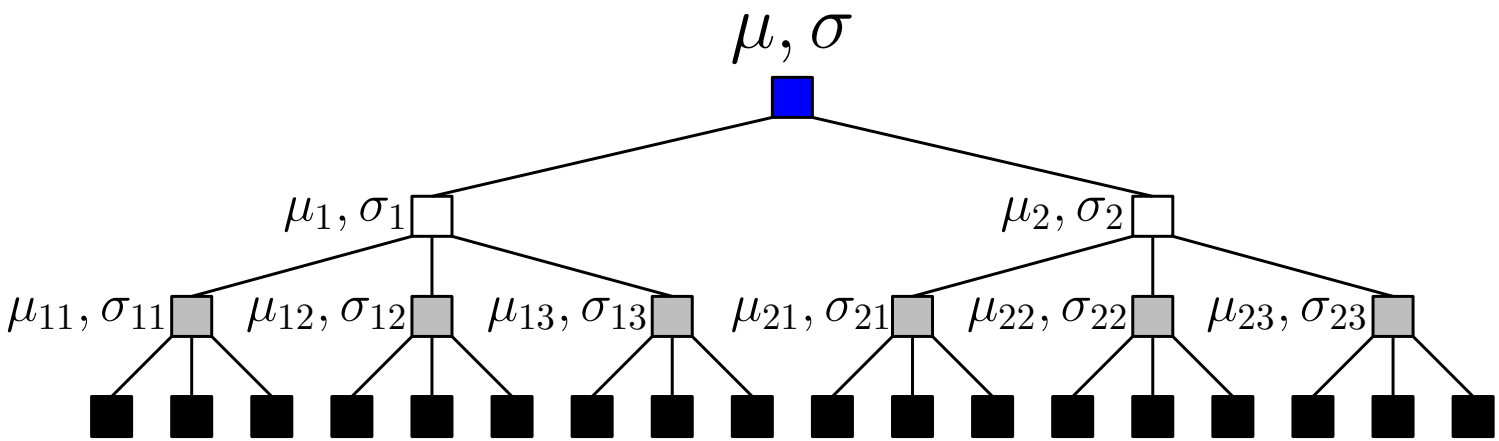
Large Data Sets: Airline Data (700K)



- (r)BCM and (g)PoE with 4096 GP experts
- Gradient time: 13 seconds (12 cores)
- Inducing inputs: Dist-VGP (Gal et al., 2014), SVI-GP (Hensman et al., 2013)

- rBCM performs best
- (g)PoE and BCM perform not worse than sparse GPs
- KD-tree data assignment clearly helps (r)BCM

Summary: Distributed Gaussian Processes



- Scale Gaussian processes to large data (beyond 10^6)
- Model conceptually straightforward and easy to train
- Key: Distributed computation
- Currently tested with $N \in \mathcal{O}(10^7)$
- Scales to arbitrarily large data sets (with enough computing power)

m.deisenroth@imperial.ac.uk

References

[1] Y. Cao and D. J. Fleet. Generalized Product of Experts for Automatic and Principled Fusion of Gaussian Process Predictions. `http://arxiv.org/abs/1410.7827`, October 2014.

[2] K. Chalupka, C. K. I. Williams, and I. Murray. A Framework for Evaluating Approximate Methods for Gaussian Process Regression. *Journal of Machine Learning Research*, 14:333–350, February 2013.

[3] M. P. Deisenroth and J. W. Ng. Distributed Gaussian Processes. In *Proceedings of the International Conference on Machine Learning*, 2015.

[4] M. P. Deisenroth, C. E. Rasmussen, and D. Fox. Learning to Control a Low-Cost Manipulator using Data-Efficient Reinforcement Learning. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2011.

[5] Y. Gal, M. van der Wilk, and C. E. Rasmussen. Distributed Variational Inference in Sparse Gaussian Process Regression and Latent Variable Models. In *Advances in Neural Information Processing Systems*. 2014.

[6] J. Hensman, N. Fusi, and N. D. Lawrence. Gaussian Processes for Big Data. In A. Nicholson and P. Smyth, editors, *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2013.

[7] T. Heskes. Selecting Weighting Factors in Logarithmic Opinion Pools. In *Advances in Neural Information Processing Systems*, pages 266–272. Morgan Kaufman, 1998.

[8] N. Lawrence. Probabilistic Non-linear Principal Component Analysis with Gaussian Process Latent Variable Models. *Journal of Machine Learning Research*, 6:1783–1816, November 2005.

[9] J. Ng and M. P. Deisenroth. Hierarchical Mixture-of-Experts Model for Large-Scale Gaussian Process Regression. `http://arxiv.org/abs/1412.3078`, December 2014.

[10] J. Quiñonero-Candela and C. E. Rasmussen. A Unifying View of Sparse Approximate Gaussian Process Regression. *Journal of Machine Learning Research*, 6(2):1939–1960, 2005.

[11] E. Snelson and Z. Ghahramani. Sparse Gaussian Processes using Pseudo-inputs. In Y. Weiss, B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1257–1264. The MIT Press, Cambridge, MA, USA, 2006.

[12] M. K. Titsias. Variational Learning of Inducing Variables in Sparse Gaussian Processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2009.

[13] V. Tresp. A Bayesian Committee Machine. *Neural Computation*, 12(11):2719–2741, 2000.