Imperial College London

Distributed Gaussian Processes for Large-Scale Probabilistic Regression

Marc Deisenroth

Department of Computing
Imperial College London

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

Joint work with Jun Wei Ng

Workshop on Gaussian Process Approximations
Copenhagen, 21 May 2015

Scaling Gaussian Processes to Large Data Sets

Two orthogonal approaches

- Sparse Gaussian processes
  - ⏩ Use (smart) subset of data.
- Distributed Gaussian processes
  - ⏩ Use full data set, distribute computations

Sparse Gaussian Processes

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

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)

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)

Sparse Gaussian Processes

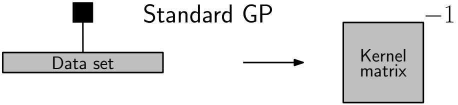
- Sparse approximations typically approximate a GP with N data points by a model with $M \ll N$ data points
- Selection of these M data points can be tricky and may involve non-trivial computations (e.g., optimizing inducing inputs)
- Simple (random) subset selection is fast and generally robust (Chalupka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

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 of the data set size is $N \in \mathcal{O}(10^6)$

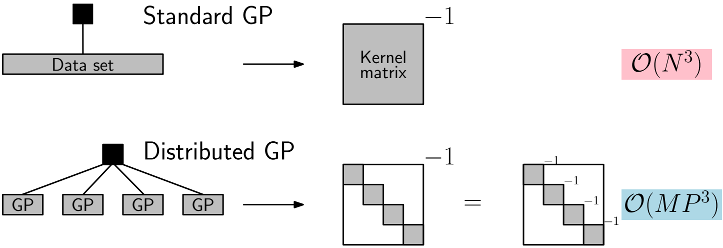
Distributed Gaussian Processes

Standard GP



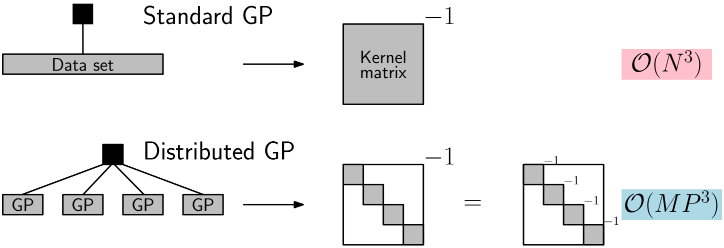
$\mathcal{O}(N^3)$

Distributed Gaussian Processes



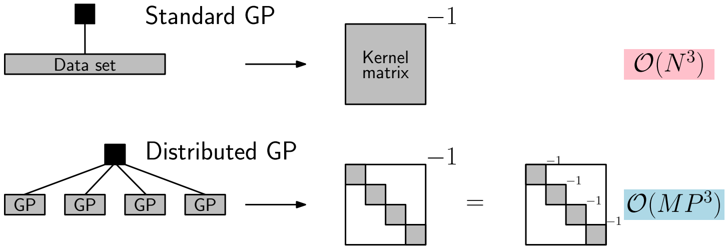
- Randomly split the full data set into M chunks of size P

Distributed Gaussian Processes



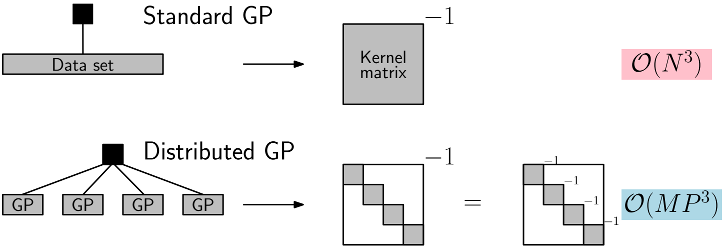
- Randomly split the full data set into M chunks of size P
- Place M independent GP experts on these small chunks

Distributed Gaussian Processes



- Randomly split the full data set into M chunks of size P
- Place M independent GP experts on these small chunks
- Block-diagonal approximation of kernel matrix $\boldsymbol{K}$ (sim. to PIC)

Distributed Gaussian Processes



- Randomly split the full data set into M chunks of size P
- Place M independent GP experts on these small chunks
- Block-diagonal approximation of kernel matrix $\boldsymbol{K}$ (sim. to PIC)
- Combine independent computations to an overall result

Training the Distributed GP

- Randomly 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

- Randomly 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
- No inducing/variational parameters ⏩ Easy optimization

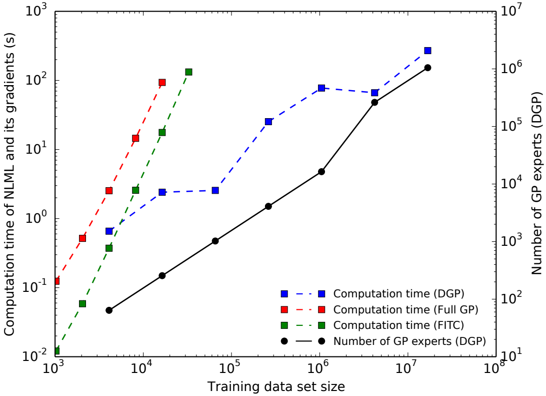
Training the Distributed GP

- Randomly 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
- No inducing/variational parameters ⏩ Easy optimization
- Computational complexity: $\mathcal{O}(MP^3)$ [instead of $\mathcal{O}(N^3)$]
- Memory footprint: $\mathcal{O}(MP^2 + ND)$ [instead of $\mathcal{O}(N^2 + ND)$], potentially distributed across M computing nodes

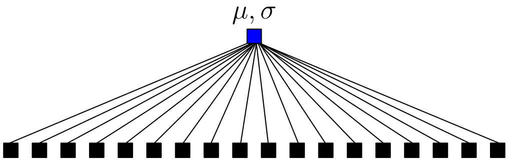
Scaling



- NLML is proportional to training time
- Full GP (16K training points) $\approx$ sparse GP (32K training points) $\approx$ distributed GP (16M training points)

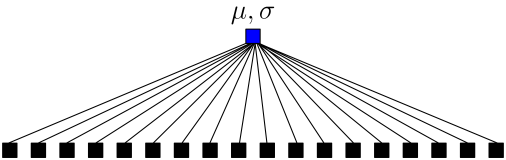
▶▶ Push practical limit by order(s) of magnitude

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 $\mathcal{N}(\mu_i, \sigma_i^2)$
- How to combine them to an overall prediction $\mathcal{N}(\mu, \sigma^2)$?

▸▸ 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

- Scale to large data sets ✓

Objectives

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

Objectives

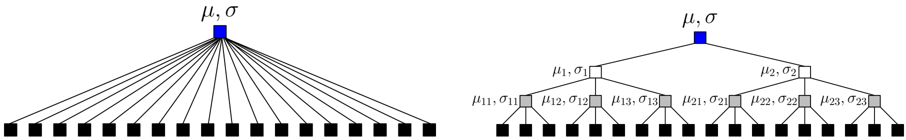


Figure: Two computational graphs

- Scale to large data sets ✓
- Good approximation of full GP ("ground truth")
- Predictions independent of computational graph
  ▶▶ Heterogeneous computing infrastructures (laptop, cluster, ...)

Objectives

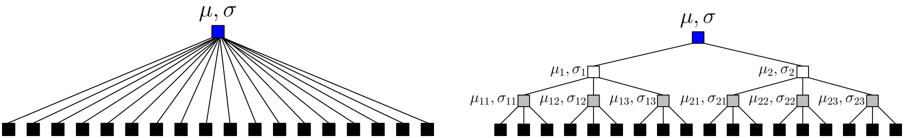
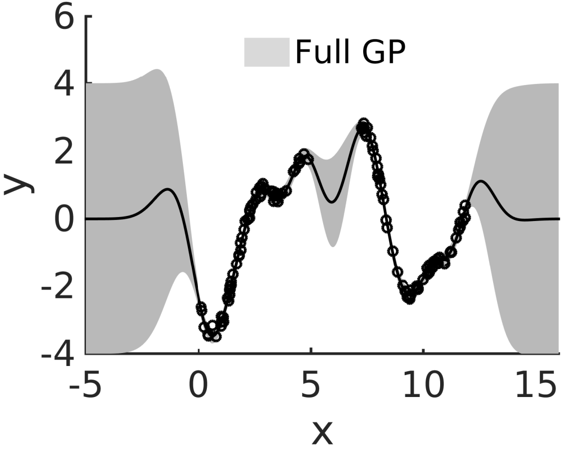


Figure: Two computational graphs

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

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

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

Product of GP Experts

- Prediction model (independent predictors):

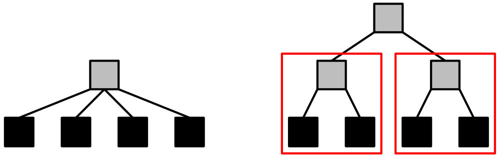
$$p(f_*|\boldsymbol{x}_*, \mathcal{D}) = \prod_{k=1}^{M} p_k(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)}),$$

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

- Predictive precision and mean:

$$\begin{aligned}(\sigma_*^{\text{poe}})^{-2} &= \textstyle\sum_k \sigma_k^{-2}(\boldsymbol{x}_*)\\ \mu_*^{\text{poe}} &= (\sigma_*^{\text{poe}})^2 \textstyle\sum_k \sigma_k^{-2}(\boldsymbol{x}_*)\mu_k(\boldsymbol{x}_*)\end{aligned}$$

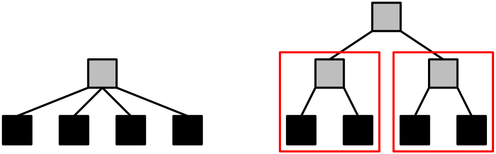
Computational Graph



Prediction:

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

Computational Graph

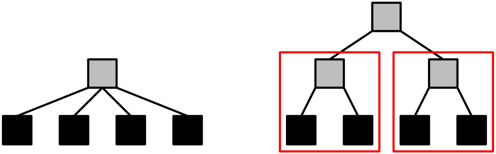


Prediction:

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

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

Computational Graph



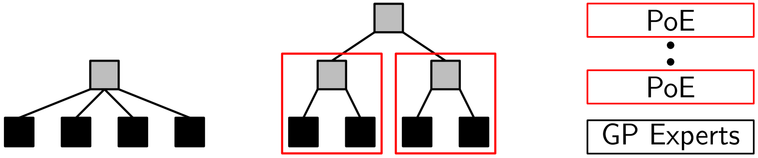
Prediction:

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

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

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

Computational Graph



Prediction:

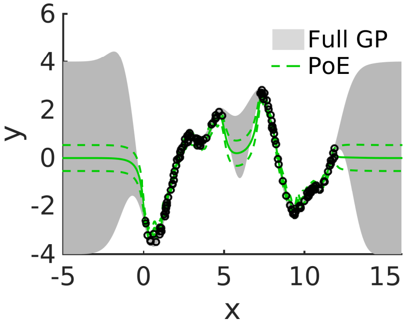
$$p(f_*|\boldsymbol{x}_*, \mathcal{D}) = \prod_{k=1}^{M} p_k(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)})$$

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 $M > 1$:

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

- 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 (Cao & Fleet, 2014)

- Weight the responsiblity of each expert in PoE with β_k

Generalized Product of GP Experts (Cao & Fleet, 2014)

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

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

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

Generalized Product of GP Experts (Cao & Fleet, 2014)

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

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

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

- 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 (Cao & Fleet, 2014)

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

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

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

- Predictive precision and mean:

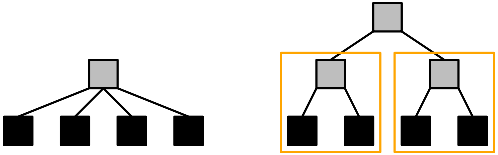
$$(\sigma_*^{\text{gpoe}})^{-2} = \sum\nolimits_k \beta_k \sigma_k^{-2}(\boldsymbol{x}_*)$$

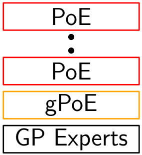
$$\mu_*^{\text{gpoe}} = (\sigma_*^{\text{gpoe}})^2 \sum\nolimits_k \beta_k \sigma_k^{-2}(\boldsymbol{x}_*) \mu_k(\boldsymbol{x}_*)$$

- With $\sum_k \beta_k = 1$, the model can fall back to the prior ✓

▶▶ Log-opinion pool model (e.g., Heskes, 1998)

Computational Graph

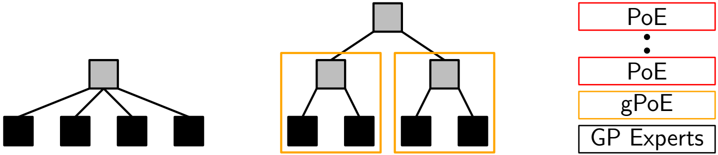




Prediction:

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

Computational Graph

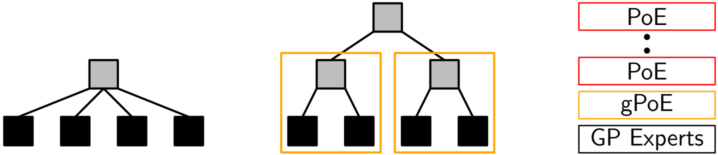


Prediction:

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

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

Computational Graph

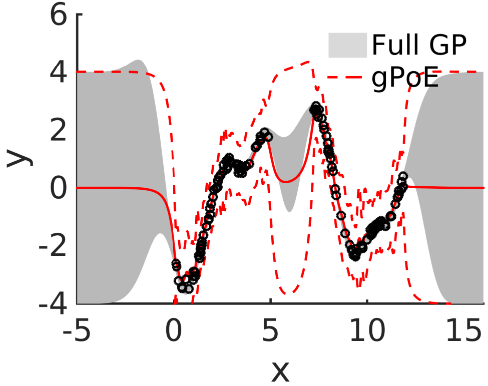


Prediction:

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

- 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 (Tresp, 2000)

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

Bayesian Committee Machine (Tresp, 2000)

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

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

Bayesian Committee Machine (Tresp, 2000)

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

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

- 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 (Tresp, 2000)

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

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

- 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 (Tresp, 2000)

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

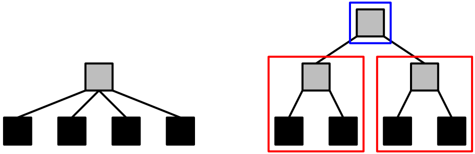
$$p(f_* | \boldsymbol{x}_*, \mathcal{D}) = \frac{\prod_{k=1}^{M} p_k(f_* | \boldsymbol{x}_*, \mathcal{D}^{(k)})}{p^{M-1}(f_*)}$$

- 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 ✓

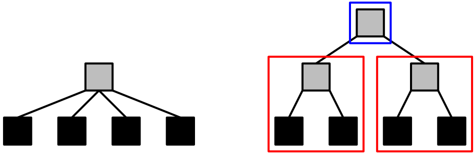
Computational Graph



Prediction:

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

Computational Graph

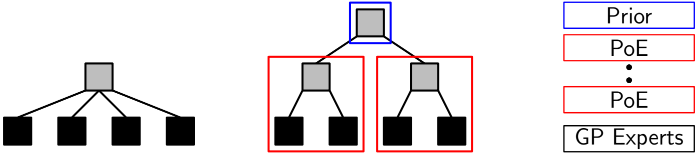


Prediction:

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

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

Computational Graph



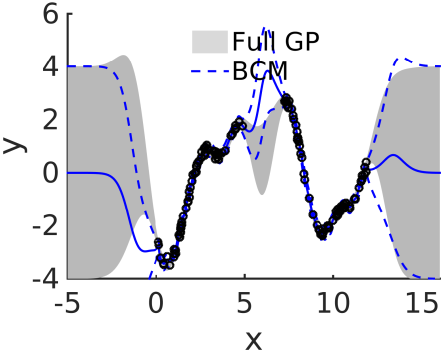
Prediction:

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

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

▸▸ Independent of computational graph ✓

Bayesian Committee Machine



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

  ▸▸ **Robustify**

Robust Bayesian Committee Machine

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

Robust Bayesian Committee Machine

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

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

Robust Bayesian Committee Machine

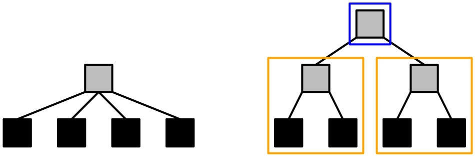
- Combine gPoE (weighting of experts) with the BCM (Bayes' theorem when combining predictions)
- Prediction model (conditional independence $\mathcal{D}^{(j)} \perp\!\!\!\perp \mathcal{D}^{(k)} | f_*$):

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

- Predictive precision and mean:

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

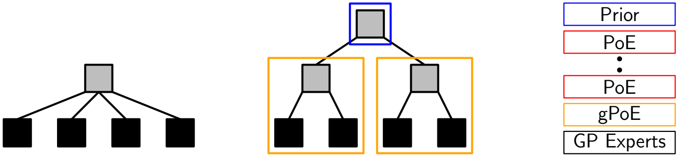
Computational Graph



Prediction:

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

Computational Graph

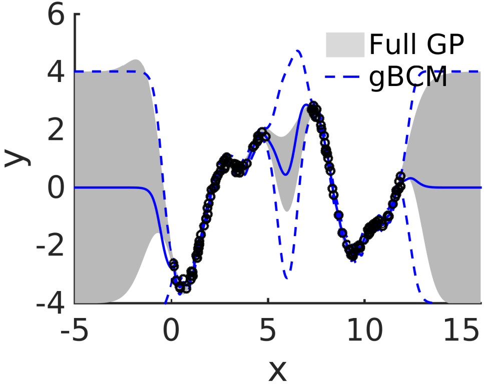


Prediction:

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

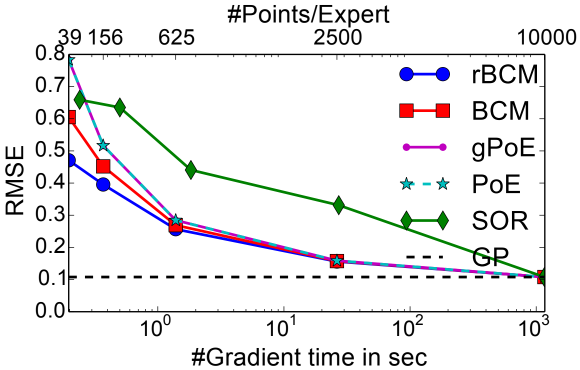
▶▶ Independent of computational graph, even with arbitrary β_k ✓

Robust Bayesian Committee Machine



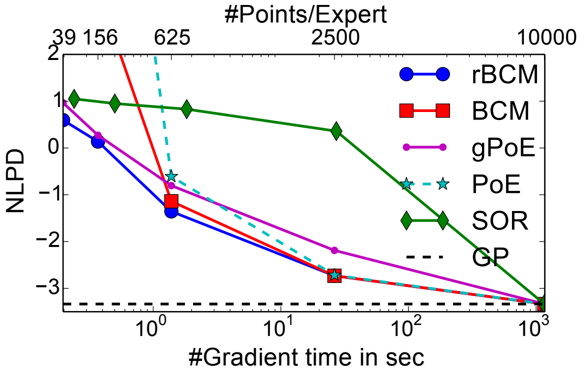
- Does not break down in case of weak experts ⏩ Robustified ✓
- Robust version of BCM ⏩ Reasonable predictions ✓
- Independent of computational graph (for all choices of β_k) ✓

Empirical Approximation Error



- Simulated robot arm data (10K training, 30K test)
- All models use 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)

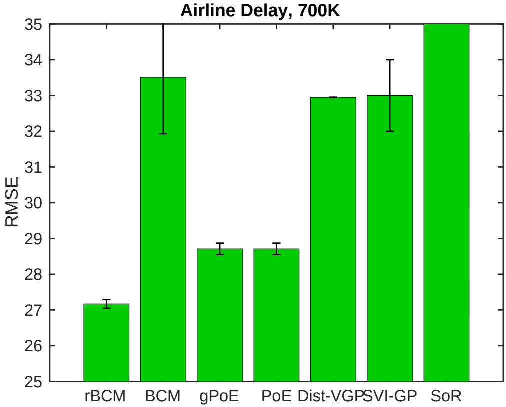


- NLPD as a function of the training time ▶▶ Mean and variance
- BCM and PoE are not robust to weak experts
- gPoE suffers from too conservative variances
- rBCM consistently outperforms other methods

Large Data

- Predict US Airline Delays (01/2008–04/2008) of commercial flights
- Inputs: age of aircraft, flight distance, departure/arrival time, airtime, day of week, day of month, month,
- Training data: 700K, 2M, 5M. Test data: 100K

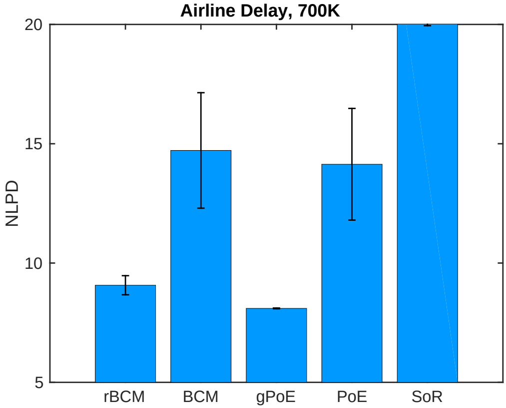
Training Data: 700K — RMSE



- (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 performs not worse sparse GPs

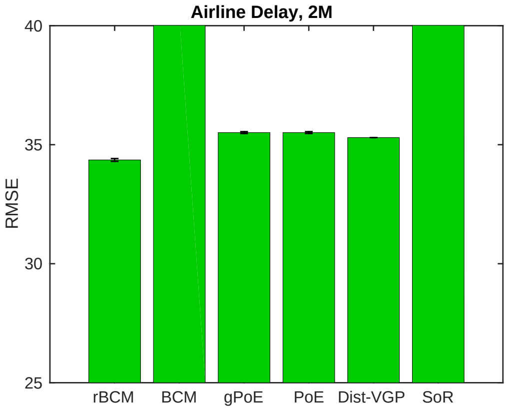
Training Data: 700K — NLPD



- (r)BCM and (g)PoE with 4096 GP experts
- Gradient time: 13 seconds (12 cores)
- No results reported for inducing input methods (Gal et al., 2014; Hensman et al., 2013)

- gPoE performs best, just ahead of rBCM

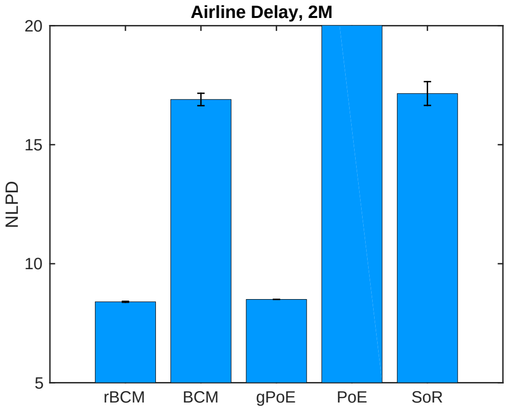
Training Data: 2M — RMSE



- (r)BCM and (g)PoE with 8192 GP experts
- Gradient time: 39 seconds (12 cores)
- Inducing inputs: Dist-VGP (Gal et al., 2014)

- rBCM performs best
- (g)PoE as good as best results reported for sparse methods
- BCM suffers from weak experts

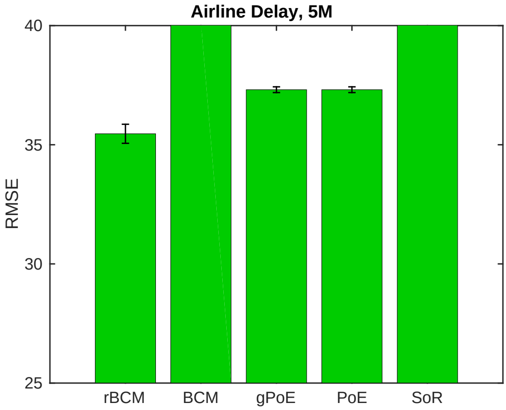
Training Data: 2M — NLPD



- (r)BCM and (g)PoE with 8192 GP experts
- Gradient time: 39 sec (12 cores)
- Inducing inputs: no results reported

- rBCM and gPoE perform best
- BCM suffers from weak experts
- PoE suffers from under-estimation of variances

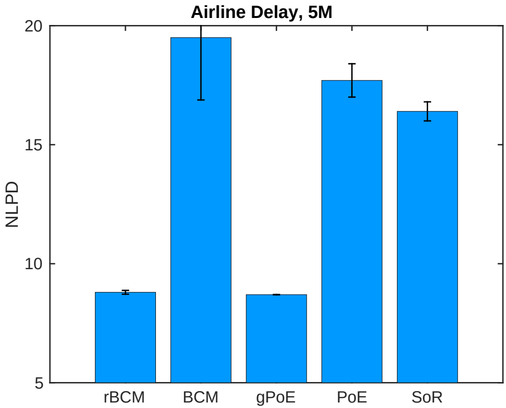
Training Data: 5M — RMSE



- (r)BCM and (g)PoE with 32768 GP experts
- Gradient time: 90 sec (12 cores)

- rBCM performs best
- (g)PoE produce good results
- BCM off the chart ⏩ suffers from weak experts

Training Data: 5M — NLPD



- (r)BCM and (g)PoE with 32768 GP experts
- Gradient time: 90 sec (12 cores)

- rBCM and gPoE perform best
- PoE and BCM significantly worse

Overview Airline Delays

- RMSE: rBCM consistently performs best
- NLPD: rBCM and gPoE approximately the same
  ▶▶ gPoE recovers because of conservative variance estimates
- BCM suffers from “wrong means”, PoE suffers from overconfident estimates
- All models: Training time is acceptable
- All experiments (DGP) run on a laptop

Summary

- Distributed product-of-experts approaches to scaling Gaussian processes to large data sets
- Robust Bayesian Committee Machine
- Model conceptually straightforward and easy to train
  - ▸▸ Only kernel hyper-parameters need to be optimized
- Independent of computational graph
- Scales to arbitrarily large data sets (in principle)

`m.deisenroth@imperial.ac.uk`

Thank you for your attention

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. Ng. Distributed Gaussian Processes. In *Proceedings of the International Conference on Machine Learning*, 2015.

[4] 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.

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

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

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