Data Analysis and Probabilistic Inference

Imperial College London

Distributed Gaussian Processes

Recommended reading:
Deisenroth & Ng (2015) [1]

Marc Deisenroth

Department of Computing
Imperial College London

February 15, 2018

Limitations of Gaussian Processes

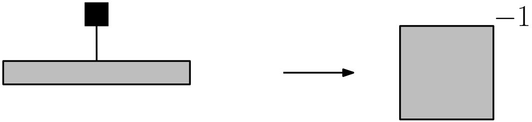
Computational and memory complexity

Training set size: N

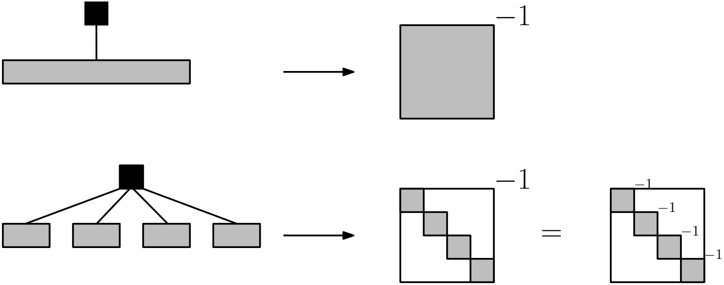
- Training scales in $\mathcal{O}(N^3)$
- Prediction (variances) scales in $\mathcal{O}(N^2)$
- Memory requirement: $\mathcal{O}(ND + N^2)$

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

Large-Scale GPs via Distributed Inference

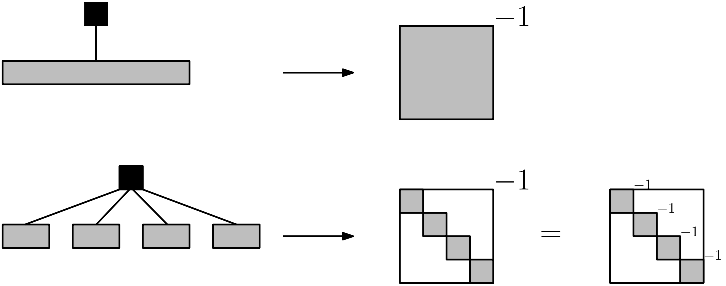


Large-Scale GPs via Distributed Inference



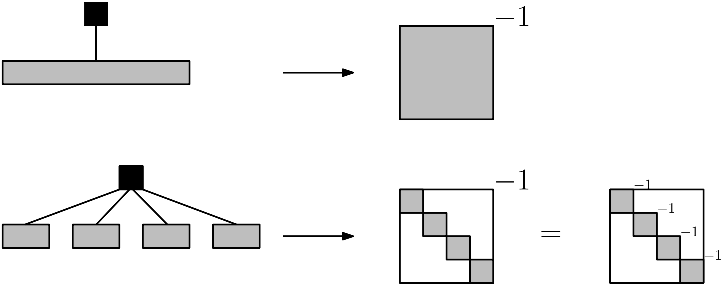
- Randomly split the full data set into M chunks

Large-Scale GPs via Distributed Inference



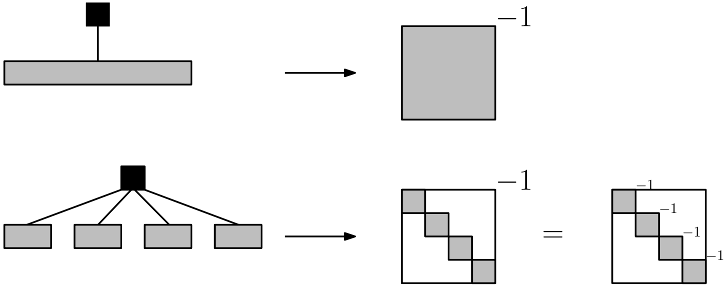
- Randomly split the full data set into M chunks
- Place M independent GP models (experts) on these small chunks

Large-Scale GPs via Distributed Inference



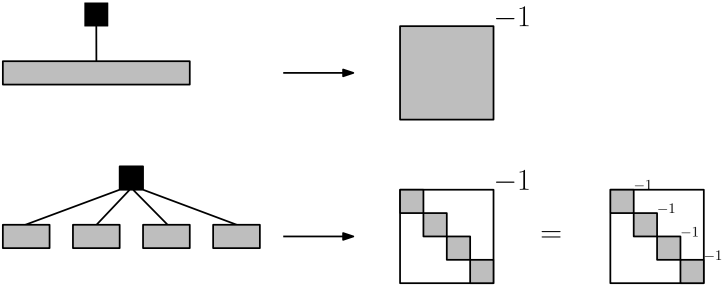
- Randomly split the full data set into M chunks
- Place M independent GP models (experts) on these small chunks
- Independent computations can be distributed

Large-Scale GPs via Distributed Inference



- 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 $\boldsymbol{K}$

Large-Scale GPs via Distributed Inference



- 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

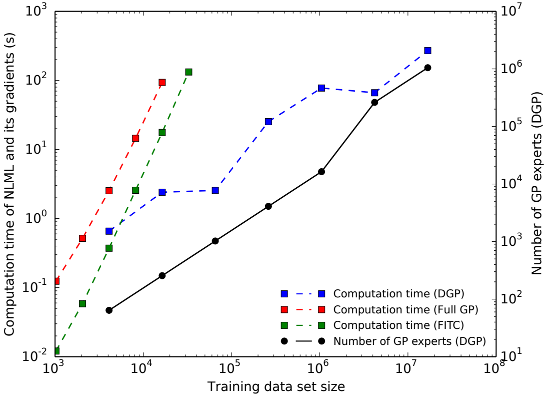
Training the Distributed GP

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

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

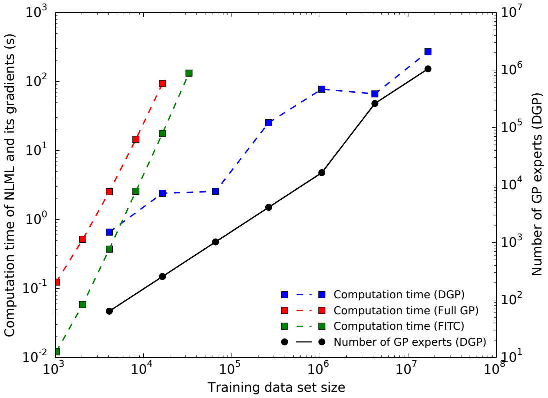
- Distributed optimization and training straightforward
- Computational complexity: $\mathcal{O}(MP^3)$ [instead of $\mathcal{O}(N^3)$]
  But distributed over many machines
- Memory footprint: $\mathcal{O}(MP^2 + ND)$ [instead of $\mathcal{O}(N^2 + ND)$]

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: $\approx$ 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: $\approx$ 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

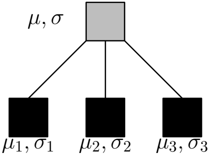
- Training* with $N = 10^6, D = 1$ on a laptop: $\approx$ 30 min
- Training* with $N = 5 \times 10^6, D = 8$ on a workstation: $\approx$ 4 hours

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

: Convergence often after 30–80 line searches*

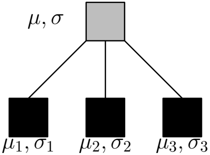
***: Line search $\approx$ 2–3 evaluations of marginal likelihood and its gradient (usually $\mathcal{O}(N^3)$)

Predictions with the Distributed GP



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

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

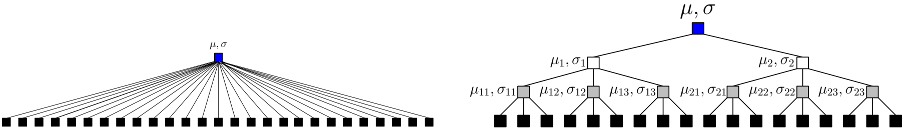


Figure: Two computational graphs

- Scale to large data sets ✓

Objectives

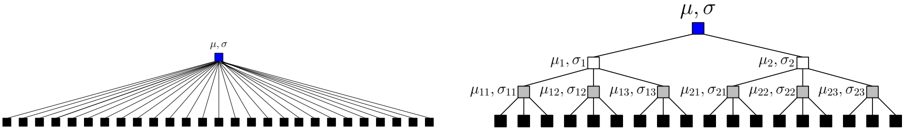


Figure: Two computational graphs

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

Objectives

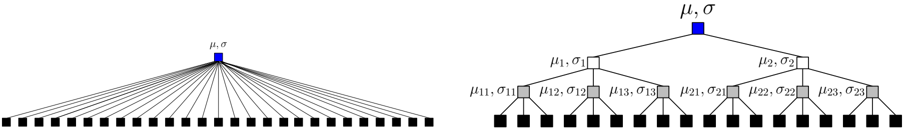


Figure: Two computational graphs

- 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

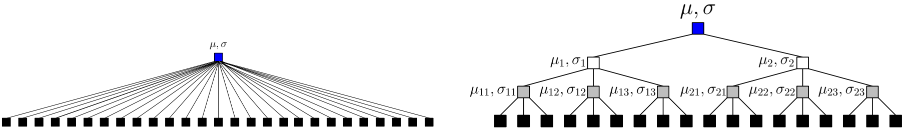
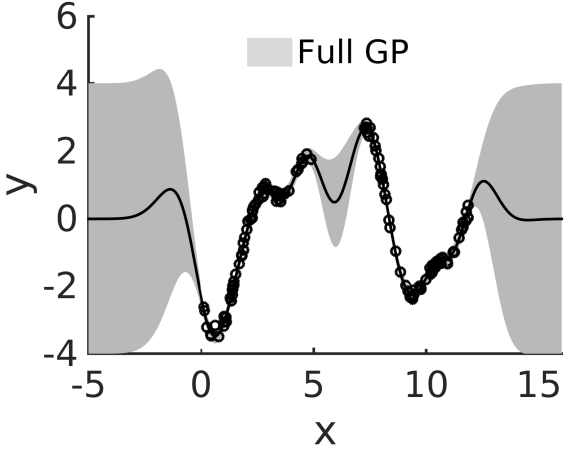


Figure: Two computational graphs

- 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):

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

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

Product of GP Experts

- Prediction model (independent predictors):

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

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

- Predictive precision (inverse variance) and mean:

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

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

Product of GP Experts

- Prediction model (independent predictors):

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

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

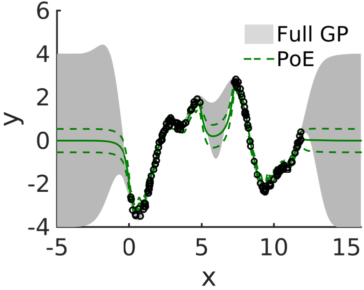
- Predictive precision (inverse variance) and mean:

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

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

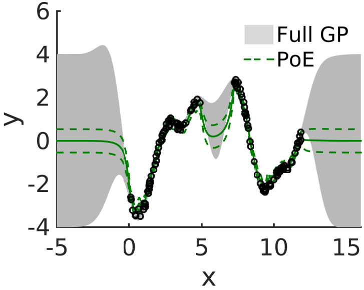
- Independent of the computational graph ✓

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 (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 (Heskes, 1998)

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

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

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

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

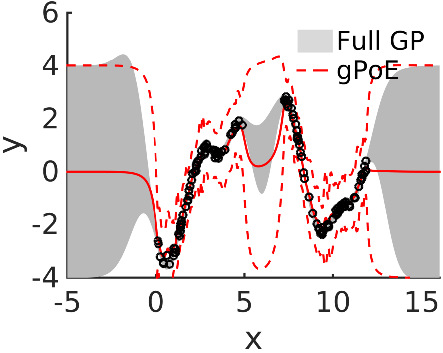
- Predictive precision and mean:

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

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

- With $\sum_k \beta_k = 1$, the model can fall back to the prior ✓
  "Log-opinion pool" model (Heskes, 1998)
- Independent of computational graph for $\beta_k = 1/M$ ✓

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



- 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 ($\mathcal{D}^{(j)} \perp\!\!\!\perp \mathcal{D}^{(k)} | f_*$):

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

Bayesian Committee Machine (Tresp, 2000)

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

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

- 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 ($\mathcal{D}^{(j)} \perp\!\!\!\perp \mathcal{D}^{(k)} | f_*$):

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

- 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 ($\mathcal{D}^{(j)} \perp\!\!\!\perp \mathcal{D}^{(k)} | f_*$):

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

- 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 ✓

Bayesian Committee Machine (Tresp, 2000)

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

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

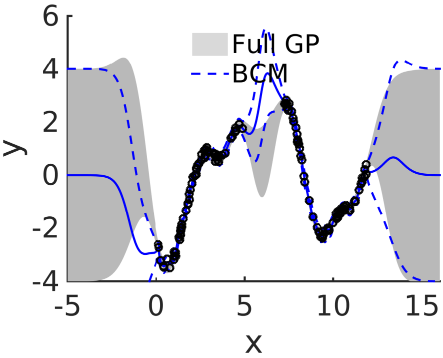
- Predictive precision and mean:

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

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

- Product of GP experts, divided by $M-1$ times the prior
- Guaranteed to fall back to the prior outside data regime ✓
- 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 $\mathcal{D}^{(j)} \perp\!\!\!\perp \mathcal{D}^{(k)} | f_*$):

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

Robust Bayesian Committee Machine

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

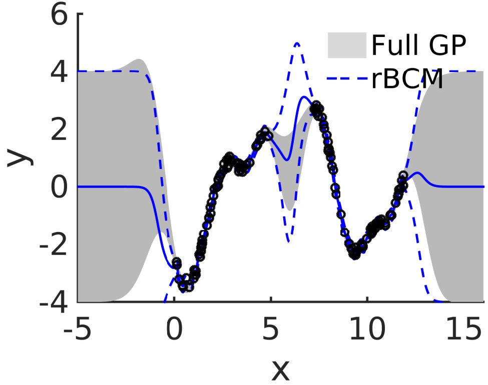
$$p(f_*|\boldsymbol{x}_*, \mathcal{D}) = \frac{\prod_{k=1}^{M} p_k^{\beta_k}(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)})}{p^{\sum_k \beta_k - 1}(f_*)}$$

- Predictive precision and mean:

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

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

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) ✓

Setting the Weighting β_k

- The gPoE and the rBCM have a β_k parameter that assigns individual experts different weights when predicting:

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

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

Setting the Weighting β_k

- The gPoE and the rBCM have a β_k parameter that assigns individual experts different weights when predicting:

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

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

- Intuition: Set $\beta_k(\boldsymbol{x}_*)$ such that "informed" GP experts get more influence

Setting the Weighting β_k

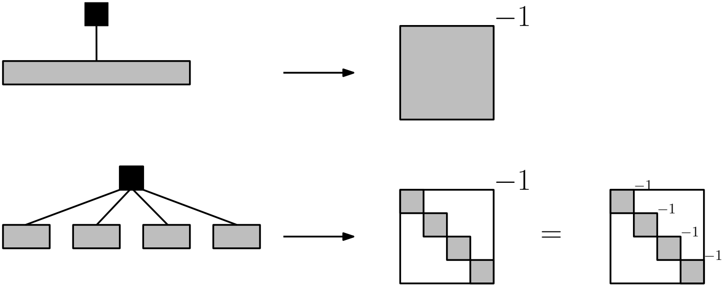
- The gPoE and the rBCM have a β_k parameter that assigns individual experts different weights when predicting:

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

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

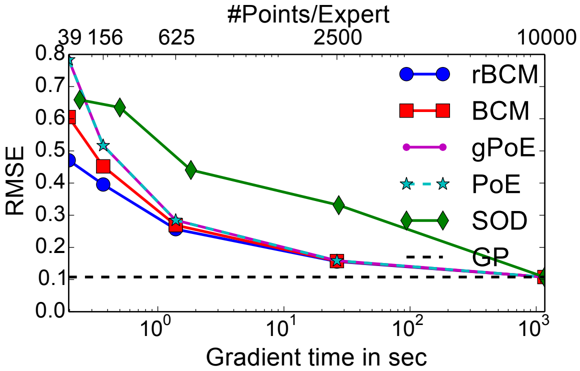
- Intuition: Set $\beta_k(\boldsymbol{x}_*)$ such that "informed" GP experts get more influence
- Use some distance/divergence between GP prior and GP posterior at test point $\boldsymbol{x}_*$
- Some options for β_k:
  - $\beta_k \propto \mathrm{KL}(\text{prior}||\text{posterior})$
  - $\beta_k \propto \mathrm{DiffEnt}(\text{prior}, \text{posterior})$

Splitting the Data



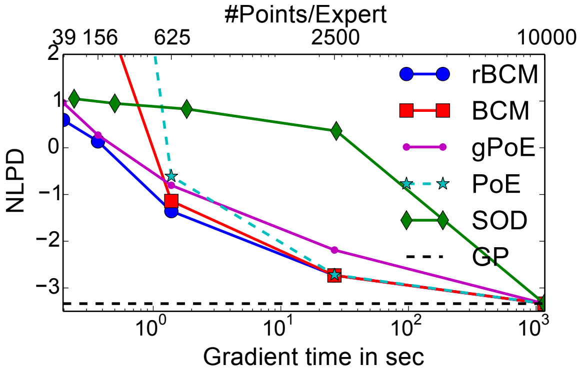
- Data sets should be of approximately the same size
- Random assignment of data points to experts
- Cluster inputs (e.g., k-means), assign clusters to experts

Empirical Approximation Error (1)



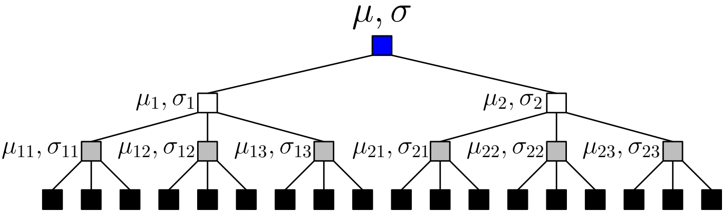
- Simulated robot arm data (10K training, 10K test)
- Hyper-parameters of ground-truth full GP
- RMSE as a function of the training time
- Subset of data (SOD) performs worse than any distributed GP
- rBCM performs best with "weak" GP experts

Empirical Approximation Error (2)



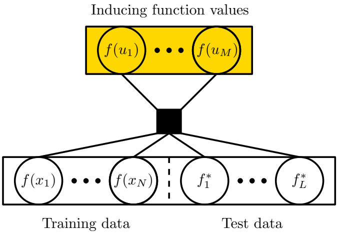
- NLPD as a function of the training time ▸▸ Mean and variance
- BCM and PoE are not robust for weak experts
- gPoE suffers from too conservative variances
- rBCM consistently outperforms other methods

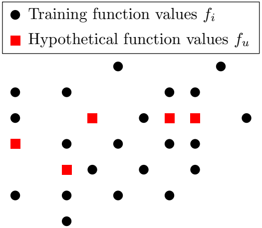
Summary: Distributed Gaussian Processes



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

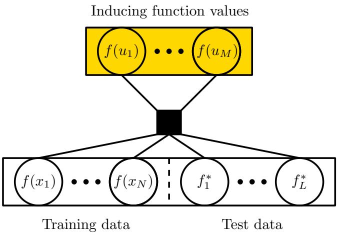
Scaling GPs using Inducing Inputs

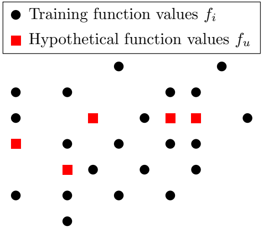




- Introduce inducing function values f_u
  - "Hypothetical" function values

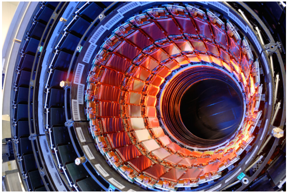
Scaling GPs using Inducing Inputs

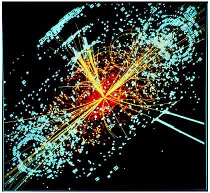




- Introduce inducing function values f_u
  - “Hypothetical” function values
- All function values are still jointly Gaussian distributed (e.g., training, test and inducing function values)
- Compress information into inducing function values
- Selected references: [6–13]

Gaussian Processes in High-Energy Physics





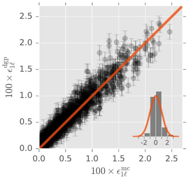
- LHC BSM simulator experiments (e.g., predicting natural supersymmetry signal events) can be very time consuming
- Sampling in a high-dimensional parameter space of theoretical models
  - Monte Carlo sampling of collision events
  - Run samples through a detector simulation
  - Compare predicted signal with real data

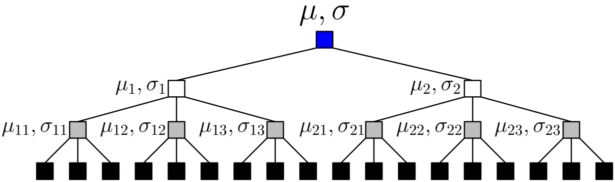
▶▶ Bottleneck for global theoretical analysis of BSM theories

Rapid Predictions

- Learn mapping between theory and data
- Rapidly predict signal region (SR) differences
- Model the relationship between BSM parameters $\boldsymbol{\theta}$ and SR efficiency ϵ with Gaussian processes: $\epsilon = f(\boldsymbol{\theta}), f \sim GP$.

GP Surrogate Model for the Full Simulation Chain

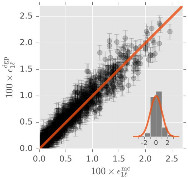


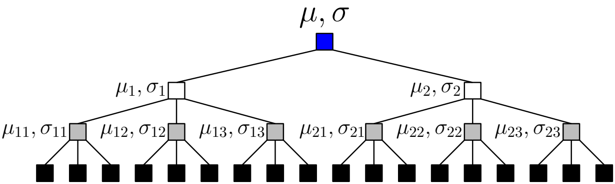


Challenges:

- Training set is moderately large (18,000) ⏩ Distributed GPs

GP Surrogate Model for the Full Simulation Chain





Challenges:

- Training set is moderately large (18,000) ⏩ Distributed GPs

Results:

- Similar to expensive MC simulator (event generator)
- 10,000-fold speedup for reconstruction of theory parameters
- Rapid reconstruction of the theory parameters of a BSM model
- New opportunities in the interpretation of LHC data

Deisenroth & Ng (ICML, 2015): *Distributed Gaussian Processes*
Bertone et al. (arXiv 1611.02704): *Accelerating the BSM Interpretation of LHC Data with Machine Learning*

References I

[1] Marc P. Deisenroth and Jun W. Ng. Distributed Gaussian Processes. In *Proceedings of the International Conference on Machine Learning*, 2015.

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

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

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

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

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

[7] Edward Snelson and Zoubin 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.

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

[9] James Hensman, Nicolò Fusi, and Neil 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.

[10] Yarin Gal, Mark van der Wilk, and Carl E. Rasmussen. Distributed Variational Inference in Sparse Gaussian Process Regression and Latent Variable Models. In *Advances in Neural Information Processing Systems*. 2014.

[11] Andrew G. Wilson and Hannes Nickisch. Kernel Interpolation for Scalable Structured Gaussian Processes (KISS-GP). In *Proceedings of the International Conference on Machine Learning*, 2015.

References II

[12] Seth R. Flaxman, Andrew G. Wilson, Daniel B. Neill, Hannes Nickisch, and Alexander J. Smola. Fast Kronecker Inference in Gaussian Processes with non-Gaussian Likelihoods. In *Proceedings of the International Conference on Machine Learning*, 2014.

[13] Hugh Salimbeni and Marc P. Deisenroth. Doubly Stochastic Variational Inference for Deep Gaussian Processes. In *Advances in Neural Information Processing Systems*, 2017.

[14] Gianfranco Bertone, Marc P. Deisenroth, Jong S. Kim, Sebastian Liem, Roberto R. de Austri, and Max Welling. Accelerating the BSM Interpretation of LHC Data with Machine Learning. arXiv preprint arXiv:1611.02704, 2016.

Appendix

BCM: Derivation

Conditional Independence Assumption (BCM)

$\mathcal{D}^{(j)} \perp\!\!\!\perp \mathcal{D}^{(k)} | f_*$

$$
\begin{aligned}
p(f_*|\mathcal{D}^{(j)}, \mathcal{D}^{(k)}) &\propto p(\mathcal{D}^{(j)}, \mathcal{D}^{(k)}|f_*)p(f_*) \\
&\overset{\text{BCM}}{=} p(\mathcal{D}^{(j)}|f_*)\, p(\mathcal{D}^{(k)}|f_*)p(f_*) \\
&= \frac{p(\mathcal{D}^{(j)}, f_*)\, p(\mathcal{D}^{(k)}, f_*)}{p(f_*)} \\
&\propto \frac{p_k(f_*|\mathcal{D}^{(k)})p_j(f_*|\mathcal{D}^{(j)})}{p(f_*)}
\end{aligned}
$$