Probabilistic Linear Solvers

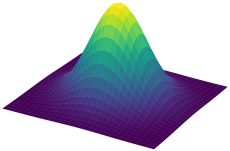
Jonathan Wenger

COLUMBIA | Zuckerman Institute

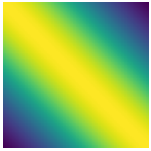
Linear Systems are Everywhere in Scientific Computing

Arguably, the most fundamental numerical task in scientific computing and machine learning.

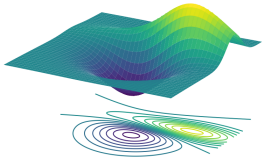
Basic Statistics



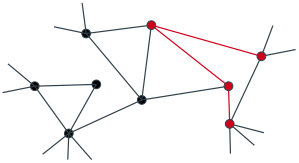
Probabilistic / Kernel Methods



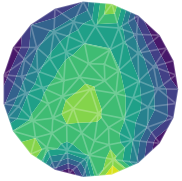
Optimization



Graphs and (Neural) Networks



Differential Equations

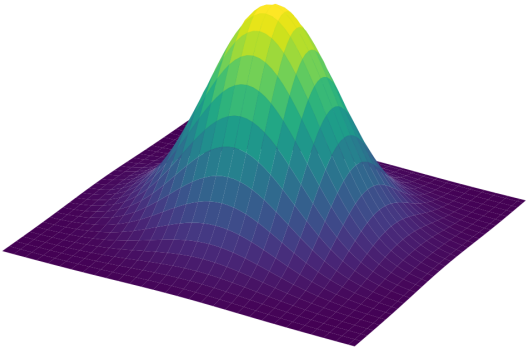


…and many more.

Normal Distribution

$$x \sim \mathcal{N}(\mu, \Sigma)$$

$$p(x) = \frac{1}{\sqrt{(2\pi)^n \det(\Sigma)}} \exp(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu))$$



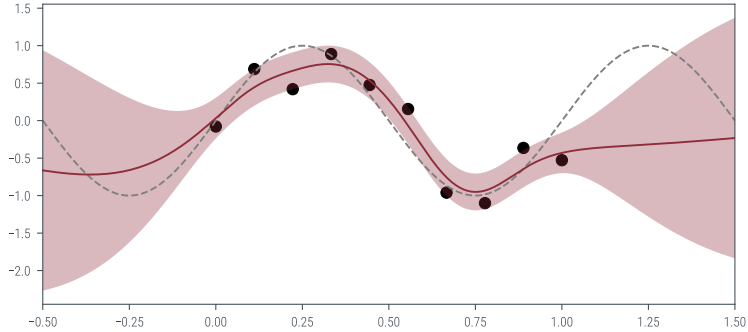
Gaussian Processes

$$f \sim \mathcal{GP}(\mu, k)$$

$$f \mid \boldsymbol{X}, \boldsymbol{y} \sim \mathcal{GP}(\mu_{\text{post}}, k_{\text{post}})$$

$$\mu_{\text{post}}(\boldsymbol{x}) = \mu(\boldsymbol{x}) + k(\boldsymbol{x}, \boldsymbol{X})(k(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 \boldsymbol{I})^{-1}(\boldsymbol{y} - \mu(\boldsymbol{X}))$$

$$k_{\text{post}}(\boldsymbol{x}_0, \boldsymbol{x}_1) = k(\boldsymbol{x}_0, \boldsymbol{x}_1) - k(\boldsymbol{x}_0, \boldsymbol{X})(k(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 \boldsymbol{I})^{-1} k(\boldsymbol{X}, \boldsymbol{x}_1)$$

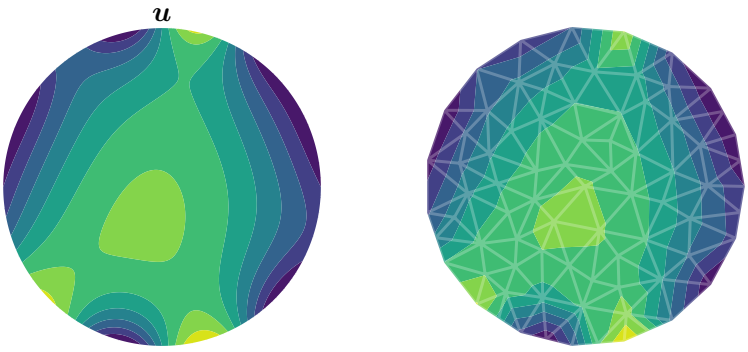


Linear Systems are Everywhere in Scientific Computing

Example: Linear Differential Equations.

Galerkin Method

$$\underbrace{Du = f}_{\text{linear differential equation}} \quad \Longrightarrow \quad \underbrace{\hat{D}\hat{u} = \hat{f}}_{\text{finite dimensional linear system}}$$

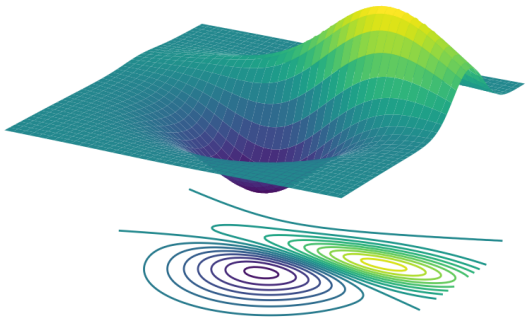


Iterative Optimization Methods

$$\boldsymbol{\theta}_i \approx \arg\min_{\boldsymbol{\theta} \in \Theta} \mathcal{L}(\boldsymbol{\theta})$$

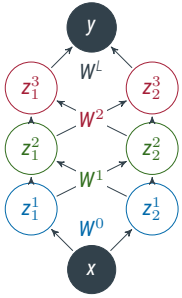
$$\boldsymbol{\theta}_i = \boldsymbol{\theta}_{i-1} + \alpha_i \boldsymbol{M}_i \boldsymbol{d}_i$$

Examples: natural / conjugate / stochastic gradient descent, (Quasi-) Newton method, …

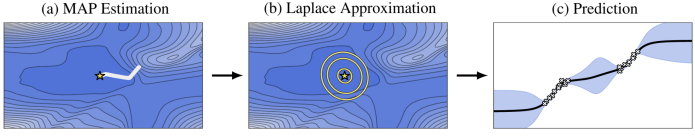


Feedforward Neural Network

$$
\begin{aligned}
z^0(x, \theta) &= x \\
z^{\ell+1}(x, \theta) &= \sigma(W^\ell z^\ell + b^\ell) \\
y \coloneqq f(x, \theta) &= z^L(x, \theta)
\end{aligned}
$$



Bayesian deep learning via Laplace approximation: $p(\theta \mid \mathcal{D}) \approx \mathcal{N}\left(\theta; \theta_{\text{MAP}}, (\nabla^2_\theta \mathcal{L}(\theta)|_{\theta_{\text{MAP}}})^{-1}\right)$



Daxberger et al. [Dax+22]

Probabilistic Linear Solvers

Learning the solution of a linear system.

[Hen15; Coc+19b; WH20]

Goal

Solve **large-scale** linear system $\boldsymbol{A}\boldsymbol{x}_* = \boldsymbol{b}$ for $\boldsymbol{x}_* \in \mathbb{R}^n$.

[Hen15; Coc+19b; WH20]

Goal

Solve **large-scale** linear system $\boldsymbol{A}\boldsymbol{x}_* = \boldsymbol{b}$ for $\boldsymbol{x}_* \in \mathbb{R}^n$.

Core Insights of Probabilistic Numerics

[HOG15; Coc+19a; HOK22]

- The solution to any numerical problem is fundamentally **uncertain**.

Probabilistic Linear Solvers for Machine Learning

Solving linear systems as probabilistic inference. [Hen15; Coc+19b; WH20]

Goal

Solve **large-scale** linear system $\boldsymbol{A}\boldsymbol{x}_* = \boldsymbol{b}$ for $\boldsymbol{x}_* \in \mathbb{R}^n$.

Core Insights of Probabilistic Numerics [HOG15; Coc+19a; HOK22]

- The solution to any numerical problem is fundamentally **uncertain**.
- Numerical algorithms are **learning agents**, which actively collect data and make predictions.

Probabilistic Linear Solvers for Machine Learning

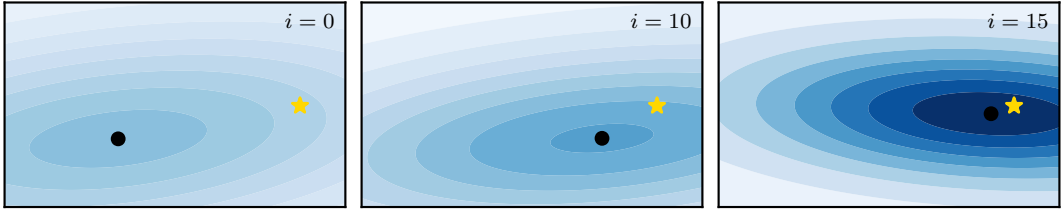
Solving linear systems as probabilistic inference. [Hen15; Coc+19b; WH20]

Goal

Solve **large-scale** linear system $Ax_* = b$ for $x_* \in \mathbb{R}^n$.

Core Insights of Probabilistic Numerics [HOG15; Coc+19a; HOK22]

- The solution to any numerical problem is fundamentally **uncertain**.
- Numerical algorithms are **learning agents**, which actively collect data and make predictions.

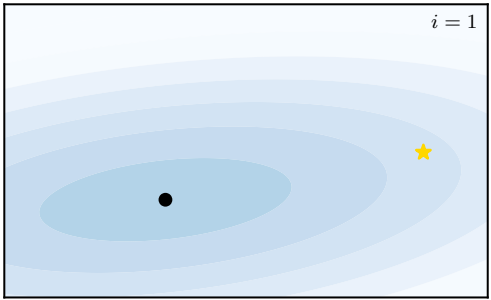


Learning The Solution

Estimating the solution of a linear system with a probabilistic linear solver. [Hen15; Coc+19b]

Goal: Solve $Ax_* = b$ for x_*.

Prior: $x_* \sim \mathcal{N}(x_0, \Sigma_0)$



Learning The Solution

Estimating the solution of a linear system with a probabilistic linear solver. [Hen15; Coc+19b]

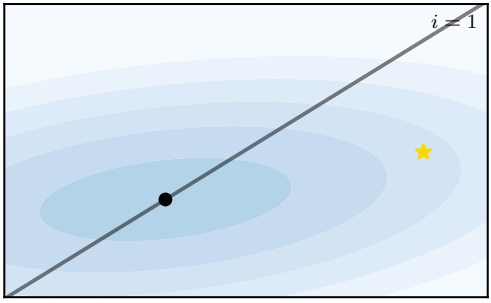
Goal: Solve $Ax_* = b$ for x_*.

Prior: $x_* \sim \mathcal{N}(x_0, \Sigma_0)$

Likelihood: Observe x_* via arbitrary actions s_i:

$$\alpha_i := s_i^\top A(x_* - x_{i-1}) = s_i^\top r_{i-1}$$

$$p(\alpha_i \mid x_*) = \lim_{\varepsilon \to 0} \mathcal{N}(\alpha_i; 0, \varepsilon)$$



Learning The Solution

Estimating the solution of a linear system with a probabilistic linear solver. [Hen15; Coc+19b]

Goal: Solve $Ax_* = b$ for x_*.

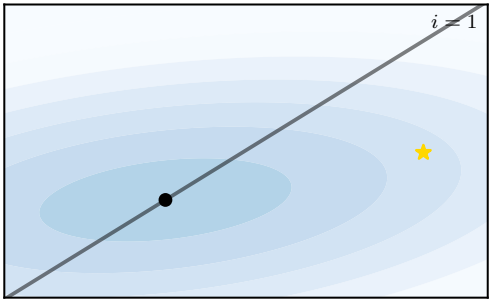
Prior: $x_* \sim \mathcal{N}(x_0, \Sigma_0)$

Likelihood: Observe x_* via arbitrary actions s_i:

$$\alpha_i := s_i^\top A(x_* - x_{i-1}) = s_i^\top r_{i-1}$$

$$p(\alpha_i \mid x_*) = \lim_{\varepsilon \to 0} \mathcal{N}(\alpha_i; 0, \varepsilon)$$

Posterior: Bayes' rule gives a closed form update!



Learning The Solution

Estimating the solution of a linear system with a probabilistic linear solver. [Hen15; Coc+19b]

Goal: Solve $Ax_* = b$ for x_*.

Prior: $x_* \sim \mathcal{N}(x_0, \Sigma_0)$

Likelihood: Observe x_* via arbitrary actions s_i:

$$\alpha_i := s_i^\intercal A(x_* - x_{i-1}) = s_i^\intercal r_{i-1}$$

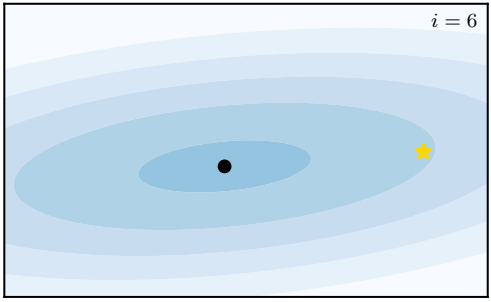
$$p(\alpha_i \mid x_*) = \lim_{\varepsilon \to 0} \mathcal{N}(\alpha_i; 0, \varepsilon)$$

Posterior:

$$x_* \mid \alpha_1, \dots, \alpha_i \sim \mathcal{N}(x_i, \Sigma_i)$$

$$x_i = x_0 + \Sigma_0 A S_i (S_i^\intercal A \Sigma_0 A S_i)^{-1} S_i^\intercal (b - A x_0)$$

$$\Sigma_i = \Sigma_0 - \Sigma_0 A S_i (S_i^\intercal A \Sigma_0 A S_i)^{-1} S_i^\intercal A \Sigma_0$$



Learning The Solution

Estimating the solution of a linear system with a probabilistic linear solver.

[Hen15; Coc+19b]

Goal: Solve $Ax_* = b$ for x_*.

Prior: $x_* \sim \mathcal{N}(x_0, \Sigma_0)$

Likelihood: Observe x_* via arbitrary actions s_i:

$$\alpha_i := s_i^\top A(x_* - x_{i-1}) = s_i^\top r_{i-1}$$

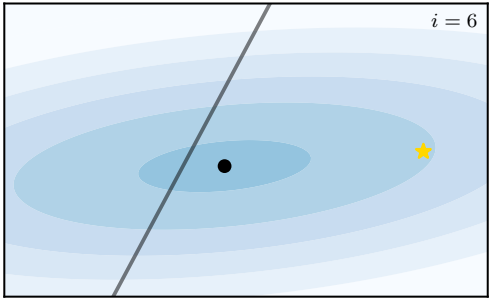
$$p(\alpha_i \mid x_*) = \lim_{\varepsilon \to 0} \mathcal{N}(\alpha_i; 0, \varepsilon)$$

Posterior:

$$x_* \mid \alpha_1, \ldots, \alpha_i \sim \mathcal{N}(x_i, \Sigma_i)$$

$$x_i = x_0 + \Sigma_0 A S_i (S_i^\top A \Sigma_0 A S_i)^{-1} S_i^\top (b - A x_0)$$

$$\Sigma_i = \Sigma_0 - \Sigma_0 A S_i (S_i^\top A \Sigma_0 A S_i)^{-1} S_i^\top A \Sigma_0$$



Learning The Solution

Estimating the solution of a linear system with a probabilistic linear solver. [Hen15; Coc+19b]

Goal: Solve $Ax_* = b$ for x_*.

Prior: $x_* \sim \mathcal{N}(x_0, \Sigma_0)$

Likelihood: Observe x_* via arbitrary actions s_i:

$$\alpha_i := s_i^\top A(x_* - x_{i-1}) = s_i^\top r_{i-1}$$

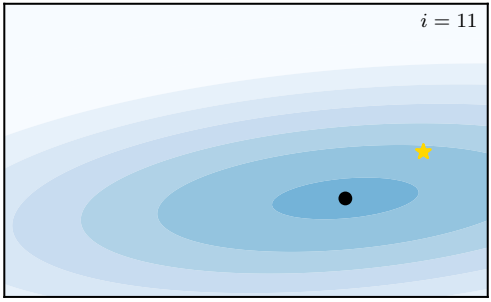
$$p(\alpha_i \mid x_*) = \lim_{\varepsilon \to 0} \mathcal{N}(\alpha_i; 0, \varepsilon)$$

Posterior:

$$x_* \mid \alpha_1, \dots, \alpha_i \sim \mathcal{N}(x_i, \Sigma_i)$$

$$x_i = x_0 + \Sigma_0 A S_i (S_i^\top A \Sigma_0 A S_i)^{-1} S_i^\top (b - A x_0)$$

$$\Sigma_i = \Sigma_0 - \Sigma_0 A S_i (S_i^\top A \Sigma_0 A S_i)^{-1} S_i^\top A \Sigma_0$$



Learning The Solution

Estimating the solution of a linear system with a probabilistic linear solver. [Hen15; Coc+19b]

Goal: Solve $Ax_* = b$ for x_*.

Prior: $x_* \sim \mathcal{N}(x_0, \Sigma_0)$

Likelihood: Observe x_* via arbitrary actions s_i:

$$\alpha_i := s_i^\intercal A(x_* - x_{i-1}) = s_i^\intercal r_{i-1}$$

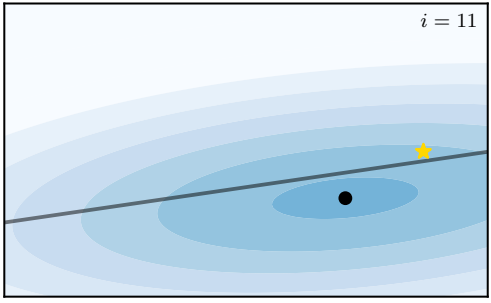
$$p(\alpha_i \mid x_*) = \lim_{\varepsilon \to 0} \mathcal{N}(\alpha_i; 0, \varepsilon)$$

Posterior:

$$x_* \mid \alpha_1, \dots, \alpha_i \sim \mathcal{N}(x_i, \Sigma_i)$$

$$x_i = x_0 + \Sigma_0 A S_i (S_i^\intercal A \Sigma_0 A S_i)^{-1} S_i^\intercal (b - A x_0)$$

$$\Sigma_i = \Sigma_0 - \Sigma_0 A S_i (S_i^\intercal A \Sigma_0 A S_i)^{-1} S_i^\intercal A \Sigma_0$$



Learning The Solution

Estimating the solution of a linear system with a probabilistic linear solver. [Hen15; Coc+19b]

Goal: Solve $Ax_* = b$ for x_*.

Prior: $x_* \sim \mathcal{N}(x_0, \Sigma_0)$

Likelihood: Observe x_* via arbitrary actions s_i:

$$\alpha_i := s_i^\top A(x_* - x_{i-1}) = s_i^\top r_{i-1}$$

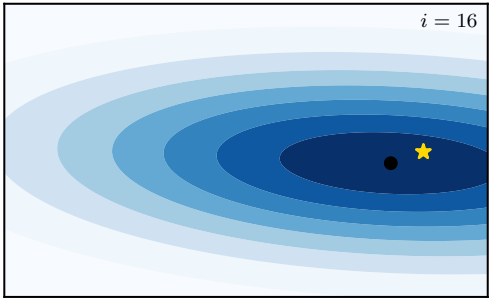
$$p(\alpha_i \mid x_*) = \lim_{\varepsilon \to 0} \mathcal{N}(\alpha_i; 0, \varepsilon)$$

Posterior:

$$x_* \mid \alpha_1, \dots, \alpha_i \sim \mathcal{N}(x_i, \Sigma_i)$$

$$x_i = x_0 + \Sigma_0 A S_i (S_i^\top A \Sigma_0 A S_i)^{-1} S_i^\top (b - A x_0)$$

$$\Sigma_i = \Sigma_0 - \Sigma_0 A S_i (S_i^\top A \Sigma_0 A S_i)^{-1} S_i^\top A \Sigma_0$$



Learning The Solution

Estimating the solution of a linear system with a probabilistic linear solver.

[Hen15; Coc+19b]

Goal: Solve $Ax_* = b$ for x_*.

Prior: $x_* \sim \mathcal{N}(x_0, \Sigma_0)$

Likelihood: Observe x_* via arbitrary actions s_i:

$$\alpha_i := s_i^\top A(x_* - x_{i-1}) = s_i^\top r_{i-1}$$

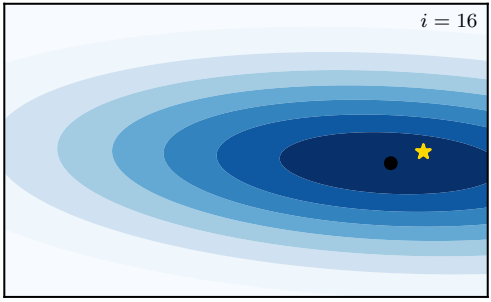
$$p(\alpha_i \mid x_*) = \lim_{\varepsilon \to 0} \mathcal{N}(\alpha_i; 0, \varepsilon)$$

Posterior:

$$x_* \mid \alpha_1, \dots, \alpha_i \sim \mathcal{N}(x_i, \Sigma_i)$$

$$x_i = x_0 + \Sigma_0 A S_i (S_i^\top A \Sigma_0 A S_i)^{-1} S_i^\top (b - A x_0)$$

$$\Sigma_i = \Sigma_0 - \Sigma_0 A S_i (S_i^\top A \Sigma_0 A S_i)^{-1} S_i^\top A \Sigma_0$$



How do we choose the linear solver **actions** S and the **prior** $\mathcal{N}(x_0, \Sigma_0)$?

Policy Choice

How do we choose the actions?

Observation: Actions "weigh" entries in the residual: $\alpha_i := s_i^\top r_{i-1} = s_i^\top A(x_* - x_{i-1})$

Policy Choice

How do we choose the actions?

Observation: Actions "weigh" entries in the residual: $\alpha_i := \boldsymbol{s}_i^\top \boldsymbol{r}_{i-1} = \boldsymbol{s}_i^\top \boldsymbol{A}(\boldsymbol{x}_* - \boldsymbol{x}_{i-1})$

Idea: Focus computation where residual is large: $\boldsymbol{s}_i = \boldsymbol{r}_{i-1} \implies \alpha_i = \|\boldsymbol{r}_{i-1}\|_2^2$

$\implies$ BayesCG [Coc+19b]

Interlude: Method of Conjugate Gradients

Efficiently solving linear systems with positive definite system matrix via matrix-vector multiplies.

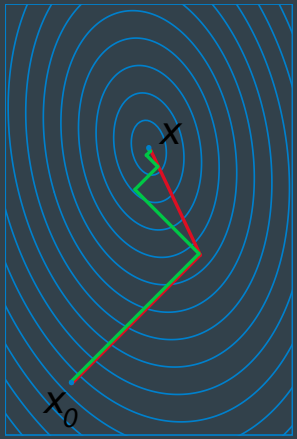
Goal: Approximately solve linear system $Ax_* = b$.

Idea: Rephrase as quadratic optimization problem and optimize. Let

$$f(x) = \frac{1}{2} x^\top A x - b^\top x$$

then $\nabla f(x_*) = 0 \iff Ax_* = b \iff r(x_*) := b - Ax_* = 0$.

Question: How should we optimize?



Oleg Alexandrov, commons.wikimedia.org/w/index.php?curid=2267598

Interlude: Method of Conjugate Gradients

Efficiently solving linear systems with positive definite system matrix via matrix-vector multiplies.

Goal: Approximately solve linear system $Ax_* = b$.

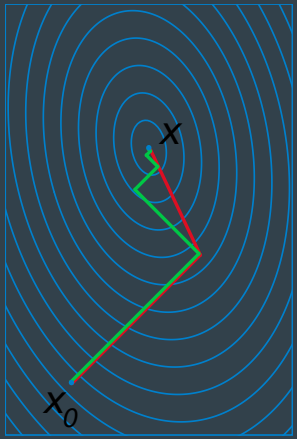
Idea: Rephrase as quadratic optimization problem and optimize. Let

$$f(x) = \frac{1}{2} x^\top A x - b^\top x$$

then $\nabla f(x_*) = 0 \iff Ax_* = b \iff r(x_*) := b - Ax_* = 0$.

Question: How should we optimize?

1. Gradient descent: Follow $d_i = r(x_i) = -\nabla f(x_i)$ s.t. $\langle d_i, d_j \rangle = 0$.



Oleg Alexandrov, commons.wikimedia.org/w/index.php?curid=2267598

Interlude: Method of Conjugate Gradients

Efficiently solving linear systems with positive definite system matrix via matrix-vector multiplies.

Goal: Approximately solve linear system $Ax_* = b$.

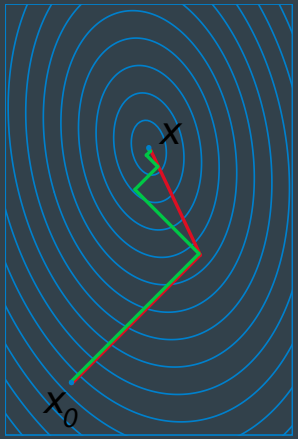
Idea: Rephrase as quadratic optimization problem and optimize. Let

$$f(x) = \frac{1}{2} x^\top A x - b^\top x$$

then $\nabla f(x_*) = 0 \iff Ax_* = b \iff r(x_*) := b - Ax_* = 0$.

Question: How should we optimize?

1. Gradient descent: Follow $d_i = r(x_i) = -\nabla f(x_i)$ s.t. $\langle d_i, d_j \rangle = 0$.
2. Conjugate direction method: Follow d_i s. t. $\langle d_j^\top d_j \rangle_A = d_i^\top A d_j = 0$ for $i \neq j$.
   $\implies$ convergence in at most n steps.



Oleg Alexandrov, commons.wikimedia.org/w/index.php?curid=2267598

Interlude: Method of Conjugate Gradients

Efficiently solving linear systems with positive definite system matrix via matrix-vector multiplies.

Goal: Approximately solve linear system $Ax_* = b$.

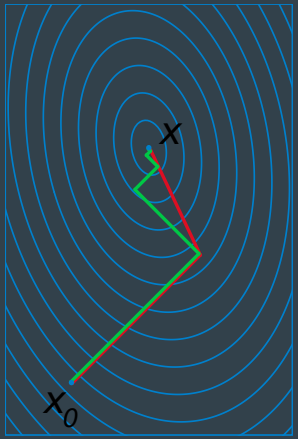
Idea: Rephrase as quadratic optimization problem and optimize. Let

$$f(x) = \frac{1}{2} x^\top A x - b^\top x$$

then $\nabla f(x_*) = 0 \iff Ax_* = b \iff r(x_*) := b - Ax_* = 0$.

Question: How should we optimize?

1. Gradient descent: Follow $d_i = r(x_i) = -\nabla f(x_i)$ s.t. $\langle d_i, d_j \rangle = 0$.
2. Conjugate direction method: Follow d_i s. t. $\langle d_i^\top d_j \rangle_A = d_i^\top A d_j = 0$ for $i \neq j$.
   $\implies$ convergence in at most n steps.
3. Conjugate gradient method: First step $d_0 = r(x_0)$.



Oleg Alexandrov, commons.wikimedia.org/w/index.php?curid=2267598

Policy Choice

How do we choose the actions?

Observation: Actions S_i "weigh" entries in the residual: $\alpha_i := s_i^\top r_{i-1} = s_i^\top A(x_* - x_{i-1})$

Idea: Focus computation where residual is large: $s_i = r_{i-1} \implies \alpha_i = \|r_{i-1}\|_2^2$

$\implies$ BayesCG [Coc+19b]

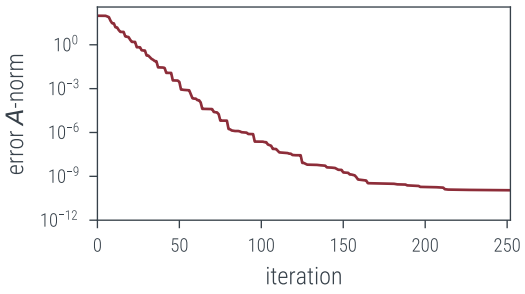
Theorem (Equivalence to Conjugate Gradient Method [Coc+19b; Wen+22])

If $x_0 = 0$, $\Sigma_0 = A^{-1}$ and the actions are either conjugate gradients $s_i = d_i^{CG}$ or gradients $s_i = r_{i-1}$, then the posterior mean $x_i = x_i^{CG}$ of BayesCG is equivalent to the approximation returned by CG.

Convergence Behavior of the Conjugate Gradient Method

The spectrum of the matrix determines the convergence speed.

$n = 10^3 \quad \kappa(A) \approx 7 \cdot 10^5$



Theorem (Convergence Rate of CG[TB97])

$$\|x - x_i\|_A \leq 2 \left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1} \right)^i \|x - x_0\|_A$$

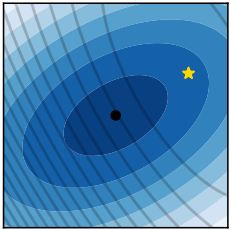
CG converges fast for a small condition number.

Prior Choice

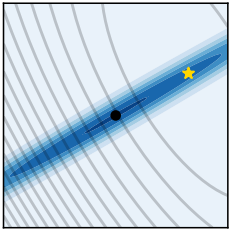
Comparing different choices of prior for BayesCG. [Coc+19b]

Prior

$$x_* \sim \mathcal{N}(x_0, \Sigma_0)$$



$\Rightarrow$

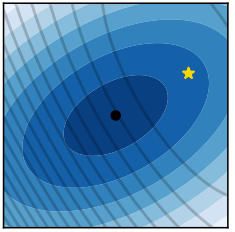


Prior Choice

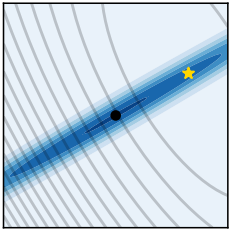
Comparing different choices of prior for BayesCG. [Coc+19b]

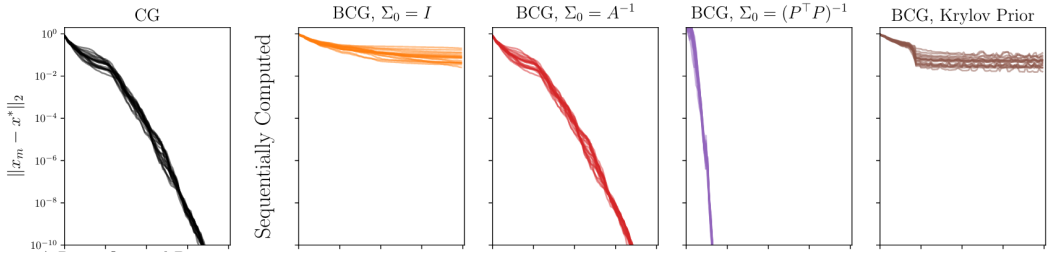
Prior

$$x_* \sim \mathcal{N}(x_0, \Sigma_0)$$



⇒





Algorithm: Probabilistic Linear Solver

Sequential formulation.

			Time	Space
1	**procedure** ProbabilisticLinearSolver($A, b, x_0 = \mathbf{0}, \mathbf{\Sigma}_0$)			
2	**while not** StoppingCriterion() **do**			
3	$s_i \leftarrow$ Policy()	Select action via policy.		
4	$r_{i-1} \leftarrow b - Ax_{i-1}$	Residual.	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
5	$\alpha_i \leftarrow s_i^\top r_{i-1}$	Observation.	$\mathcal{O}(n)$	$\mathcal{O}(1)$
6	$z_i \leftarrow As_i$		$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
7	$d_i \leftarrow \mathbf{\Sigma}_{i-1} As_i \; = \mathbf{\Sigma}_{i-1} z_i$	Search direction.	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
8	$\eta_i \leftarrow s_i^\top A \mathbf{\Sigma}_{i-1} As_i \; = z_i^\top d_i$		$\mathcal{O}(n)$	$\mathcal{O}(1)$
9	$C_i \leftarrow C_{i-1} + \frac{1}{\eta_i} d_i d_i^\top$		$\mathcal{O}(n)$	$\mathcal{O}(ni)$
10	$x_i \leftarrow x_{i-1} + \frac{\alpha_i}{\eta_i} d_i$	Solution estimate.	$\mathcal{O}(n)$	$\mathcal{O}(n)$
11	$\mathbf{\Sigma}_i \leftarrow \mathbf{\Sigma}_0 - C_i$	Uncertainty.		
12	**return** $\mathcal{N}(x_i, \mathbf{\Sigma}_i)$			

Application: Gaussian Processes

Scaling Gaussian processes via probabilistic linear solvers.

Gaussian Process Regression

Learning an unknown function from data.

Goal: Supervised learning from n data points $(\boldsymbol{X}, \boldsymbol{y})$

Prior: Gaussian process $f \sim \mathcal{GP}(\mu, k)$

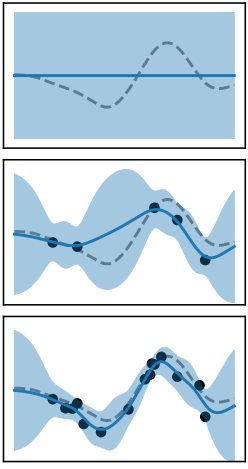
Likelihood: Observations $\boldsymbol{y} = f(\boldsymbol{X}) + \boldsymbol{\varepsilon} \sim \mathcal{N}\left(f(\boldsymbol{X}), \sigma^2 \boldsymbol{I}\right)$

Posterior: $f \mid \boldsymbol{X}, \boldsymbol{y} \sim \mathcal{GP}(\mu_*, k_*)$ with

$$\mu_*(\cdot) = \mu(\cdot) + K(\cdot, \boldsymbol{X})\hat{\boldsymbol{K}}^{-1}(\boldsymbol{y} - \mu(\boldsymbol{X}))$$

$$K_*(\cdot, \cdot) = K(\cdot, \cdot) - K(\cdot, \boldsymbol{X})\hat{\boldsymbol{K}}^{-1}K(\boldsymbol{X}, \cdot)$$

where $\hat{\boldsymbol{K}} = \boldsymbol{K} + \sigma^2 \boldsymbol{I} \in \mathbb{R}^{n \times n}$.



Gaussian Process Regression

Learning an unknown function from data.

Goal: Supervised learning from n data points $(\boldsymbol{X}, \boldsymbol{y})$

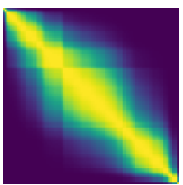
Prior: Gaussian process $f \sim \mathcal{GP}(\mu, k)$

Likelihood: Observations $\boldsymbol{y} = f(\boldsymbol{X}) + \boldsymbol{\varepsilon} \sim \mathcal{N}\left(f(\boldsymbol{X}), \sigma^2 \boldsymbol{I}\right)$

Posterior: $f \mid \boldsymbol{X}, \boldsymbol{y} \sim \mathcal{GP}(\mu_*, k_*)$ with

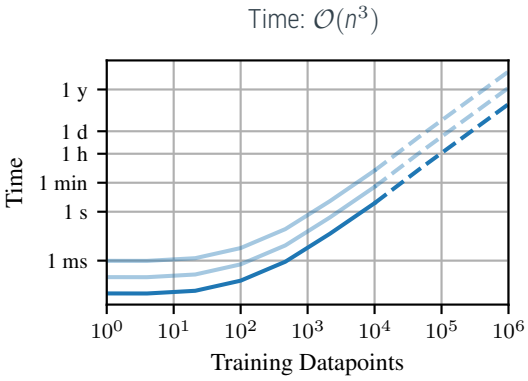
$$\mu_*(\cdot) = \mu(\cdot) + K(\cdot, \boldsymbol{X})\hat{\boldsymbol{K}}^{-1}(\boldsymbol{y} - \mu(\boldsymbol{X}))$$
$$K_*(\cdot, \cdot) = K(\cdot, \cdot) - K(\cdot, \boldsymbol{X})\hat{\boldsymbol{K}}^{-1}K(\boldsymbol{X}, \cdot)$$

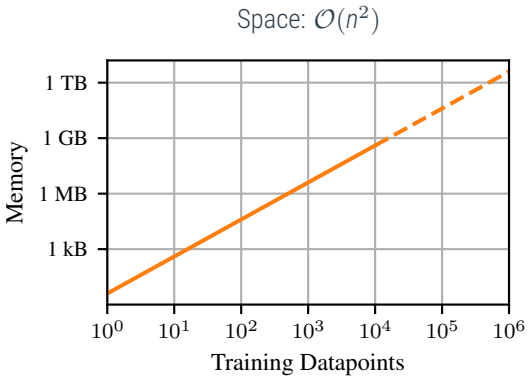
where $\hat{\boldsymbol{K}} = \boldsymbol{K} + \sigma^2 \boldsymbol{I} \in \mathbb{R}^{n \times n}$.

$\hat{\boldsymbol{K}} =$ 

Computational Cost of Gaussian Processes

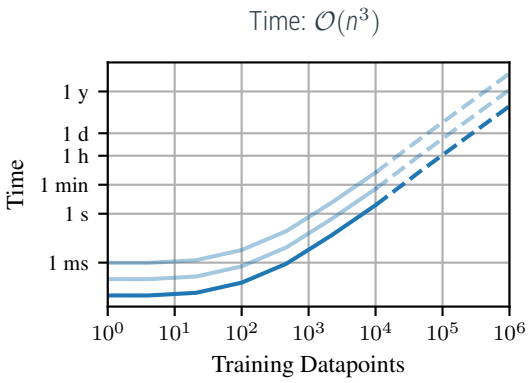
Gaussian processes scale prohibitively with the size n of the dataset.

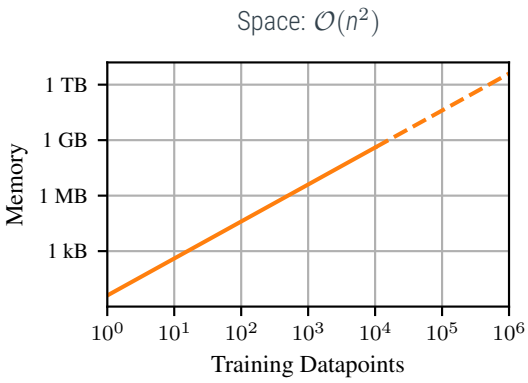




Computational Cost of Gaussian Processes

Gaussian processes scale prohibitively with the size n of the dataset.

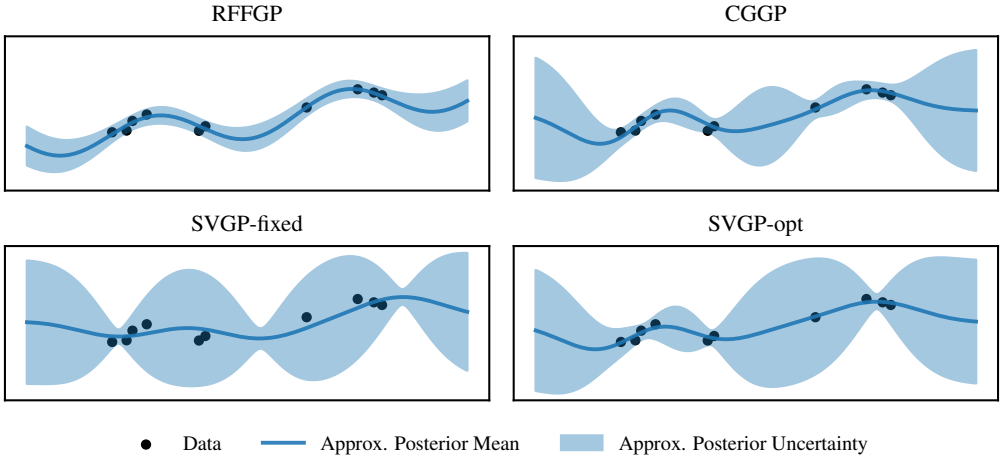




We need to **approximate** the posterior.

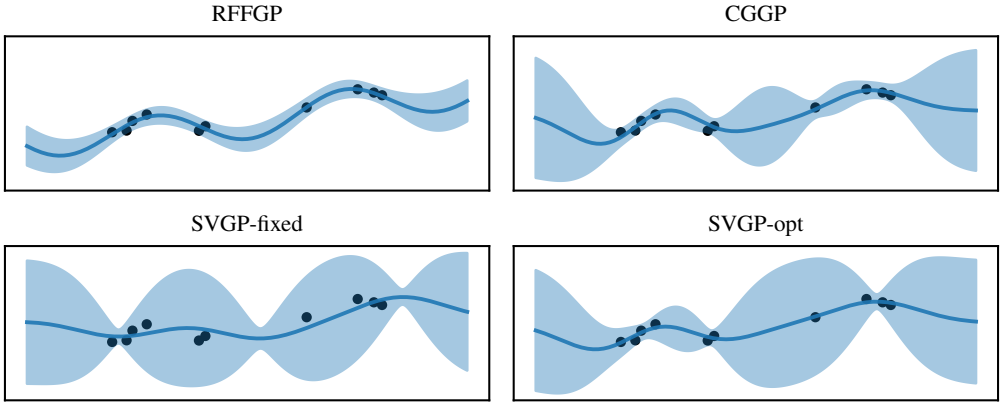
Approximate Gaussian Process Inference

Impact of approximations on uncertainty quantification and decision-making.



Approximate Gaussian Process Inference

Impact of approximations on uncertainty quantification and decision-making.



Approximations introduce **error**, which **impacts downstream decisions**.

Fundamental Questions

Question 1:

How can we perform **Gaussian process inference at scale**?

Fundamental Questions

Question 1:

How can we perform **Gaussian process inference at scale**?

Question 2:

How can we **quantify** the inevitable **approximation error**?

Q1: Gaussian Process Inference at Scale?

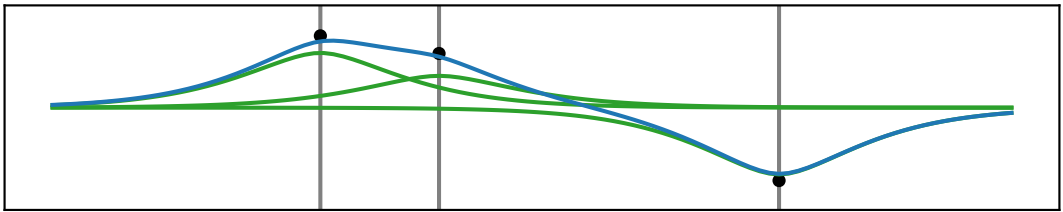
Efficiently approximating the posterior of a Gaussian process.

Representer Weights

The posterior mean is a linear combination of kernel functions centered at data points.

$$f \mid X, y \sim \mathcal{GP}(\mu_*, k_*)$$

$$\mu_*(\cdot) = \mu(\cdot) + k(\cdot, X) \underbrace{\hat{K}^{-1}(y - \mu(X))}_{\text{representer weights } v_*} = \mu(\cdot) + \sum_{j=1}^{n} k(\cdot, x_j)(v_*)_j$$

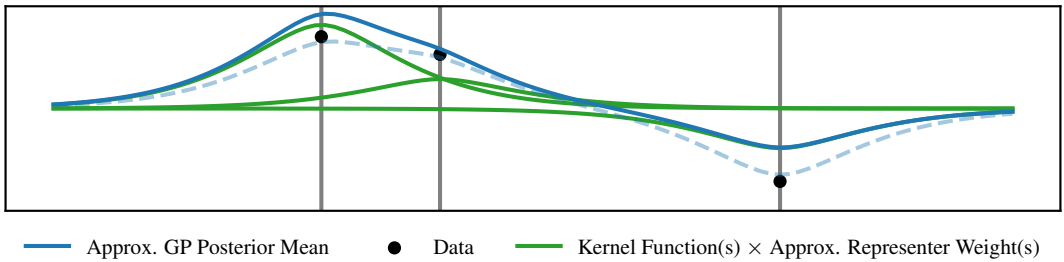


Approximating Representer Weights

Iterative linear solvers can be used to approximate the representer weights.

$$\mu_*(\cdot) = \mu(\cdot) + k(\cdot, X)\underbrace{\hat{K}^{-1}(y - \mu(X))}_{\text{representer weights } v_*} \approx \mu(\cdot) + k(\cdot, X)v_i$$

Known: Can use iterative linear solvers (e.g. CG) to approximate representer weights $v_* \approx v_i$.

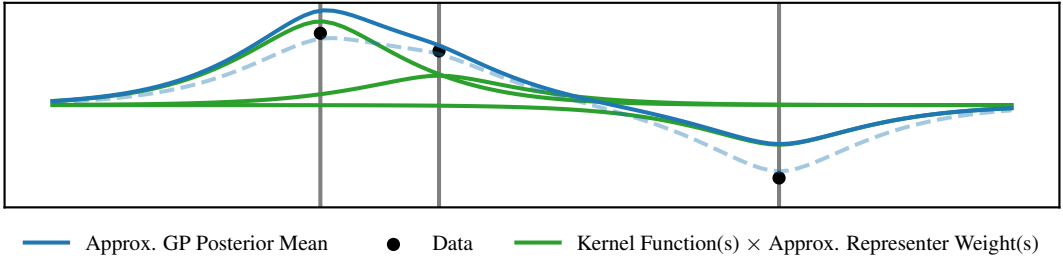


Approximating Representer Weights

Iterative linear solvers can be used to approximate the representer weights.

$$\mu_*(\cdot) = \mu(\cdot) + k(\cdot, X)\underbrace{\hat{K}^{-1}(y - \mu(X))}_{\text{representer weights } v_*} \approx \mu(\cdot) + k(\cdot, X)v_i$$

Known: Can use iterative linear solvers (e.g. CG) to approximate representer weights $v_* \approx v_i$.



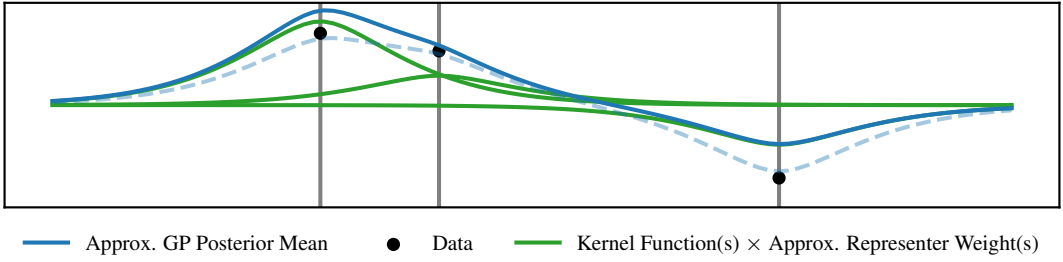
Benefit: Time complexity $\mathcal{O}(n^2)$ and space complexity $\mathcal{O}(nd)$.

Approximating Representer Weights

Iterative linear solvers can be used to approximate the representer weights.

$$\mu_*(\cdot) = \mu(\cdot) + k(\cdot, X)\underbrace{\hat{K}^{-1}(y - \mu(X))}_{\text{representer weights } v_*} \approx \mu(\cdot) + k(\cdot, X)v_i$$

Known: Can use iterative linear solvers (e.g. CG) to approximate representer weights $v_* \approx v_i$.



Problem: Approximation error of the linear solve.

Q2: Can We Quantify Approximation Error?

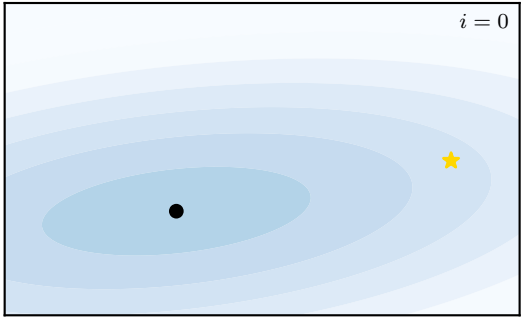
Probabilistic error quantification at prediction time using probabilistic linear solvers.

Linear Solver Prior for GP Inference

The Gaussian process prior makes assumptions about the representer weights. [Wen+22]

Observation:
GP prior induces representer weights prior:

$$y - \mu \sim \mathcal{N}\left(0, \hat{K}\right)$$



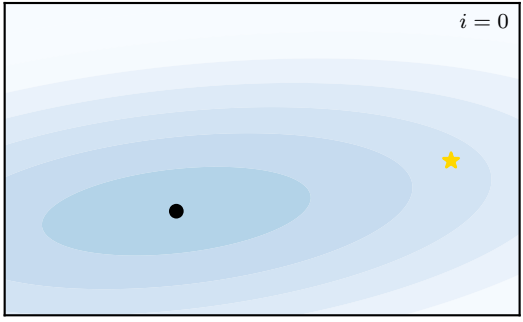
Linear Solver Prior for GP Inference

The Gaussian process prior makes assumptions about the representer weights. [Wen+22]

Observation:
GP prior induces representer weights prior:

$$y - \mu \sim \mathcal{N}\left(\mathbf{0}, \hat{K}\right)$$

$$\Rightarrow v_* = \hat{K}^{-1}(y - \mu) \sim \mathcal{N}\left(\underbrace{\mathbf{0}}_{=v_0}, \underbrace{\hat{K}^{-1}}_{=\Sigma_0}\right)$$



Linear Solver Posterior for GP Inference

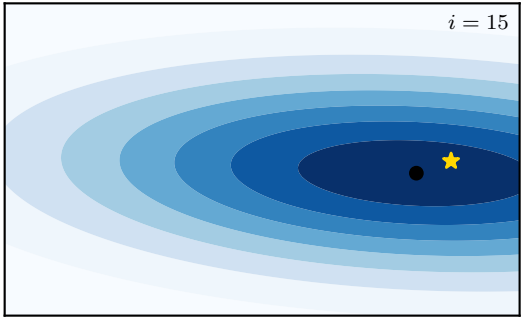
Estimation of representer weights with a probabilistic linear solver.

[Wen+22]

Representer weights posterior $v_* \sim \mathcal{N}(v_i, \Sigma_i)$, s.t.

$$v_i = C_i(y - \mu)$$
$$\Sigma_i = \hat{K}^{-1} - C_i$$

$$\hat{K}^{-1} = \approx = C_i$$



Linear Solver Posterior for GP Inference

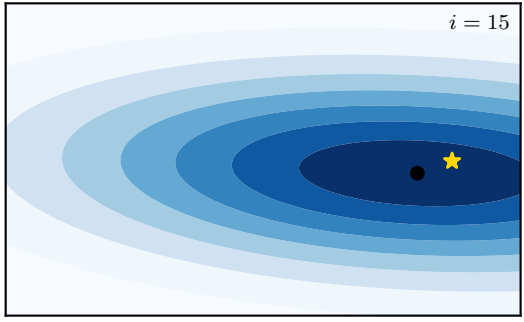
Estimation of representer weights with a probabilistic linear solver.

[Wen+22]

Representer weights posterior $v_* \sim \mathcal{N}(v_i, \Sigma_i)$, s.t.

$$v_i = C_i(y - \mu)$$
$$\Sigma_i = \hat{K}^{-1} - C_i$$

$$\hat{K}^{-1} = \approx = C_i$$



Chicken & Egg Problem: How can we get a probabilistic error estimate for $v_i \approx v_*$, if we need $\hat{K}^{-1}$?

IterGP: Computation-Aware Gaussian Process Inference

Quantifying uncertainty arising from observing finite data and performing a finite amount of computation. [Wen+22]

Goal: Approximate the Gaussian process posterior $f \mid \boldsymbol{y} \sim \mathcal{GP}(\mu_*, k_*)$.

IterGP: Computation-Aware Gaussian Process Inference

Quantifying uncertainty arising from observing finite data and performing a finite amount of computation. [Wen+22]

Goal: Approximate the Gaussian process posterior $f \mid \boldsymbol{y} \sim \mathcal{GP}(\mu_*, k_*)$.

Obtained: Belief about representer weights $\boldsymbol{v}_* = \hat{\boldsymbol{K}}^{-1}(\boldsymbol{y} - \boldsymbol{\mu}) \sim \mathcal{N}(\boldsymbol{v}_i, \boldsymbol{\Sigma}_i) = \mathcal{N}\left(\boldsymbol{v}_i, \hat{\boldsymbol{K}}^{-1} - \boldsymbol{C}_i\right)$

IterGP: Computation-Aware Gaussian Process Inference

Quantifying uncertainty arising from observing finite data and performing a finite amount of computation. [Wen+22]

Goal: Approximate the Gaussian process posterior $f \mid \boldsymbol{y} \sim \mathcal{GP}(\mu_*, k_*)$.

Obtained: Belief about representer weights $\boldsymbol{v}_* = \hat{\boldsymbol{K}}^{-1}(\boldsymbol{y} - \boldsymbol{\mu}) \sim \mathcal{N}(\boldsymbol{v}_i, \boldsymbol{\Sigma}_i) = \mathcal{N}\left(\boldsymbol{v}_i, \hat{\boldsymbol{K}}^{-1} - \boldsymbol{C}_i\right)$

Idea: Propagate uncertainty about representer weights to posterior.

IterGP: Computation-Aware Gaussian Process Inference

Quantifying uncertainty arising from observing finite data and performing a finite amount of computation. [Wen+22]

Goal: Approximate the Gaussian process posterior $f \mid \boldsymbol{y} \sim \mathcal{GP}(\mu_*, k_*)$.

Obtained: Belief about representer weights $\boldsymbol{v}_* = \hat{\boldsymbol{K}}^{-1}(\boldsymbol{y} - \boldsymbol{\mu}) \sim \mathcal{N}(\boldsymbol{v}_i, \boldsymbol{\Sigma}_i) = \mathcal{N}\left(\boldsymbol{v}_i, \hat{\boldsymbol{K}}^{-1} - \boldsymbol{C}_i\right)$

Idea: Propagate uncertainty about representer weights to posterior.

1. Pathwise form of posterior: $(f \mid \boldsymbol{y})(\cdot) \overset{d}{=} f(\cdot) + k(\cdot, \boldsymbol{X}) \underbrace{\hat{\boldsymbol{K}}^{-1}(\boldsymbol{y} - \boldsymbol{\mu})}_{=\boldsymbol{v}_*} \overset{d}{=} (f \mid \boldsymbol{v}_*)(\cdot)$

IterGP: Computation-Aware Gaussian Process Inference

Quantifying uncertainty arising from observing finite data and performing a finite amount of computation. [Wen+22]

Goal: Approximate the Gaussian process posterior $f \mid \boldsymbol{y} \sim \mathcal{GP}(\mu_*, k_*)$.

Obtained: Belief about representer weights $\boldsymbol{v}_* = \hat{\boldsymbol{K}}^{-1}(\boldsymbol{y} - \boldsymbol{\mu}) \sim \mathcal{N}(\boldsymbol{v}_i, \boldsymbol{\Sigma}_i) = \mathcal{N}\left(\boldsymbol{v}_i, \hat{\boldsymbol{K}}^{-1} - \boldsymbol{C}_i\right)$

Idea: Propagate uncertainty about representer weights to posterior.

1. Pathwise form of posterior: $(f \mid \boldsymbol{y})(\cdot) \overset{d}{=} f(\cdot) + k(\cdot, \boldsymbol{X}) \underbrace{\hat{\boldsymbol{K}}^{-1}(\boldsymbol{y} - \boldsymbol{\mu})}_{=\boldsymbol{v}_*} \overset{d}{=} (f \mid \boldsymbol{v}_*)(\cdot)$

IterGP: Computation-Aware Gaussian Process Inference

Quantifying uncertainty arising from observing finite data and performing a finite amount of computation. [Wen+22]

Goal: Approximate the Gaussian process posterior $f \mid \boldsymbol{y} \sim \mathcal{GP}(\mu_*, k_*)$.

Obtained: Belief about representer weights $\boldsymbol{v}_* = \hat{\boldsymbol{K}}^{-1}(\boldsymbol{y} - \boldsymbol{\mu}) \sim \mathcal{N}(\boldsymbol{v}_i, \boldsymbol{\Sigma}_i) = \mathcal{N}\left(\boldsymbol{v}_i, \hat{\boldsymbol{K}}^{-1} - \boldsymbol{C}_i\right)$

Idea: Propagate uncertainty about representer weights to posterior.

1. Pathwise form of posterior: $(f \mid \boldsymbol{y})(\cdot) \overset{d}{=} f(\cdot) + k(\cdot, \boldsymbol{X}) \underbrace{\hat{\boldsymbol{K}}^{-1}(\boldsymbol{y} - \boldsymbol{\mu})}_{=\boldsymbol{v}_*} \overset{d}{=} (f \mid \boldsymbol{v}_*)(\cdot)$
2. Marginalize representer weights belief: $p(f(\cdot)) = \int p(f(\cdot) \mid \boldsymbol{v}_*) p(\boldsymbol{v}_*)\, d\boldsymbol{v}_* = \mathcal{GP}(f; \mu_i, k_i)$

IterGP: Computation-Aware Gaussian Process Inference

Quantifying uncertainty arising from observing finite data and performing a finite amount of computation. [Wen+22]

Goal: Approximate the Gaussian process posterior $f \mid \boldsymbol{y} \sim \mathcal{GP}(\mu_*, k_*)$.

Obtained: Belief about representer weights $\boldsymbol{v}_* = \hat{\boldsymbol{K}}^{-1}(\boldsymbol{y} - \boldsymbol{\mu}) \sim \mathcal{N}(\boldsymbol{v}_i, \boldsymbol{\Sigma}_i) = \mathcal{N}\left(\boldsymbol{v}_i, \hat{\boldsymbol{K}}^{-1} - \boldsymbol{C}_i\right)$

Idea: Propagate uncertainty about representer weights to posterior.

1. Pathwise form of posterior: $(f \mid \boldsymbol{y})(\cdot) \overset{d}{=} f(\cdot) + k(\cdot, \boldsymbol{X}) \underbrace{\hat{\boldsymbol{K}}^{-1}(\boldsymbol{y} - \boldsymbol{\mu})}_{=\boldsymbol{v}_*} \overset{d}{=} (f \mid \boldsymbol{v}_*)(\cdot)$
2. Marginalize representer weights belief: $p(f(\cdot)) = \int p(f(\cdot) \mid \boldsymbol{v}_*) p(\boldsymbol{v}_*)\, d\boldsymbol{v}_* = \mathcal{GP}(f; \mu_i, k_i)$

$$\mu_i(\cdot) = \mu(\cdot) + K(\cdot, \boldsymbol{X})\boldsymbol{v}_i$$

IterGP: Computation-Aware Gaussian Process Inference

Quantifying uncertainty arising from observing finite data and performing a finite amount of computation. [Wen+22]

Goal: Approximate the Gaussian process posterior $f \mid \boldsymbol{y} \sim \mathcal{GP}(\mu_*, k_*)$.

Obtained: Belief about representer weights $\boldsymbol{v}_* = \hat{\boldsymbol{K}}^{-1}(\boldsymbol{y} - \boldsymbol{\mu}) \sim \mathcal{N}(\boldsymbol{v}_i, \boldsymbol{\Sigma}_i) = \mathcal{N}\left(\boldsymbol{v}_i, \hat{\boldsymbol{K}}^{-1} - \boldsymbol{C}_i\right)$

Idea: Propagate uncertainty about representer weights to posterior.

1. Pathwise form of posterior: $(f \mid \boldsymbol{y})(\cdot) \overset{d}{=} f(\cdot) + k(\cdot, \boldsymbol{X}) \underbrace{\hat{\boldsymbol{K}}^{-1}(\boldsymbol{y} - \boldsymbol{\mu})}_{=\boldsymbol{v}_*} \overset{d}{=} (f \mid \boldsymbol{v}_*)(\cdot)$
2. Marginalize representer weights belief: $p(f(\cdot)) = \int p(f(\cdot) \mid \boldsymbol{v}_*) p(\boldsymbol{v}_*)\, d\boldsymbol{v}_* = \mathcal{GP}(f; \mu_i, k_i)$

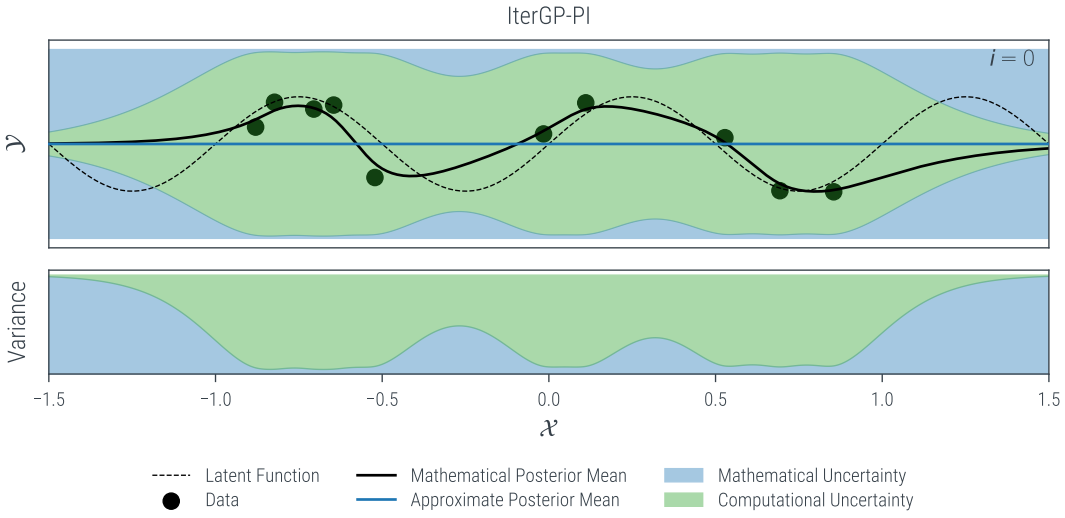
$$\mu_i(\cdot) = \mu(\cdot) + K(\cdot, \boldsymbol{X})\boldsymbol{v}_i$$

$$k_i(\cdot,\cdot) = \underbrace{K(\cdot,\cdot) - K(\cdot,\boldsymbol{X})\hat{\boldsymbol{K}}^{-1}K(\boldsymbol{X},\cdot)}_{=\mathbb{E}\left((f(\cdot)-\mu_*(\cdot))^2\right)} + \underbrace{K(\cdot,\boldsymbol{X})\boldsymbol{\Sigma}_i K(\boldsymbol{X},\cdot)}_{=\mathbb{E}\left((\mu_*(\cdot)-\mu_i(\cdot))^2\right)} = \underbrace{K(\cdot,\cdot) - K(\cdot,\boldsymbol{X})\boldsymbol{C}_i K(\boldsymbol{X},\cdot)}_{=\mathbb{E}\left((f(\cdot)-\mu_i(\cdot))^2\right)}$$

mathematical uncertainty — computational uncertainty — combined uncertainty

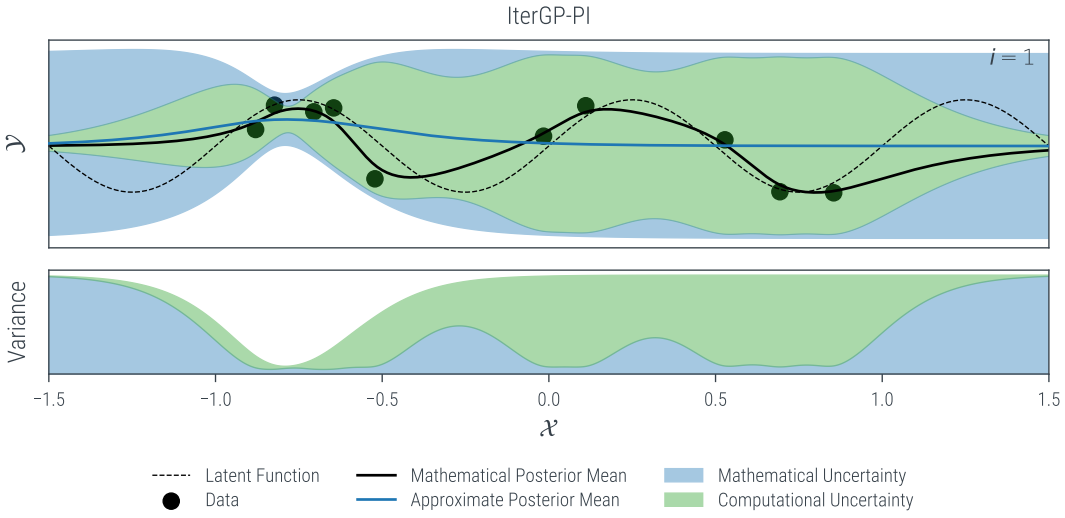
Computation-Aware GP Inference Illustrated

Interpreting computational and combined uncertainty as error quantification.



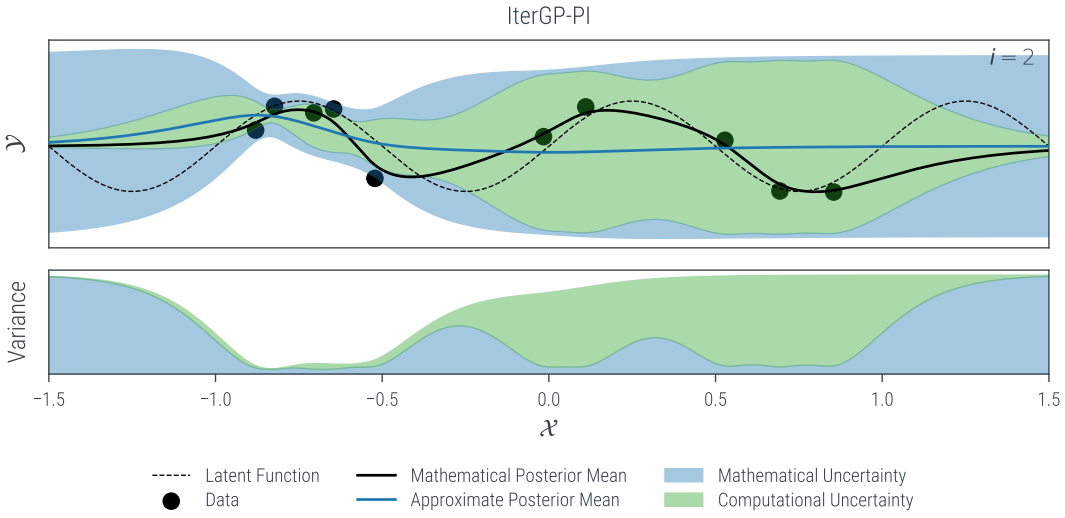
Computation-Aware GP Inference Illustrated

Interpreting computational and combined uncertainty as error quantification.



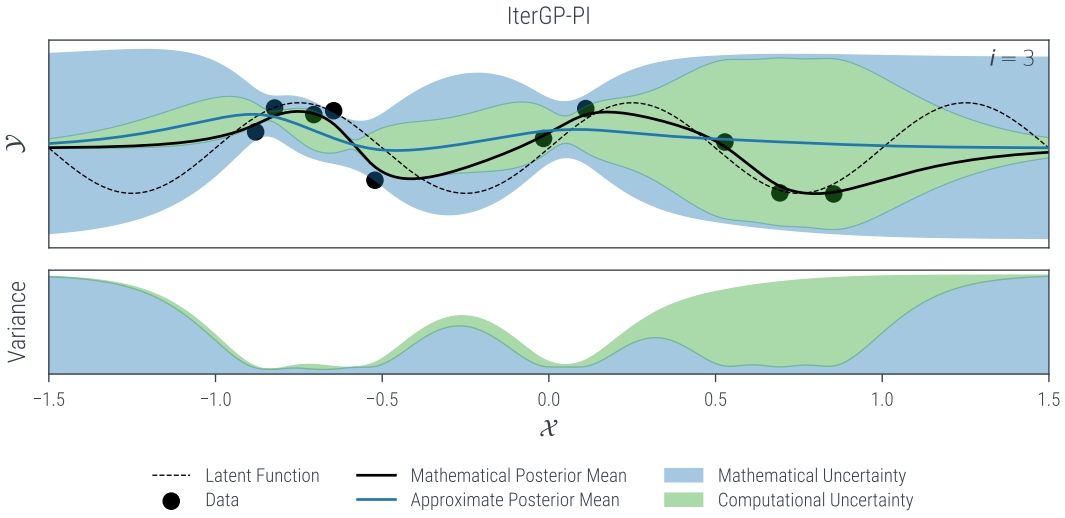
Computation-Aware GP Inference Illustrated

Interpreting computational and combined uncertainty as error quantification.



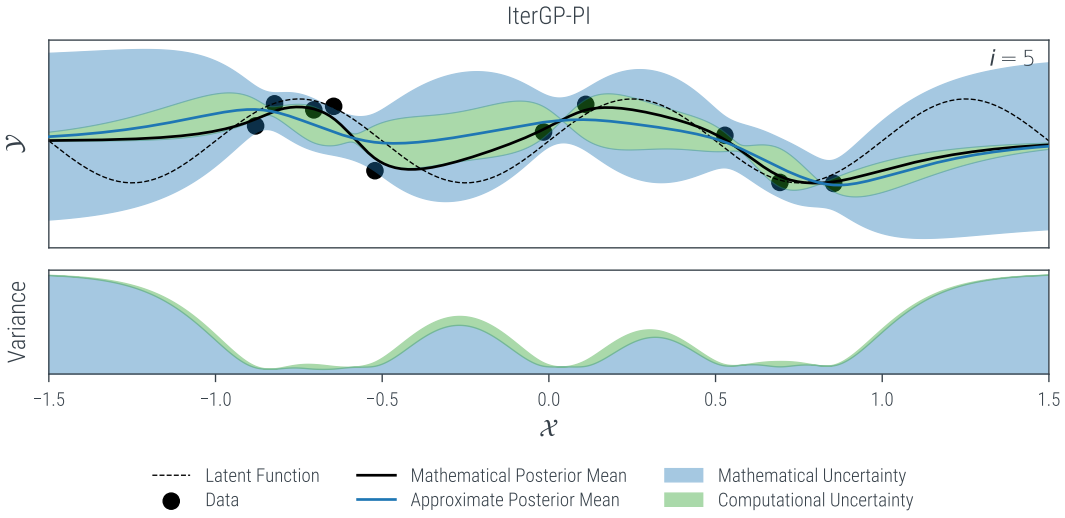
Computation-Aware GP Inference Illustrated

Interpreting computational and combined uncertainty as error quantification.



Computation-Aware GP Inference Illustrated

Interpreting computational and combined uncertainty as error quantification.



Algorithm: IterGP

			Time	Space
1	**procedure** IterGP($\mu, K, X, y, C_0 = \mathbf{0}$)			
2	**while not** StoppingCriterion() **do**			
3	$s_i \leftarrow$ Policy()	Select action via policy.		
4	$r_{i-1} \leftarrow (y - \mu) - \hat{K} v_{i-1}$	Residual.	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
5	$\alpha_i \leftarrow s_i^\top r_{i-1}$	Observation.	$\mathcal{O}(n)$	$\mathcal{O}(1)$
6	$z_i \leftarrow \hat{K} s_i$		$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
7	$d_i \leftarrow \Sigma_{i-1} \hat{K} s_i = s_i - C_{i-1} z_i$	Search direction.	$\mathcal{O}(ni)$	$\mathcal{O}(n)$
8	$\eta_i \leftarrow s_i^\top \hat{K} \Sigma_{i-1} \hat{K} s_i = z_i^\top d_i$		$\mathcal{O}(n)$	$\mathcal{O}(1)$
9	$C_i \leftarrow C_{i-1} + \frac{1}{\eta_i} d_i d_i^\top$	Precision matrix approx. $C_i \approx \hat{K}^{-1}$.	$\mathcal{O}(n)$	$\mathcal{O}(ni)$
10	$v_i \leftarrow v_{i-1} + \frac{\alpha_i}{\eta_i} d_i$	Representer weights estimate.	$\mathcal{O}(n)$	$\mathcal{O}(n)$
11	$\Sigma_i \leftarrow \Sigma_0 - C_i$	Representer weights uncertainty.		
12	$\mu_i(\cdot) \leftarrow \mu(\cdot) + K(\cdot, X) v_i$	Approximate posterior mean.	$\mathcal{O}(n_\diamond n)$	$\mathcal{O}(n_\diamond)$
13	$K_i(\cdot, \cdot) \leftarrow K(\cdot, \cdot) - K(\cdot, X) C_i K(X, \cdot)$	Combined covariance function.	$\mathcal{O}(n_\diamond n i)$	$\mathcal{O}(n_\diamond^2)$
14	**return** $\mathcal{GP}(\mu_i, K_i)$			

Theoretical Analysis

Uncertainty as a tight bound on the relative error.

Theorem (Kanagawa et al. [Kan+18])

$$\sup_{g \in \mathcal{H}_{k^\sigma} : \|g\|_{\mathcal{H}_{k^\sigma}} \leq 1} \underbrace{g(x) - \mu_*^g(x)}_{\text{error of math. post. mean}} = \sup_{g \in \mathcal{H}_{k^\sigma}} \frac{|g(x) - \mu_*^g(x)|}{\|g\|_{\mathcal{H}_{k^\sigma}}} = \sqrt{k_*(x,x) + \sigma^2}$$

Theoretical Analysis

Uncertainty as a tight bound on the relative error.

Theorem (Kanagawa et al. [Kan+18])

$$\sup_{g \in \mathcal{H}_{k^\sigma} : \|g\|_{\mathcal{H}_{k^\sigma}} \leq 1} \underbrace{g(x) - \mu_*^g(x)}_{\text{error of math. post. mean}} = \sup_{g \in \mathcal{H}_{k^\sigma}} \frac{|g(x) - \mu_*^g(x)|}{\|g\|_{\mathcal{H}_{k^\sigma}}} = \sqrt{k_*(x,x) + \sigma^2}$$

Theorem (Wenger et al. [Wen+22])

$$\sup_{g \in \mathcal{H}_{k^\sigma} : \|g\|_{\mathcal{H}_{k^\sigma}} \leq 1} \overbrace{\underbrace{g(x) - \mu_*^g(x)}_{\text{error of math. post. mean}} + \underbrace{\mu_*^g(x) - \mu_i^g(x)}_{\text{computational error}}}^{\text{error of approximate posterior mean} + } = \sqrt{k_i(x,x) + \sigma^2}$$

Theoretical Analysis

Uncertainty as a tight bound on the relative error.

Theorem (Kanagawa et al. [Kan+18])

$$\sup_{g \in \mathcal{H}_{k^\sigma} : \|g\|_{\mathcal{H}_{k^\sigma}} \leq 1} \underbrace{g(x) - \mu_*^g(x)}_{\text{error of math. post. mean}} = \sup_{g \in \mathcal{H}_{k^\sigma}} \frac{|g(x) - \mu_*^g(x)|}{\|g\|_{\mathcal{H}_{k^\sigma}}} = \sqrt{k_*(x,x) + \sigma^2}$$

Theorem (Wenger et al. [Wen+22])

$$\sup_{g \in \mathcal{H}_{k^\sigma} : \|g\|_{\mathcal{H}_{k^\sigma}} \leq 1} \overbrace{\underbrace{g(x) - \mu_*^g(x)}_{\text{error of math. post. mean}} + \underbrace{\mu_*^g(x) - \mu_i^g(x)}_{\text{computational error}}}^{\text{error of approximate posterior mean} + } = \sqrt{k_i(x,x) + \sigma^2}$$

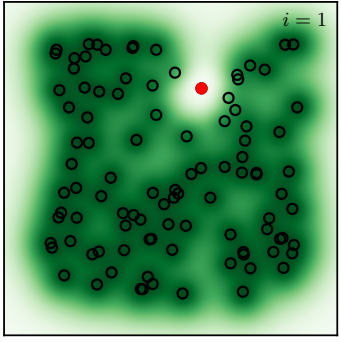
Exact quantification of uncertainty from **limited data** and **limited computation**.

Policy Choice and Connection to Other Approximations

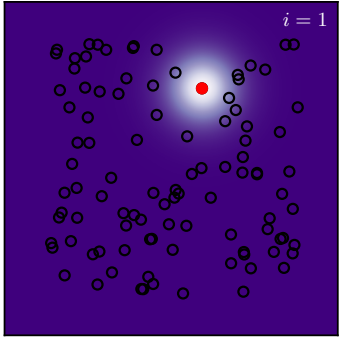
IterGP extends the most commonly used GP approximations to include computational uncertainty, with at most quadratic cost.

	Actions s_i	Classic Analog
IterGP-Cholesky	e_i	(Partial) Cholesky / Subset of data
IterGP-CG	$\hat{P}^{-1} r_i$	(Preconditioned) CG
IterGP-PseudoInput	$k(X, z_i)$	$\approx$ SVGP

Computational Uncertainty



Combined Uncertainty

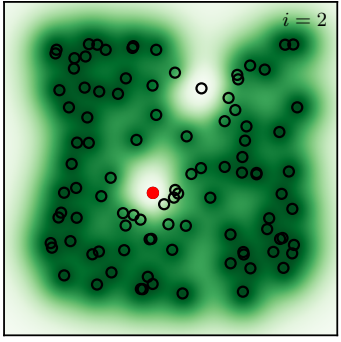


Policy Choice and Connection to Other Approximations

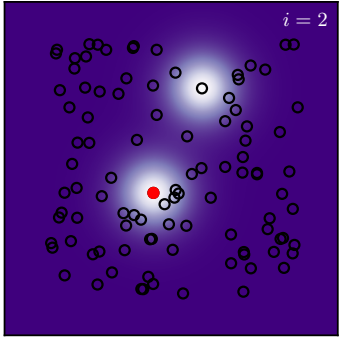
IterGP extends the most commonly used GP approximations to include computational uncertainty, with at most quadratic cost.

	Actions s_i	Classic Analog
IterGP-Cholesky	e_i	(Partial) Cholesky / Subset of data
IterGP-CG	$\hat{P}^{-1} r_i$	(Preconditioned) CG
IterGP-PseudoInput	$k(X, z_i)$	$\approx$ SVGP

Computational Uncertainty



Combined Uncertainty

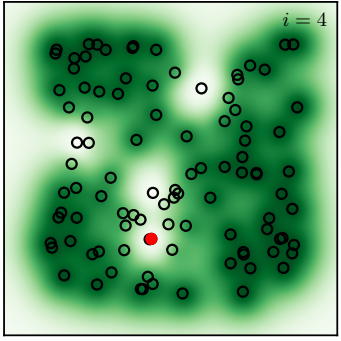


Policy Choice and Connection to Other Approximations

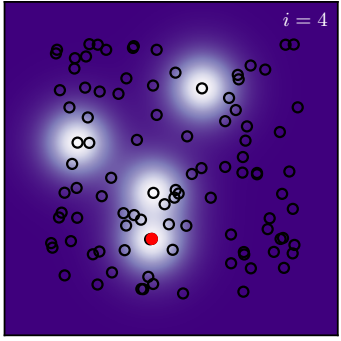
IterGP extends the most commonly used GP approximations to include computational uncertainty, with at most quadratic cost.

	Actions s_i	Classic Analog
IterGP-Cholesky	e_i	(Partial) Cholesky / Subset of data
IterGP-CG	$\hat{P}^{-1} r_i$	(Preconditioned) CG
IterGP-PseudoInput	$k(X, z_i)$	$\approx$ SVGP

Computational Uncertainty



Combined Uncertainty

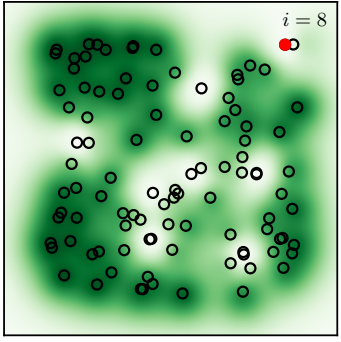


Policy Choice and Connection to Other Approximations

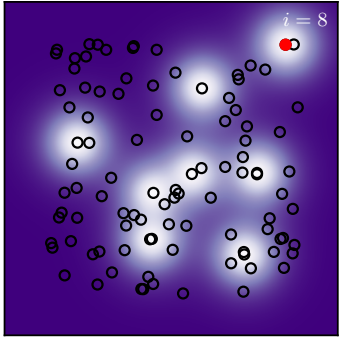
IterGP extends the most commonly used GP approximations to include computational uncertainty, with at most quadratic cost.

	Actions s_i	Classic Analog
IterGP-Cholesky	e_i	(Partial) Cholesky / Subset of data
IterGP-CG	$\hat{P}^{-1}r_i$	(Preconditioned) CG
IterGP-PseudoInput	$k(X, z_i)$	$\approx$ SVGP

Computational Uncertainty



Combined Uncertainty

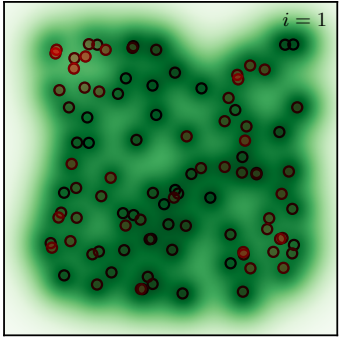


Policy Choice and Connection to Other Approximations

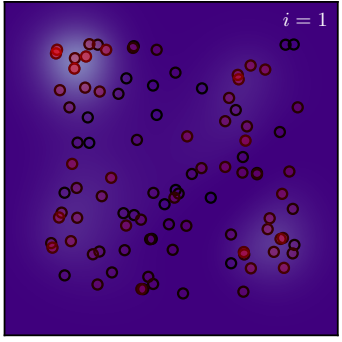
IterGP extends the most commonly used GP approximations to include computational uncertainty, with at most quadratic cost.

	Actions s_i	Classic Analog
IterGP-Cholesky	e_i	(Partial) Cholesky / Subset of data
IterGP-CG	$\hat{P}^{-1}r_i$	(Preconditioned) CG
IterGP-PseudoInput	$k(X, z_i)$	$\approx$ SVGP

Computational Uncertainty



Combined Uncertainty

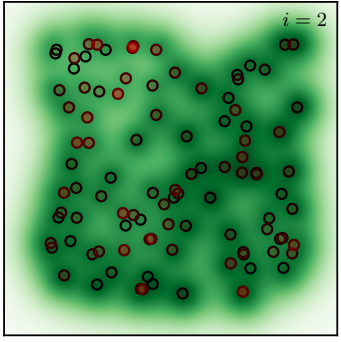


Policy Choice and Connection to Other Approximations

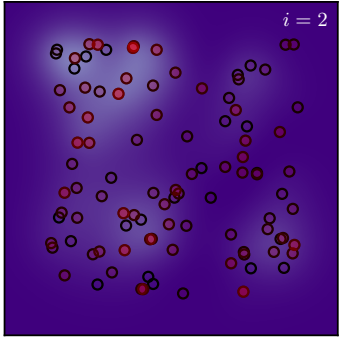
IterGP extends the most commonly used GP approximations to include computational uncertainty, with at most quadratic cost.

	Actions s_i	Classic Analog
IterGP-Cholesky	e_i	(Partial) Cholesky / Subset of data
IterGP-CG	$\hat{P}^{-1}r_i$	(Preconditioned) CG
IterGP-PseudoInput	$k(X, z_i)$	$\approx$ SVGP

Computational Uncertainty



Combined Uncertainty

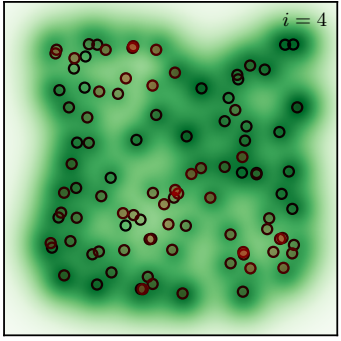


Policy Choice and Connection to Other Approximations

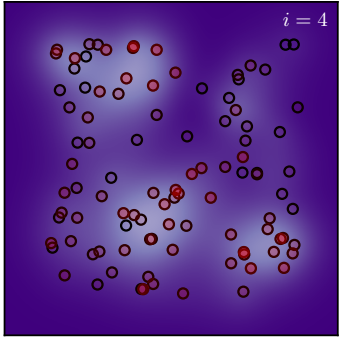
IterGP extends the most commonly used GP approximations to include computational uncertainty, with at most quadratic cost.

	Actions s_i	Classic Analog
IterGP-Cholesky	e_i	(Partial) Cholesky / Subset of data
IterGP-CG	$\hat{P}^{-1}r_i$	(Preconditioned) CG
IterGP-PseudoInput	$k(X, z_i)$	$\approx$ SVGP

Computational Uncertainty



Combined Uncertainty

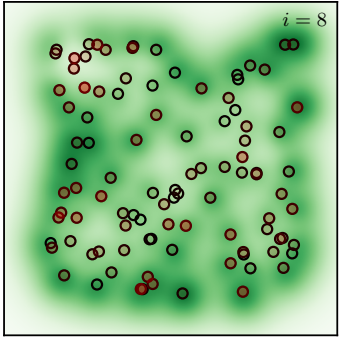


Policy Choice and Connection to Other Approximations

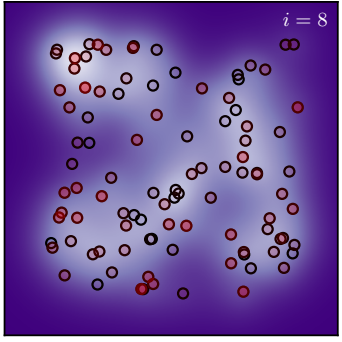
IterGP extends the most commonly used GP approximations to include computational uncertainty, with at most quadratic cost.

	Actions s_i	Classic Analog
IterGP-Cholesky	e_i	(Partial) Cholesky / Subset of data
IterGP-CG	$\hat{P}^{-1}r_i$	(Preconditioned) CG
IterGP-PseudoInput	$k(X, z_i)$	$\approx$ SVGP

Computational Uncertainty



Combined Uncertainty

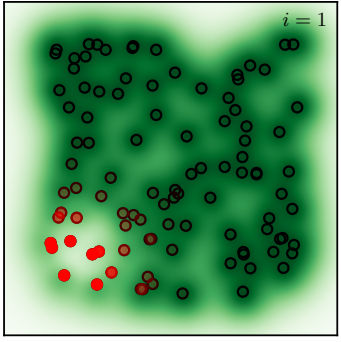


Policy Choice and Connection to Other Approximations

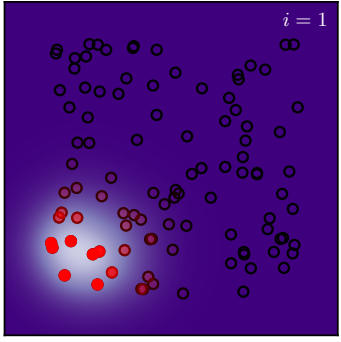
IterGP extends the most commonly used GP approximations to include computational uncertainty, with at most quadratic cost.

	Actions s_i	Classic Analog
IterGP-Cholesky	e_i	(Partial) Cholesky / Subset of data
IterGP-CG	$\hat{P}^{-1} r_i$	(Preconditioned) CG
IterGP-PseudoInput	$k(X, z_i)$	$\approx$ SVGP

Computational Uncertainty



Combined Uncertainty

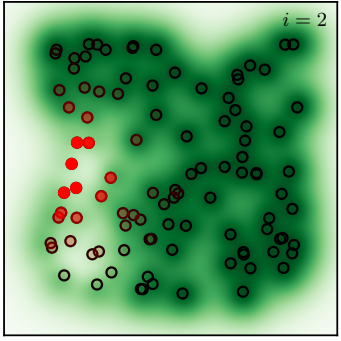


Policy Choice and Connection to Other Approximations

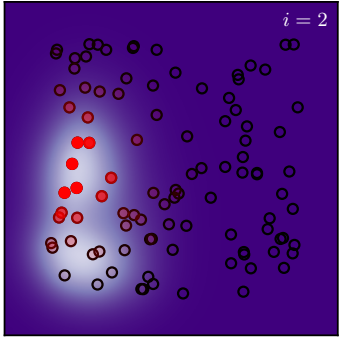
IterGP extends the most commonly used GP approximations to include computational uncertainty, with at most quadratic cost.

	Actions s_i	Classic Analog
IterGP-Cholesky	e_i	(Partial) Cholesky / Subset of data
IterGP-CG	$\hat{P}^{-1} r_i$	(Preconditioned) CG
IterGP-PseudoInput	$k(X, z_i)$	$\approx$ SVGP

Computational Uncertainty



Combined Uncertainty

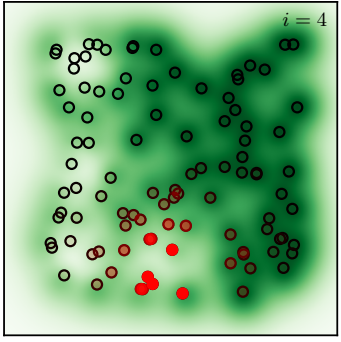


Policy Choice and Connection to Other Approximations

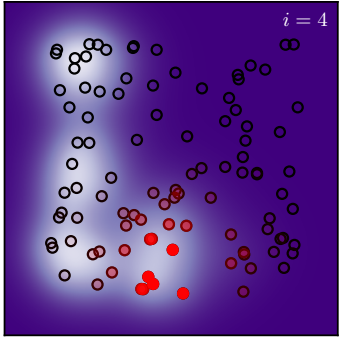
IterGP extends the most commonly used GP approximations to include computational uncertainty, with at most quadratic cost.

	Actions s_i	Classic Analog
IterGP-Cholesky	e_i	(Partial) Cholesky / Subset of data
IterGP-CG	$\hat{P}^{-1} r_i$	(Preconditioned) CG
IterGP-PseudoInput	$k(X, z_i)$	$\approx$ SVGP

Computational Uncertainty



Combined Uncertainty

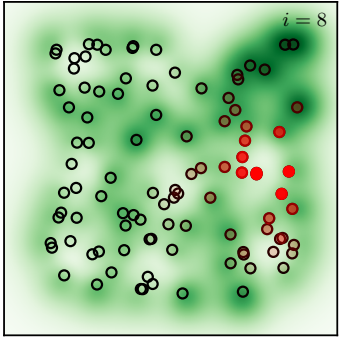


Policy Choice and Connection to Other Approximations

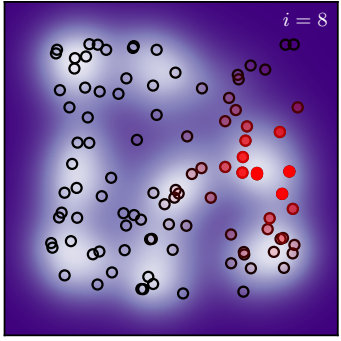
IterGP extends the most commonly used GP approximations to include computational uncertainty, with at most quadratic cost.

	Actions s_i	Classic Analog
IterGP-Cholesky	e_i	(Partial) Cholesky / Subset of data
IterGP-CG	$\hat{P}^{-1} r_i$	(Preconditioned) CG
IterGP-PseudoInput	$k(X, z_i)$	$\approx$ SVGP

Computational Uncertainty

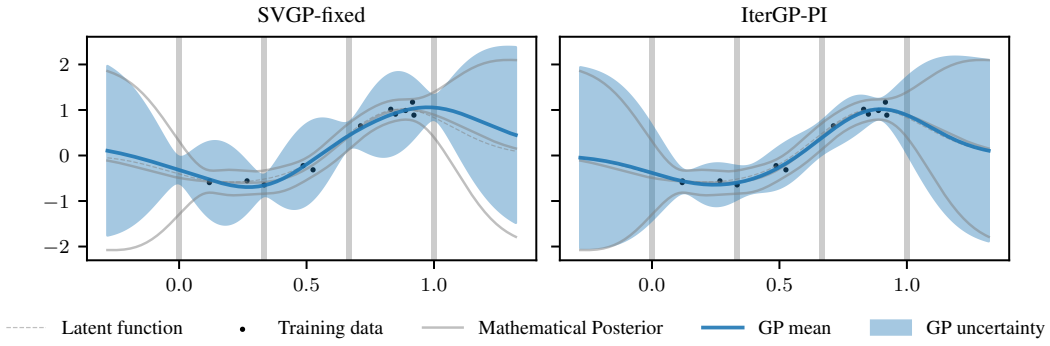


Combined Uncertainty



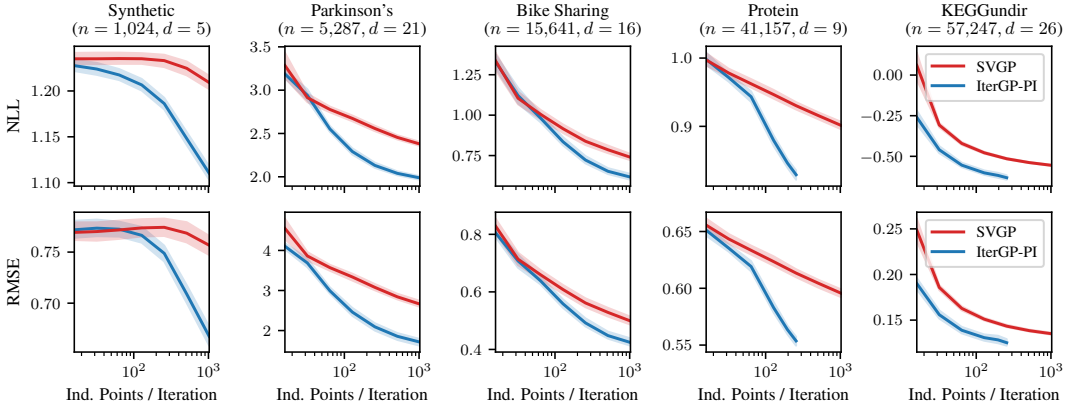
SVGP versus IterGP-PI

Quantifying computational uncertainty improves generalization of inducing point methods like SVGP. [Tit09; HFL13]



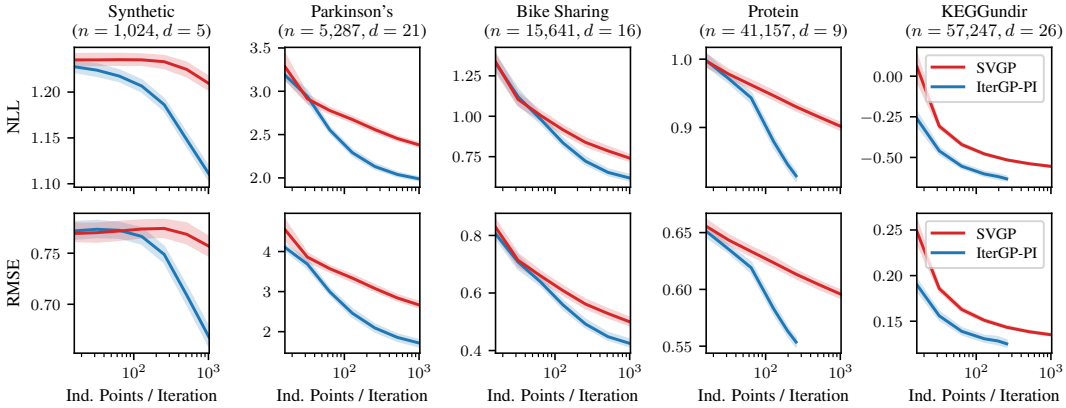
SVGP versus IterGP-PI

Quantifying computational uncertainty improves generalization of inducing point methods like SVGP. [Tit09; HFL13]



SVGP versus IterGP-PI

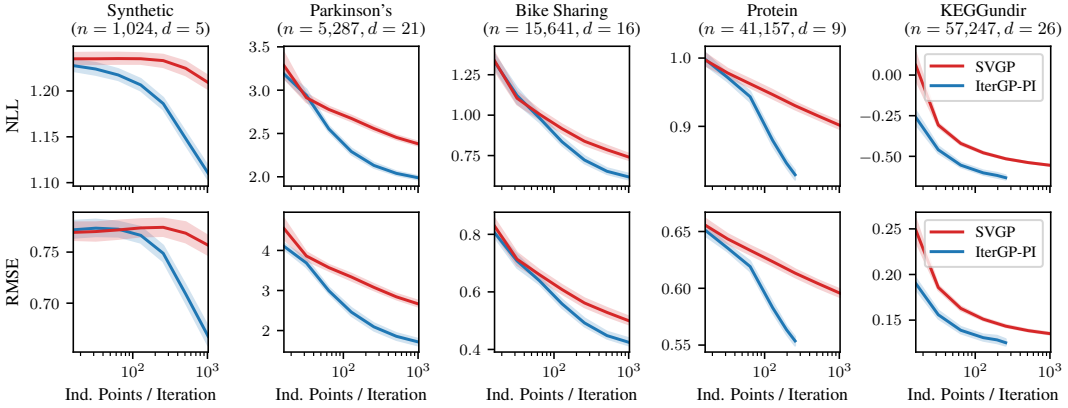
Quantifying computational uncertainty improves generalization of inducing point methods like SVGP. [Tit09; HFL13]



What about optimizing inducing point locations?

SVGP versus IterGP-PI

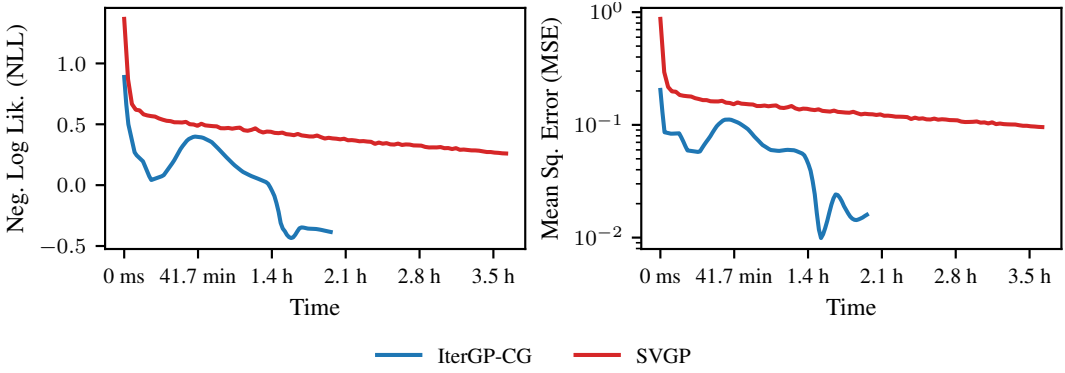
Quantifying computational uncertainty improves generalization of inducing point methods like SVGP. [Tit09; HFL13]



What about **computational cost**? SVGP: $\mathcal{O}(ni^2)$ versus IterGP: $\mathcal{O}(n^2 i)$.

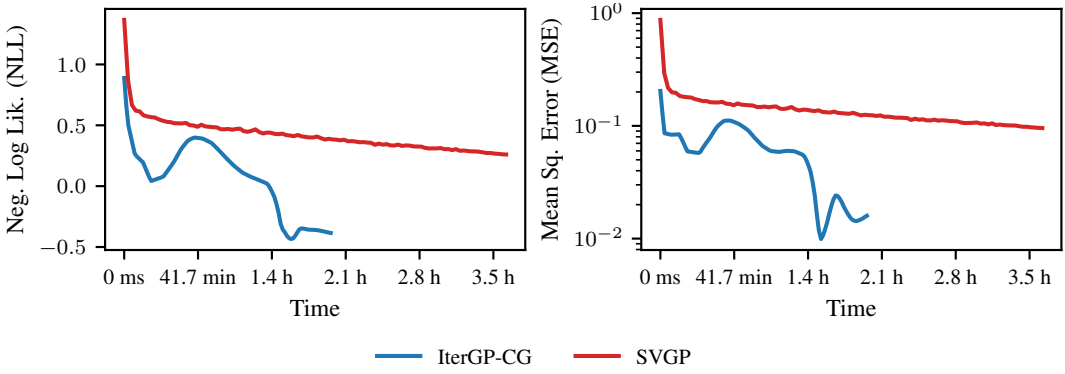
Training Gaussian Processes on Large-Scale Data

Kernel hyperparameter optimization with SVGP and IterGP on a problem with $n \approx 4 \cdot 10^5$ data points. [Wen+24, Unpublished work]



Training Gaussian Processes on Large-Scale Data

Kernel hyperparameter optimization with SVGP and IterGP on a problem with $n \approx 4 \cdot 10^5$ data points. [Wen+24, Unpublished work]



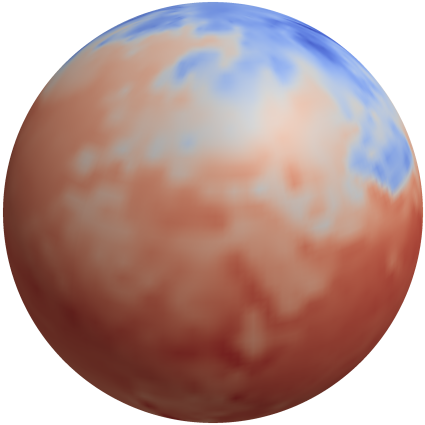
Faster large-scale Gaussian processes with **better generalization**!

Other Applications

Extending these ideas beyond what we've seen.

Spatiotemporal Modeling

Spatio-temporal regression of Earth surface temperature via computation-aware filtering and smoothing. [Pfö+24, Unpublished Work]



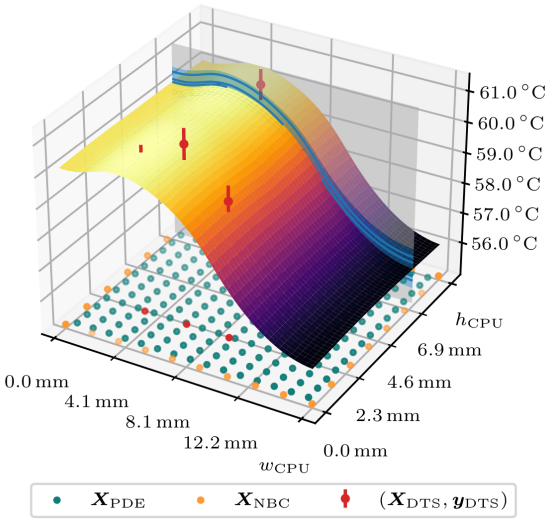
(a) Prediction

(b) Uncertainty

Physics-Informed GP Regression

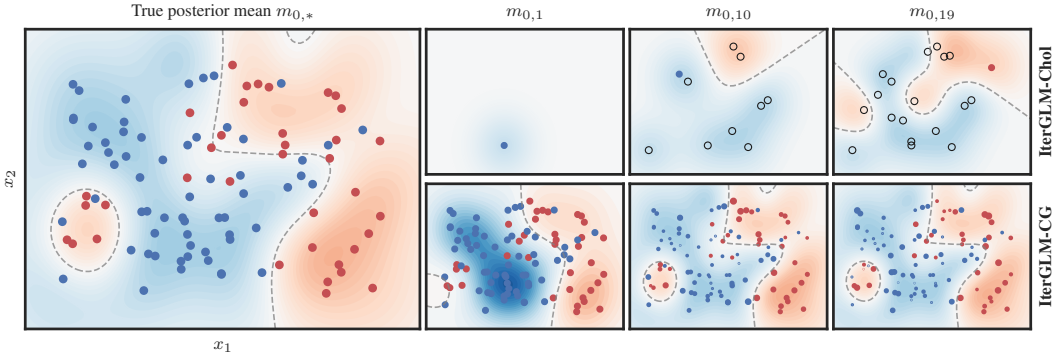
Learning to solve linear partial differential equations.

[Pfö+23]



Generalized Linear Models

Gaussian process classification with IterGLM using two different policies. [Tat+23]



Wrapping Up

Summary

- Large-scale linear systems are ubiquitous in scientific computation.

Wrapping Up

Summary

- Large-scale linear systems are ubiquitous in scientific computation.
- Probabilistic linear solvers allow us to explicitly trade off **speed for precision**.

Wrapping Up

Summary

- Large-scale linear systems are ubiquitous in scientific computation.
- Probabilistic linear solvers allow us to explicitly trade off **speed for precision**.

Application: Gaussian Processes

- Large-scale GP models are often as much about the approximation as they are about the data.

Wrapping Up

Summary

- Large-scale linear systems are ubiquitous in scientific computation.
- Probabilistic linear solvers allow us to explicitly trade off **speed for precision**.

Application: Gaussian Processes

- Large-scale GP models are often as much about the approximation as they are about the data.

Wrapping Up

Summary

- Large-scale linear systems are ubiquitous in scientific computation.
- Probabilistic linear solvers allow us to explicitly trade off **speed for precision**.

Application: Gaussian Processes

- Large-scale GP models are often as much about the approximation as they are about the data.
- We can exactly quantify the error from **finite data** and from the **approximation** via a combined uncertainty estimate. $\Longrightarrow$ IterGP

Wrapping Up

Summary

- Large-scale linear systems are ubiquitous in scientific computation.
- Probabilistic linear solvers allow us to explicitly trade off **speed for precision**.

Application: Gaussian Processes

- Large-scale GP models are often as much about the approximation as they are about the data.
- We can exactly quantify the error from **finite data** and from the **approximation** via a combined uncertainty estimate. $\Longrightarrow$ IterGP
- **Explicit trade-off** between computation and uncertainty via probabilistic linear solver.

Wrapping Up

Summary

- Large-scale linear systems are ubiquitous in scientific computation.
- Probabilistic linear solvers allow us to explicitly trade off **speed for precision**.

Application: Gaussian Processes

- Large-scale GP models are often as much about the approximation as they are about the data.
- We can exactly quantify the error from **finite data** and from the **approximation** via a combined uncertainty estimate. $\Longrightarrow$ IterGP
- **Explicit trade-off** between computation and uncertainty via probabilistic linear solver.

Open Research Questions / Future Directions

- Calibration.
- Policy design for downstream tasks (Active learning, Bayesian optimization, …).
- …?

Table of Contents

Additional Material

Observation: Uncertainty quantification of probabilistic linear solvers can be conservative!

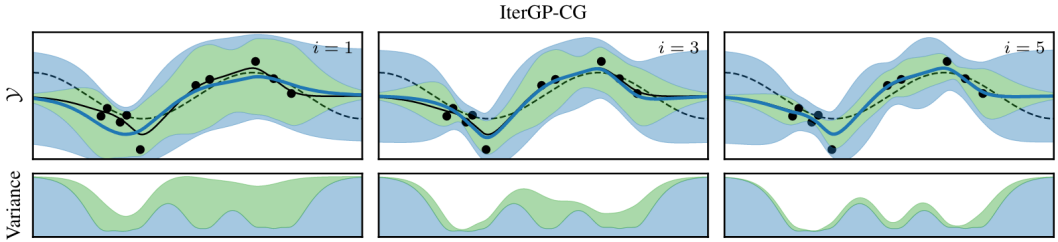


Figure: IterGP using a (conjugate) gradient policy.

Observation: Uncertainty quantification of probabilistic linear solvers can be conservative!

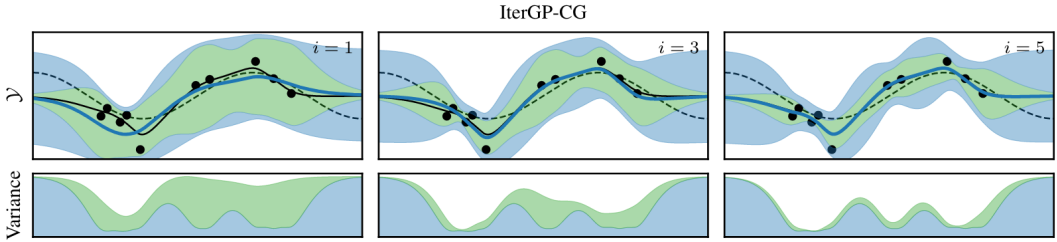


Figure: IterGP using a (conjugate) gradient policy.

Why is that? We conditioned on $\alpha_i = s_i^\top r_{i-1} = s_i^\top A(x_* - x_{i-1})$.

But: We've "cheated" for a gradient policy, since $s_i = b - Ax_{i-1} = A(x_* - x_{i-1}) = s_i(x_*)$.

Working with Infinite Data

For IterGP it does not matter how large the dataset is, or whether we have it stored on our machine.

[Wen+22] ToC

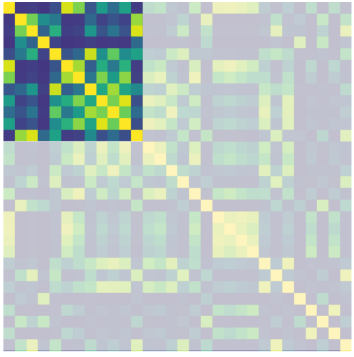
Theorem (Online GP Approximation with IterGP)

Let $n, n' \in \mathbb{N}$ and consider training data sets $\boldsymbol{X} \in \mathbb{R}^{n \times d}, \boldsymbol{y} \in \mathbb{R}^n$ and $\boldsymbol{X}' \in \mathbb{R}^{n' \times d}, \boldsymbol{y}' \in \mathbb{R}^{n'}$. Consider two sequences of actions $(\boldsymbol{s}_i)_{i=1}^{n} \in \mathbb{R}^n$ and $(\tilde{\boldsymbol{s}}_i)_{i=1}^{n+n'} \in \mathbb{R}^{n+n'}$ such that

$$\tilde{\boldsymbol{s}}_i = \begin{pmatrix} \boldsymbol{s}_i \\ \boldsymbol{0} \end{pmatrix} \tag{1}$$

Then the posterior returned by IterGP for the dataset $(\boldsymbol{X}, \boldsymbol{y})$ using actions $\boldsymbol{s}_i$ is identical to the posterior returned by IterGP for the extended dataset using actions $\tilde{\boldsymbol{s}}_i$:

$$\textsc{IterGP}(\mu, k, \boldsymbol{X}, \boldsymbol{y}, (\boldsymbol{s}_i)_i) = \textsc{IterGP}\left(\mu, k, \begin{pmatrix} \boldsymbol{X} \\ \boldsymbol{X}' \end{pmatrix}, \begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{y}' \end{pmatrix}, (\tilde{\boldsymbol{s}}_i)_i\right).$$



An Approximation Method or a Better Model?

An alternative view of IterGP as a better model for the way we do inference instead of an approximation.

Observation: Only once we perform computation on data, does it enter our prediction.

An Approximation Method or a Better Model?

An alternative view of IterGP as a better model for the way we do inference instead of an approximation.

ToC

Observation: Only once we perform computation on data, does it enter our prediction.

The distinction between data and computation vanishes.

An Approximation Method or a Better Model?

An alternative view of IterGP as a better model for the way we do inference instead of an approximation.

Observation: Only once we perform computation on data, does it enter our prediction.

The distinction between data and computation vanishes.

What if we modeled this situation with a Gaussian process?

$$
\begin{aligned}
f &\sim \mathcal{GP}(\mu, k) \\
\tilde{\boldsymbol{y}} \mid f(\boldsymbol{X}) &\sim \mathcal{N}\left(\boldsymbol{S}_i^\intercal f(\boldsymbol{X}), \sigma^2 \boldsymbol{S}_i^\intercal \boldsymbol{S}_i\right) \\
f \mid \boldsymbol{X}, \tilde{\boldsymbol{y}} &\sim \mathcal{GP}(\mu_i, k_i)
\end{aligned}
$$

An Approximation Method or a Better Model?

An alternative view of IterGP as a better model for the way we do inference instead of an approximation.

ToC

Observation: Only once we perform computation on data, does it enter our prediction.

The distinction between data and computation vanishes.

What if we modeled this situation with a Gaussian process?

$$
\begin{aligned}
f &\sim \mathcal{GP}(\mu, k) \\
\tilde{\boldsymbol{y}} \mid f(\boldsymbol{X}) &\sim \mathcal{N}\left(\boldsymbol{S}_i^\intercal f(\boldsymbol{X}), \sigma^2 \boldsymbol{S}_i^\intercal \boldsymbol{S}_i\right) \\
f \mid \boldsymbol{X}, \tilde{\boldsymbol{y}} &\sim \mathcal{GP}(\mu_i, k_i)
\end{aligned}
$$

IterGP's combined posterior is equivalent to exact GP regression for linearly projected data.

References I

- E. Daxberger, A. Kristiadi, A. Immer, R. Eschenhagen, M. Bauer, and P. Hennig. *Laplace Redux – Effortless Bayesian Deep Learning*. 2022. DOI: 10.48550/arXiv.2106.14806. URL: http://arxiv.org/abs/2106.14806 (cit. on p. 7).
- P. Hennig, M. A. Osborne, and M. Girolami. "Probabilistic numerics and uncertainty in computations". In: *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 471.2179 (2015) (cit. on pp. 9–12).
- J. Cockayne, C. J. Oates, T. J. Sullivan, and M. Girolami. "Bayesian Probabilistic Numerical Methods". In: *SIAM Review* 61.4 (2019), pp. 756–789. DOI: 10.1137/17M1139357 (cit. on pp. 9–12).
- P. Hennig, M. A. Osborne, and H. P. Kersting. *Probabilistic Numerics: Computation as Machine Learning*. CUP, 2022. ISBN: 978-1-316-68141-1. DOI: 10.1017/9781316681411 (cit. on pp. 9–12).
- P. Hennig. "Probabilistic Interpretation of Linear Solvers". In: *SIAM Journal on Optimization* 25.1 (2015), pp. 234–260 (cit. on pp. 9–21).

References II

- J. Cockayne, C. J. Oates, I. C. Ipsen, and M. Girolami. "A Bayesian Conjugate Gradient Method (with Discussion)". In: *Bayesian Analysis* 14.3 (2019), pp. 937–1012. DOI: 10.1214/19-BA1145 (cit. on pp. 9–23, 28, 30, 31).
- J. Wenger and P. Hennig. "Probabilistic Linear Solvers for Machine Learning". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020 (cit. on pp. 9–12).
- J. Wenger, G. Pleiss, M. Pförtner, P. Hennig, and J. P. Cunningham. "Posterior and Computational Uncertainty in Gaussian Processes". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2022 (cit. on pp. 28, 48–59, 66–68, 102).
- L. N. Trefethen and D. Bau. *Numerical Linear Algebra*. Society for Industrial and Applied Mathematics (SIAM), 1997 (cit. on p. 29).
- M. Kanagawa, P. Hennig, D. Sejdinovic, and B. K. Sriperumbudur. *Gaussian processes and kernel methods: A review on connections and equivalences*. 2018. arXiv: 1807.02582 (cit. on pp. 66–68).

References III

- M. Titsias. "Variational learning of inducing variables in sparse Gaussian processes". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2009 (cit. on pp. 81–84).
- J. Hensman, N. Fusi, and N. D. Lawrence. "Gaussian processes for big data". In: *Conference on Uncertainty in Artificial Intelligence (UAI)*. 2013 (cit. on pp. 81–84).
- M. Pförtner, I. Steinwart, P. Hennig, and J. Wenger. *Physics-Informed Gaussian Process Regression Generalizes Linear PDE Solvers*. 2023. DOI: 10.48550/arXiv.2212.12474. URL: http://arxiv.org/abs/2212.12474 (cit. on p. 89).
- L. Tatzel, J. Wenger, F. Schneider, and P. Hennig. *Accelerating Generalized Linear Models by Trading off Computation for Uncertainty*. 2023. DOI: 10.48550/arXiv.2310.20285. URL: http://arxiv.org/abs/2310.20285 (cit. on p. 90).