Computation-Aware Gaussian Process Inference

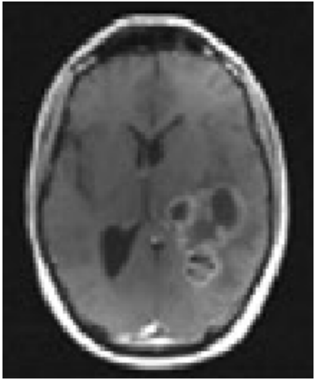
Jonathan Wenger

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

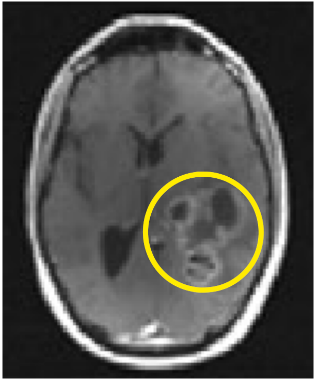
imprs-is

(Radmanesh et al., 2022)

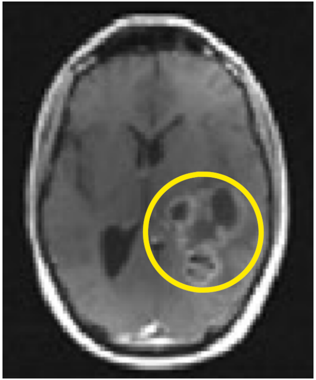
Accurate Reconstruction



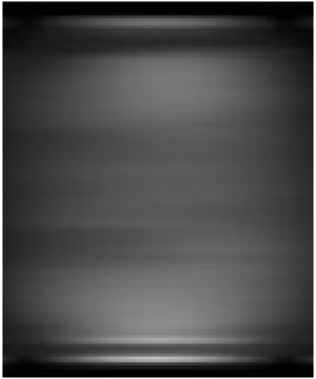
Accurate Reconstruction



Accurate Reconstruction



Subsampled Reconstruction (100x)

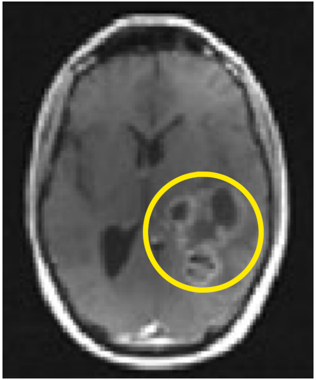


Motivation

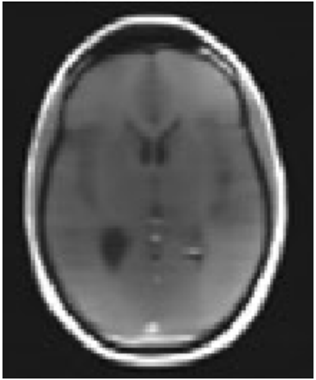
Accelerated MRI Reconstruction

(Radmanesh et al., 2022)

Accurate Reconstruction



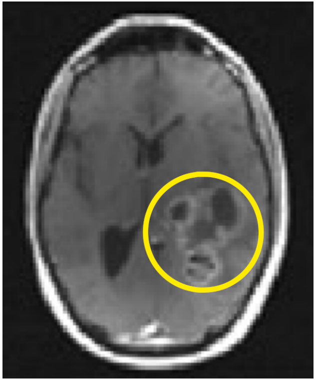
Learned Reconstruction (100x)



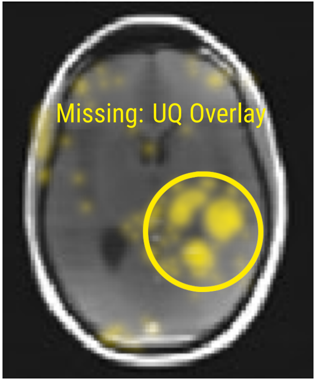
Importance of Uncertainty Quantification

Crucial information to benefit from the 100x acceleration is missing!

Accurate Reconstruction



Learned Reconstruction (100x)



Uncertainty quantification is essential to make critical decisions.

Gaussian Process Regression

Supervised learning of an unknown function $f : \mathbb{R}^d \to \mathbb{R}$ with uncertainty quantification.

Gaussian Process Regression

Learning an unknown function from data.

Goal: Supervised learning from n data points (X, y)

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

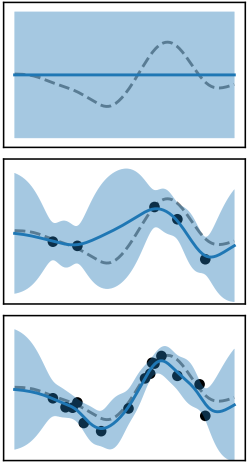
Likelihood: Observations $y = f(X) + \varepsilon \sim \mathcal{N}\left(f(X), \sigma^2 I\right)$

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

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

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

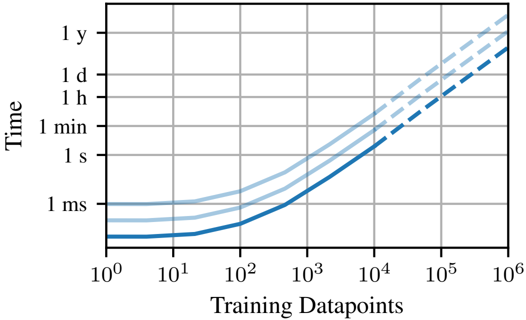
where $\hat{K} = K + \sigma^2 I \in \mathbb{R}^{n \times n}$.



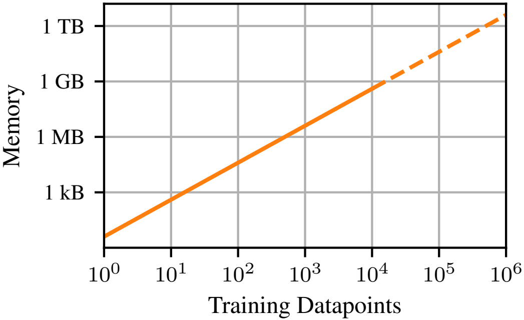
Computational Cost of Gaussian Processes

Uncertainty quantification can be expensive.

Time: $\mathcal{O}(n^3)$

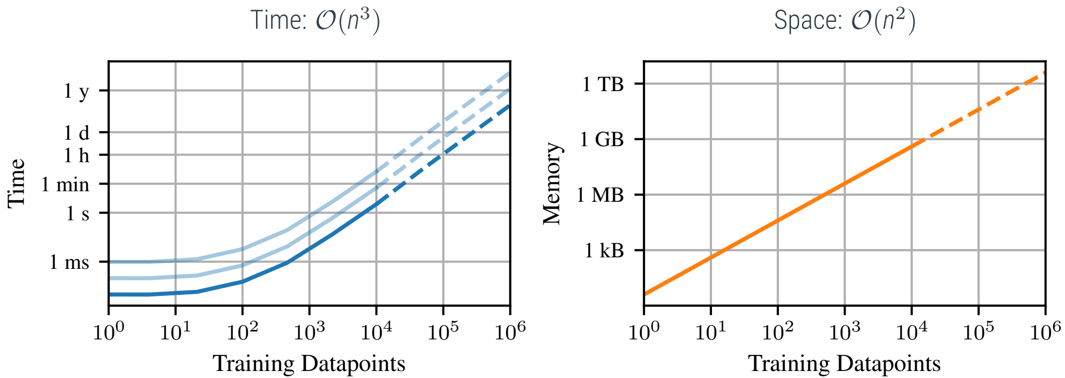


Space: $\mathcal{O}(n^2)$



Computational Cost of Gaussian Processes

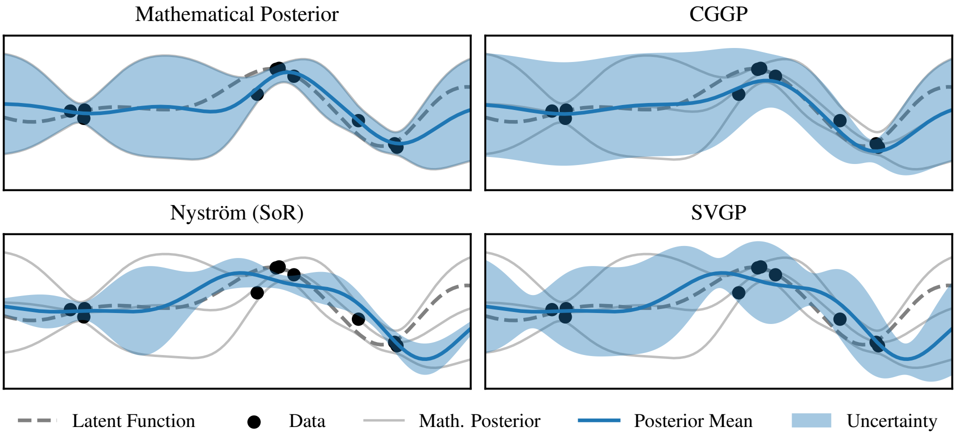
Uncertainty quantification can be expensive.



We need to **approximate** the posterior.

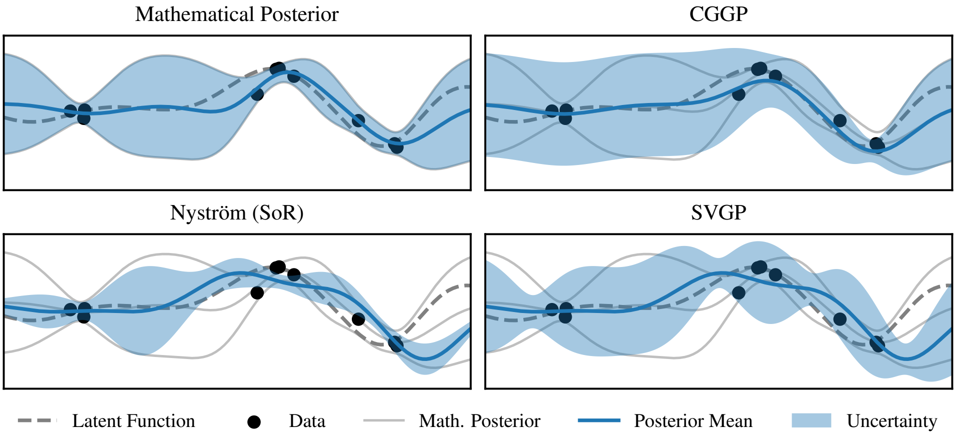
Approximate Gaussian Process Inference

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

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

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?

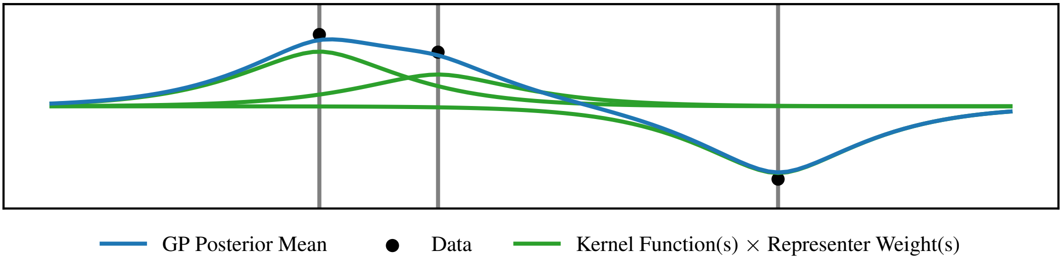
Efficiently approximating the posterior of a Gaussian process.

Representer Weights

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

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

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



Interlude: Method of Conjugate Gradients

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

Goal: Approximately solve linear system $Ax = b$, where A symmetric positive definite.

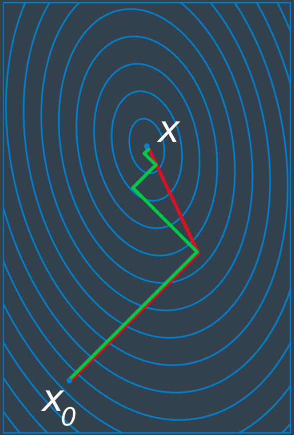
Idea: Rephrase as quadratic optimization problem and optimize. Let

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

then $\nabla f(x) = \mathbf{0} \iff Ax = b \iff r(x) := b - Ax = \mathbf{0}$.

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. $\langle d_i^\intercal d_j \rangle_A = d_i^\intercal A d_j = 0$ 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

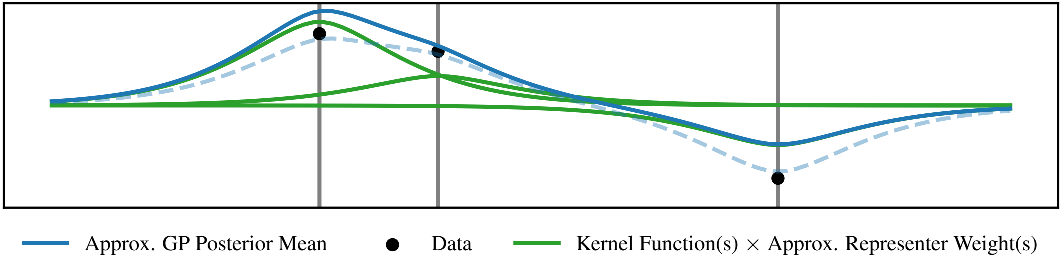
Approximating Representer Weights

Iterative linear solvers can approximate the representer weights.

(Gardner et al., 2018; Charlier et al., 2021)

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

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



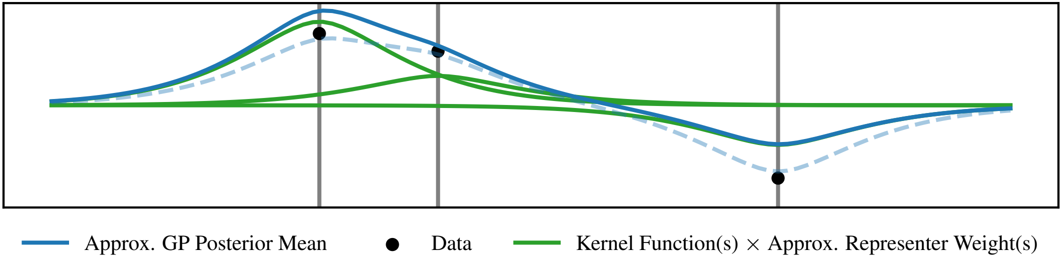
Approximating Representer Weights

Iterative linear solvers can approximate the representer weights.

(Gardner et al., 2018; Charlier et al., 2021)

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

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



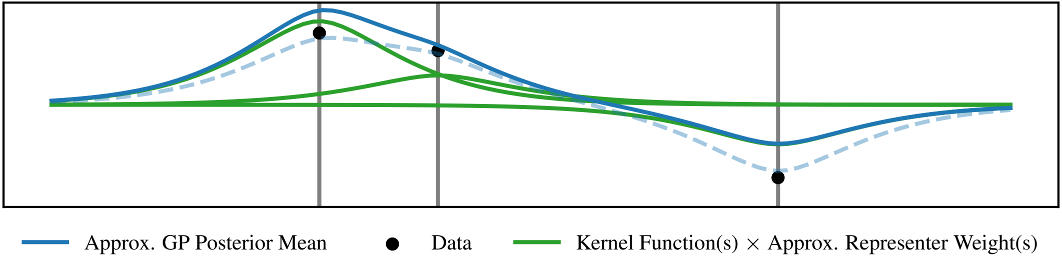
Benefit: Time complexity $\mathcal{O}(n^2)$ and space complexity $\mathcal{O}(nd)$.

Approximating Representer Weights

Iterative linear solvers can approximate the representer weights. (Gardner et al., 2018; Charlier et al., 2021)

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

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

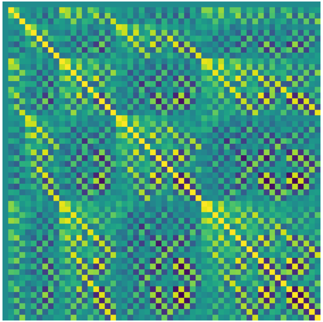


Question: Can we quantify the impact of this approximation on the posterior?

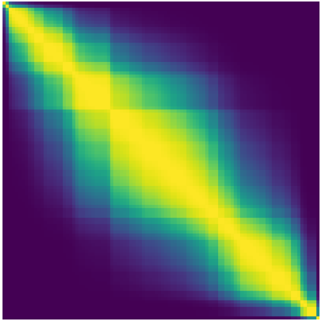
Q2: Can We Quantify Approximation Error?

Probabilistic error quantification at prediction time using probabilistic linear solvers.

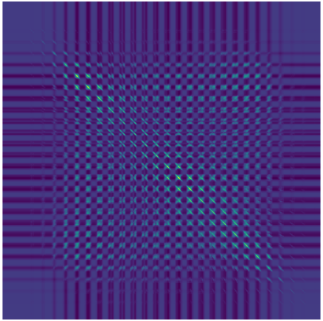
Problem: Solve linear system(s) $Ax_* = b$ for $x_* \in \mathbb{R}^n$.



(a) Gram matrix $X^\top X$



(b) Kernel matrix $\hat{K} = k(X, X) + \sigma^2 I$

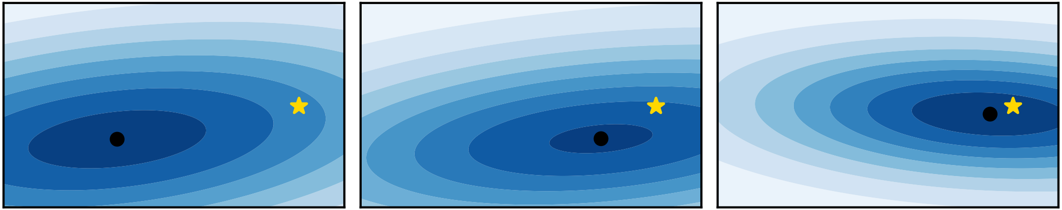


(c) Hessian matrix $\nabla^2 \ell(y, f(X))$

Linear systems in ML are **large-scale**, have **model-induced structure** and are often **solved repeatedly**.

Core Insights of Probabilistic Numerics

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



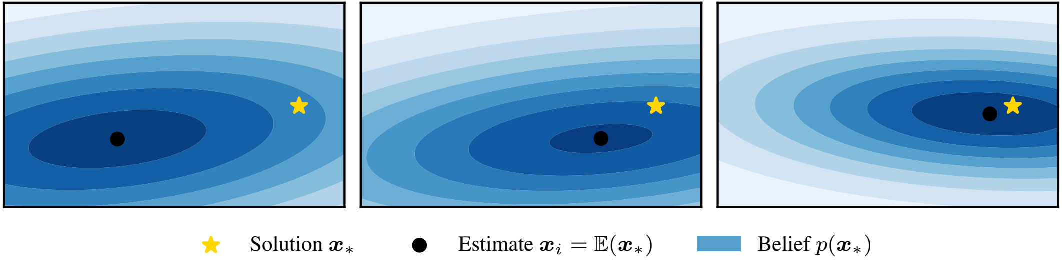
★ Solution $\boldsymbol{x}_*$ ● Estimate $\boldsymbol{x}_i = \mathbb{E}(\boldsymbol{x}_*)$ ▬ Belief $p(\boldsymbol{x}_*)$

Probabilistic Linear Solvers

Interpreting solving linear systems numerically as statistical inference. (Hennig, 2015; Cockayne et al., 2019; Wenger et al., 2020)

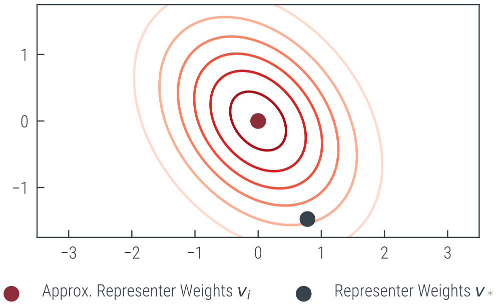
Core Insights of Probabilistic Numerics

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



Goal: Solve $\hat{K}v_* = y$ approximately.

Prior: $v_* \sim \mathcal{N}(v_0, \Sigma_0)$



Learning Representer Weights

Estimating representer weights with a probabilistic linear solver.

(Wenger et al., 2022a)

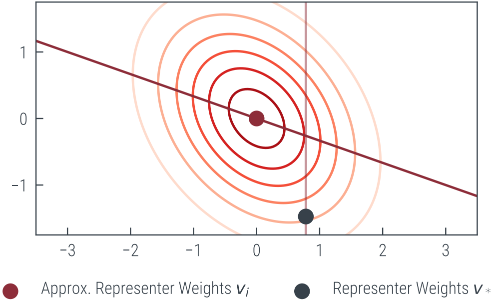
Goal: Solve $\hat{K}v_* = y$ approximately.

Prior: $v_* \sim \mathcal{N}(v_0, \Sigma_0)$

Likelihood: Observe representer weights via arbitrarily chosen actions $s_i \in \mathbb{R}^n$:

$$\alpha_i := s_i^\top r_{i-1} = s_i^\top((y-\mu) - \hat{K}v_{i-1}) = s_i^\top \hat{K}(v_* - v_{i-1})$$

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



Learning Representer Weights

Estimating representer weights with a probabilistic linear solver.

(Wenger et al., 2022a)

Goal: Solve $\hat{K} v_* = y$ approximately.

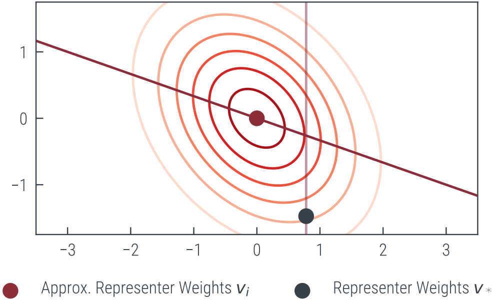
Prior: $v_* \sim \mathcal{N}(v_0, \Sigma_0)$

Likelihood: Observe representer weights via arbitrarily chosen actions $s_i \in \mathbb{R}^n$:

$$\begin{aligned}\alpha_i &:= s_i^\top r_{i-1} = s_i^\top\big((y-\mu) - \hat{K} v_{i-1}\big) \\ &= s_i^\top \hat{K}(v_* - v_{i-1})\end{aligned}$$

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

Posterior: Affine Gaussian inference!



Goal: Solve $\hat{K} v_* = y$ approximately.

Prior: $v_* \sim \mathcal{N}(v_0, \Sigma_0)$

Likelihood: Observe representer weights via arbitrarily chosen actions $s_i \in \mathbb{R}^n$:

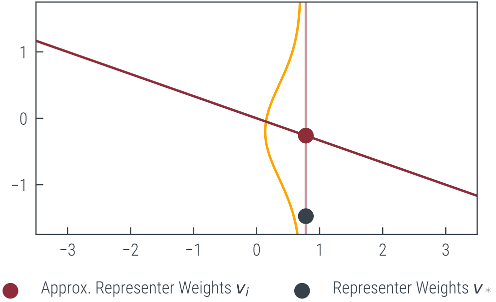
$$\alpha_i := s_i^\top r_{i-1} = s_i^\top ((y - \mu) - \hat{K} v_{i-1}) = s_i^\top \hat{K}(v_* - v_{i-1})$$

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

Posterior: $v_* \mid \alpha_i \sim \mathcal{N}(v_i, \Sigma_i)$, where

$$v_i = v_{i-1} + \Sigma_{i-1}\hat{K} s_i (s_i^\top \hat{K} \Sigma_{i-1} \hat{K} s_i)^{-1} s_i^\top \hat{K}(v_* - v_{i-1})$$

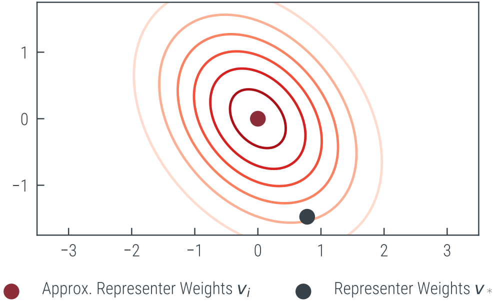
$$\Sigma_i = \Sigma_{i-1} - \Sigma_{i-1}\hat{K} s_i (s_i^\top \hat{K} \Sigma_{i-1} \hat{K} s_i)^{-1} s_i^\top \hat{K} \Sigma_{i-1}$$



Choosing the Linear Solver Prior

The Gaussian process prior makes assumptions about the representer weights.

Question: How to choose the linear solver prior?



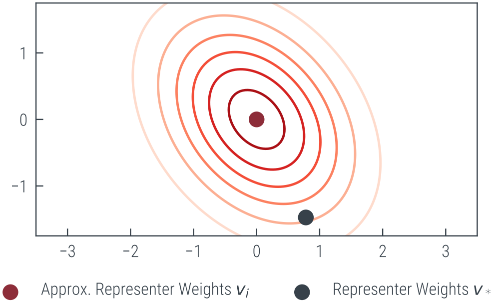
Choosing the Linear Solver Prior

The Gaussian process prior makes assumptions about the representer weights.

Question: How to choose the linear solver prior?

Remember: $\boldsymbol{y} = f(\boldsymbol{X}) + \boldsymbol{\varepsilon}$, where $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})$.

$$\implies \boldsymbol{y} - \boldsymbol{\mu} \sim \mathcal{N}(\mathbf{0}, k(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 \boldsymbol{I}) = \mathcal{N}\left(\mathbf{0}, \hat{\boldsymbol{K}}\right)$$



Choosing the Linear Solver Prior

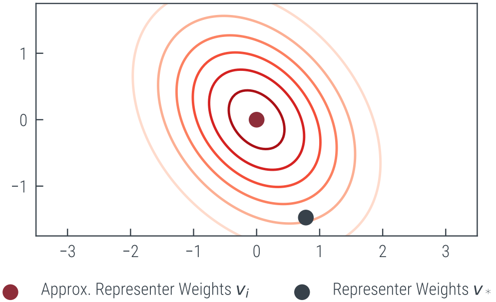
The Gaussian process prior makes assumptions about the representer weights.

Question: How to choose the linear solver prior?

Remember: $y = f(X) + \varepsilon$, where $\varepsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2 I)$.

$$\implies y - \boldsymbol{\mu} \sim \mathcal{N}(\mathbf{0}, k(X, X) + \sigma^2 I) = \mathcal{N}\left(\mathbf{0}, \hat{K}\right)$$

$$\implies v_* = \hat{K}^{-1}(y - \boldsymbol{\mu}) \sim \mathcal{N}\left(\underbrace{\mathbf{0}}_{=v_0}, \underbrace{\hat{K}^{-1}}_{=\boldsymbol{\Sigma}_0}\right)$$



Choosing the Linear Solver Prior

The Gaussian process prior makes assumptions about the representer weights.

Question: How to choose the linear solver prior?

$$\implies y - \boldsymbol{\mu} \sim \mathcal{N}(\mathbf{0}, k(X,X) + \sigma^2 I) = \mathcal{N}\left(\mathbf{0}, \hat{K}\right)$$

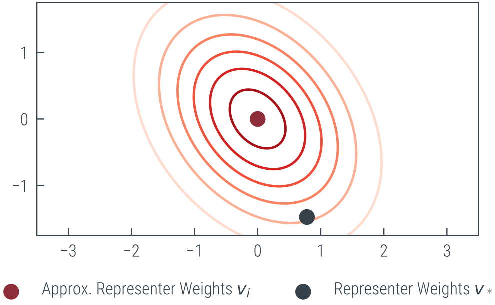
$$\implies v_* = \hat{K}^{-1}(y - \boldsymbol{\mu}) \sim \mathcal{N}\left(\underbrace{\mathbf{0}}_{=v_0}, \underbrace{\hat{K}^{-1}}_{=\boldsymbol{\Sigma}_0}\right)$$

Setting $v_0 = 0$ and $\boldsymbol{\Sigma}_0 = \hat{K}^{-1}$, we have

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

$$\boldsymbol{\Sigma}_i = \boldsymbol{\Sigma}_{i-1} - S_i(S_i^\top \hat{K} S_i)^{-1} S_i^\top = \boldsymbol{\Sigma}_0 - C_i$$

where S_i is the matrix of actions $s_1, \ldots, s_i$.



Choosing the Linear Solver Prior

The Gaussian process prior makes assumptions about the representer weights.

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Question: How to choose the linear solver prior?

$$\implies y - \boldsymbol{\mu} \sim \mathcal{N}(\mathbf{0}, k(X, X) + \sigma^2 I) = \mathcal{N}\left(\mathbf{0}, \hat{K}\right)$$

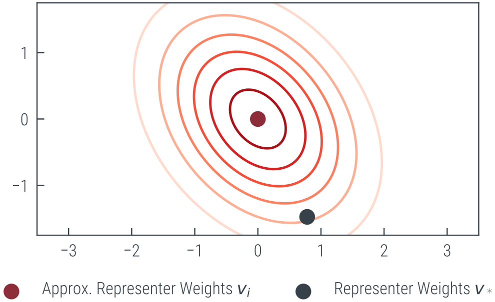
$$\implies v_* = \hat{K}^{-1}(y - \boldsymbol{\mu}) \sim \mathcal{N}\left(\underbrace{\mathbf{0}}_{=v_0}, \underbrace{\hat{K}^{-1}}_{=\boldsymbol{\Sigma}_0}\right)$$

Setting $v_0 = 0$ and $\boldsymbol{\Sigma}_0 = \hat{K}^{-1}$, we have

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

$$\boldsymbol{\Sigma}_i = \boldsymbol{\Sigma}_{i-1} - S_i(S_i^\top \hat{K} S_i)^{-1} S_i^\top = \boldsymbol{\Sigma}_0 - C_i$$

where S_i is the matrix of actions $s_1, \ldots, s_i$.



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. (Wenger et al., 2022a)

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

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

IterGP: Computation-Aware Gaussian Process Inference

Quantifying uncertainty arising from observing finite data and performing a finite amount of computation. (Wenger et al., 2022a)

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

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

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. (Wenger et al., 2022a)

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

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

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. (Wenger et al., 2022a)

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

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

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: $(f \mid \boldsymbol{y})(\cdot) \stackrel{d}{=} f(\cdot) + k(\cdot, \boldsymbol{X})\hat{\boldsymbol{K}}^{-1}(\boldsymbol{y} - \boldsymbol{\mu})$

IterGP: Computation-Aware Gaussian Process Inference

Quantifying uncertainty arising from observing finite data and performing a finite amount of computation. (Wenger et al., 2022a)

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

$$\mu_*(\cdot) = \mu(\cdot) + k(\cdot, \mathbf{X})\hat{\mathbf{K}}^{-1}(\mathbf{y} - \boldsymbol{\mu}), \quad \text{and} \quad k_*(\cdot,\cdot) = k(\cdot,\cdot) - k(\cdot, \mathbf{X})\hat{\mathbf{K}}^{-1}k(\mathbf{X}, \cdot)$$

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

Idea: Propagate uncertainty about representer weights to posterior.

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

IterGP: Computation-Aware Gaussian Process Inference

Quantifying uncertainty arising from observing finite data and performing a finite amount of computation. (Wenger et al., 2022a)

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

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

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

Idea: Propagate uncertainty about representer weights to posterior.

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

$$\begin{aligned}
\mu_i(\cdot) &= \mu(\cdot) + k(\cdot, \boldsymbol{X})\boldsymbol{v}_i \\
k_i(\cdot,\cdot) &= \underbrace{k(\cdot,\cdot) - k(\cdot, \boldsymbol{X})\hat{K}^{-1}k(\boldsymbol{X}, \cdot)}_{\text{mathematical uncertainty}} + \underbrace{k(\cdot, \boldsymbol{X})\boldsymbol{\Sigma}_i k(\boldsymbol{X}, \cdot)}_{\text{computational uncertainty}} = \underbrace{k(\cdot,\cdot) - k(\cdot, \boldsymbol{X})\boldsymbol{C}_i k(\boldsymbol{X}, \cdot)}_{\text{combined uncertainty}}
\end{aligned}$$

Probabilistic Quantification of Approximation Error

The covariance can be interpreted as a squared error.

Combined Uncertainty

Belief about the latent function is captured by $f \sim \mathcal{GP}(\mu_i, k_i)$, s.t.

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

$$k_i(\cdot,\cdot) = \underbrace{k(\cdot,\cdot) - k(\cdot, \boldsymbol{X})\hat{\boldsymbol{K}}^{-1}k(\boldsymbol{X},\cdot)}_{\text{mathematical uncertainty}} + \underbrace{k(\cdot, \boldsymbol{X})\boldsymbol{\Sigma}_i k(\boldsymbol{X},\cdot)}_{\text{computational uncertainty}} = \underbrace{k(\cdot,\cdot) - k(\cdot, \boldsymbol{X})\boldsymbol{C}_i k(\boldsymbol{X},\cdot)}_{\text{combined uncertainty}}$$

Remember: $k(\boldsymbol{x}, \boldsymbol{x}) = \mathrm{Cov}(f(\boldsymbol{x}), f(\boldsymbol{x})) = \mathbb{E}\big((f(\boldsymbol{x}) - \mathbb{E}(f(\boldsymbol{x})))^2\big)$

Probabilistic Quantification of Approximation Error

The covariance can be interpreted as a squared error.

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Combined Uncertainty

Belief about the latent function is captured by $f \sim \mathcal{GP}(\mu_i, k_i)$, s.t.

$$\mu_i(\cdot) = \mu(\cdot) + k(\cdot, X)v_i$$

$$k_i(\cdot,\cdot) = \underbrace{k(\cdot,\cdot) - k(\cdot, X)\hat{K}^{-1}k(X,\cdot)}_{\text{mathematical uncertainty}} + \underbrace{k(\cdot, X)\Sigma_i k(X,\cdot)}_{\text{computational uncertainty}} = \underbrace{k(\cdot,\cdot) - k(\cdot, X)C_i k(X,\cdot)}_{\text{combined uncertainty}}$$

Remember: $k(x, x) = \mathrm{Cov}(f(x), f(x)) = \mathbb{E}\big((f(x) - \mathbb{E}(f(x)))^2\big)$

$$k_*(x, x) = \underbrace{k(x,x) - k(x, X)\hat{K}^{-1}k(X, x)}_{\text{mathematical uncertainty}} = \mathbb{E}\big((f(x) - \mu_*(x))^2\big)$$

Probabilistic Quantification of Approximation Error

The covariance can be interpreted as a squared error.

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Combined Uncertainty

Belief about the latent function is captured by $f \sim \mathcal{GP}(\mu_i, k_i)$, s.t.

$$\mu_i(\cdot) = \mu(\cdot) + k(\cdot, X)v_i$$

$$k_i(\cdot,\cdot) = \underbrace{k(\cdot,\cdot) - k(\cdot, X)\hat{K}^{-1}k(X,\cdot)}_{\text{mathematical uncertainty}} + \underbrace{k(\cdot, X)\boldsymbol{\Sigma}_i k(X,\cdot)}_{\text{computational uncertainty}} = \underbrace{k(\cdot,\cdot) - k(\cdot, X)C_i k(X,\cdot)}_{\text{combined uncertainty}}$$

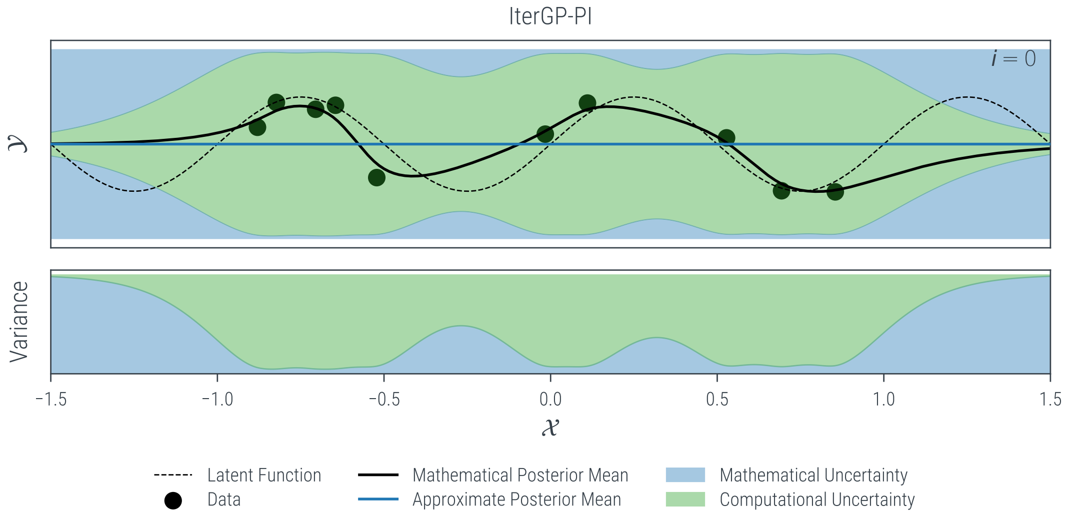
Remember: $k(x,x) = \mathrm{Cov}(f(x), f(x)) = \mathbb{E}\big((f(x) - \mathbb{E}(f(x)))^2\big)$

$$k_*(x,x) = \underbrace{k(x,x) - k(x,X)\hat{K}^{-1}k(X,x)}_{\text{mathematical uncertainty}} = \mathbb{E}\big((f(x) - \mu_*(x))^2\big)$$

$$k_i^{\text{comp}}(x,x) = \underbrace{k(x,X)\boldsymbol{\Sigma}_i k(X,x)}_{\text{computational uncertainty}} \underbrace{=}_{\boldsymbol{\Sigma}_i = \mathrm{Cov}(v_*) = \mathbb{E}((v_* - v_i)(v_* - v_i)^\intercal)} \mathbb{E}\big((\mu_*(x) - \mu_i(x))^2\big)$$

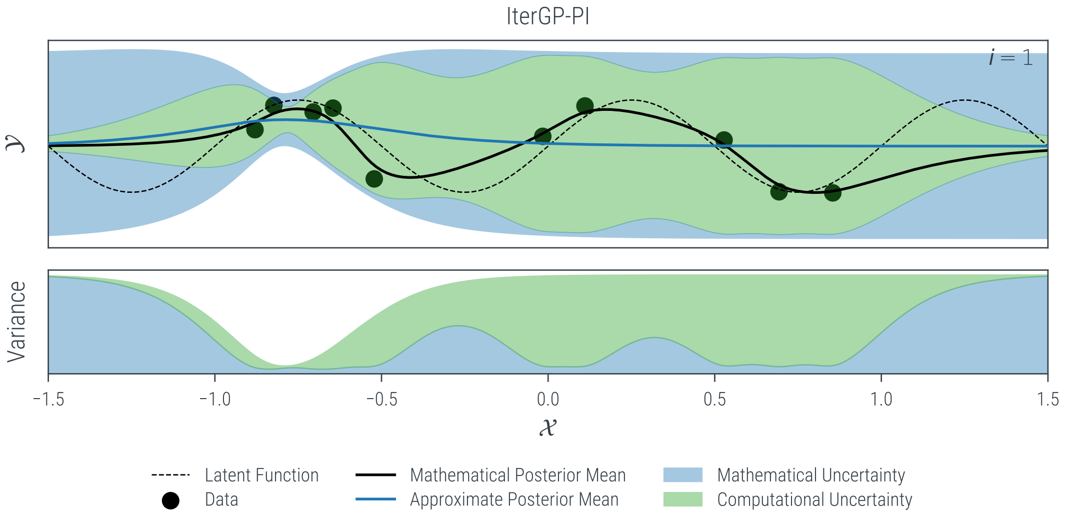
Computation-Aware GP Inference Illustrated

Interpreting computational and combined uncertainty as error quantification.



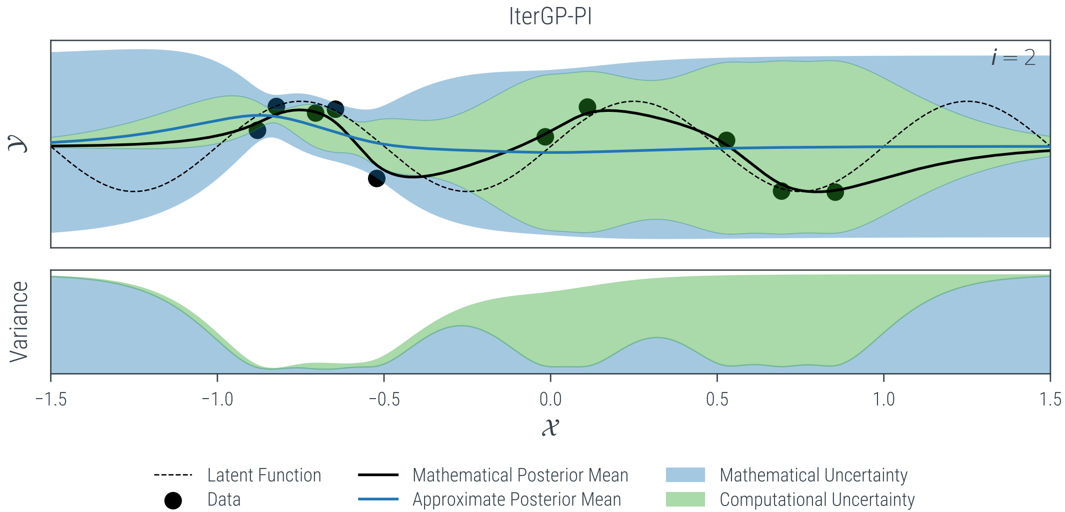
Computation-Aware GP Inference Illustrated

Interpreting computational and combined uncertainty as error quantification.



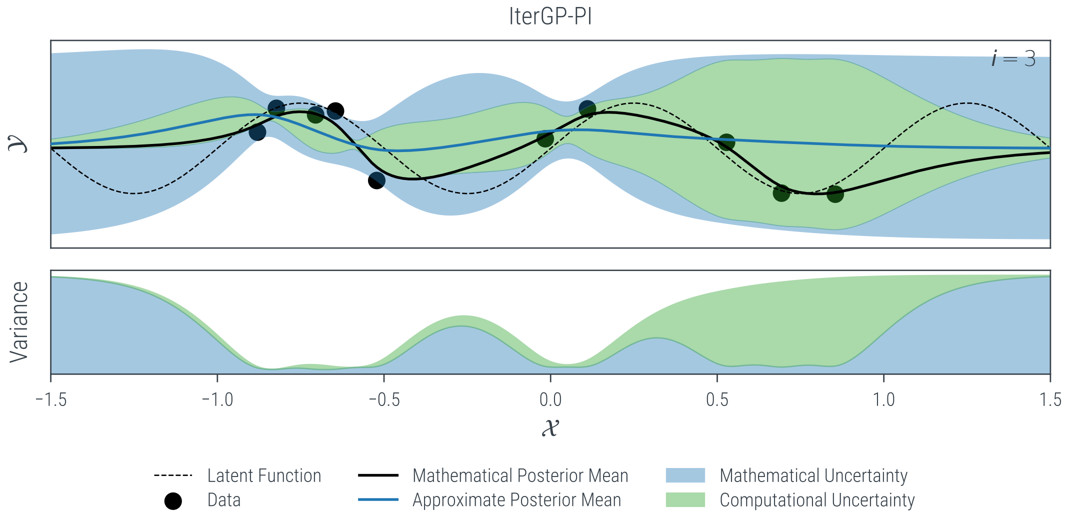
Computation-Aware GP Inference Illustrated

Interpreting computational and combined uncertainty as error quantification.



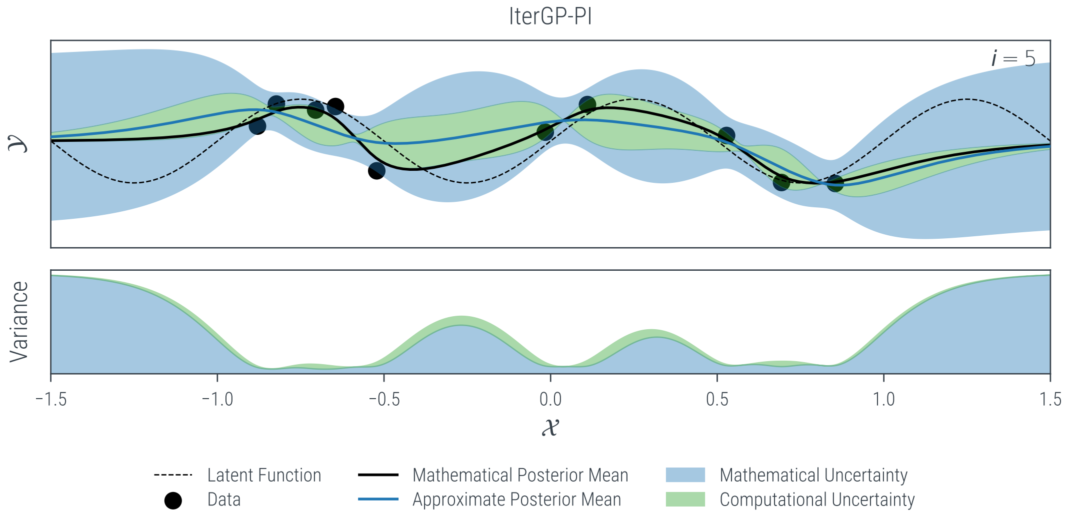
Computation-Aware GP Inference Illustrated

Interpreting computational and combined uncertainty as error quantification.



Computation-Aware GP Inference Illustrated

Interpreting computational and combined uncertainty as error quantification.



Theoretical Analysis

Uncertainty as a bound on the relative predictive error.

Theorem (Relative Error Bound)

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

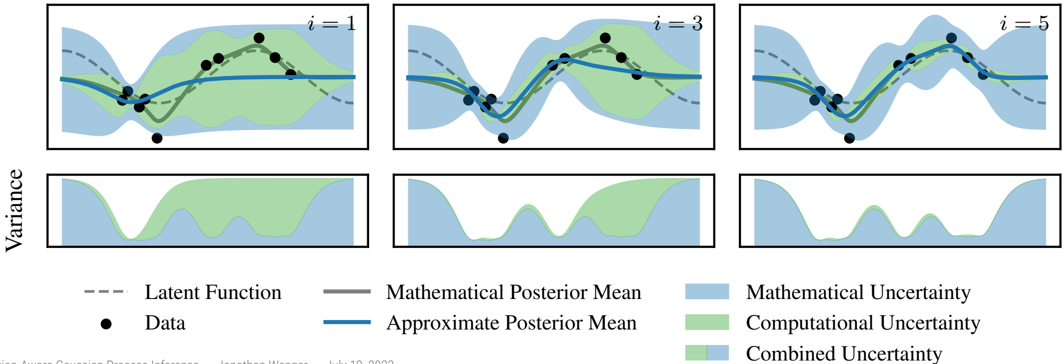
Theoretical Analysis

The combined uncertainty is a tight worst case bound on the relative error to the latent function.

(Wenger et al., 2022a)

Theorem (Relative Error Bound)

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



What Have We Learned?

So Far:

- Gaussian process inference is prohibitive for large datasets.

What Have We Learned?

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

So Far:

- Gaussian process inference is prohibitive for large datasets.
- Iterative methods can reduce the necessary computations from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$.

What Have We Learned?

So Far:

- Gaussian process inference is prohibitive for large datasets.
- Iterative methods can reduce the necessary computations from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$.
- Using probabilistic numerics we can quantify the error when approximating Gaussian processes.

What Have We Learned?

So Far:

- Gaussian process inference is prohibitive for large datasets.
- Iterative methods can reduce the necessary computations from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$.
- Using probabilistic numerics we can quantify the error when approximating Gaussian processes.
- **Explicit trade-off** between computation and uncertainty.

What Have We Learned?

So Far:

- Gaussian process inference is prohibitive for large datasets.
- Iterative methods can reduce the necessary computations from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$.
- Using probabilistic numerics we can quantify the error when approximating Gaussian processes.
- **Explicit trade-off** between computation and uncertainty.

What About:

- How does IterGP relate to other numerical (approximation) methods, e.g. Cholesky, CGGP, SVGP?

What Have We Learned?

So Far:

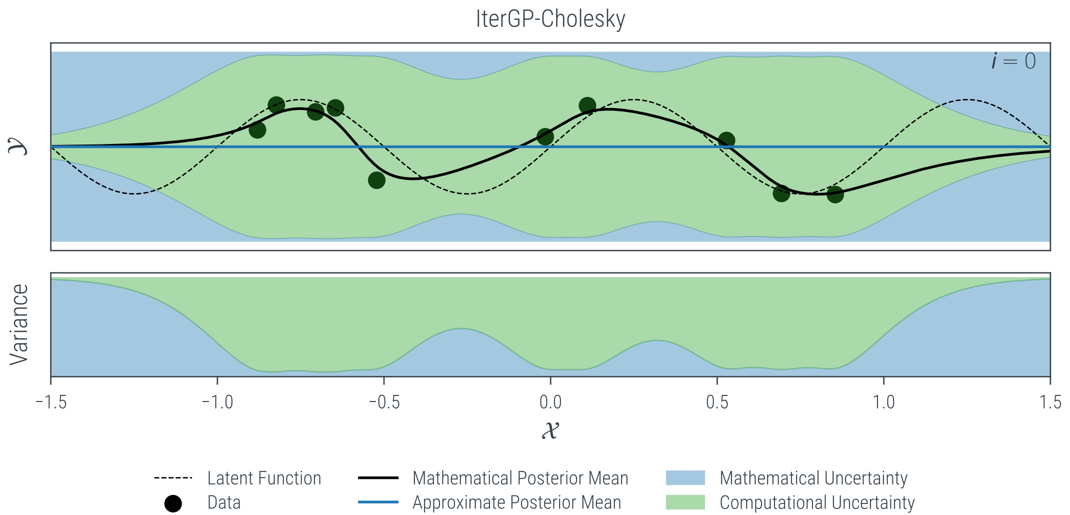
- Gaussian process inference is prohibitive for large datasets.
- Iterative methods can reduce the necessary computations from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$.
- Using probabilistic numerics we can quantify the error when approximating Gaussian processes.
- **Explicit trade-off** between computation and uncertainty.

What About:

- How does IterGP relate to other numerical (approximation) methods, e.g. Cholesky, CGGP, SVGP?
- Is quadratic time $\mathcal{O}(n^2)$ the limit? Can we approximate more cheaply?

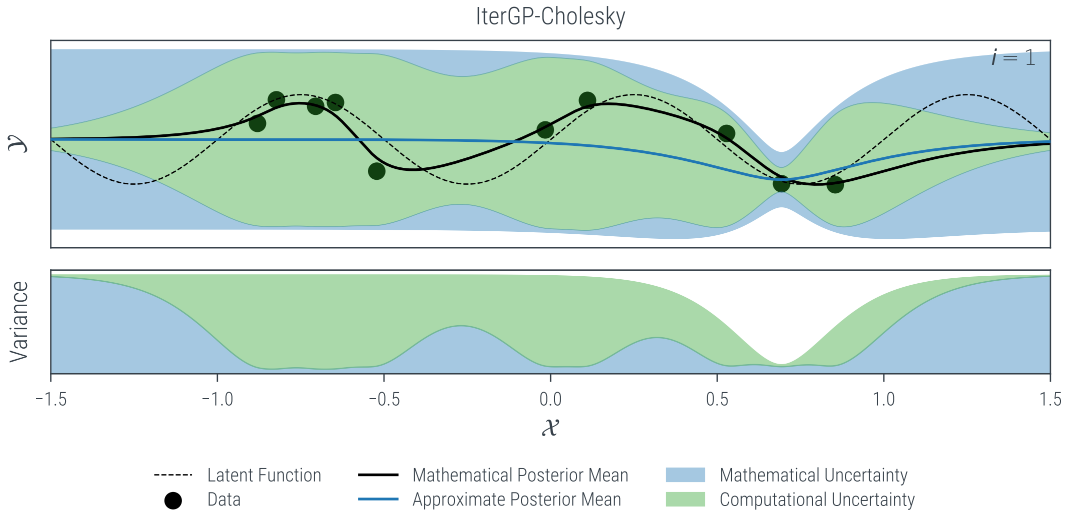
Subset of Data versus IterGP-Cholesky

IterGP with unit vector actions recovers vanilla GP inference.



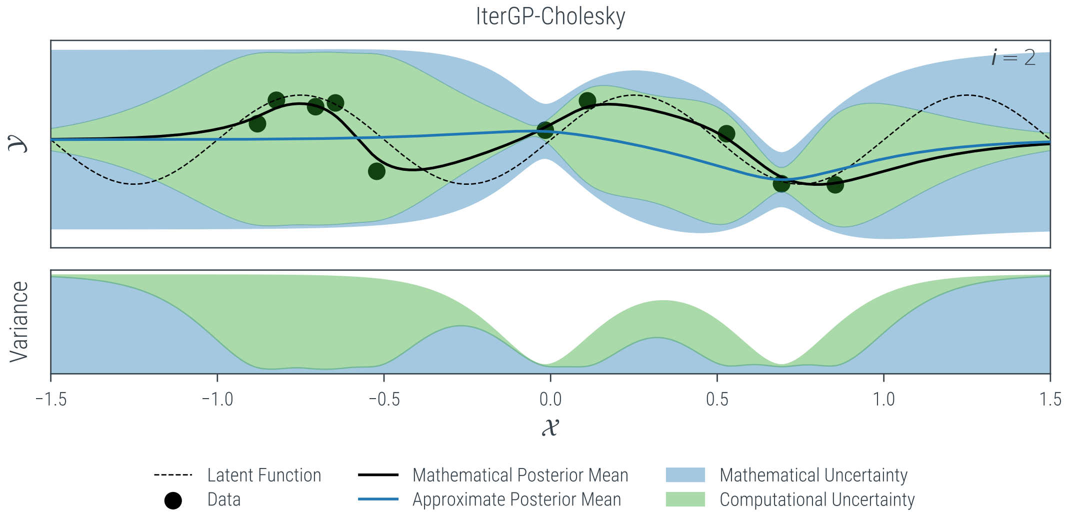
Subset of Data versus IterGP-Cholesky

IterGP with unit vector actions recovers vanilla GP inference.



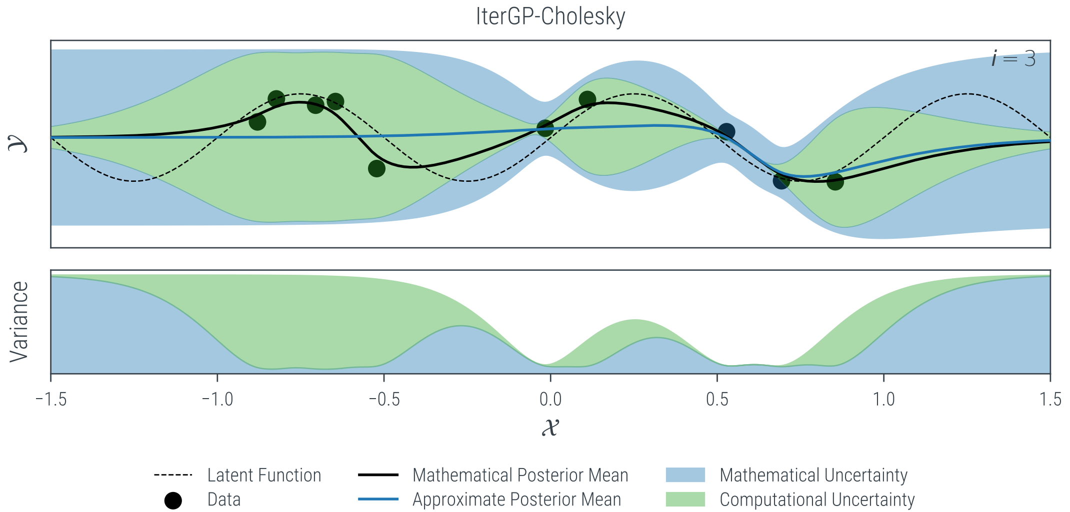
Subset of Data versus IterGP-Cholesky

IterGP with unit vector actions recovers vanilla GP inference.



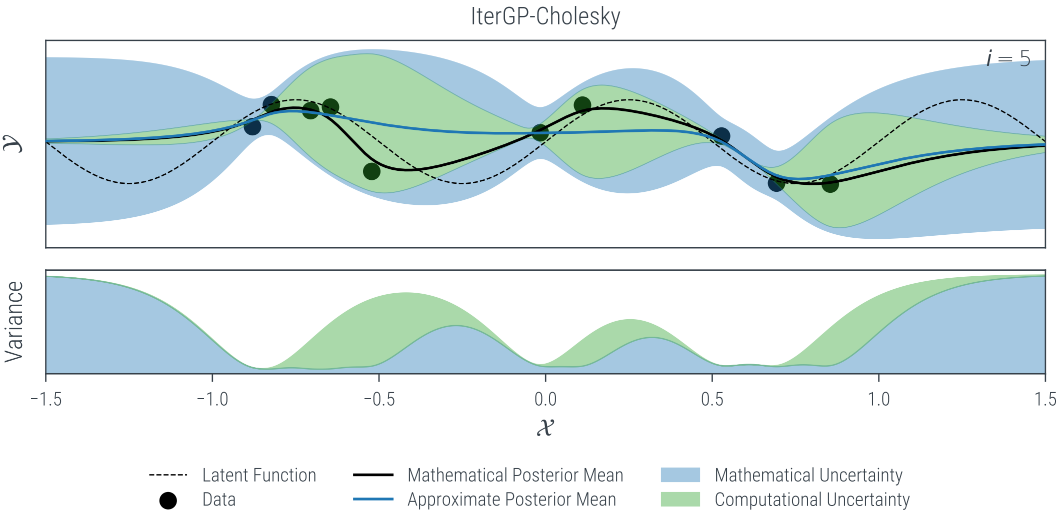
Subset of Data versus IterGP-Cholesky

IterGP with unit vector actions recovers vanilla GP inference.



Subset of Data versus IterGP-Cholesky

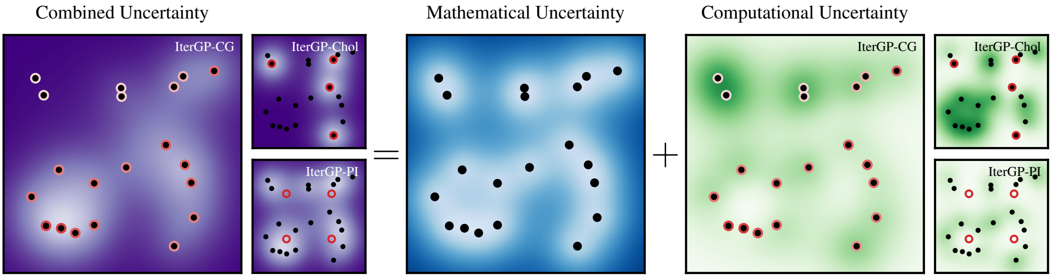
IterGP with unit vector actions recovers vanilla GP inference.



Policy Choice and Connection to Other Approximations

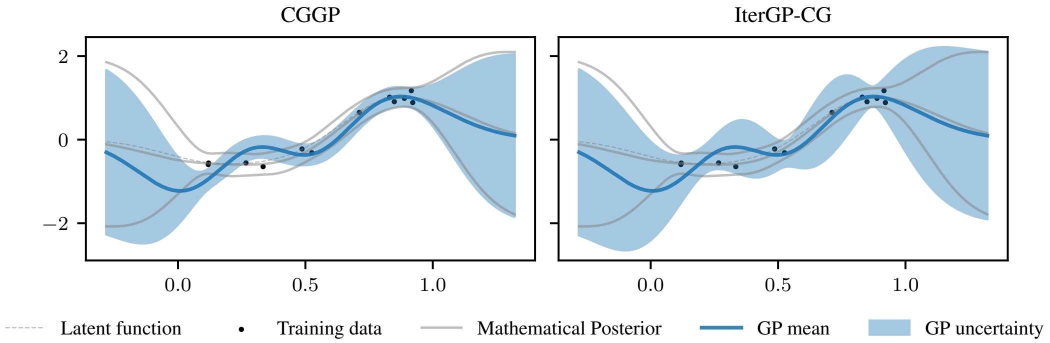
IterGP extends the most commonly used GP approximations to include computational uncertainty, with at most quadratic cost.

Method	Actions $\boldsymbol{s}_i$	Classic Analog
IterGP-Cholesky	$\boldsymbol{e}_i$	(Partial) Cholesky / subset of data
IterGP-EVD	$\mathrm{ev}_i(\hat{\boldsymbol{K}})$	(Partial) Eigenvalue decomposition
IterGP-CG	$\boldsymbol{s}_i^{\mathrm{PCG}}$ or $\hat{\boldsymbol{P}}^{-1}\boldsymbol{r}_i$	(Preconditioned) CG
IterGP-PseudoInput	$k(\boldsymbol{X}, \boldsymbol{z}_i)$	$\approx$ SVGP



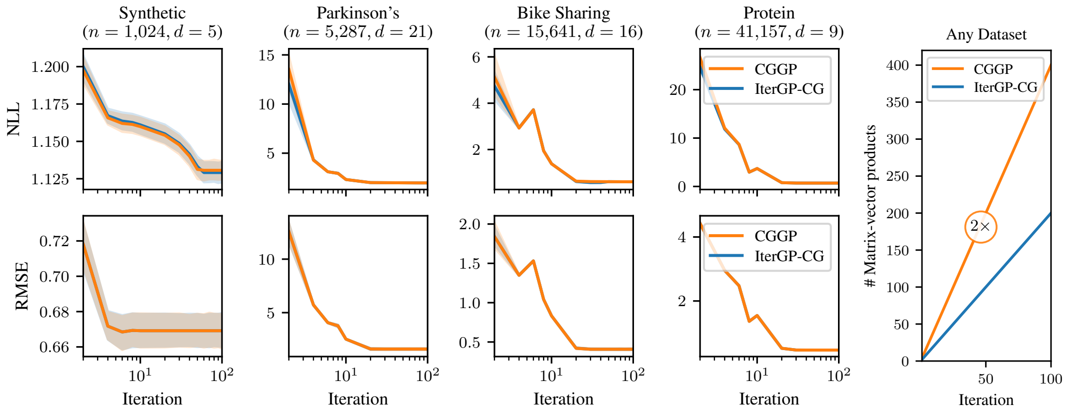
CGGP versus IterGP-CG

IterGP reduces the necessary computations for CG-based GP inference.



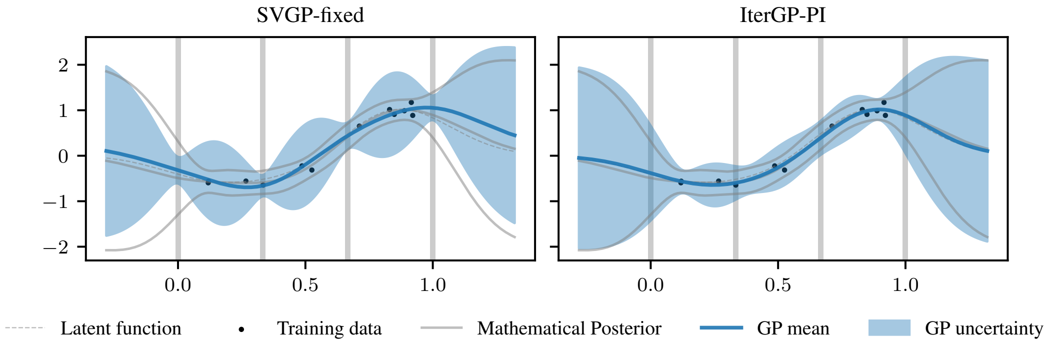
CGGP versus IterGP-CG

IterGP reduces the necessary computations for CG-based GP inference.



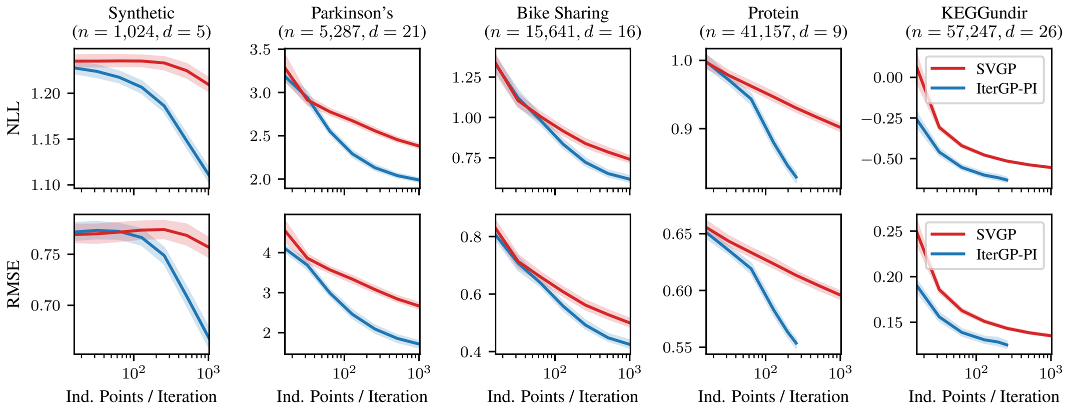
SVGP versus IterGP-PI

Quantifying computational uncertainty improves generalization of inducing point methods like SVGP (Titsias, 2009; Hensman et al., 2013).



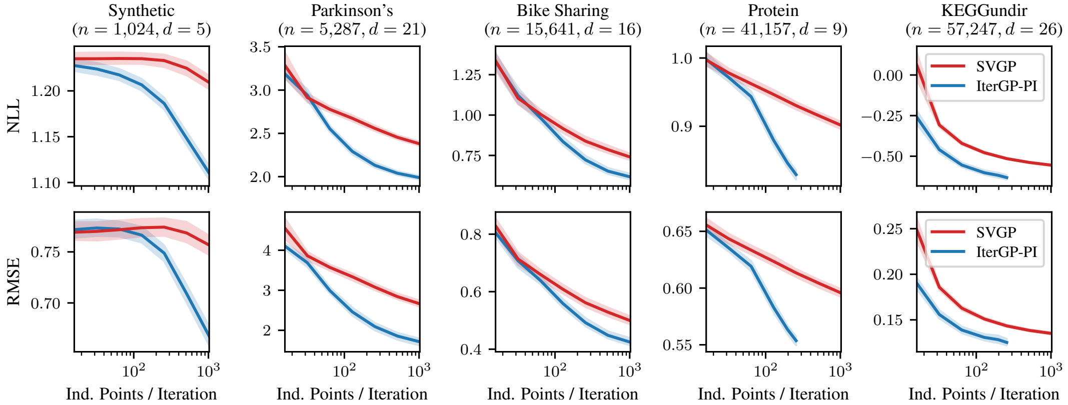
SVGP versus IterGP-PI

Quantifying computational uncertainty improves generalization of inducing point methods like SVGP (Titsias, 2009; Hensman et al., 2013).



SVGP versus IterGP-PI

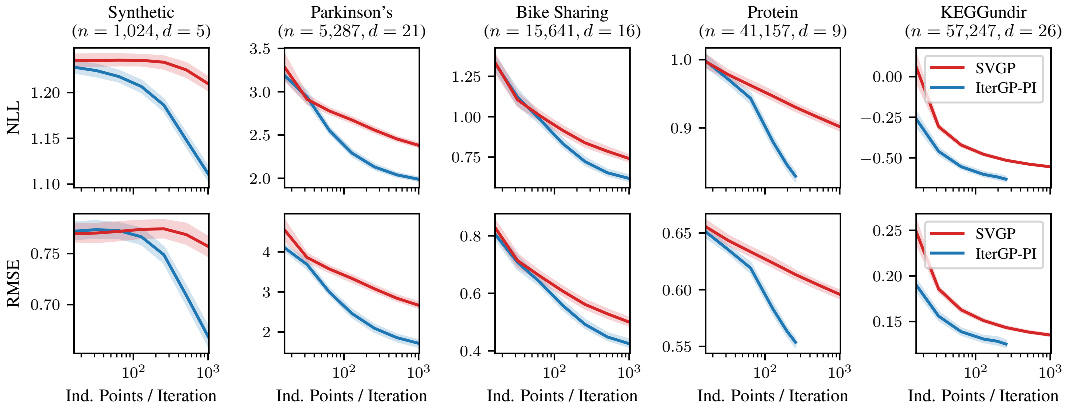
Quantifying computational uncertainty improves generalization of inducing point methods like SVGP (Titsias, 2009; Hensman et al., 2013).



What about optimizing inducing point locations?

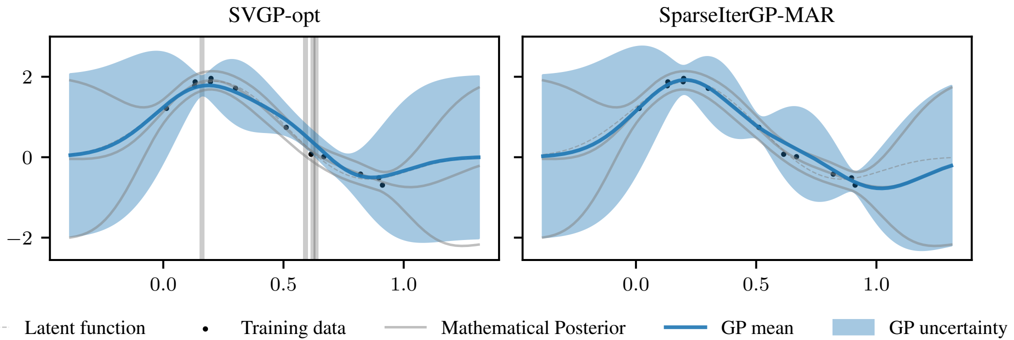
SVGP versus IterGP-PI

Quantifying computational uncertainty improves generalization of inducing point methods like SVGP (Titsias, 2009; Hensman et al., 2013).



What about **computational cost**? SVGP: $\mathcal{O}(nm^2)$ versus IterGP-PI: $\mathcal{O}(n^2 m)$.

Policy: Unit vector actions $\boldsymbol{s}_i = \boldsymbol{e}_j$ which select points greedily as $j = \arg\max \boldsymbol{r}_{i-1} \implies \mathcal{O}(nm^2)$.

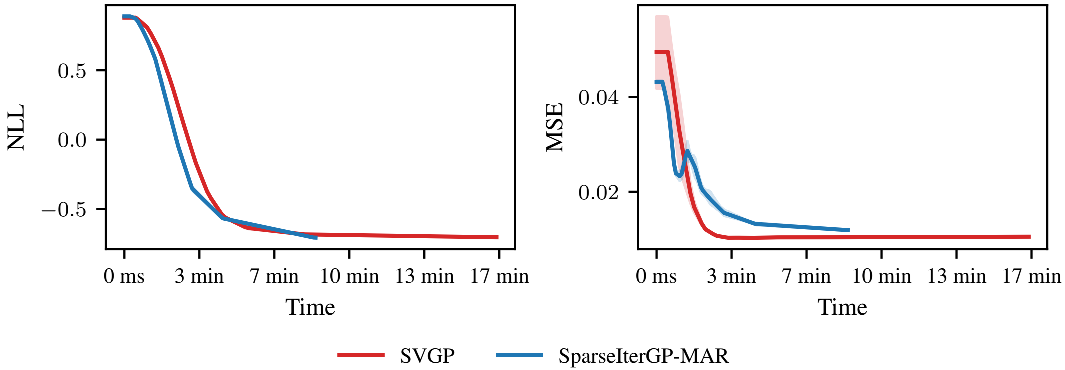


SVGP versus IterGP-MAR: Large-Scale Problem

Linear-time computation-aware GP inference with IterGP on a problem with $n \approx 10^5$ datapoints.

Unpublished work

Policy: Unit vector actions $\boldsymbol{s}_i = \boldsymbol{e}_j$ which select points greedily as $j = \arg\max \boldsymbol{r}_{i-1} \implies \mathcal{O}(nm^2)$.

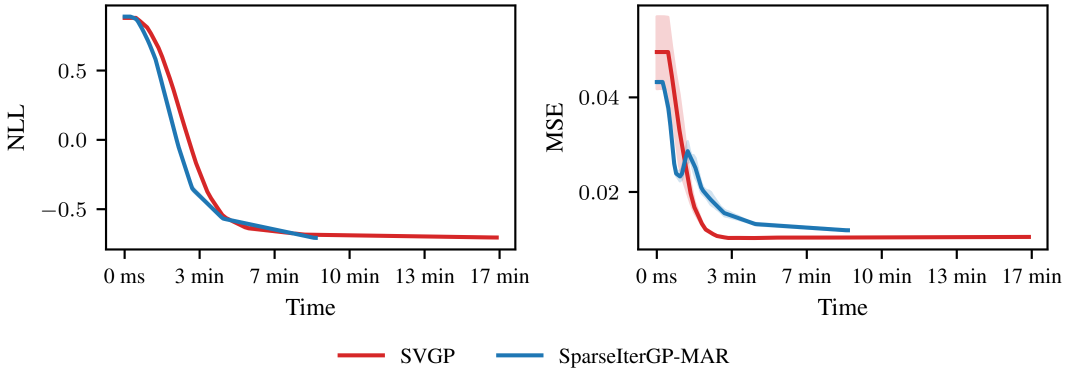


SVGP versus IterGP-MAR: Large-Scale Problem

Linear-time computation-aware GP inference with IterGP on a problem with $n \approx 10^5$ datapoints.

Unpublished work

Policy: Unit vector actions $\boldsymbol{s}_i = \boldsymbol{e}_j$ which select points greedily as $j = \arg\max \boldsymbol{r}_{i-1} \implies \mathcal{O}(nm^2)$.



Scalable GP approximation **without inadvertently comprimising uncertainty quantification.**

Bonus: Getting Philosophical

Blurring the lines between data and computation.

Working with Infinite Data

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

(Wenger et al., 2022a)

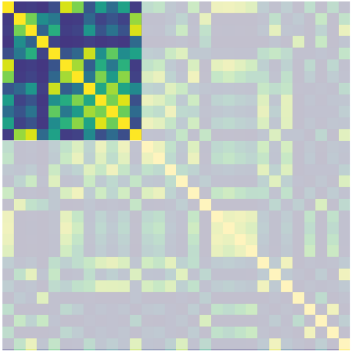
Theorem (Online GP Approximation with IterGP)

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

$$\tilde{\boldsymbol{s}}_i = \begin{pmatrix} \boldsymbol{s}_i \\ \boldsymbol{0} \end{pmatrix} \tag{2}$$

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 $\tilde{\boldsymbol{s}}_i$:

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



Data is as Data Does

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.

Data is as Data Does

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 from this perspective.

Data is as Data Does

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 from this perspective.

What if we modelled this situation with a Gaussian process?

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

Data is as Data Does

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 from this perspective.

What if we modelled this situation with a Gaussian process?

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

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

Summary

Takeaways

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

Summary

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Takeaways

- Large-scale models are often as much about the approximation as they are about the data.
- **Uncertainty** arises from **limited data** and from **limited computation**.

Summary

Takeaways

- Large-scale models are often as much about the approximation as they are about the data.
- **Uncertainty** arises from **limited data** and from **limited computation**.
- For GPs, we can exactly quantify the approximation error given arbitrary resources $\implies$ IterGP.

Summary

Takeaways

- Large-scale models are often as much about the approximation as they are about the data.
- **Uncertainty** arises from **limited data** and from **limited computation**.
- For GPs, we can exactly quantify the approximation error given arbitrary resources $\implies$ IterGP.
- **Explicit trade-off** between computation and uncertainty.

Summary

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Takeaways

- Large-scale models are often as much about the approximation as they are about the data.
- **Uncertainty** arises from **limited data** and from **limited computation**.
- For GPs, we can exactly quantify the approximation error given arbitrary resources $\implies$ IterGP.
- **Explicit trade-off** between computation and uncertainty.

Open Questions

- Model selection / hyperparameter optimization?

Summary

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Takeaways

- Large-scale models are often as much about the approximation as they are about the data.
- **Uncertainty** arises from **limited data** and from **limited computation**.
- For GPs, we can exactly quantify the approximation error given arbitrary resources $\implies$ IterGP.
- **Explicit trade-off** between computation and uncertainty.

Open Questions

- Model selection / hyperparameter optimization?
- Policy design for downstream tasks and decision making problems.
  - Active learning
  - Bayesian optimization
  - ...

Summary

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Takeaways

- Large-scale models are often as much about the approximation as they are about the data.
- **Uncertainty** arises from **limited data** and from **limited computation**.
- For GPs, we can exactly quantify the approximation error given arbitrary resources $\implies$ IterGP.
- **Explicit trade-off** between computation and uncertainty.

Open Questions

- Model selection / hyperparameter optimization?
- Policy design for downstream tasks and decision making problems.
  - Active learning
  - Bayesian optimization
  - …
- Extension to non-Gaussian likelihoods.

Additional Material

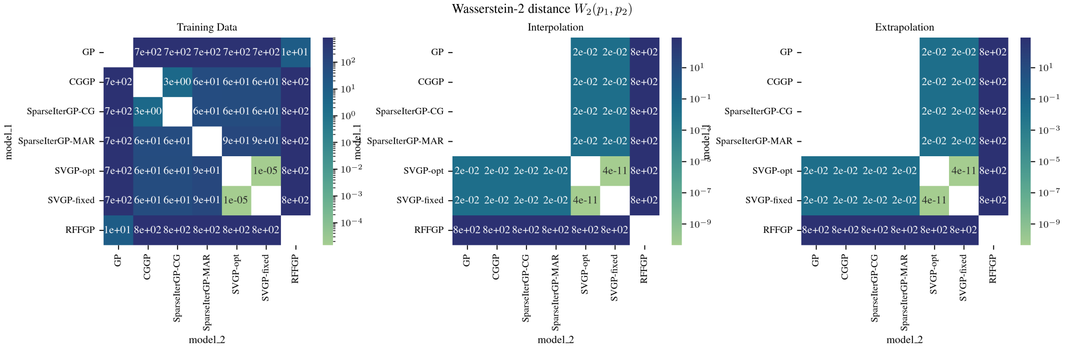
Comparison of GP Approximations

Gaussian Process Classification

Large-scale Model Selection

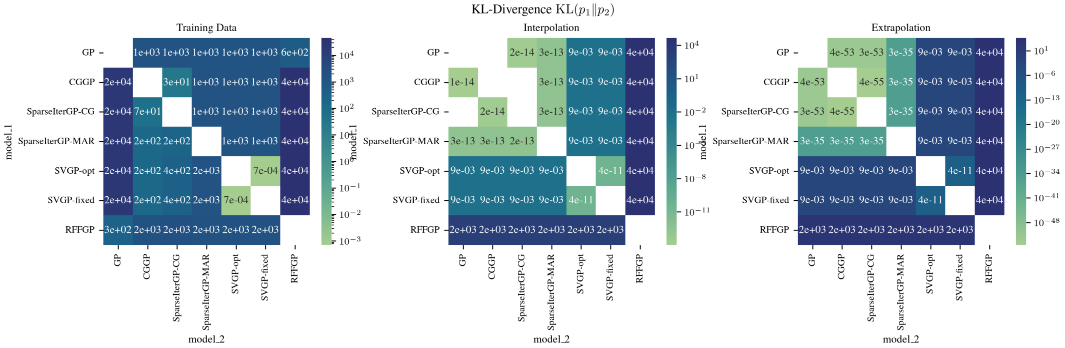
Comparison of GP Approximations: Wasserstein-2 Distance

Comparison of different GP approximations at the training data, for interpolation and extrapolation.



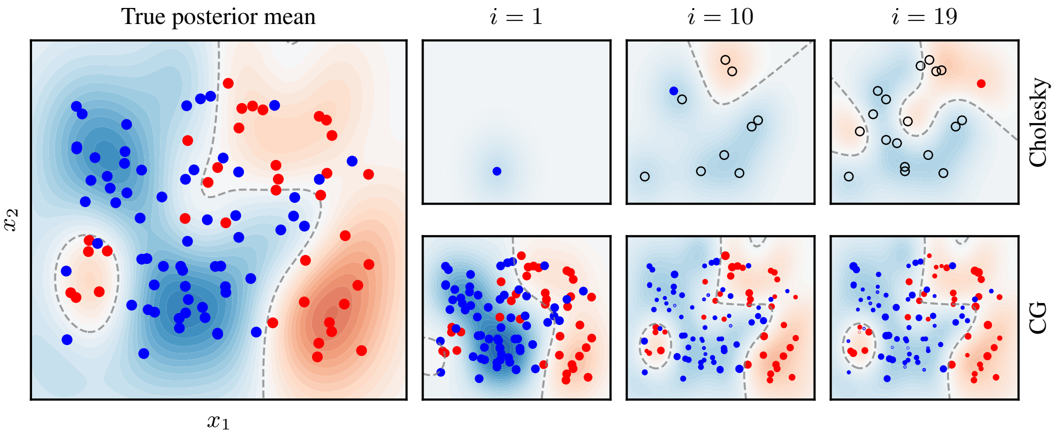
Comparison of GP Approximations: KL-Divergence

Comparison of different GP approximations at the training data, for interpolation and extrapolation.



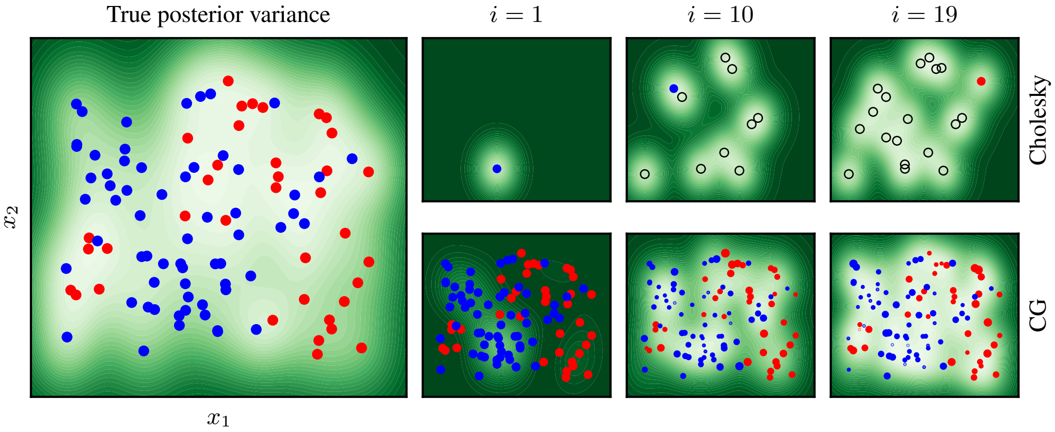
Gaussian Process Classification

Extension to non-Gaussian likelihoods via Laplace Approximation.



Gaussian Process Classification

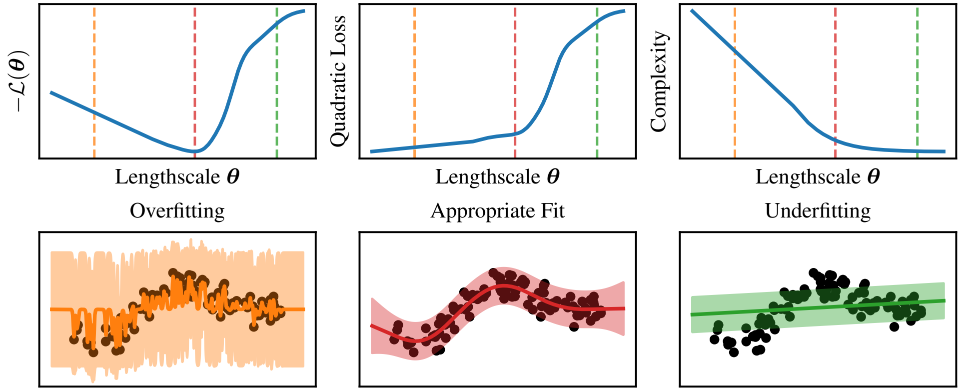
Extension to non-Gaussian likelihoods via Laplace Approximation.



Model Selection for Gaussian Processes

We can identify kernel hyperparameters by optimizing the log-marginal likelihood.

$$\boldsymbol{\theta}_* = \arg\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{y} \mid \boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} \Big(\underbrace{(\boldsymbol{y}-\boldsymbol{\mu})^\intercal \hat{\boldsymbol{K}}^{-1}(\boldsymbol{y}-\boldsymbol{\mu})}_{\text{quadratic loss}} + \underbrace{\log\det(\hat{\boldsymbol{K}})}_{\text{model complexity}} \Big)$$



Large-scale GP Hyperparameter Optimization

A numerical linear algebra bottleneck.

(Ubaru et al., 2017; Gardner et al., 2018)

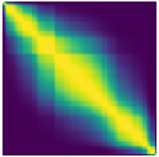
Need to: Evaluate log-marginal likelihood and its derivative repeatedly.

- log-marginal likelihood $\mathcal{L}(\boldsymbol{\theta}) = -\frac{1}{2}\left(\boldsymbol{y}^\intercal \hat{K}^{-1}\boldsymbol{y} + \log\det(\hat{K}) + n\log(2\pi)\right)$
- derivative $\frac{\partial}{\partial\theta}\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2}\boldsymbol{y}^\intercal \hat{K}^{-1}\frac{\partial\hat{K}}{\partial\theta}\hat{K}^{-1}\boldsymbol{y} - \frac{1}{2}\operatorname{tr}(\hat{K}^{-1}\frac{\partial\hat{K}}{\partial\theta})$

Challenge: Computationally costly operations with the kernel matrix.

- linear solves $\boldsymbol{v} \mapsto \hat{K}^{-1}\boldsymbol{v}$
- matrix traces $\log\det(\hat{K}) = \operatorname{tr}(\log(\hat{K}))$ and $\operatorname{tr}(\hat{K}^{-1}\frac{\partial\hat{K}}{\partial\boldsymbol{\theta}_i})$

$\hat{K} =$



$n \times n$

Linear solves and matrix traces can be computed solely via *matrix-vector multiplication*!

This is great because ...

- matrix-vector multiplies have complexity $\mathcal{O}(n^2)$.
- structured or sparse matrices are efficient to multiply with.
- the kernel matrix does not need to be stored in memory explicitly (Charlier et al., 2021).
- we can exploit parallelization and modern hardware (GPUs).

lower time and space complexity

Preconditioning

How to encode and leverage structural prior knowledge about matrices.

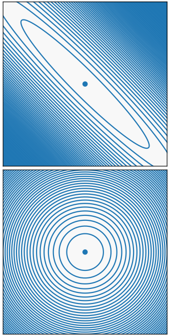
EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Preconditioner

$$\hat{P} \approx \hat{K}$$

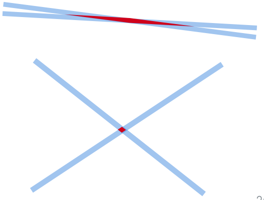
such that $\kappa(\hat{P}^{-1}\hat{K}) \ll \kappa(\hat{K})$ and $\hat{P}$ is computationally tractable.

- Computing and storing $\hat{P}$ is cheap.
- Linear solves $v \mapsto \hat{P}^{-1}v$ are efficient.
- Derived properties, such as the determinant or spectrum are known.



Asymptotic approx. error $g(\ell) \to 0$ of sequence of preconditioners $\hat{P}_\ell \to \hat{K}$:

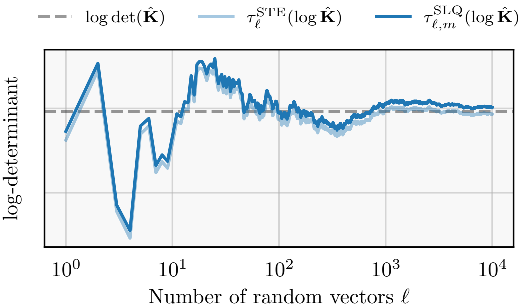
$$\kappa(\hat{P}_\ell^{-1}\hat{K}) \leq (1 + \mathcal{O}(g(\ell))\|\hat{K}\|_F)^2$$



Known Use: Accelerate and stabilize linear solves via CG $\Rightarrow$ bias reduction

Stochastic Trace Estimation

Computing matrix traces $\operatorname{tr}(f(\hat{K}))$ via matrix-vector multiplication. (Hutchinson, 1989; Golub et al., 2009; Ubaru et al., 2017)



$$\operatorname{tr}(f(\hat{K})) = n\mathbb{E}[z_i^\top f(\hat{K}) z_i]$$
$$\approx \tau_\ell^{\mathrm{STE}}(f(\hat{K})) = \frac{n}{\ell}\sum_{i=1}^{\ell} z_i^\top f(\hat{K}) z_i$$
$$\approx \tau_{\ell,m}^{\mathrm{SLQ}}(f(\hat{K}))$$

Problems:

- Worst-case convergence in the number of random vectors is $\mathcal{O}(\ell^{-\frac{1}{2}})$
- Introduces stochasticity into hyperparameter optimization

$\implies$ slows down training

Preconditioned Log-Determinant Estimation

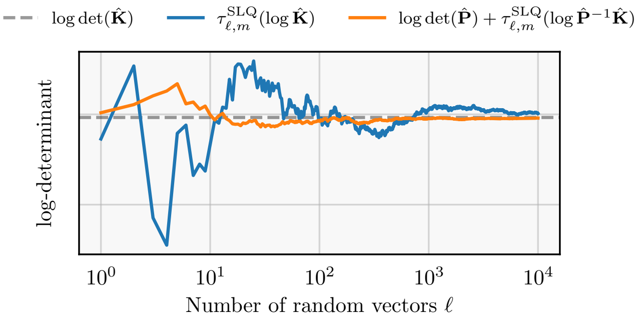
Variance-reduced stochastic trace estimation via preconditioning.

(Wenger et al., 2022b)

Idea: Decompose log-determinant into deterministic and stochastic approximation.

$$\log\det(\hat{K}) = \log\det(\hat{P}_\ell \hat{P}_\ell^{-1}\hat{K}) = \underbrace{\log\det(\hat{P}_\ell)}_{\text{known}} + \underbrace{\operatorname{tr}(\log(\hat{K}) - \log(\hat{P}_\ell))}_{\approx \text{ stochastic trace estimate}}$$

The better the preconditioner, the smaller the stochastic approximation $\Rightarrow$ variance reduction



- Backward pass analogously via automatic differentiation.
- If we compute a preconditioner for CG, we can simply reuse it at negligible overhead.
- If $\hat{P}_\ell \to \hat{K}$ at rate $g(\ell)$, then the STE only requires $\mathcal{O}(\ell^{-\frac{1}{2}} g(\ell))$ random vectors.

Convergence Rates for Kernel – Preconditioner Combinations

The faster the preconditioner converges to the kernel matrix (i.e. $g(\ell) \to 0$) the fewer random vectors are needed.

If $\hat{P}_\ell \to \hat{K}$ at rate $g(\ell)$, then the STE only requires $\mathcal{O}(\ell^{-\frac{1}{2}} g(\ell))$ random vectors.

Kernel	d	Preconditioner	$g(\ell)$	Condition
any	$\mathbb{N}$	none	1	
any	$\mathbb{N}$	truncated SVD	$\ell^{-\frac{1}{2}}$	
any	$\mathbb{N}$	random. SVD	$\ell^{-\frac{1}{2}} + \mathcal{O}(\ell^{\frac{1}{4}} s^{-\frac{1}{4}})$	w/ high prob. for s samples
any	$\mathbb{N}$	random. Nyström	$\ell^{-\frac{1}{2}} + \mathcal{O}(\ell^{\frac{1}{4}} s^{-\frac{1}{4}})$	w/ high prob. for s samples
any	$\mathbb{N}$	RFF	$\ell^{-\frac{1}{2}}$	w/ high prob.
RBF	1	partial Cholesky	$\exp(-c\ell)$	for some $c > 0$
RBF	$\mathbb{N}$	QFF	$\exp(-b\ell^{\frac{1}{d}})$	for some $b > 0$ if $\ell^{\frac{1}{d}} > 2\gamma^{-2}$
Matérn(ν)	$\mathbb{N}$	partial Cholesky	$\ell^{-(\frac{2\nu}{d}+1)}$	$2\nu \in \mathbb{N}$, maximin ordering **Schaefer2021a**
Matérn(ν)	1	QFF	$\ell^{-(s(\nu)+1)}$	where $s(\nu) \in \mathbb{N}$
mod. Matérn(ν)	$\mathbb{N}$	QFF	$\ell^{-\frac{s(\nu)+1}{d}}$	where $s(\nu) \in \mathbb{N}$
additive	$\mathbb{N}$	any	$dg(\ell)$	all summands have rate $g(\ell)$
any	$\mathbb{N}$	any kernel approx.	$g(\ell)$	$\exists$ uniform convergence bound

Theoretical Guarantees

Probabilistic error bounds for the estimates of the log-marginal likelihood and its derivative.

Theorem (Log-marginal likelihood)

[...] Then with probability $1-\delta$, the error in the estimate η of the log-marginal likelihood $\mathcal{L}$ satisfies

$$|\eta - \mathcal{L}| \leq \varepsilon_{\mathrm{CG}} + \tfrac{1}{2}(\varepsilon_{\mathrm{Lanczos}} + \varepsilon_{\mathrm{STE}})\|\log(\hat{K})\|_F,$$

where the individual errors are bounded by

$$\varepsilon_{\mathrm{CG}}(\kappa, i) \leq K_3 \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^i \tag{3}$$

$$\varepsilon_{\mathrm{Lanczos}}(\kappa, i) \leq K_1 \left(\frac{\sqrt{2\kappa+1}-1}{\sqrt{2\kappa+1}+1}\right)^{2i} \tag{4}$$

$$\boxed{\varepsilon_{\mathrm{STE}}(\delta, \ell) \leq C_1 \sqrt{\log(\delta^{-1})}\ell^{-\frac{1}{2}} g(\ell)} \tag{5}$$

Theorem (Derivative)

[...] Then with probability $1-\delta$, the error in the estimate ϕ of the derivative of the log-marginal likelihood $\frac{\partial}{\partial\theta}\mathcal{L}$ satisfies

$$|\phi - \tfrac{\partial}{\partial\theta}\mathcal{L}| \leq \varepsilon_{\mathrm{CG}} + \tfrac{1}{2}(\varepsilon_{\mathrm{CG}'} + \varepsilon_{\mathrm{STE}})\|\hat{K}^{-1}\tfrac{\partial\hat{K}}{\partial\theta}\|_F$$

where the individual errors are bounded by

$$\varepsilon_{\mathrm{CG}}(\kappa, i) \leq K_4 \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^i \tag{6}$$

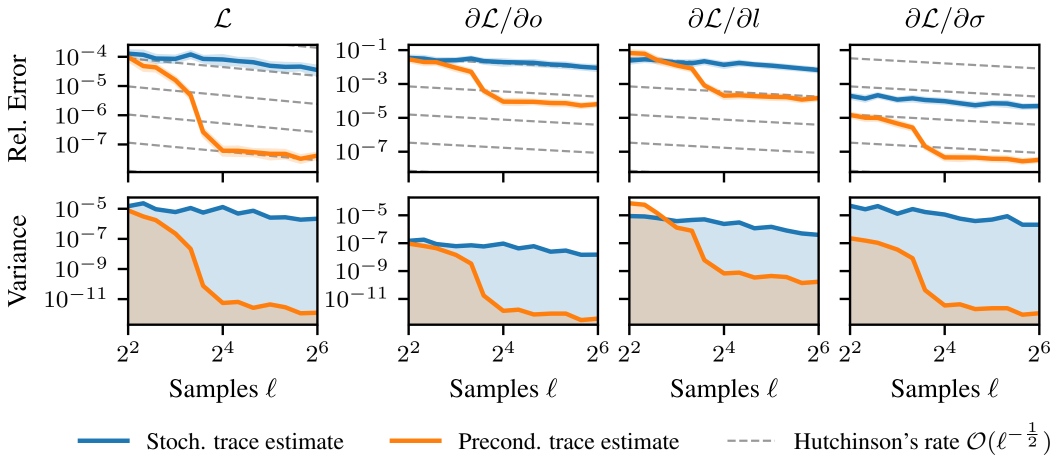
$$\varepsilon_{\mathrm{CG}'}(\kappa, i) \leq K_2 \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^i \tag{7}$$

$$\boxed{\varepsilon_{\mathrm{STE}}(\delta, \ell) \leq C_1 \sqrt{\log(\delta^{-1})}\ell^{-\frac{1}{2}} g(\ell)} \tag{8}$$

We leverage preconditioning not only to reduce bias, but crucially also to reduce variance.

Preconditioning Reduces Bias and Variance

Estimating the log-marginal likelihood and its derivatives on synthetic data.

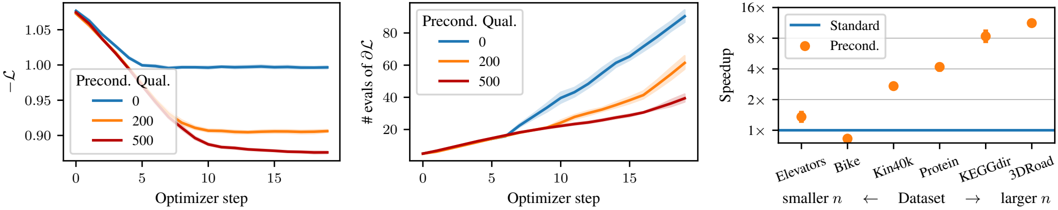


Experiment Details:

- Randomly sampled synthetic data ($n = 10{,}000$, $d = 1$)
- RBF kernel with noise scale $\sigma^2 = 10^{-2}$

Preconditioning Accelerates Hyperparameter Optimization

Gaussian process hyperparameter optimization on UCI data.



(a) Training loss (Protein).

(b) Line search computations (Protein).

(c) Speedup on UCI datasets.

Experiment Details:

- UCI datasets ($n = 12{,}449$ to $n = 326{,}155$)
- Matérn($\frac{3}{2}$) kernel with noise scale $\sigma^2 = 10^{-2}$
- Partial Cholesky preconditioner of size 500
- $\ell = 50$ random vectors

References I

- Alireza Radmanesh, Matthew J. Muckley, Tullie Murrell, Emma Lindsey, Anuroop Sriram, Florian Knoll, Daniel K. Sodickson, and Yvonne W. Lui. “Exploring the Acceleration Limits of Deep Learning Variational Network–based Two-dimensional Brain MRI”. In: *Radiology: Artificial Intelligence* 4.6 (2022). DOI: `10.1148/ryai.210313` (cit. on pp. 2–5).
- Jacob R Gardner, Geoff Pleiss, David Bindel, Kilian Q Weinberger, and Andrew Gordon Wilson. “GPyTorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration”. In: *Advances in Neural Information Processing Systems (NeurIPS)* (2018) (cit. on pp. 18–20, 90).
- Benjamin Charlier, Jean Feydy, Joan Alexis Glaunès, François-David Collin, and Ghislain Durif. “Kernel Operations on the GPU, with Autodiff, without Memory Overflows”. In: *Journal of Machine Learning Research* 22.74 (2021), pp. 1–6 (cit. on pp. 18–20, 90).
- Philipp Hennig. “Probabilistic Interpretation of Linear Solvers”. In: *SIAM Journal on Optimization* 25.1 (2015), pp. 234–260 (cit. on pp. 22–24).
- Jon Cockayne, Chris J. Oates, Ilse C.F. Ipsen, and Mark 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. 22–24).

References II

- Jonathan Wenger and Philipp Hennig. "Probabilistic Linear Solvers for Machine Learning". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020 (cit. on pp. 22–24).
- Jonathan Wenger, Geoff Pleiss, Marvin Pförtner, Philipp Hennig, and John P. Cunningham. "Posterior and Computational Uncertainty in Gaussian Processes". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2022 (cit. on pp. 25–28, 34–39, 49, 72).
- Michalis Titsias. "Variational learning of inducing variables in sparse Gaussian processes". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2009 (cit. on pp. 64–67).
- James Hensman, Nicolò Fusi, and Neil D Lawrence. "Gaussian processes for big data". In: *Conference on Uncertainty in Artificial Intelligence (UAI)*. 2013 (cit. on pp. 64–67).
- Shashanka Ubaru, Jie Chen, and Yousef Saad. "Fast estimation of $\operatorname{tr}(f(A))$ via stochastic Lanczos quadrature". In: *SIAM Journal on Matrix Analysis and Applications* 38.4 (2017), pp. 1075–1099 (cit. on pp. 90, 92).
- Michael F Hutchinson. "A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines". In: *Communications in Statistics-Simulation and Computation* 18.3 (1989), pp. 1059–1076 (cit. on p. 92).

References III

- Gene H Golub and Gérard Meurant. *Matrices, moments and quadrature with applications*. Vol. 30. Princeton University Press, 2009 (cit. on p. 92).
- Jonathan Wenger, Geoff Pleiss, Philipp Hennig, John P. Cunningham, and Jacob R. Gardner. "Preconditioning for Scalable Gaussian Process Hyperparameter Optimization". In: *International Conference on Machine Learning (ICML)*. 2022 (cit. on p. 93).