Bayesian optimization of time-varying functions

Petar M. Djurić

Department of Electrical Engineering
Stony Brook University
Stony Book, NY

(*In collaboration with Fernando Llorente*)

October 30, 2024

Some geography first

Outline of the presentation

Problem statement

- Objective: Time-varying optimization (TVO), i.e., solving

$$\boldsymbol{x}_t^{\mathtt{opt}} = \arg\max_{\boldsymbol{x}\in\mathcal{X}_t} f_t(\boldsymbol{x}), \quad f_t(\boldsymbol{x}) : \mathcal{X}_t \subset \mathbb{R}^{d_x} \mapsto \mathbb{R}, \quad t \in \mathbb{N}$$

- Assumptions: $f_t(\boldsymbol{x})$ is unknown, $f_t(\boldsymbol{x})$ is continuous for every t, $f_t(\boldsymbol{x})$ is slowly varying over time.
- Allowance: The estimation of $\mathbf{x}_t^{\mathtt{opt}}$ for $t = 1, 2, \ldots$ is carried out by probing the function N times at each step.

Previous works

Previous works of TVO and BO:

- C. Cruz, J. R. González, and D. A. Pelta. Optimization in dynamic environments: a survey on problems, methods and measures. *Soft Computing*, 15:1427–1448, 2011.

  provides review of non-probabilistic approaches for TVO.

- I. Bogunovic, J. Scarlett, and V. Cevher. Time-varying Gaussian process bandit optimization. *Artificial Intelligence and Statistics*, pp. 314–323. PMLR, 2016.

  introduces algorithms for TV-BO, but does not use sparse, online approaches.

- F. M. Nyikosa, M. A. Osborne, and S. J. Roberts. Bayesian optimization for dynamic problems. *arXiv preprint arXiv*:1803.03432, 2018.

  This paper also presents algorithms for TV-BO, but these algorithms do not use sparse, online approaches.

- Q. Lu, K. D. Polyzos, B. Li, and G. B. Giannakis. Surrogate modeling for Bayesian optimization beyond a single Gaussian process. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

  uses sparse, online approaches for BO but *not* in the TVO setting.

- S. Van Vaerenbergh, M. Lázaro-Gredilla, and I. Santamaría. Kernel recursive least-squares tracker for time-varying regression. *IEEE Transactions on Neural Networks and Learning Systems*, 23(8):1313–1326, 2012.

  considers learning a time-varying function instead of TVO.

Contributions

Our contributions:

1. We use a Bayesian approach for solving the TVO problem based on online Gaussian processes with forgetting.
2. We study different forgetting mechanisms in this framework.
3. We propose the use of windows of past data to build the time-varying acquisition function.

Applications

Some real-world problems where we need to perform TVO

– *robot/drone/self-driving vehicles learning in changing environments* (adaptations needed because of changing obstacles, shifting weather conditions, or varying traffic. TV-BO helps optimize paths, control parameters, or sensor placements in real time)

– *smart grids and energy systems* (needed for optimization of energy generation and consumption in real-time to balance supply and demand)

Applications (cont'd)

– *financial markets* (adaptation to changing market conditions; TV-BO enables dynamic portfolio optimization)

– *wireless networks and communications* (used to adapt to changing network traffic, interference patterns, and user demand, resource allocation)

– *healthcare and medical imaging* (adaptive radiation therapy, electrical stimulation of the brain)

– *aerospace engineering* (flight path optimization, satellite trajectory control, rocket launch and re-entry optimization, UAV swarm coordination)

Gaussian processes

Gaussian processes (GPs) are a model for learning unknown functions

$$f : \ \boldsymbol{x} \mapsto y$$

from observed input-outputs $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$.

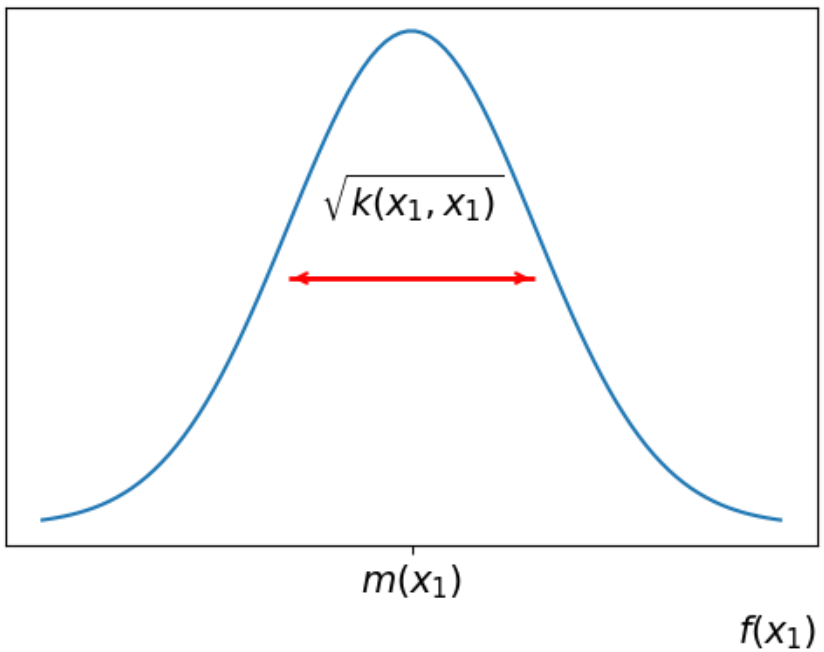
Gaussian processes

GPs are the generalization of Gaussian distributions to functions (infinite dimensional):

$$f \sim GP(m, k)$$

where m is the mean function and k is the covariance function.

$$f(\boldsymbol{x}_1) \sim N(m(\boldsymbol{x}_1), k(\boldsymbol{x}_1, \boldsymbol{x}_1))$$



Gaussian processes

GPs are attractive for many reasons:

Bayesian	☺
Nonparametric	☺
Gaussian	☺
Kernel design	☺

However, they have drawbacks:

Cost	☹
Kernel design	☹

Bayesian inference

Given some data $\boldsymbol{y}$, we aim to infer the unknown $\boldsymbol{\theta}$ by computing

$$p(\boldsymbol{\theta}|\boldsymbol{y}) \propto p(\boldsymbol{y}|\boldsymbol{\theta}) \times p(\boldsymbol{\theta})$$

$$\text{posterior} \propto \text{likelihood} \times \text{prior}$$

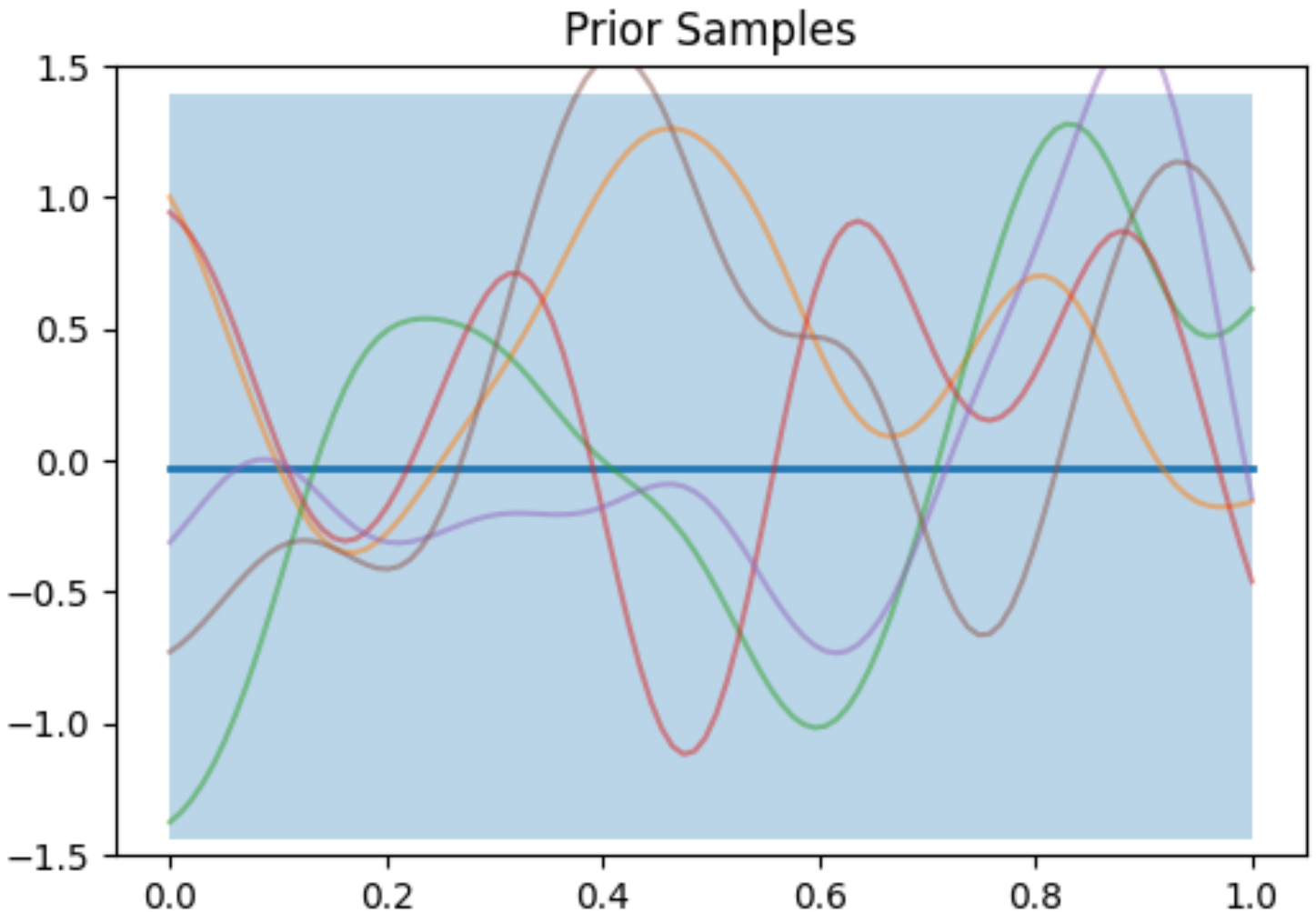
In GPs,

$$p(f|\boldsymbol{y}) \propto p(\boldsymbol{y}|f) \times p(f)$$

$$\text{GP posterior} \propto \text{likelihood} \times \text{GP prior}$$

Example of GPs

The distribution over plausible functions shrinks as we gather more data



Kernels

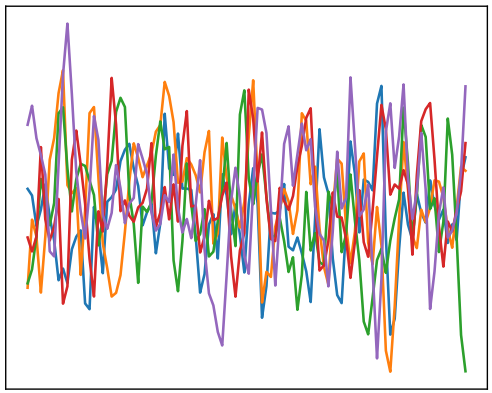
The covariance between function values is expressed by a kernel function, e.g., RBF kernel:

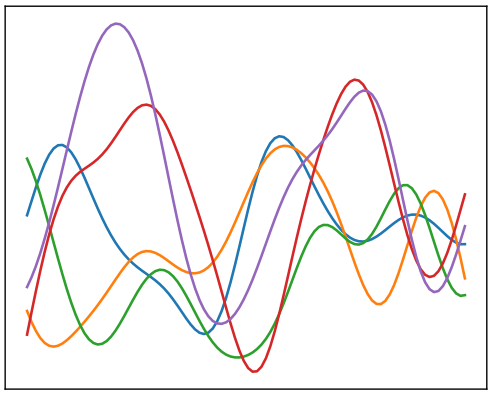
$$\text{Cov}(f(x), f(x')) = k(x, x'; \boldsymbol{\lambda}) = \sigma_f^2 \exp\left(-\frac{(x - x')^2}{2\ell^2}\right)$$

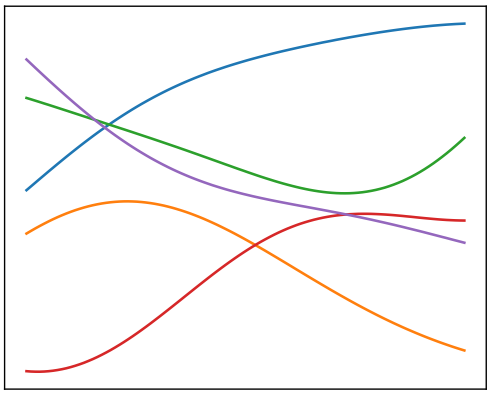
where $\boldsymbol{\lambda} = [\sigma_f, \ell]$ are hyperparameters.

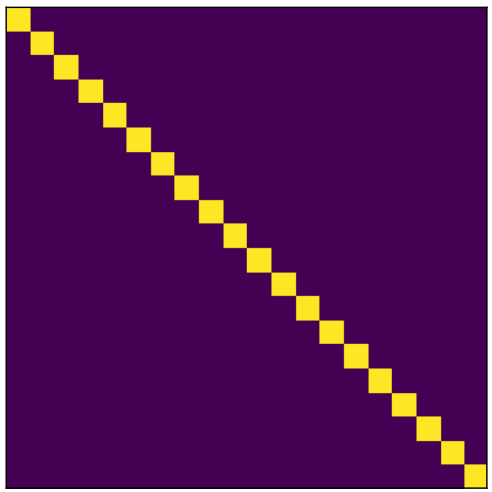
- Covariance functions include strong assumptions about $f(\boldsymbol{x})$.
- Hyperparameters allow to interpret the data.

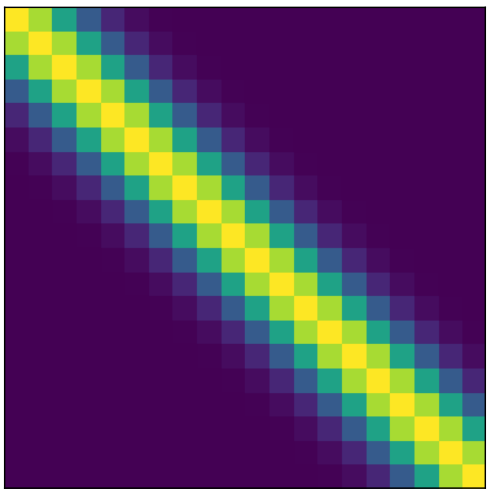
Kernels

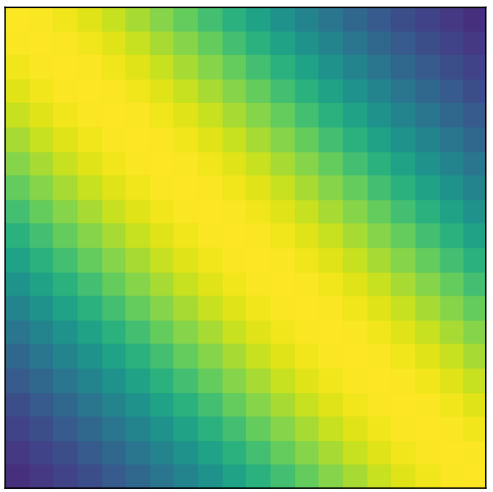












$\ell = 0.01$ $\ell = 0.1$ $\ell = 0.5$

Learning of hyperparameters

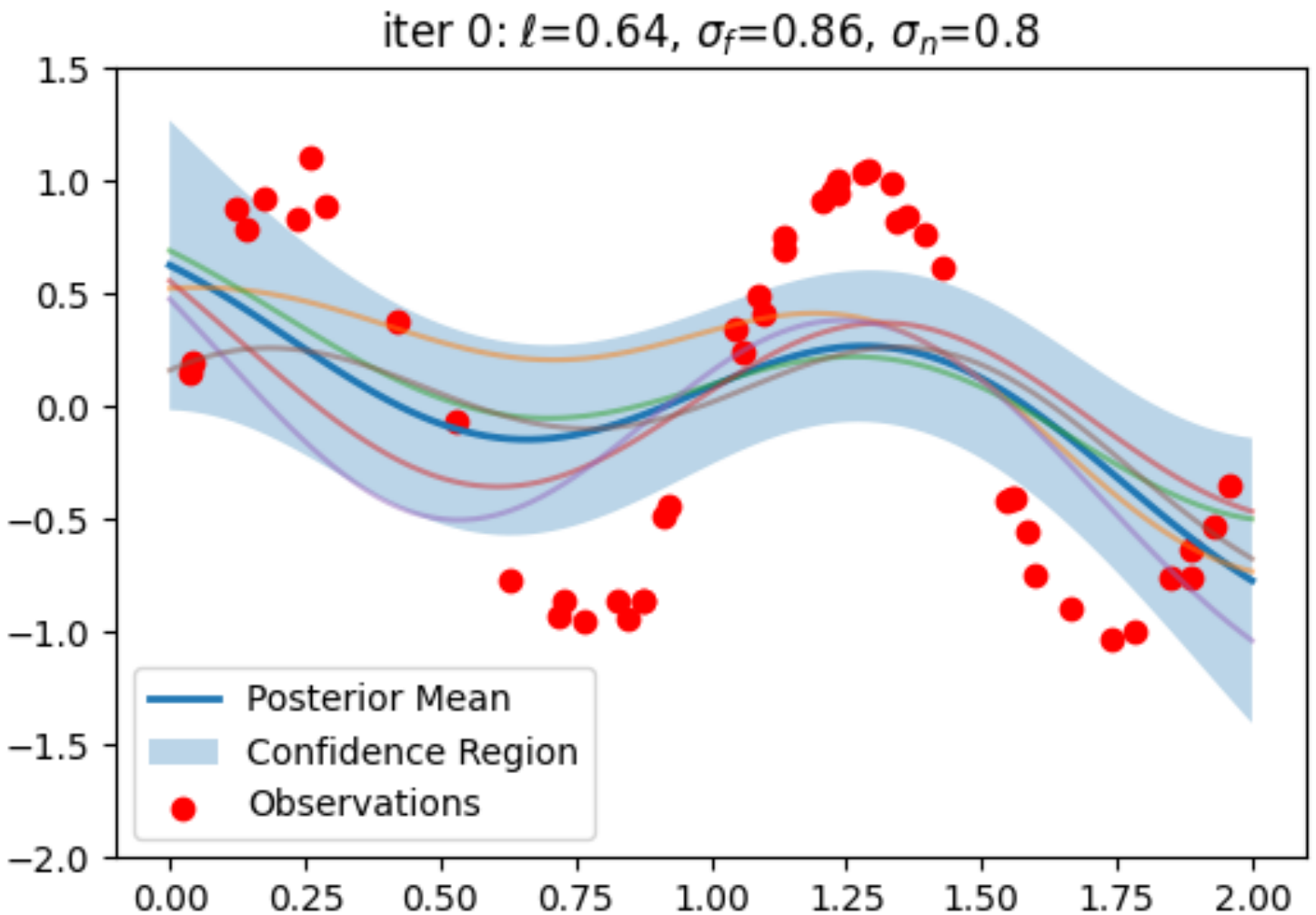
The performance of GPs depends a lot on the hyperparameters. ☹

Bayesian model selection: choose them by

$$\boldsymbol{\lambda}^* = \arg\max_{\boldsymbol{\lambda}} p(\boldsymbol{y}|\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\lambda}} -\log p(\boldsymbol{y}|\boldsymbol{\lambda})$$

The derivatives $\nabla_{\boldsymbol{\lambda}} \log p(\boldsymbol{y}|\boldsymbol{\lambda})$ are available so we can apply gradient descent algorithms. ☺

Learning of hyperparameters



Computation costs

GPs require the inversion of a $N \times N$ kernel matrix

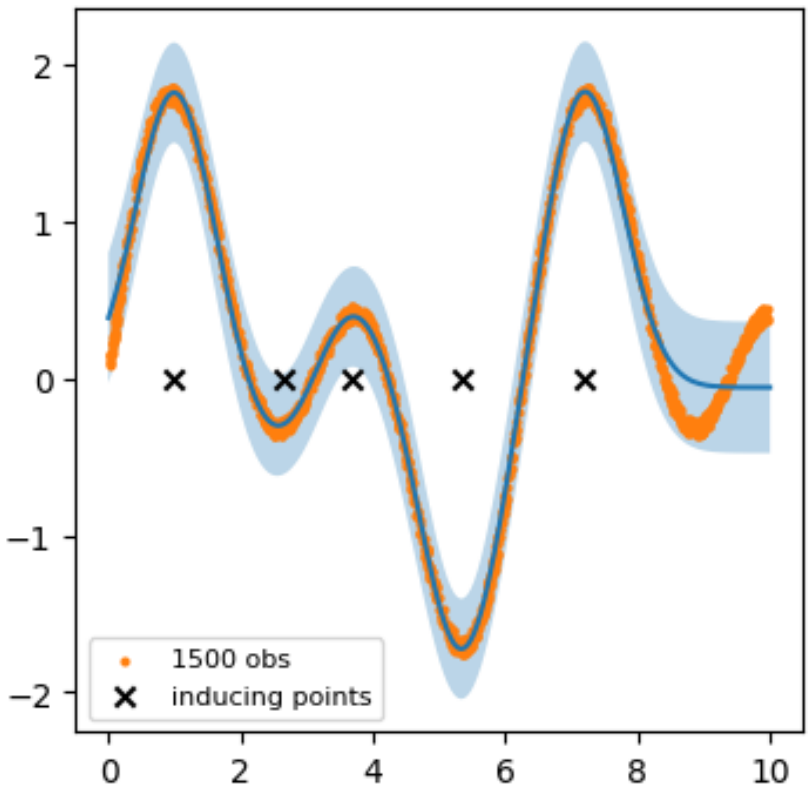
$$\boldsymbol{K} = \begin{bmatrix} k(x_1, x_1) & \dots & k(x_1, x_N) \\ \vdots & \ddots & \dots \\ k(x_N, x_1) & \dots & k(x_N, x_N) \end{bmatrix}$$

The cost scales cubically with the number of datapoints N. Several successful approaches exist for reducing this cost, based on:

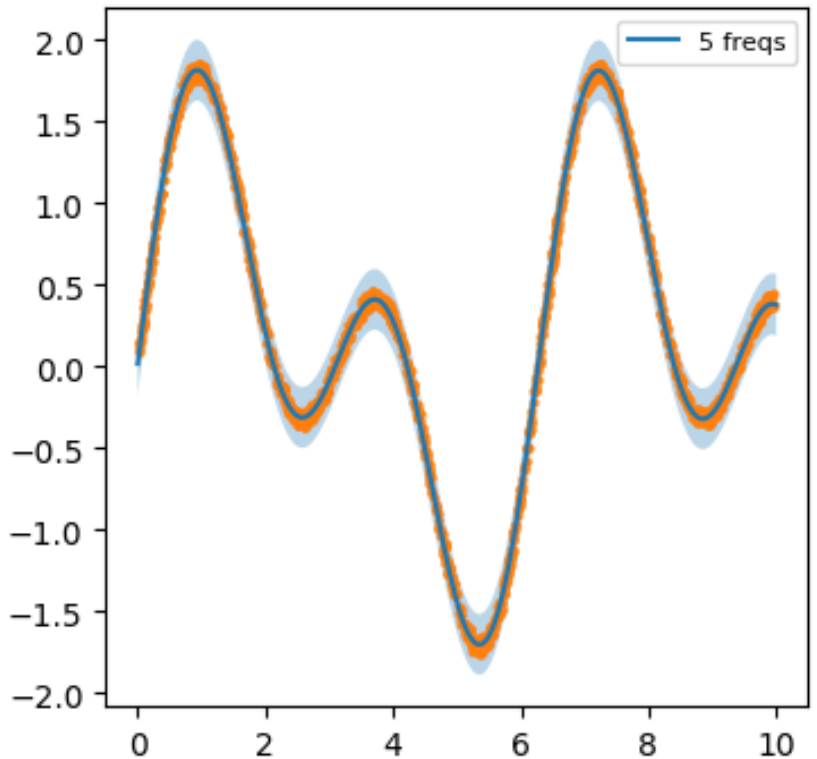
- inducing points
- random-feature expansions

Computation costs

Inducing points



Random features



Random-feature Gaussian process

When $k(\boldsymbol{x}, \boldsymbol{x}')$ is a (normalized) stationary kernel,

$$k(\boldsymbol{x}, \boldsymbol{x}') = \int e^{i\mathbf{v}^\top(\boldsymbol{x}-\boldsymbol{x}')} p(\mathbf{v}) d\mathbf{v}$$

we can sample a set of random Fourier features $\{\mathbf{v}_j\}_{j=1}^J \sim p(\mathbf{v})$ from the power spectral density of the kernel to build the map

$$\boldsymbol{\phi}(\boldsymbol{x}) = \frac{1}{\sqrt{J}}[\sin(\boldsymbol{x}^\top\mathbf{v}_1), \cos(\boldsymbol{x}^\top\mathbf{v}_1), \ldots, \sin(\boldsymbol{x}^\top\mathbf{v}_J), \cos(\boldsymbol{x}^\top\mathbf{v}_J)]^\top \in \mathbb{R}^{2J}$$

and obtain a low-rank approximation of the kernel matrix

$$\mathbf{K} \approx \sigma^2 \boldsymbol{\Phi}^\top \boldsymbol{\Phi}$$

$$\boldsymbol{\Phi} = [\boldsymbol{\phi}(\boldsymbol{x}_1), \ldots, \boldsymbol{\phi}(\boldsymbol{x}_N)] \in \mathbb{R}^{2J \times N}$$

Random-feature Gaussian process

For (normalized) RBF kernel

$$k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}')^\top \Lambda^{-1}(\boldsymbol{x} - \boldsymbol{x}')\right)$$

the power spectral density is

$$p(\mathbf{v}) = \mathcal{N}\left(\mathbf{0}, \Lambda^{-1}\right)$$

where

$$\Lambda = \text{diag}\left([\ell_1^2, \ldots, \ell_d^2]\right)$$

Random-feature Gaussian process

RF-GP is a Bayesian linear regression model with parameters $\boldsymbol{\theta} \in \mathbb{R}^{2J}$,

$$y = \boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\theta} + \epsilon$$

where

$$p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\mathbf{0}, \sigma_\theta^2 \mathbf{I}_{2J})$$
$$p(y|\boldsymbol{\theta}, \boldsymbol{x}) = \mathcal{N}(y|\boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\theta}, \sigma_{\text{obs}}^2)$$
$$p(\boldsymbol{\theta}|\boldsymbol{y}, \boldsymbol{X}) \propto p(\boldsymbol{\theta}) \prod_{i=1}^{N} p(y_i|\boldsymbol{\theta}, \boldsymbol{x}_i) \propto \mathcal{N}(\boldsymbol{\theta}|\widehat{\boldsymbol{\theta}}, \widehat{\boldsymbol{\Sigma}})$$

so

$$p(y_\star|\boldsymbol{y}, \boldsymbol{x}_\star) = \int p(y_\star|\boldsymbol{\theta}, \boldsymbol{x}_\star) p(\boldsymbol{\theta}|\boldsymbol{y}, \boldsymbol{X}) d\boldsymbol{\theta} = \mathcal{N}(y_\star|\widehat{y}(\boldsymbol{x}_\star), \widehat{\sigma}^2(\boldsymbol{x}_\star))$$

Random-feature Gaussian process

Bayesian update: prediction + update

$$p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t+1}, \boldsymbol{X}_{1:t+1}) = \frac{p(y_{t+1}|\boldsymbol{\theta}, \boldsymbol{x}_{t+1})p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t}, \boldsymbol{X}_{1:t})}{p(y_{t+1}|\boldsymbol{y}_{1:t}, \boldsymbol{X}_{1:t+1})}$$

is possible in closed-form with constant per-step computation cost.

Random-feature Gaussian process

- **Prediction step.** Let $\boldsymbol{\Phi}_t = [\boldsymbol{\phi}(\boldsymbol{x}_1) \dots \boldsymbol{\phi}(\boldsymbol{x}_t)] \in \mathbb{R}^{2J \times t}$. The predictive density is

$$p(y_{t+1}|\mathcal{D}_t, \boldsymbol{x}_{t+1}) = \mathcal{N}(y_{t+1}|\widehat{y}_{t+1}, \widehat{\sigma}^2_{t+1})$$
$$\widehat{y}_{t+1} = \boldsymbol{\phi}(\boldsymbol{x}_{t+1})^\top \widehat{\boldsymbol{\theta}}_t$$
$$\widehat{\sigma}^2_{t+1} = \boldsymbol{\phi}(\boldsymbol{x}_{t+1})^\top \widehat{\boldsymbol{\Sigma}}_t \boldsymbol{\phi}(\boldsymbol{x}_{t+1}) + \sigma^2_{\text{obs}}$$

where $\widehat{\boldsymbol{\Sigma}}_t = \left(\frac{1}{\sigma^2_{\text{obs}}} \boldsymbol{\Phi}_t \boldsymbol{\Phi}_t^\top + \frac{1}{\sigma^2_\theta} \mathbf{I}_{2J} \right)^{-1}$ and $\widehat{\boldsymbol{\theta}}_t = \frac{1}{\sigma^2_{\text{obs}}} \widehat{\boldsymbol{\Sigma}}_t \boldsymbol{\Phi}_t \boldsymbol{y}_t$ are the moments of the posterior $p(\boldsymbol{\theta}|\mathcal{D}_t) = \mathcal{N}(\boldsymbol{\theta}|\widehat{\boldsymbol{\theta}}_t, \widehat{\boldsymbol{\Sigma}}_t)$.

Random-feature Gaussian process

- **Update step.** Upon receiving $(\boldsymbol{x}_{t+1}, y_{t+1})$, the update formulae are

$$\widehat{\boldsymbol{\theta}}_{t+1} = \widehat{\boldsymbol{\theta}}_t + \widehat{\sigma}_{t+1}^{-2} \widehat{\boldsymbol{\Sigma}}_t \boldsymbol{\phi}(\boldsymbol{x}_{t+1})(y_{t+1} - \widehat{y}_{t+1})$$
$$\widehat{\boldsymbol{\Sigma}}_{t+1} = \widehat{\boldsymbol{\Sigma}}_t - \widehat{\sigma}_{t+1}^{-2} \widehat{\boldsymbol{\Sigma}}_t \boldsymbol{\phi}(\boldsymbol{x}_{t+1}) \boldsymbol{\phi}(\boldsymbol{x}_{t+1})^\top \widehat{\boldsymbol{\Sigma}}_t$$

Time dynamics

- We consider a general Markov transition model where, *before* receiving the $t+1$ datapoint, $\boldsymbol{\theta}_{t+1}$ is a linear combination of the parameter at time t, a drift term, and random noise.
- Letting $\boldsymbol{\theta}_0 \sim p_0(\boldsymbol{\theta}_0)$. For $t = 0, 1, \ldots$, the parameter vector evolves as

$$\boldsymbol{\theta}_{t+1} = a_1 \boldsymbol{\theta}_t + a_2 \mathbf{m}_t + a_3 \mathbf{u}_t$$

where $\boldsymbol{\theta}_t \sim p(\boldsymbol{\theta}_t | \mathcal{D}_t)$, $\mathbf{m}_t$ is a time-varying vector, and $\mathbf{u}_t$ is a noise process.

Time dynamics

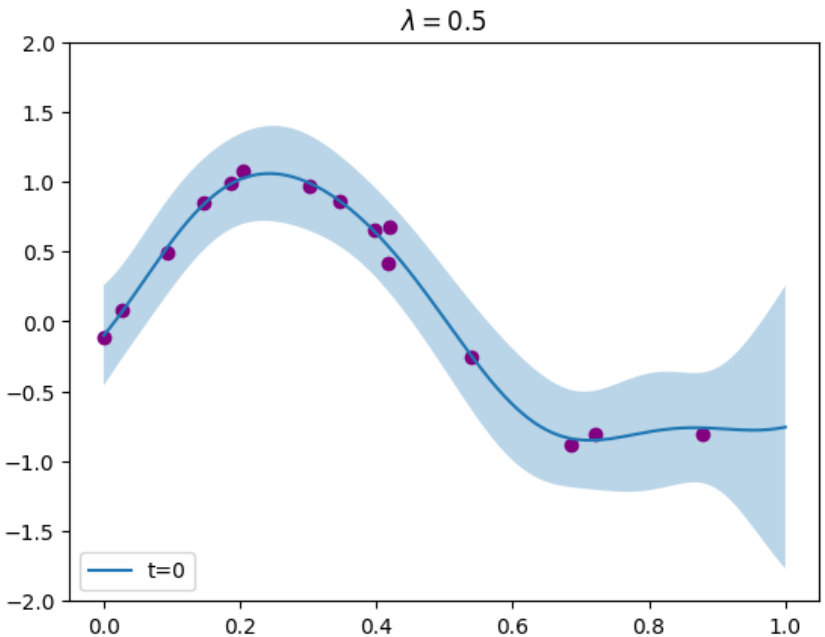
- We can employ popular “forgetting” mechanisms

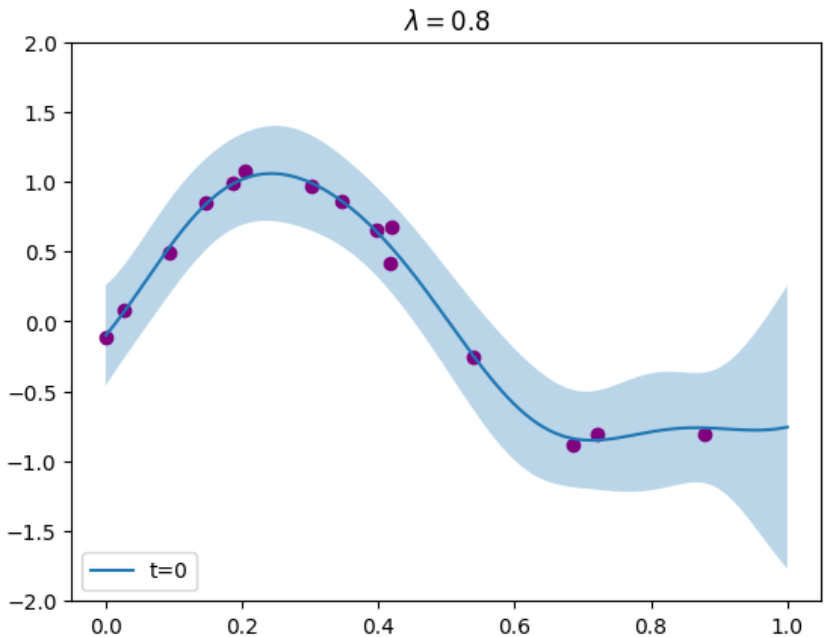
$$\text{B2P:} \quad \boldsymbol{\theta}_{t+1} = \sqrt{\lambda}\,\boldsymbol{\theta}_t + \sqrt{1-\lambda}\,\mathbf{u}_t, \qquad \mathbf{u}_t \sim \mathcal{N}(\mathbf{0}, \sigma_\theta^2 \mathbf{I}_{2J}), \;\; \lambda \in [0,1]$$

$$\text{UI:} \quad \boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \sqrt{\frac{1}{\gamma} - 1}\mathbf{u}_t, \qquad \mathbf{u}_t \sim \mathcal{N}(\mathbf{0}, \widehat{\boldsymbol{\Sigma}}_t), \;\; \gamma \in (0,1]$$

Time dynamics

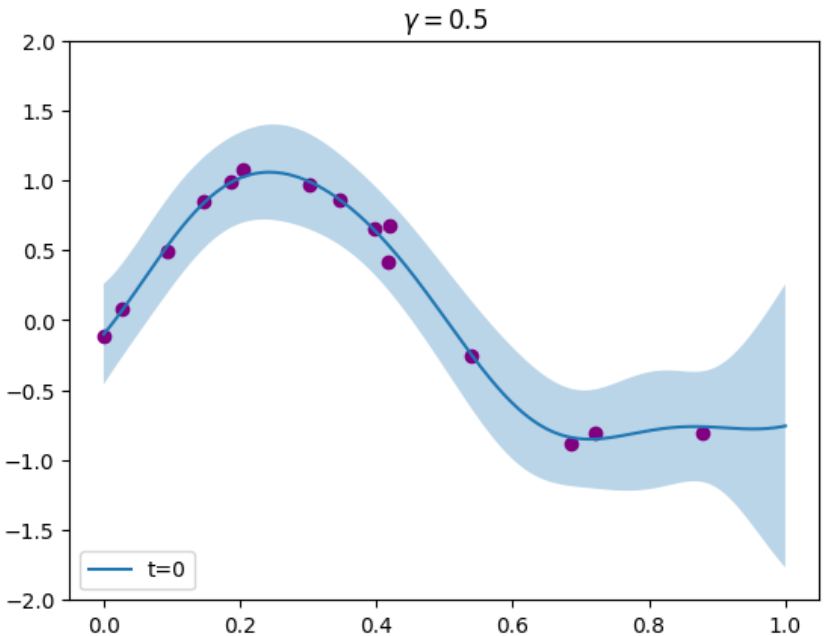
"Back to Prior" forgetting

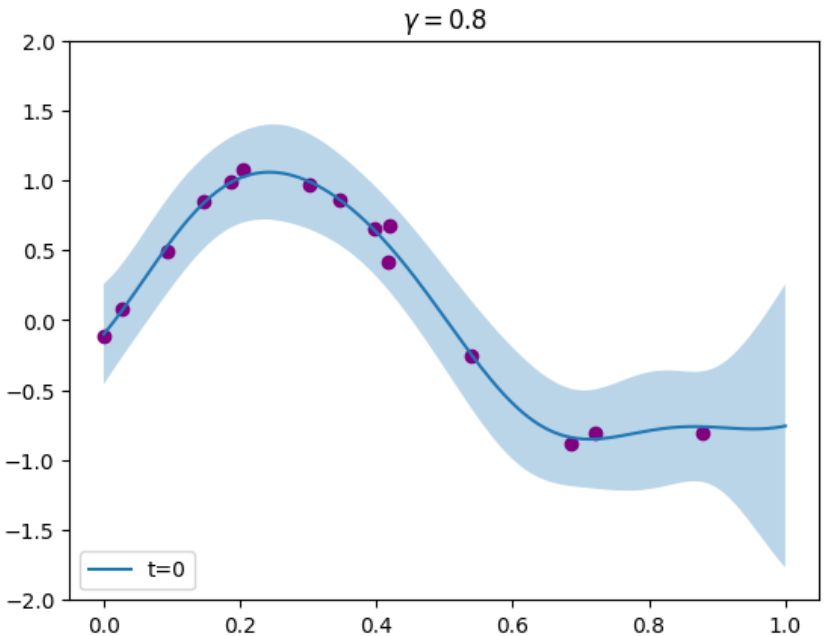




Time dynamics

"Uncertainty Injection" forgetting





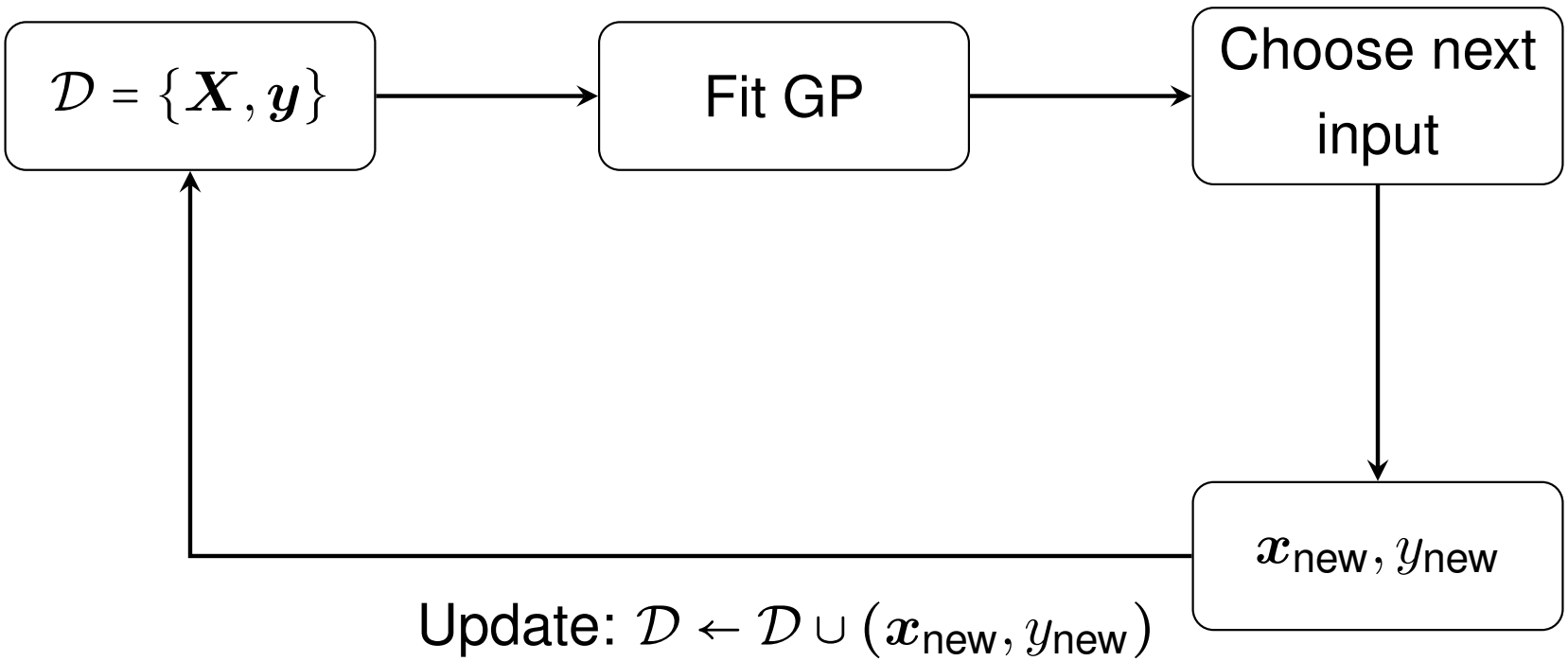
Bayesian optimization

- (Time-invariant) Bayesian optimization solves

$$\boldsymbol{x}_* = \arg\max_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x})$$

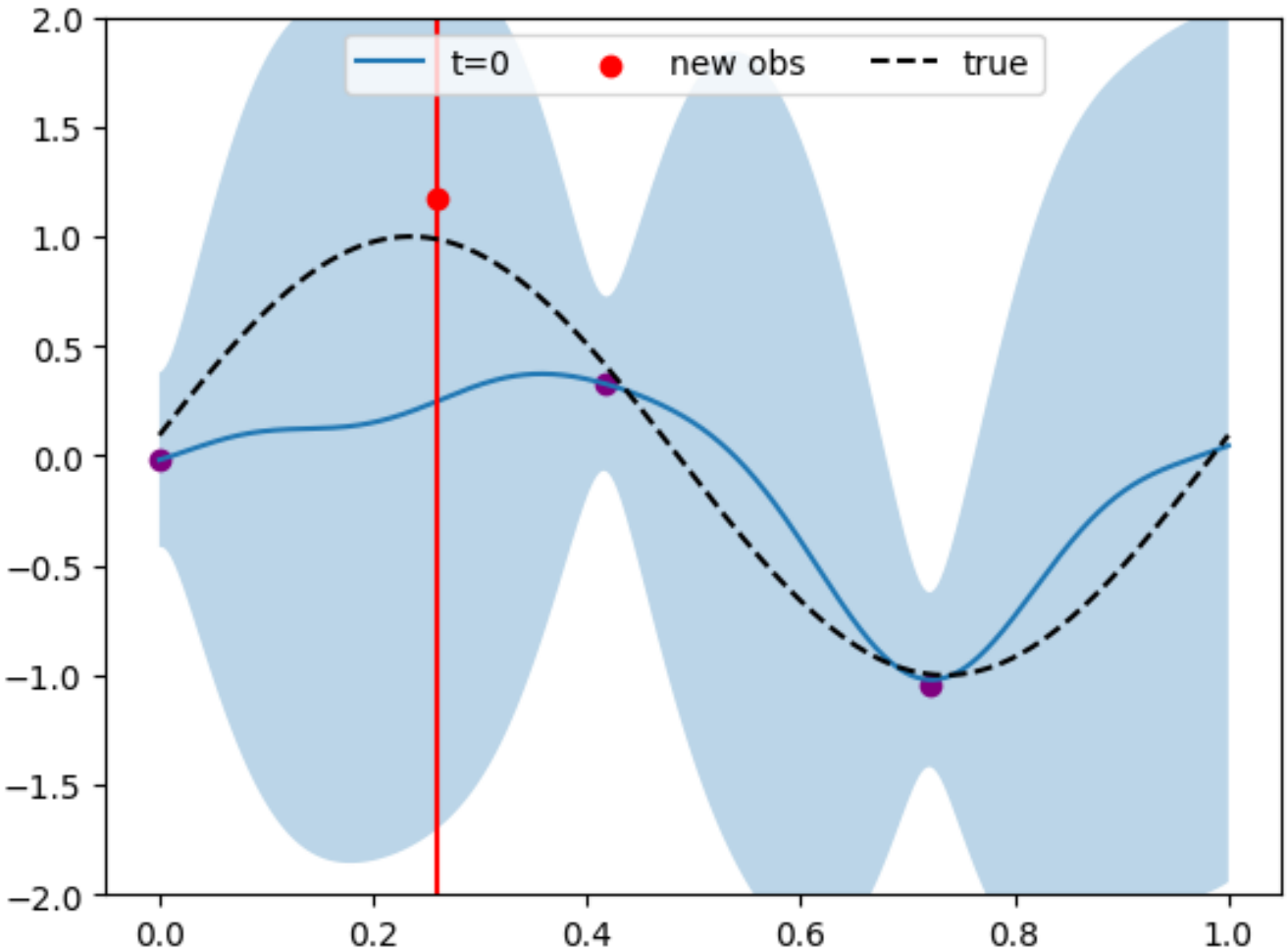
  using a probabilistic surrogate $p(f|\mathcal{D})$ and an acquisition function $\alpha(\boldsymbol{x}; \mathcal{D})$.
- BO sequentially acquires points that are "useful" by repeatedly doing:
  1. $\boldsymbol{x}_{t+1} = \arg\max_{\boldsymbol{x} \in \mathcal{X}} \alpha(\boldsymbol{x}; \mathcal{D}_t)$
  2. $\mathcal{D}_{t+1} \leftarrow \mathcal{D}_t \cup (\boldsymbol{x}_{t+1}, y_{t+1})$

Bayesian optimization



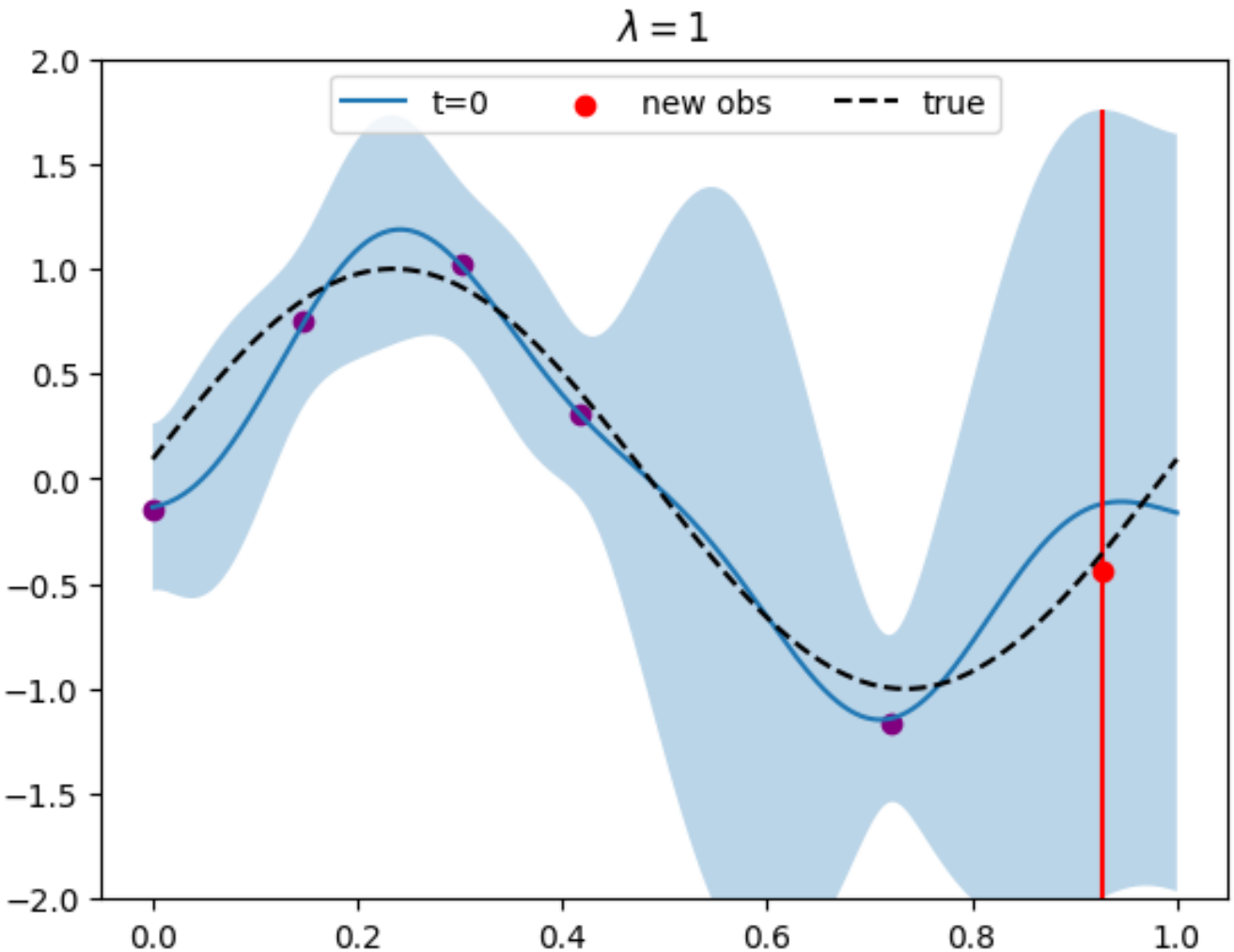
Bayesian optimization

Vanilla BO with UCB acquisition function



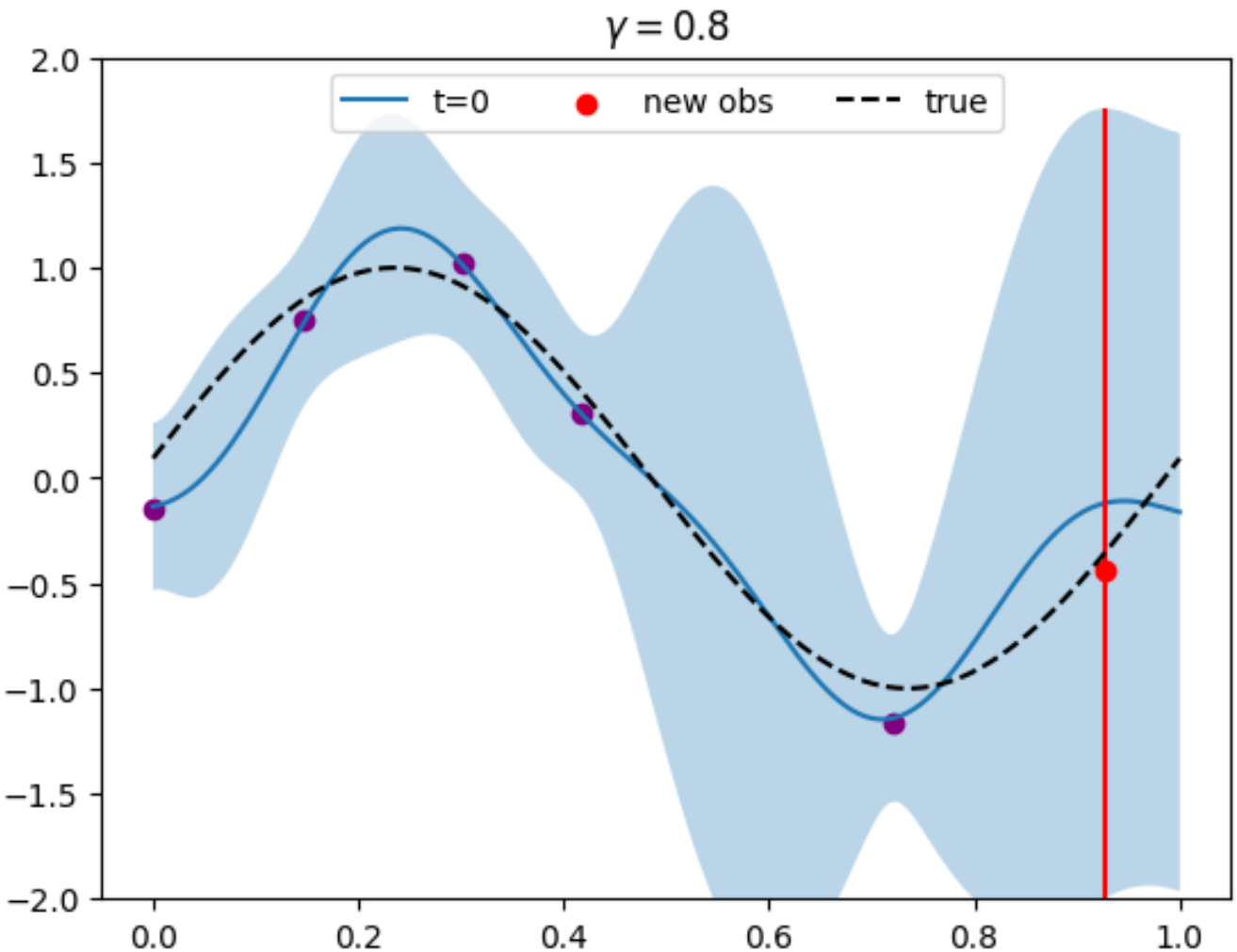
Bayesian optimization

Time-invariant BO on a time-varying function



Bayesian optimization

TVBO with UI



Bayesian optimization with dynamic RF-GP

- The model

$$\boldsymbol{\theta}_t \sim p(\boldsymbol{\theta}_t|\mathcal{D}_t)$$
$$\boldsymbol{\theta}_{t+1} = a_1\boldsymbol{\theta}_t + a_2\mathbf{m}_t + a_3\mathbf{u}_t$$
$$y_{t+1} = \boldsymbol{\phi}^\top(\boldsymbol{x})\boldsymbol{\theta}_{t+1} + \epsilon_{\text{obs}}$$

- For instance, the GP-UCB acquisition function with B2P forgetting is

$$\alpha_{t+1}^{UCB}(\boldsymbol{x}) = \sqrt{\lambda}\boldsymbol{\phi}(\boldsymbol{x})^\top\widehat{\boldsymbol{\theta}}_t + \beta_{t+1}\sqrt{\lambda\boldsymbol{\phi}(\boldsymbol{x})^\top\widehat{\boldsymbol{\Sigma}}_t\boldsymbol{\phi}(\boldsymbol{x}) + (1-\lambda)\sigma_\theta^2\boldsymbol{\phi}(\boldsymbol{x})^\top\boldsymbol{\phi}(\boldsymbol{x})}$$

Bayesian optimization with dynamic RF-GP

- Setting $a_1 = \sqrt{\lambda}$, $a_2 = 1 - \sqrt{\lambda}$, and $a_3 = \sqrt{1-\lambda}$ with $\lambda \in (0, 1)$, we get

$$\alpha_{t+1}^{UCB}(\boldsymbol{x}) = \sqrt{\lambda}\boldsymbol{\phi}(\boldsymbol{x})^\top \widehat{\boldsymbol{\theta}}_t + (1-\sqrt{\lambda})\boldsymbol{\phi}(\boldsymbol{x})^\top \mathbf{m}_t + \beta_{t+1}\sqrt{\lambda\boldsymbol{\phi}(\boldsymbol{x})^\top \widehat{\boldsymbol{\Sigma}}_t \boldsymbol{\phi}(\boldsymbol{x}) + (1-\lambda)\boldsymbol{\phi}(\boldsymbol{x})^\top \mathbf{C}_t \boldsymbol{\phi}(\boldsymbol{x})}.$$

Time-varying BO with windows of data

We propose to use a sliding window of w samples $\mathcal{D}_t^w = \{(\boldsymbol{x}_\tau, y_\tau)\}_{\tau=t-w+1}^{t}$ to compute the parameters $\{\mathbf{m}_t, \mathbf{C}_t\}$,

$$\mathbf{m}_t = \left(\boldsymbol{\Phi}_t^w (\boldsymbol{\Phi}_t^w)^\top + \frac{\sigma_{\text{obs}}^2}{\sigma_\theta^2}\mathbf{I}_{2J}\right)^{-1} \boldsymbol{\Phi}_t^w \boldsymbol{y}_t^w,$$

$$\mathbf{C}_t = \left(\frac{1}{\sigma_{\text{obs}}^2}\boldsymbol{\Phi}_t^w (\boldsymbol{\Phi}_t^w)^\top + \frac{1}{\sigma_\theta^2}\mathbf{I}_{2J}\right)^{-1},$$

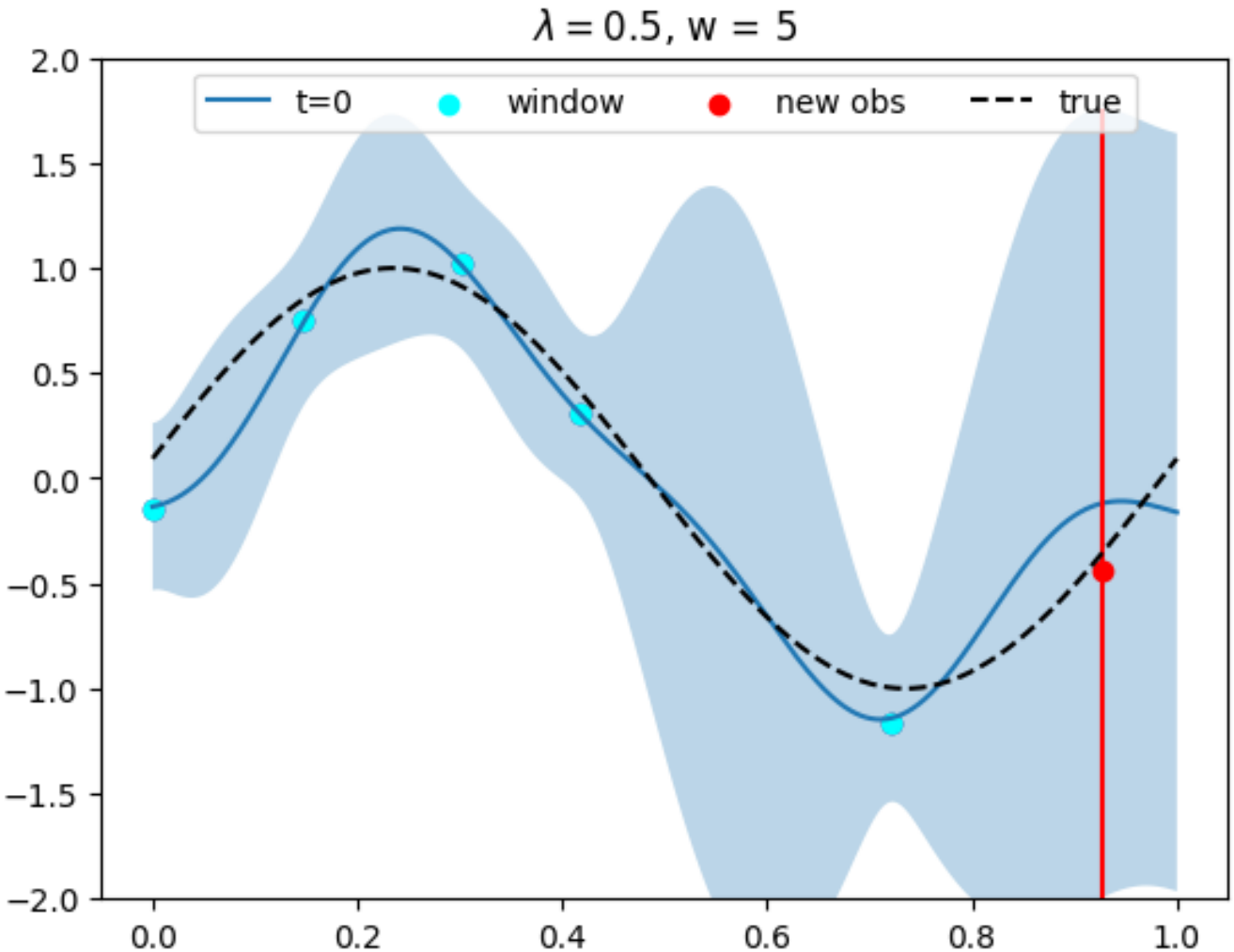
where $\boldsymbol{\Phi}_t^w$ and $\boldsymbol{y}_t^w$ are built using $\mathcal{D}_t^w$.

Note that:

- when $w = 1 \Longrightarrow \mathbf{m}_t = 0$ and $\mathbf{C}_t = \sigma_\theta^2 \mathbf{I}_{2J} \Longrightarrow$ we recover BO with prior forgetting.
- when $w = t \Longrightarrow \mathbf{m}_t = \widehat{\boldsymbol{\theta}}_t$ and $\mathbf{C}_t = \widehat{\boldsymbol{\Sigma}}_t \Longrightarrow$ we recover time-invariant BO.

Time-varying BO with windows of data

TVBO with “Back 2 Window”



Toy problem

- Let $x \in [0, 1]$

$$f_t(x) = \sin[2\pi(x + 0.15 - 0.01t)]$$

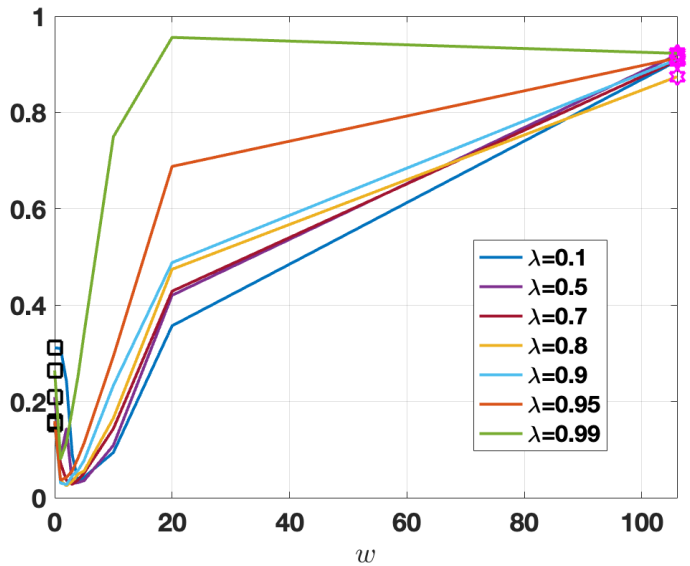
where t goes from $t = 1$ to $t = 85$.

- We compare the average regret,

$$\bar{R}_T = \frac{1}{T} \sum_{t=1}^{T} |f_t^* - f_t(\widehat{x}_t)|$$

where $f_t^* = \max_x f_t(x) = 1$ for all t.

- Goal: study window size w and forgetting strength λ.



The optimal value of w increased as λ decreased: when we allow the current model to forget more and more (smaller λ), it is beneficial that the forgetting factor carries an increasing amount of past data (bigger w). However, for all λ, increasing w eventually hampers performance.

Localization example

- Target Position $\boldsymbol{x} \in \mathbb{R}^2$. The network has $L = 3$ sensors. We observed M measurements at every sensor,

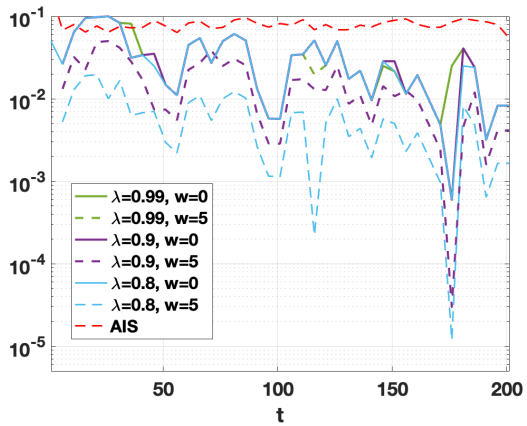
$$z_{m,l} \sim \mathcal{N}\left(A \log\left(\| \boldsymbol{x} - \mathbf{s}_l \|\right), \sigma_z^2\right),$$

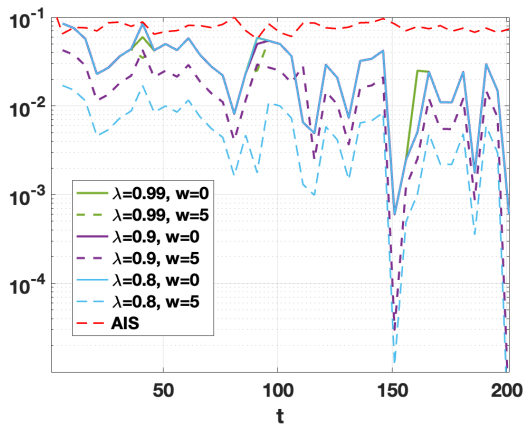
where $l = 1, 2, 3$, $m = 1, \ldots, M$, and A is a constant.

- Goal: track target over time by estimating the MAP of the sequence of posterior distributions,

$$\boldsymbol{x}_{\text{map},t} = \arg\max_{\boldsymbol{x}} \bar{\pi}(\boldsymbol{x}|\mathbf{Z}_t),$$

where $\mathbf{Z}_t$ are measurements obtained at $t = 0, \ldots, T$.





Conclusions

- We have presented a framework for time-varying optimization based on dynamic random-feature Gaussian processes.
- Random-feature Gaussian processes have several advantages such as scalability and capacity for online learning. Moreover, time dynamics can be injected by considering a Markov transition model.
- Combining random-feature Gaussian processes with a general Markov time evolution, we have proposed a framework for time-varying Bayesian optimization and studied the resulting time-varying acquisition function.
- More specifically, we proposed a novel time-varying Bayesian optimization algorithm that uses windows of past data to build the additional exploration and exploitation terms. The time-invariant setting, as well as time-varying Bayesian optimization with prior forgetting, are special cases.