Scalable Bandit Methods for Hyper-parameter Tuning

Kirthevasan Kandasamy

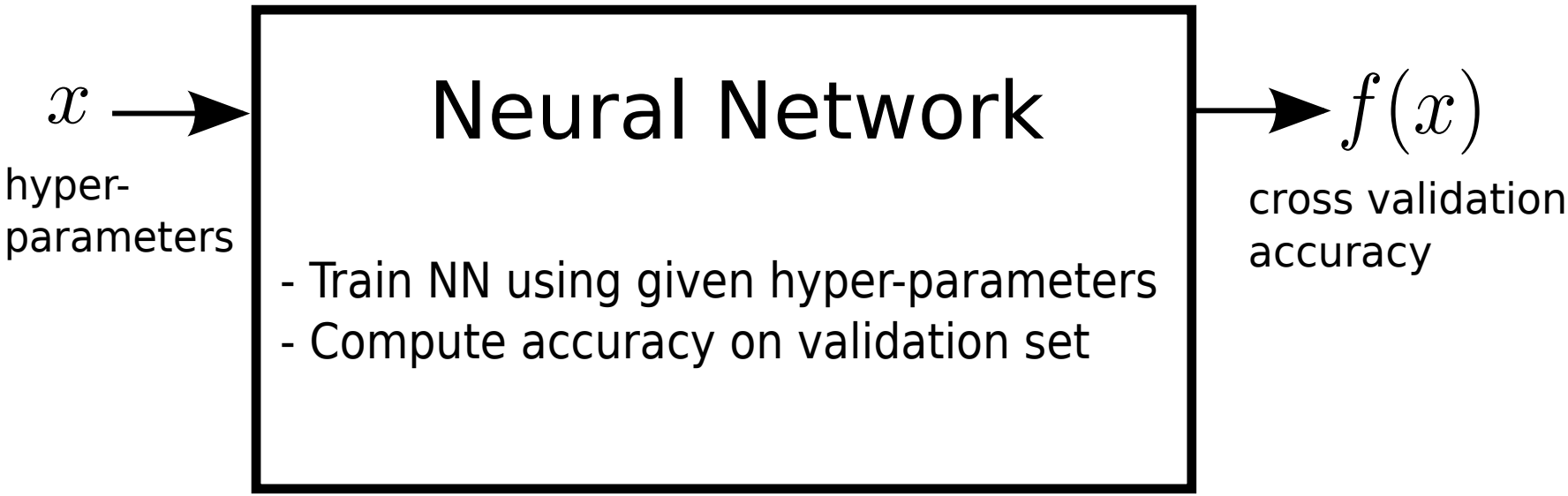
Carnegie Mellon University

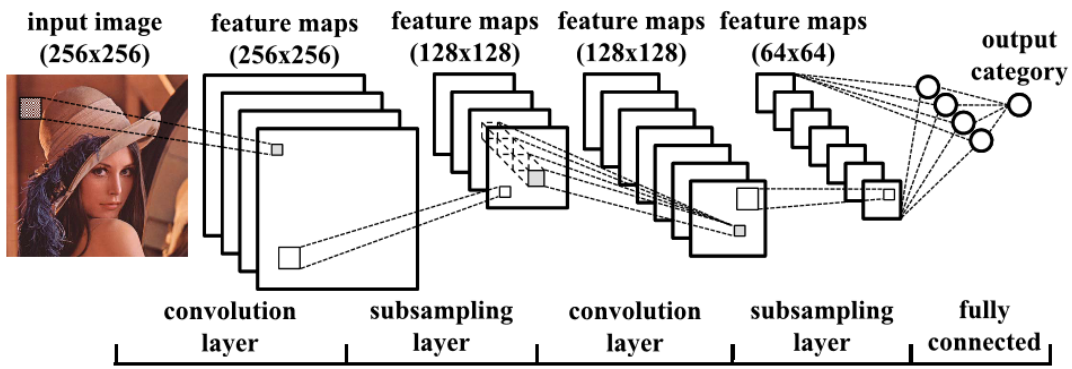
Guest Lecture - Scalable Machine Learning for Big Data Biology

University of Pittsburgh, Pittsburgh, PA

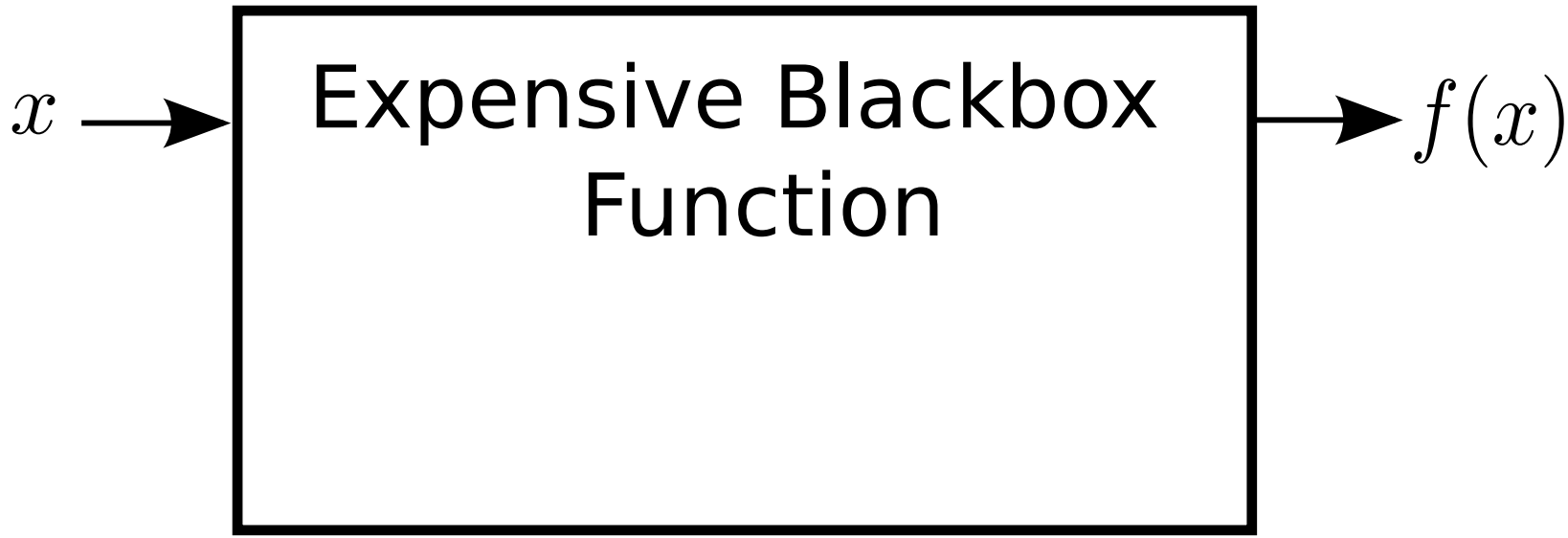
November 3, 2017

Hyper-parameter Tuning

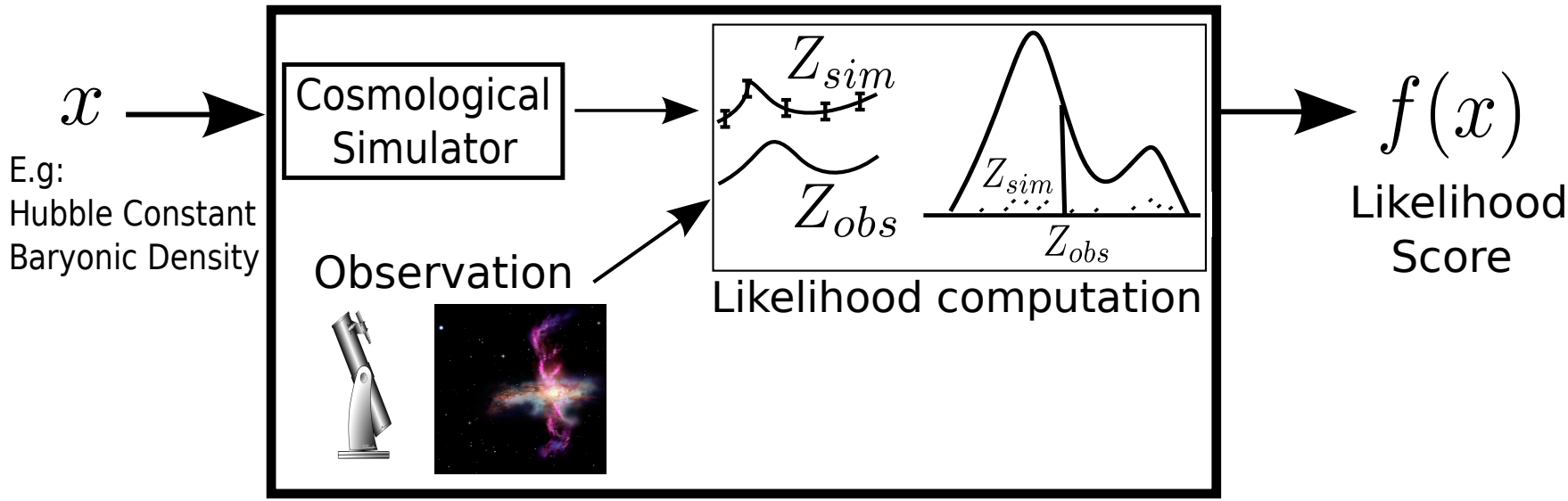


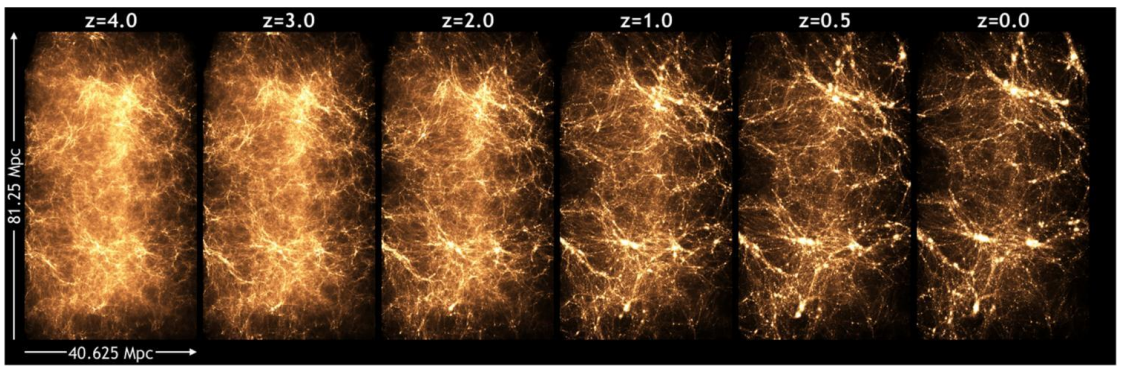


Black-box Optimisation

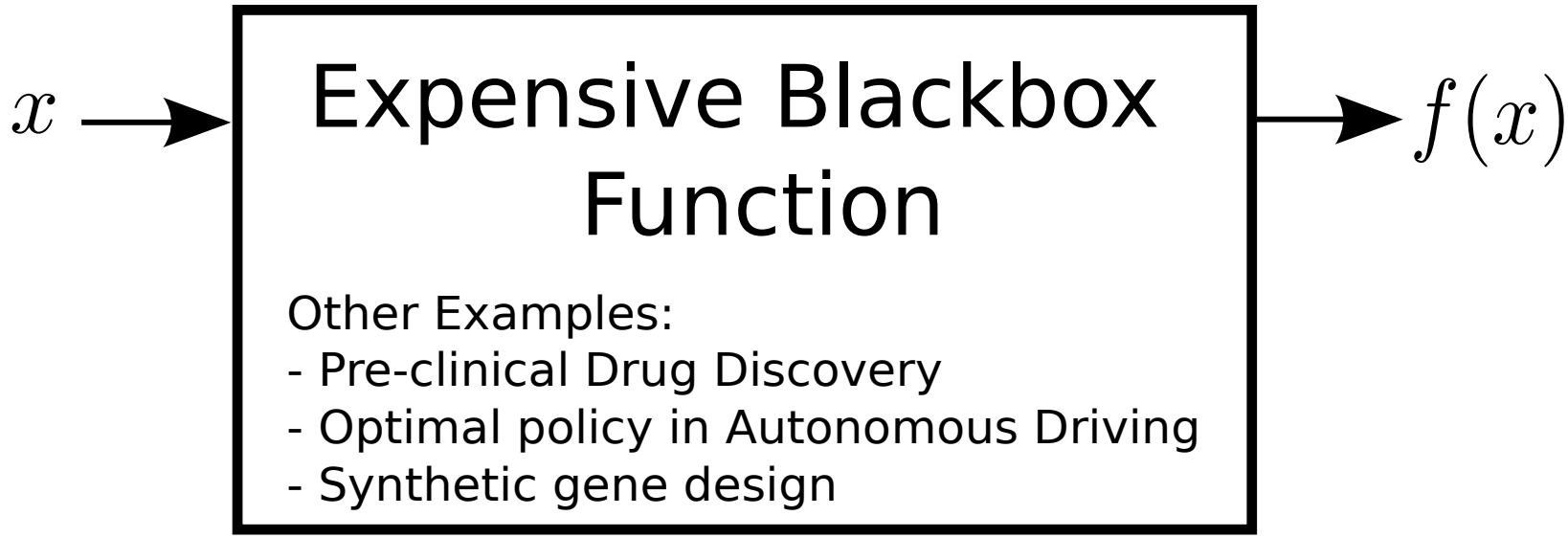


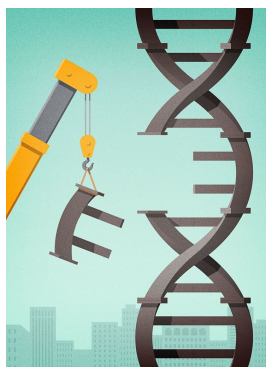
Maximum Likelihood estimation in Astrophysics





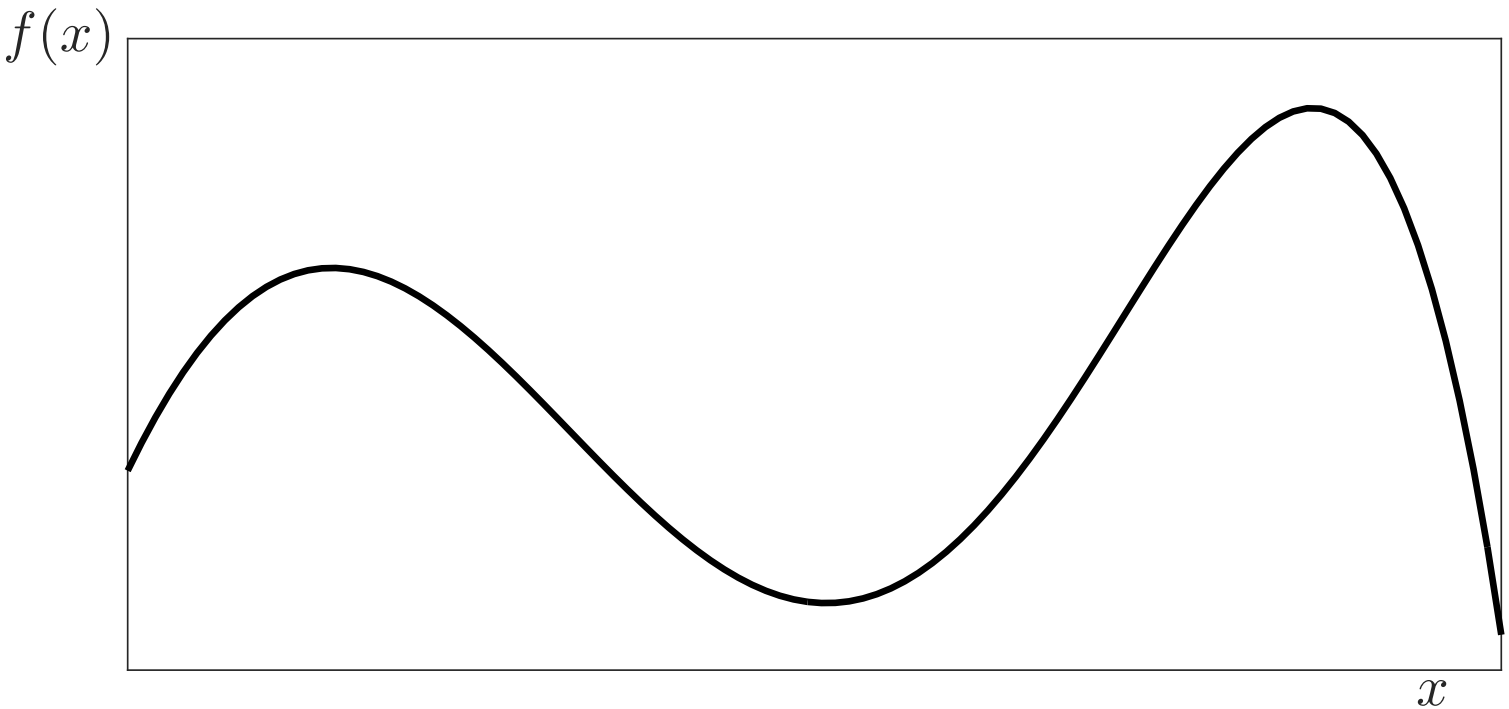
Black-box Optimisation





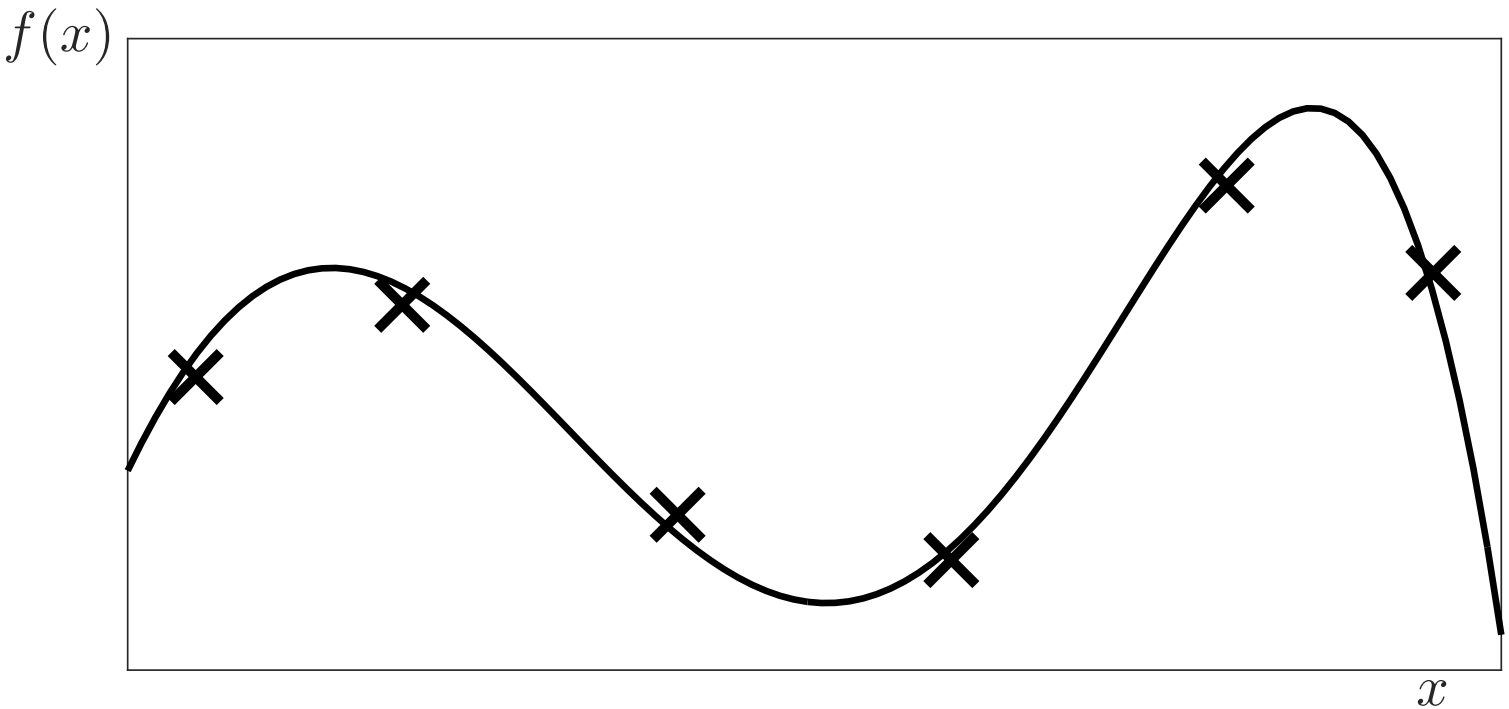
Black-box Optimisation

$f : \mathcal{X} \to \mathbb{R}$ is a black-box function that is accessible only via noisy evaluations.



Black-box Optimisation

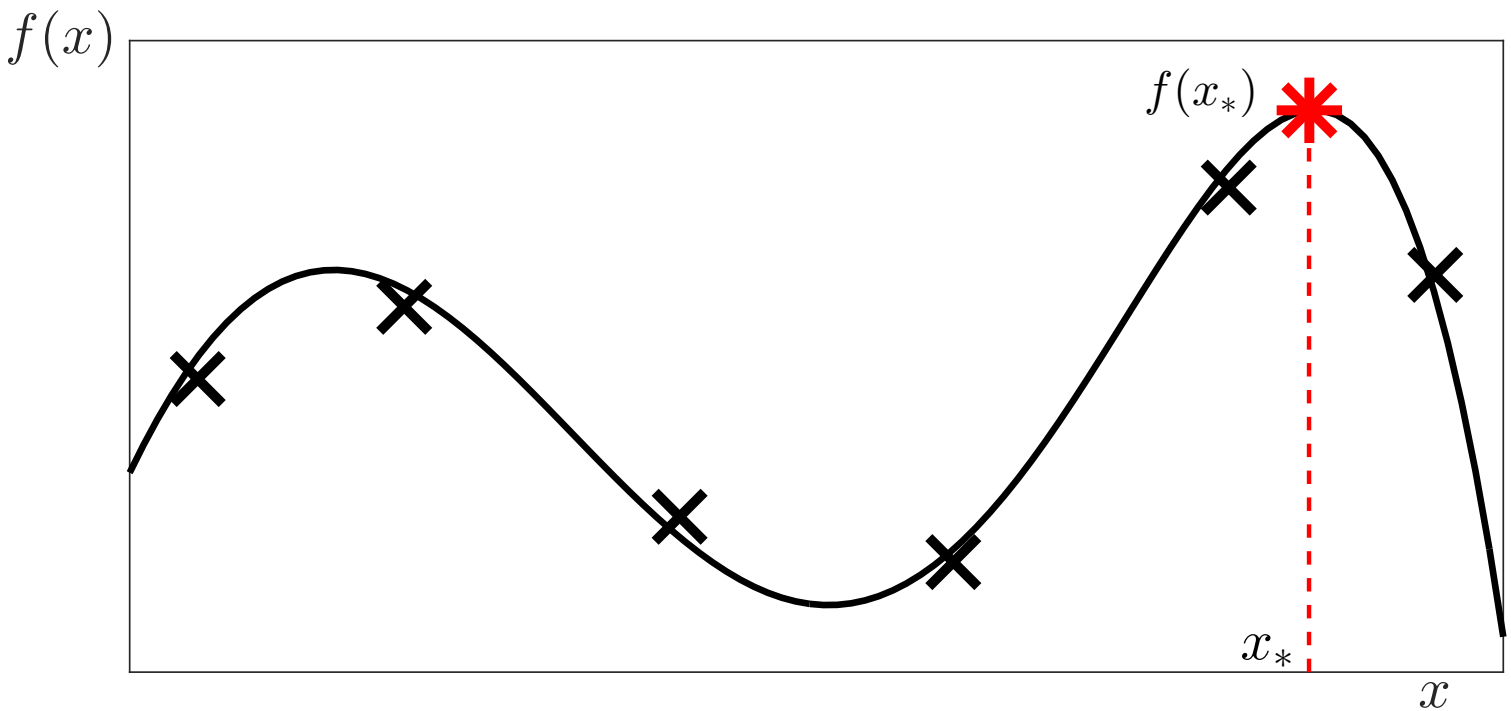
$f : \mathcal{X} \to \mathbb{R}$ is a black-box function that is accessible only via noisy evaluations.



Black-box Optimisation

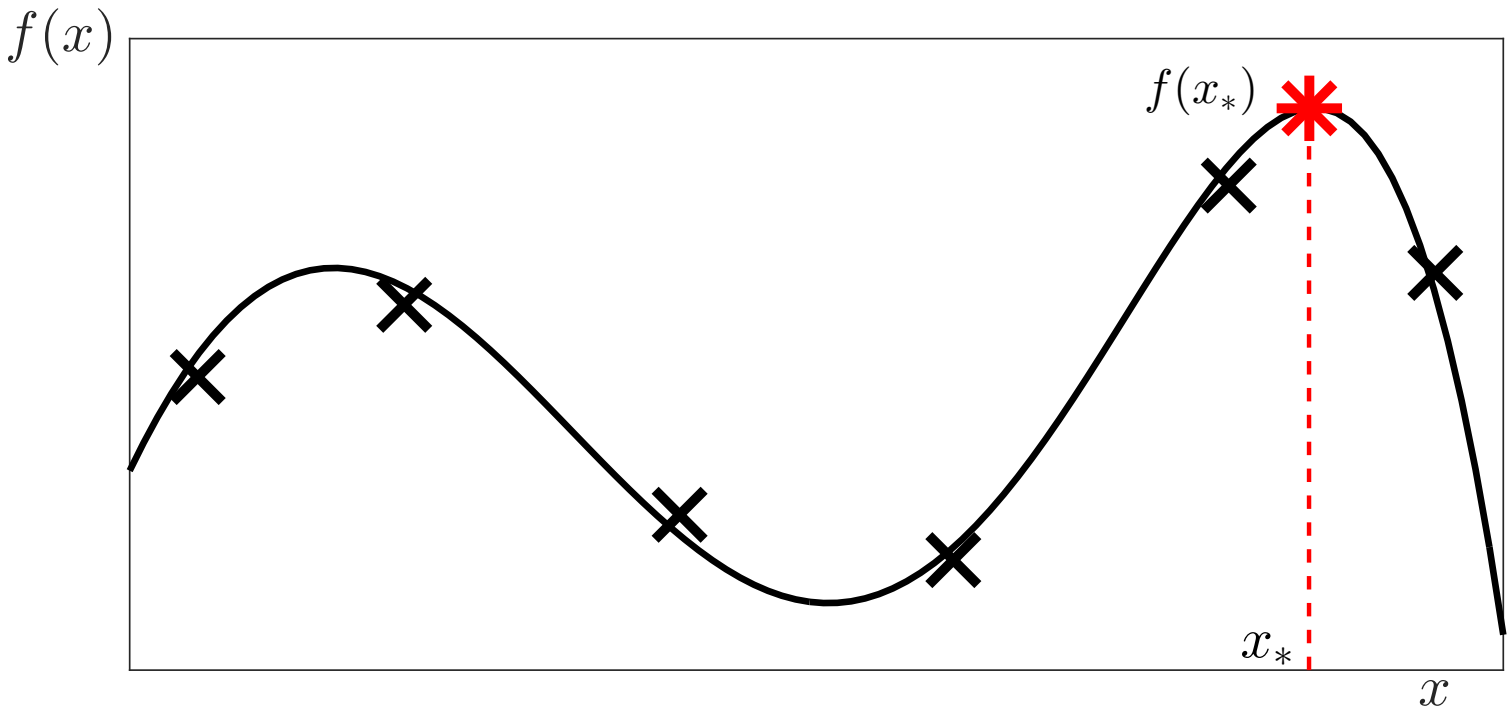
$f : \mathcal{X} \to \mathbb{R}$ is a black-box function that is accessible only via noisy evaluations.

Let $x_\star = \operatorname{argmax}_x f(x)$.



Black-box Optimisation

$f : \mathcal{X} \to \mathbb{R}$ is a black-box function that is accessible only via noisy evaluations.
Let $x_\star = \operatorname{argmax}_x f(x)$.



Simple Regret after n evaluations

$$S_n = f(x_\star) - \max_{t=1,\ldots,n} f(x_t).$$

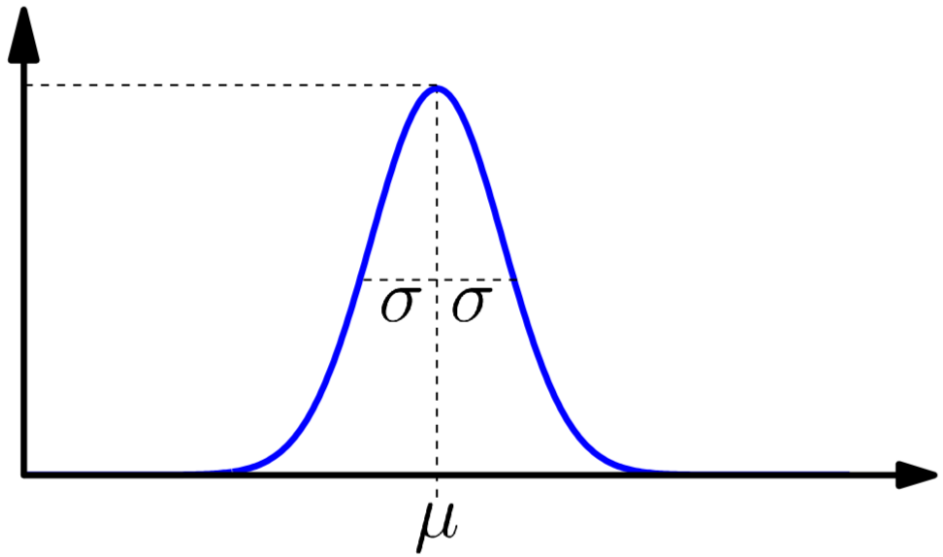
Outline

- Part I: Bandits in the Bayesian Paradigm
  1. Gaussian processes
  2. Algorithms: Upper Confidence Bound (UCB) & Thompson Sampling (TS)
- Part II: Scaling up Bandits
  1. Multi-fidelity bandit: cheap approximations to an expensive experiment
  2. Parallelising function evaluations
  3. High dimensional input spaces

Outline

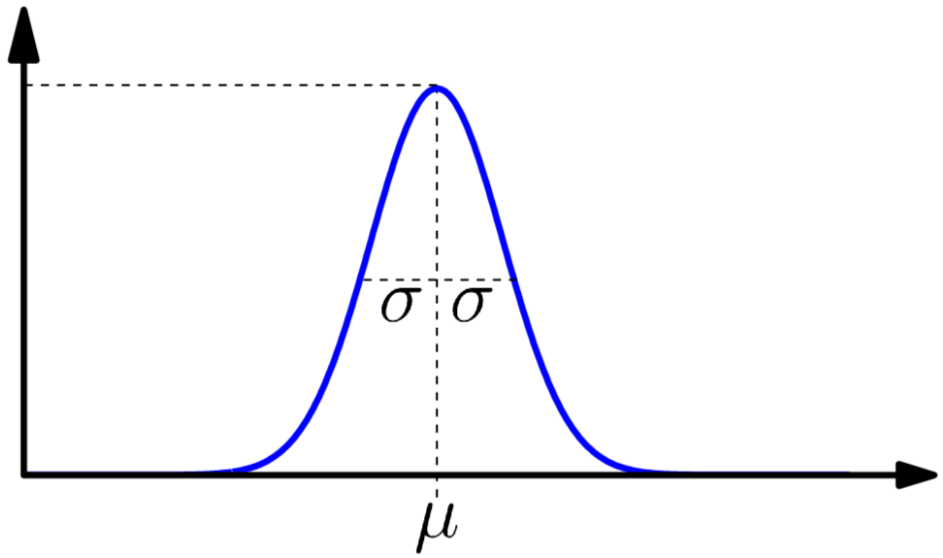
- Part I: Bandits in the Bayesian Paradigm
  1. Gaussian processes
  2. Algorithms: Upper Confidence Bound (UCB) & Thompson Sampling (TS)
- Part II: Scaling up Bandits
  1. Multi-fidelity bandit: cheap approximations to an expensive experiment
  2. Parallelising function evaluations
  3. High dimensional input spaces

Gaussian (Normal) distribution $\mathcal{N}(\mu, \sigma^2)$



- A probability distribution for real valued random variables.
- Mean μ and variance σ^2 completely characterises distribution.

Gaussian (Normal) distribution $\mathcal{N}(\mu, \sigma^2)$



- A probability distribution for real valued random variables.
- Mean μ and variance σ^2 completely characterises distribution.
- For samples $X_1, \ldots, X_n$, let $\hat{\mu} = \frac{1}{n}\sum_i X_i$ be the sample mean. Then, $\hat{\mu} \pm 1.96\frac{\sigma}{\sqrt{n}}$ is a 95% confidence interval for μ.
- Can draw samples (e.g. in Matlab: `mu + sigma * randn()`).

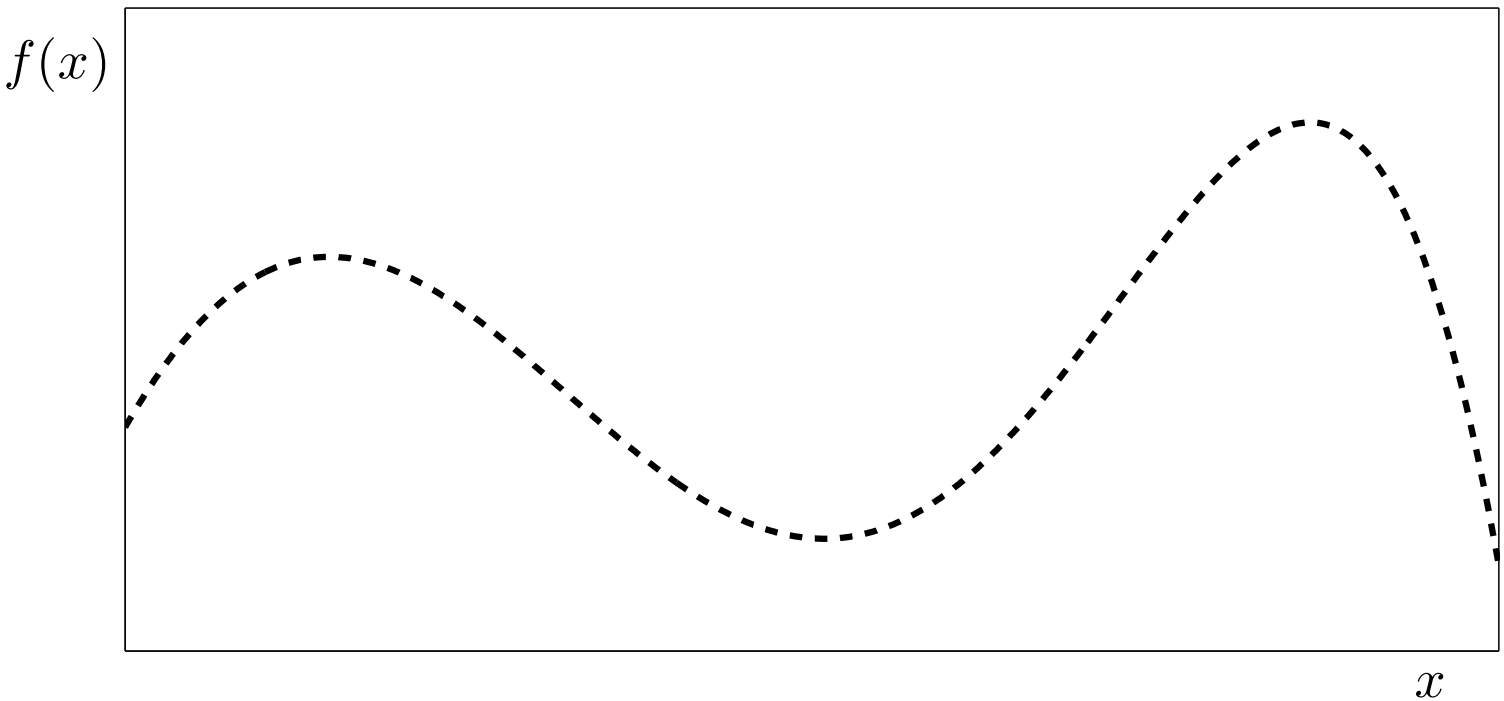
Gaussian Processes ($\mathcal{GP}$)

$\mathcal{GP}(\mu, \kappa)$: A distribution over functions from $\mathcal{X}$ to $\mathbb{R}$.

Gaussian Processes ($\mathcal{GP}$)

$\mathcal{GP}(\mu, \kappa)$: A distribution over functions from $\mathcal{X}$ to $\mathbb{R}$.

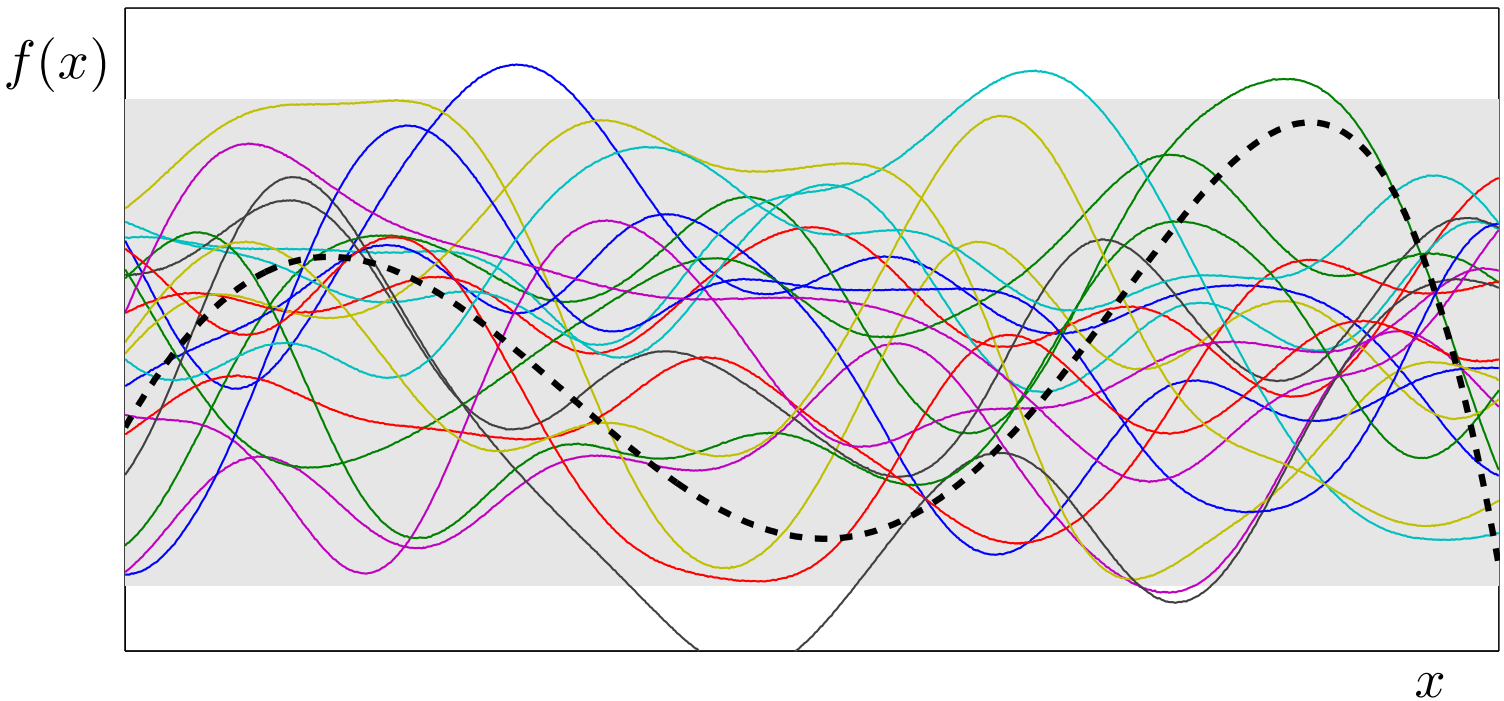
Functions with no observations



Gaussian Processes ($\mathcal{GP}$)

$\mathcal{GP}(\mu, \kappa)$: A distribution over functions from $\mathcal{X}$ to $\mathbb{R}$.

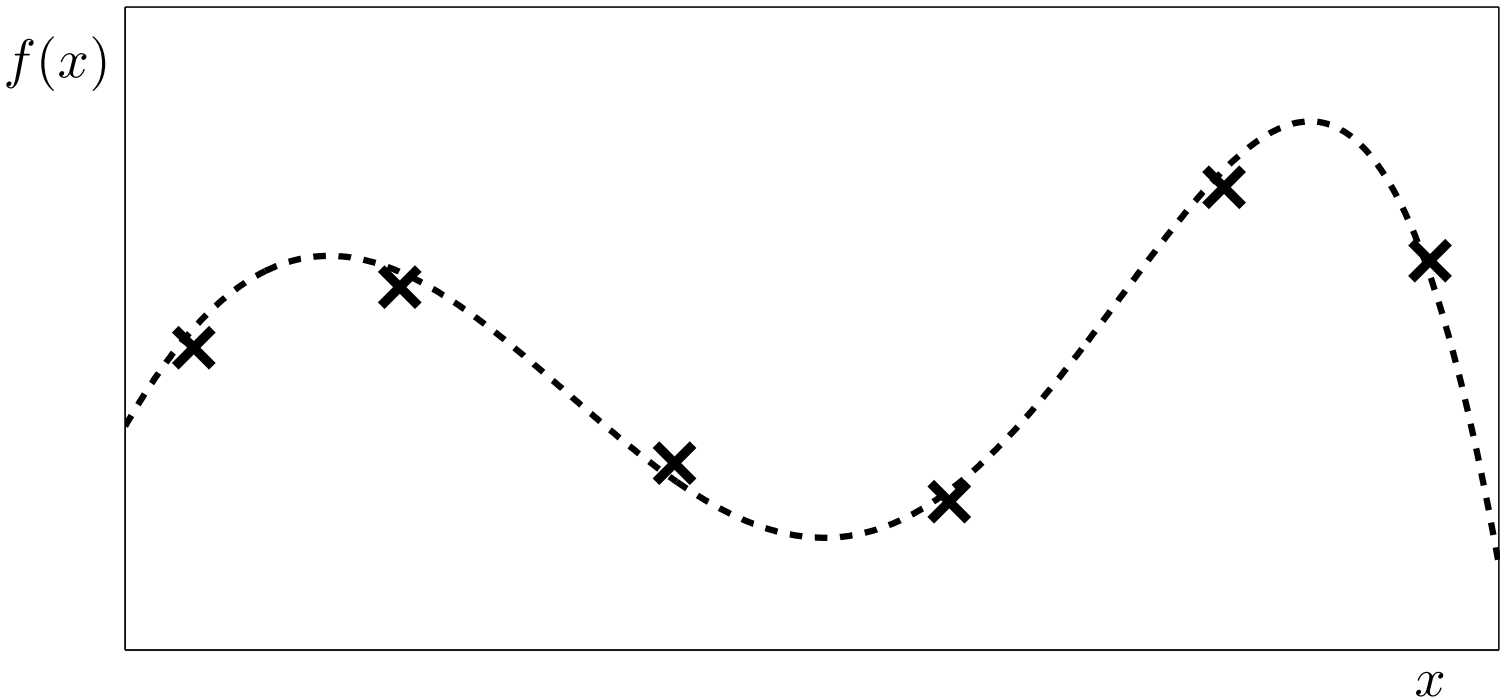
Prior $\mathcal{GP}$



Gaussian Processes ($\mathcal{GP}$)

$\mathcal{GP}(\mu, \kappa)$: A distribution over functions from $\mathcal{X}$ to $\mathbb{R}$.

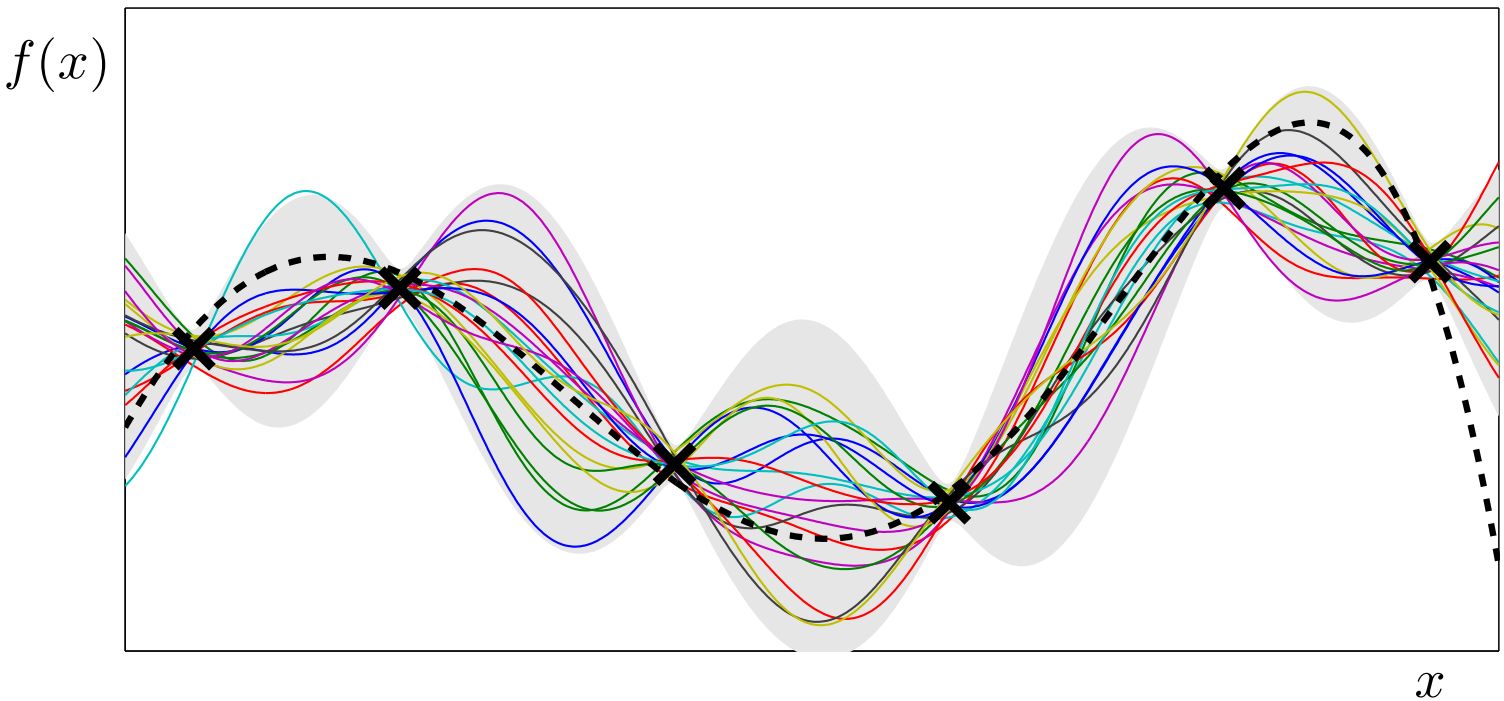
Observations



Gaussian Processes ($\mathcal{GP}$)

$\mathcal{GP}(\mu, \kappa)$: A distribution over functions from $\mathcal{X}$ to $\mathbb{R}$.

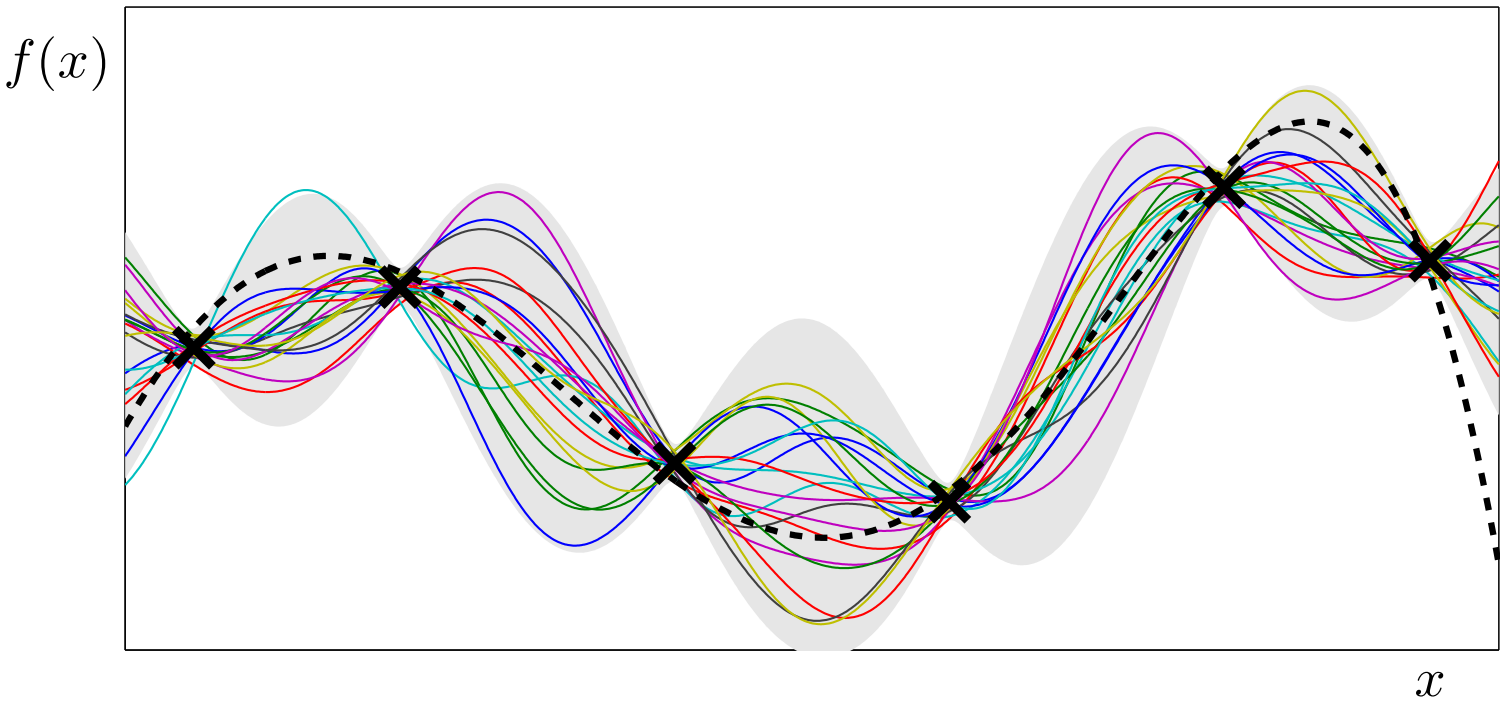
Posterior $\mathcal{GP}$ given observations



Gaussian Processes ($\mathcal{GP}$)

$\mathcal{GP}(\mu, \kappa)$: A distribution over functions from $\mathcal{X}$ to $\mathbb{R}$.

Posterior $\mathcal{GP}$ given observations



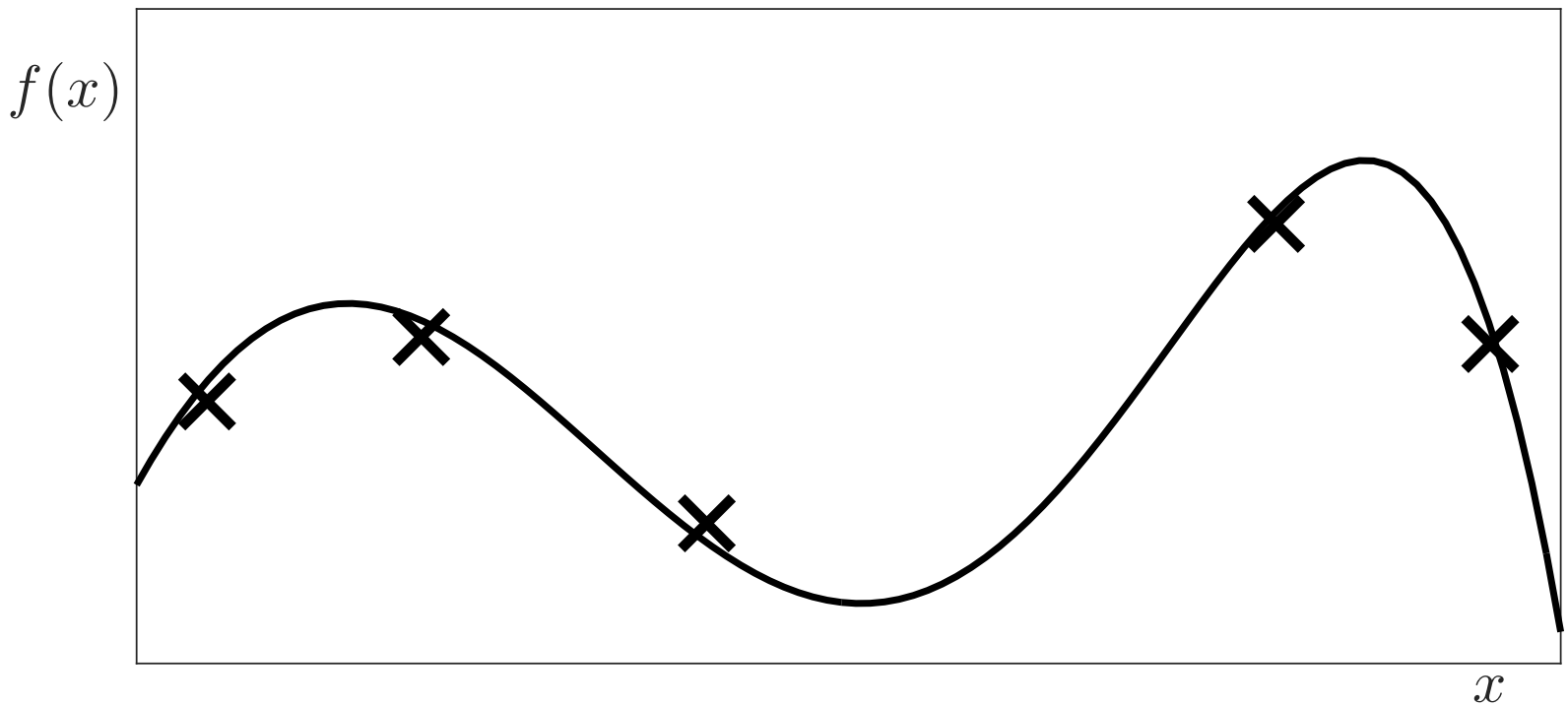
Completely characterised by mean function $\mu : \mathcal{X} \to \mathbb{R}$, and covariance kernel $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$.
After t observations, $\quad f(x) \sim \mathcal{N}(\, \mu_t(x), \sigma_t^2(x)\,)$.

Algorithm 1: Upper Confidence Bounds in GP Bandits

Model $f \sim \mathcal{GP}(\mathbf{0}, \kappa)$.

Gaussian Process Upper Confidence Bound (GP-UCB)

(Srinivas et al. 2010)

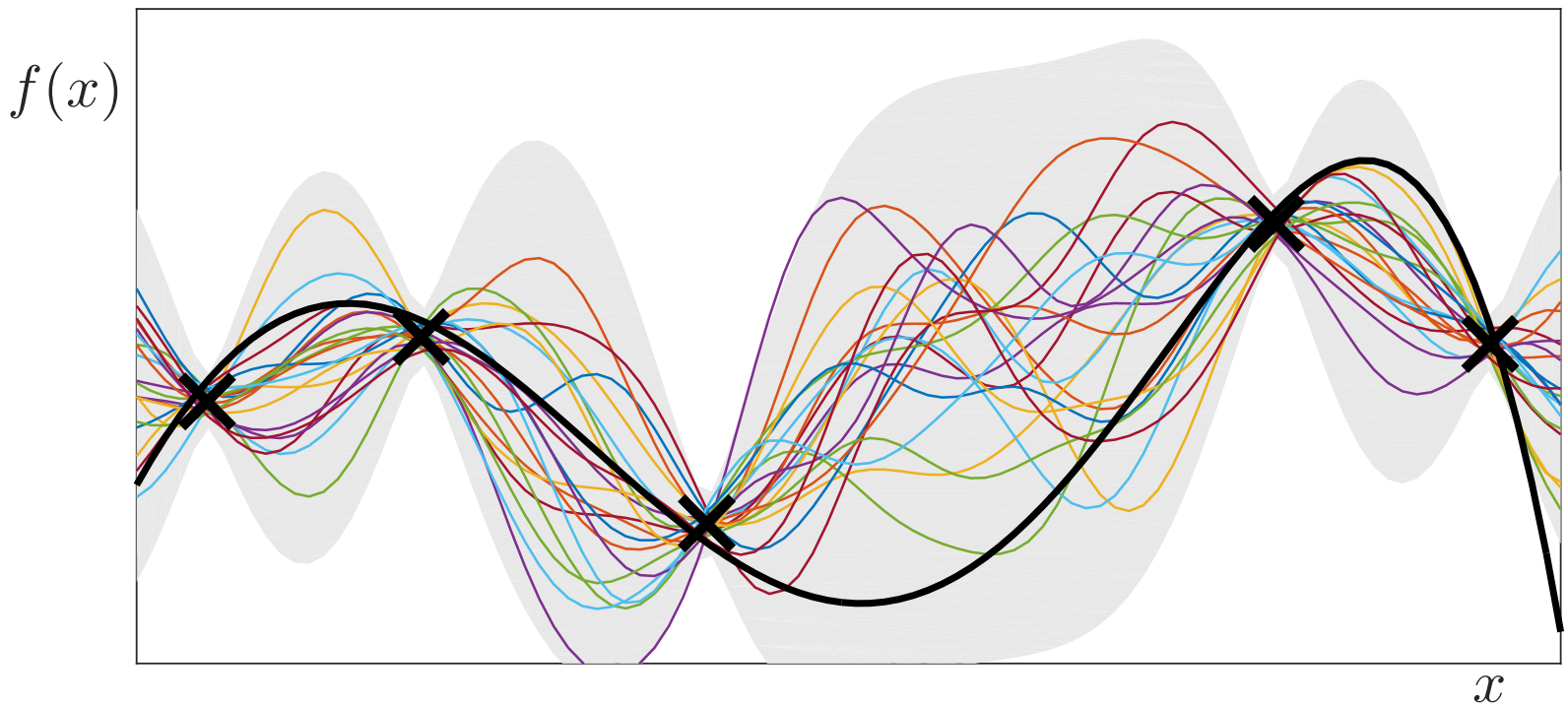


Algorithm 1: Upper Confidence Bounds in GP Bandits

Model $f \sim \mathcal{GP}(\mathbf{0}, \kappa)$.

Gaussian Process Upper Confidence Bound (GP-UCB)

(Srinivas et al. 2010)



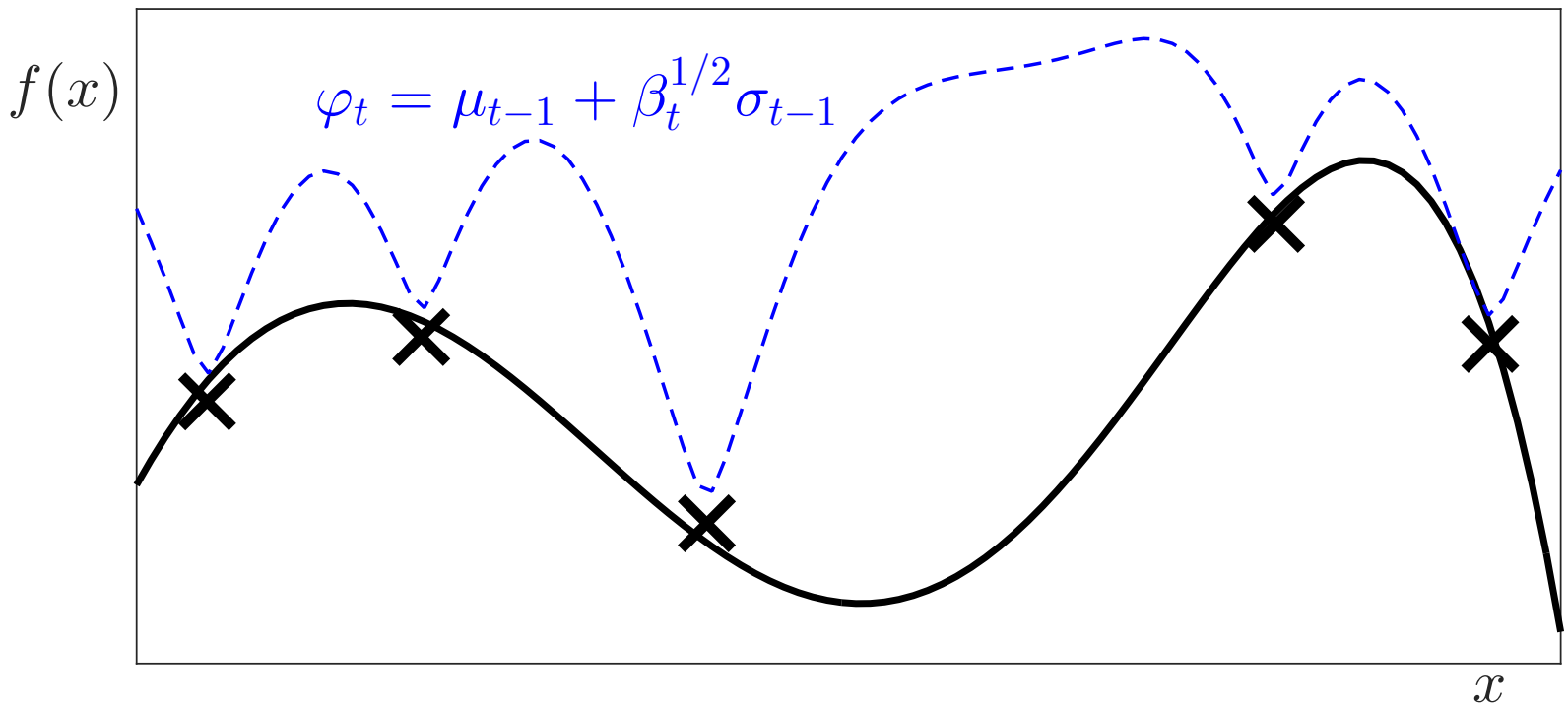
1) Construct posterior $\mathcal{GP}$.

Algorithm 1: Upper Confidence Bounds in GP Bandits

Model $f \sim \mathcal{GP}(\mathbf{0}, \kappa)$.

Gaussian Process Upper Confidence Bound (GP-UCB)

(Srinivas et al. 2010)



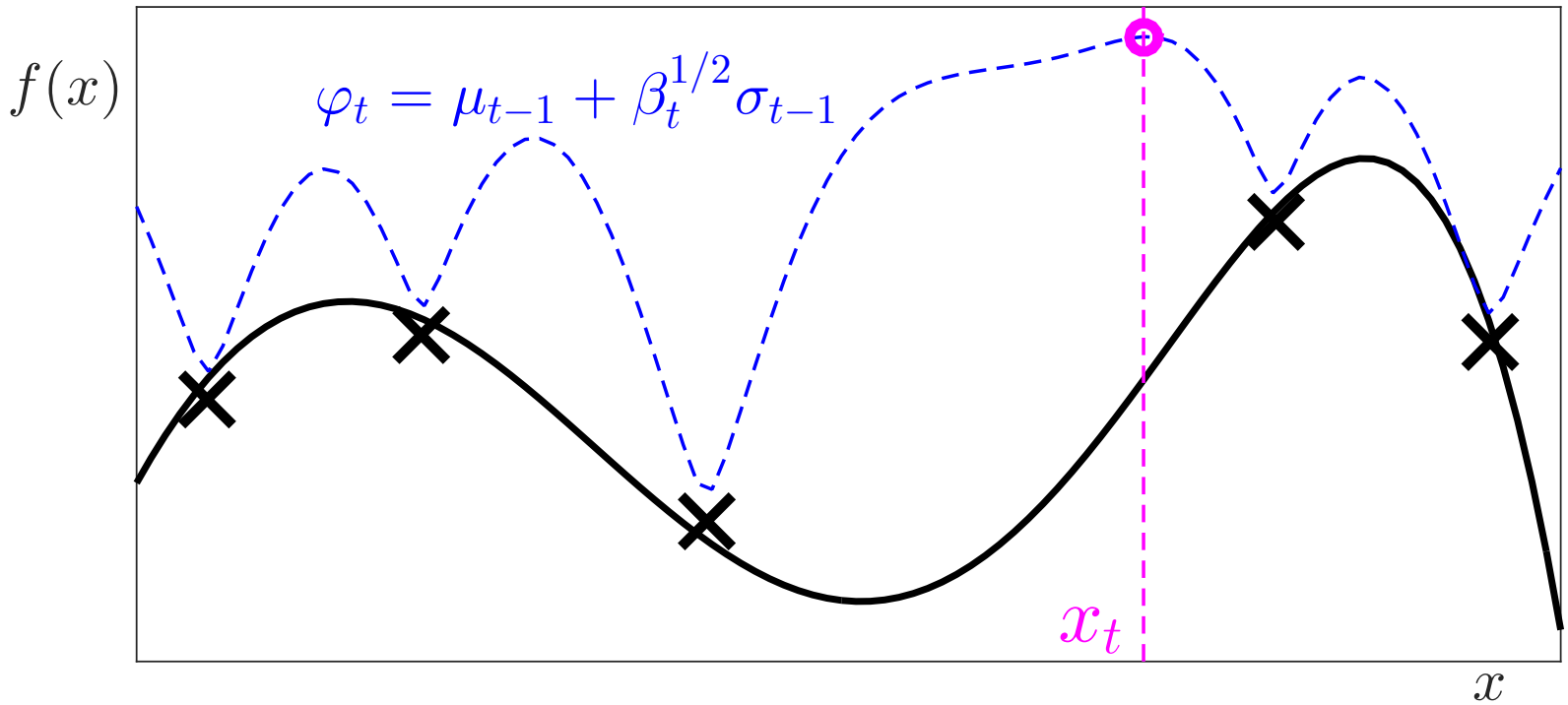
1) Construct posterior $\mathcal{GP}$. 2) $\varphi_t = \mu_{t-1} + \beta_t^{1/2}\sigma_{t-1}$ is a UCB.

Algorithm 1: Upper Confidence Bounds in GP Bandits

Model $f \sim \mathcal{GP}(\mathbf{0}, \kappa)$.

Gaussian Process Upper Confidence Bound (GP-UCB)

(Srinivas et al. 2010)



1) Construct posterior $\mathcal{GP}$. 2) $\varphi_t = \mu_{t-1} + \beta_t^{1/2}\sigma_{t-1}$ is a UCB.

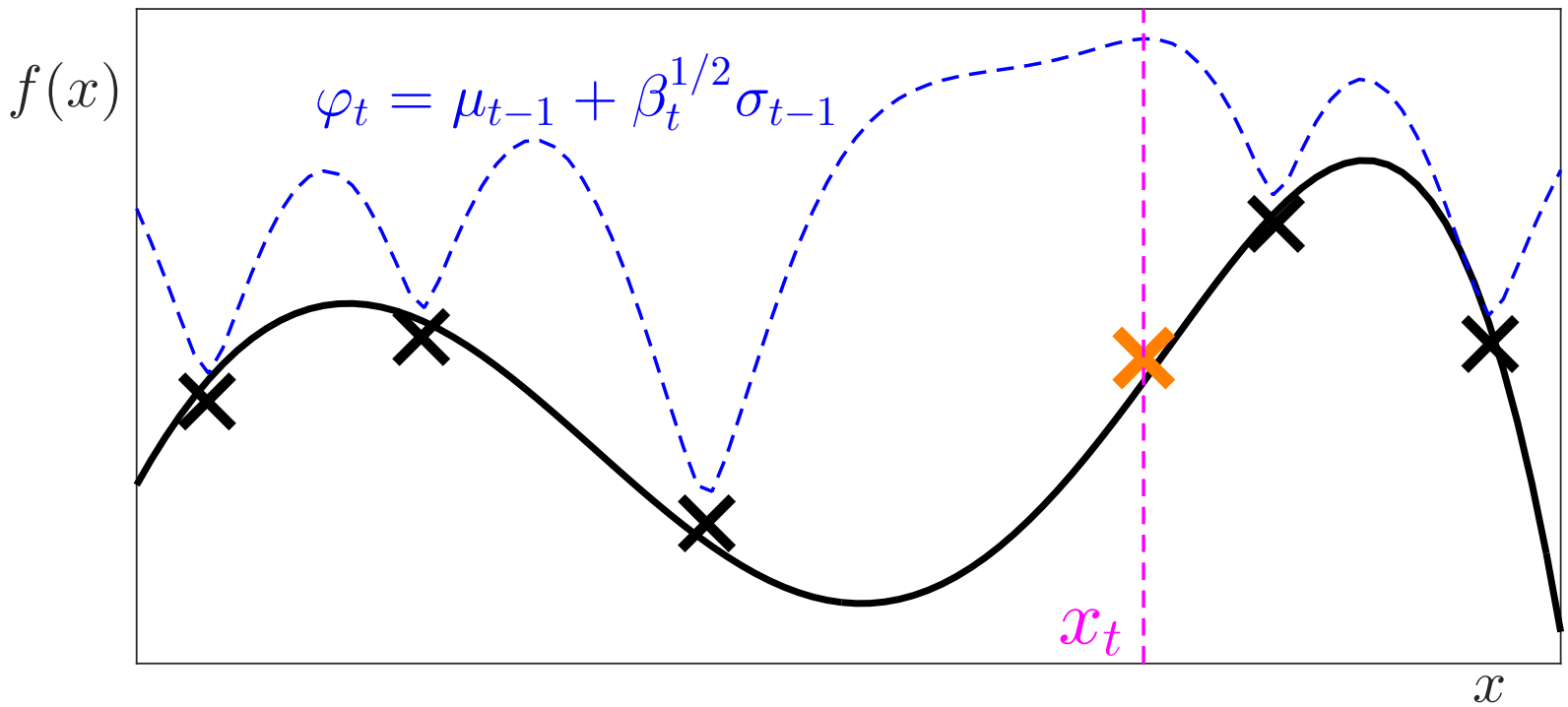
3) Choose $x_t = \operatorname{argmax}_x \varphi_t(x)$.

Algorithm 1: Upper Confidence Bounds in GP Bandits

Model $f \sim \mathcal{GP}(\mathbf{0}, \kappa)$.

Gaussian Process Upper Confidence Bound (GP-UCB)

(Srinivas et al. 2010)



1) Construct posterior $\mathcal{GP}$. 2) $\varphi_t = \mu_{t-1} + \beta_t^{1/2}\sigma_{t-1}$ is a UCB.

3) Choose $x_t = \operatorname{argmax}_x \varphi_t(x)$. 4) Evaluate f at x_t.

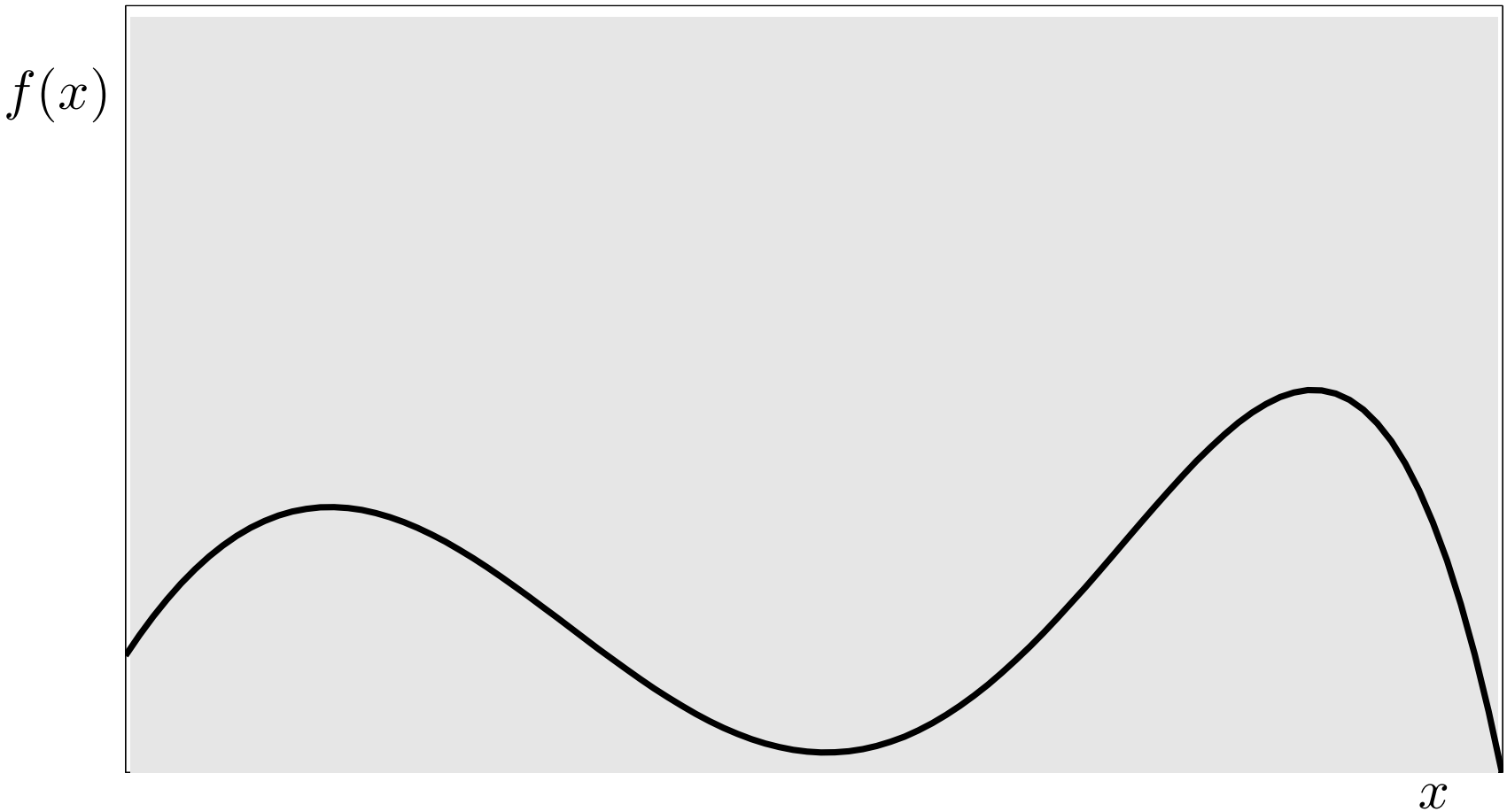
GP-UCB

$$x_t = \operatorname*{argmax}_x \quad \mu_{t-1}(x) + \beta_t^{1/2} \sigma_{t-1}(x)$$

- μ_{t-1}: Exploitation
- σ_{t-1}: Exploration
- β_t controls the tradeoff. $\beta_t \asymp \log t$.

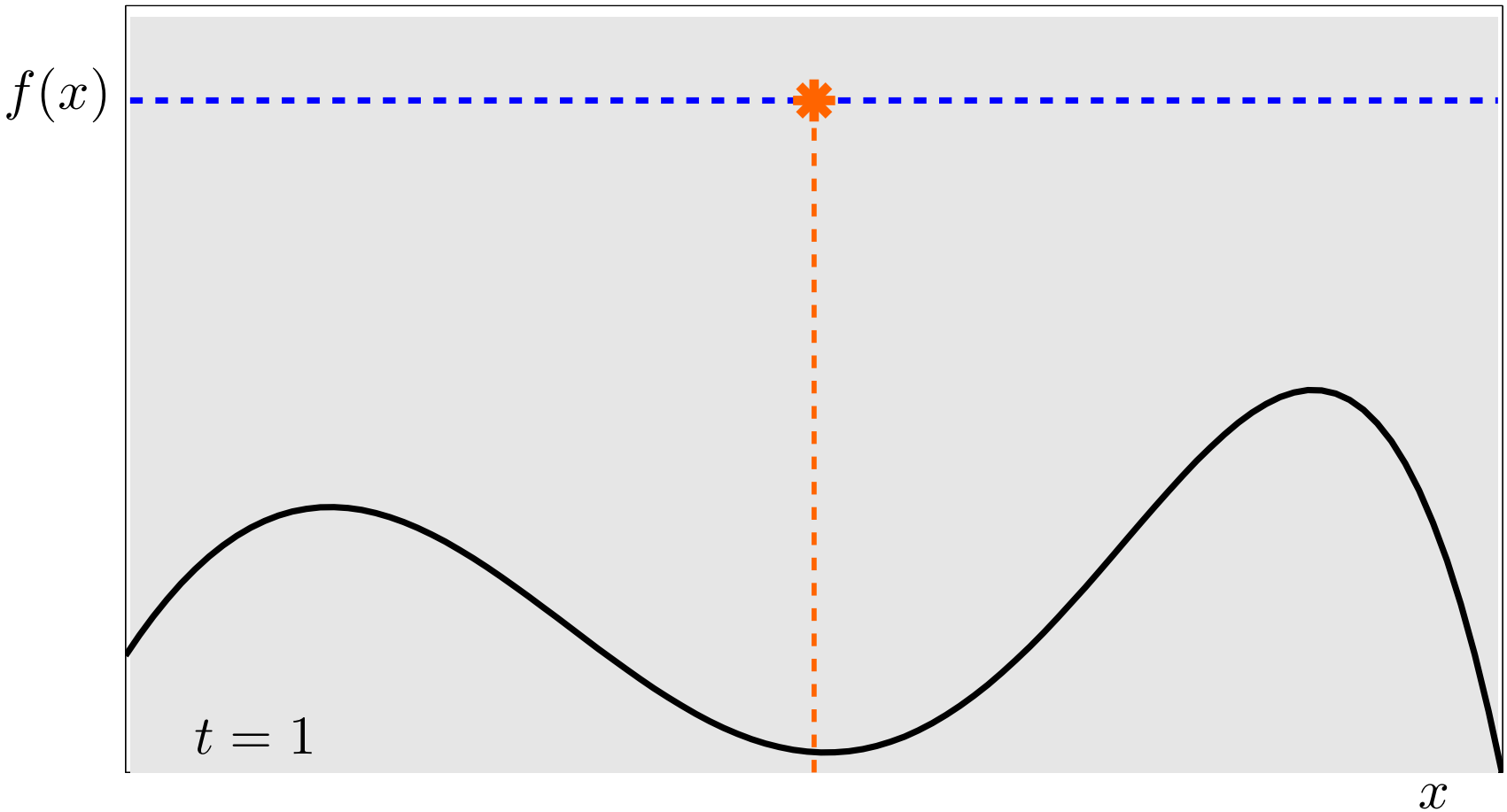
GP-UCB

(Srinivas et al. 2010)



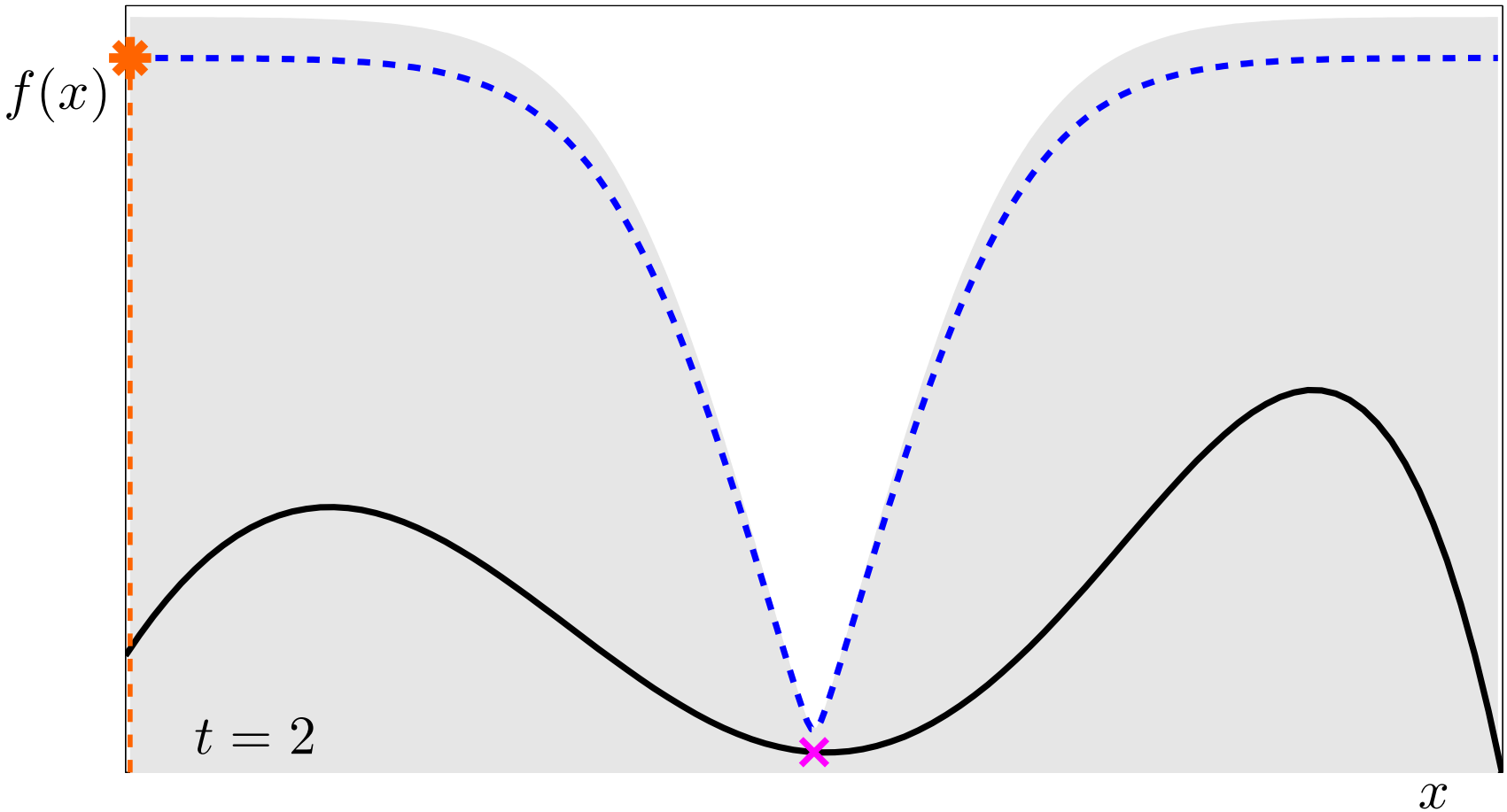
GP-UCB

(Srinivas et al. 2010)



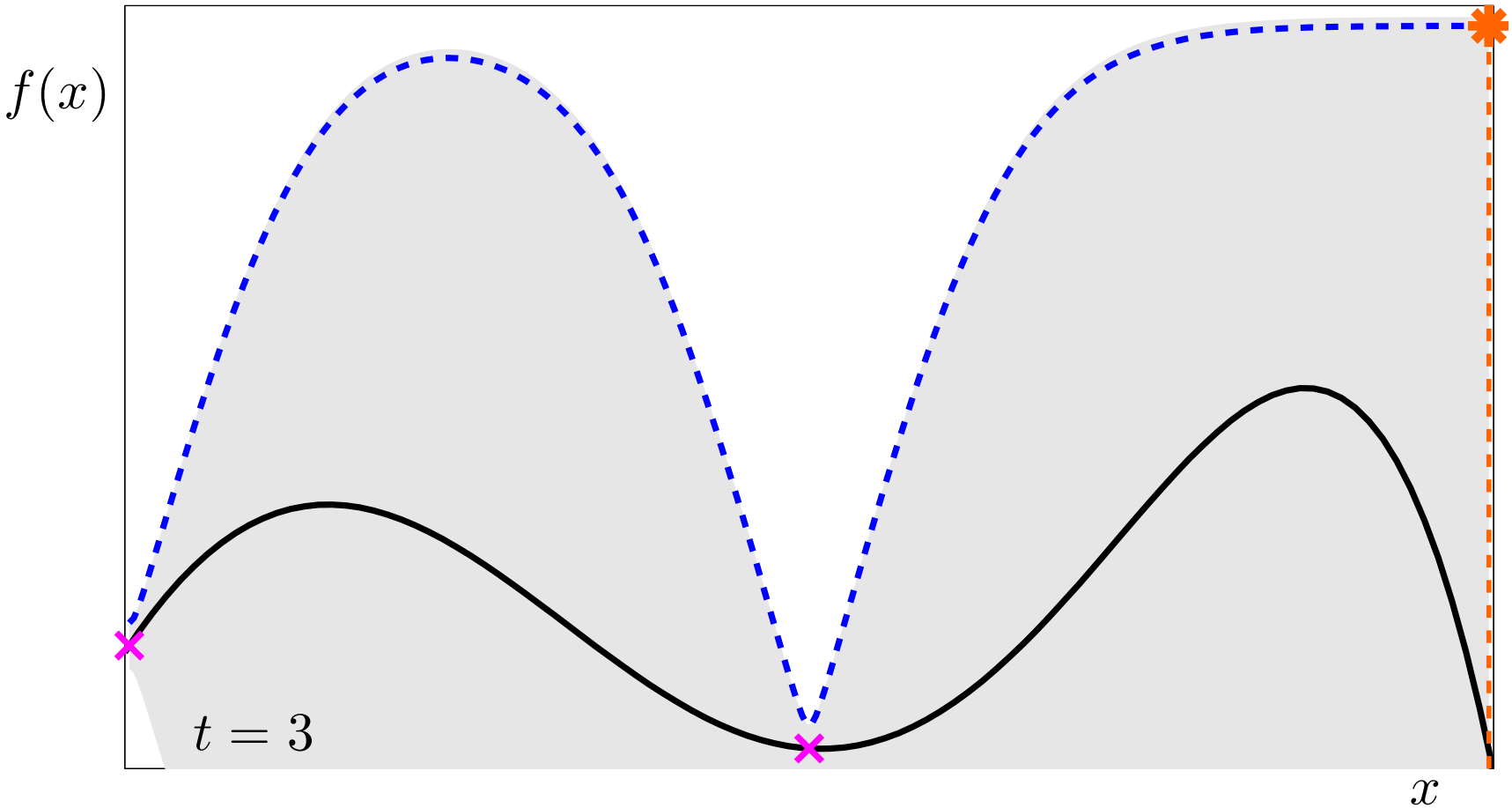
GP-UCB

(Srinivas et al. 2010)



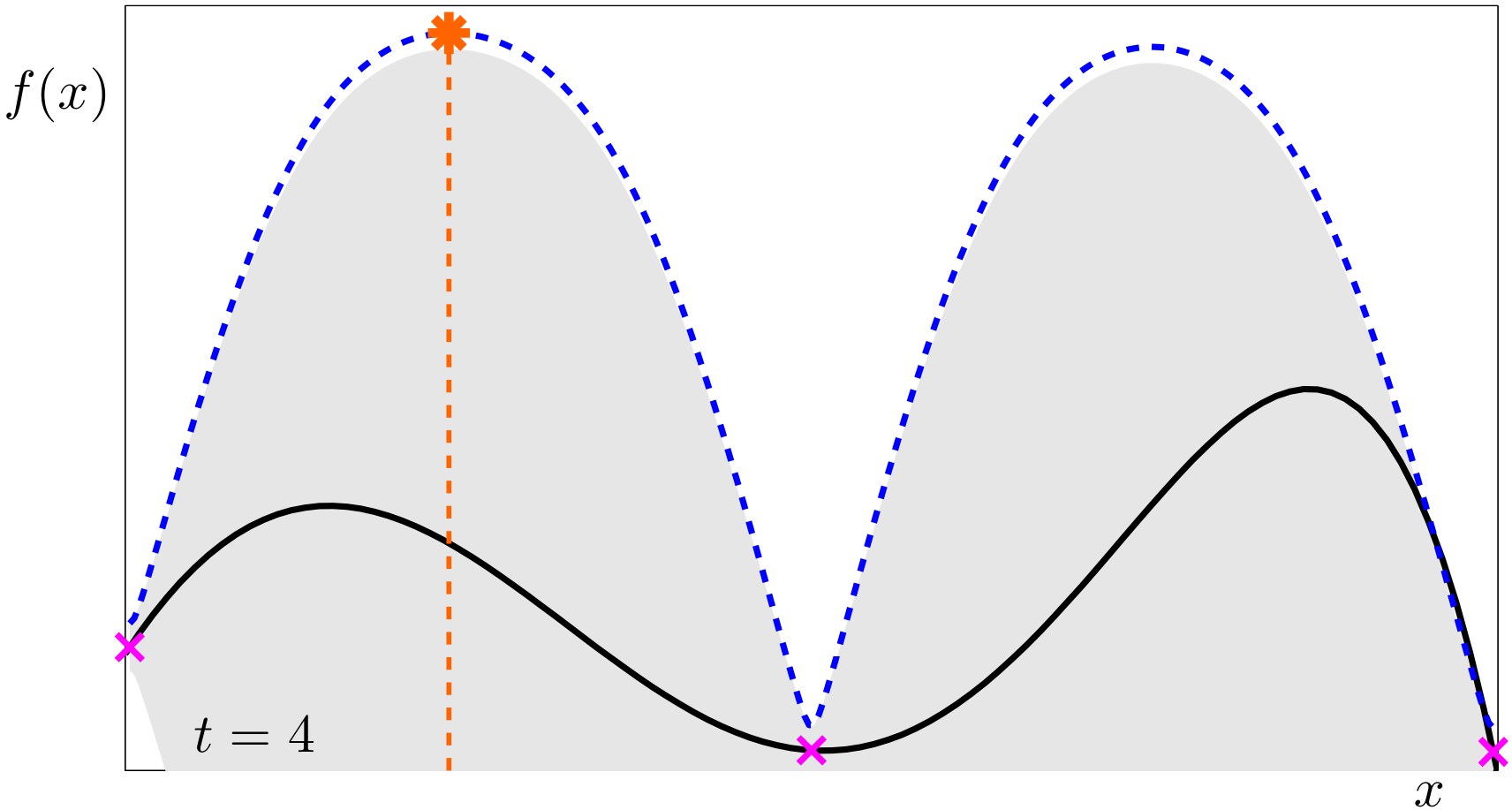
GP-UCB

(Srinivas et al. 2010)



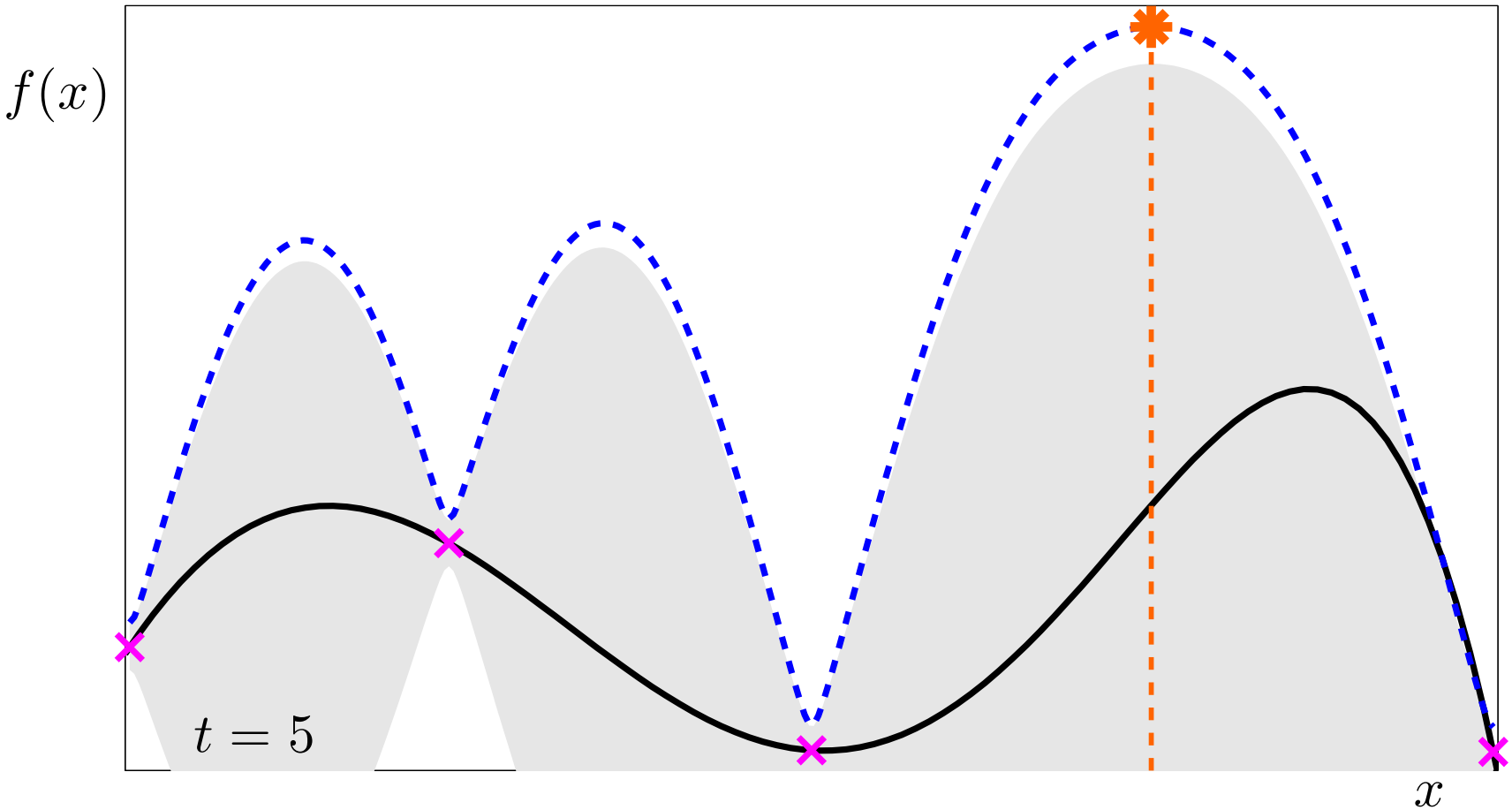
GP-UCB

(Srinivas et al. 2010)



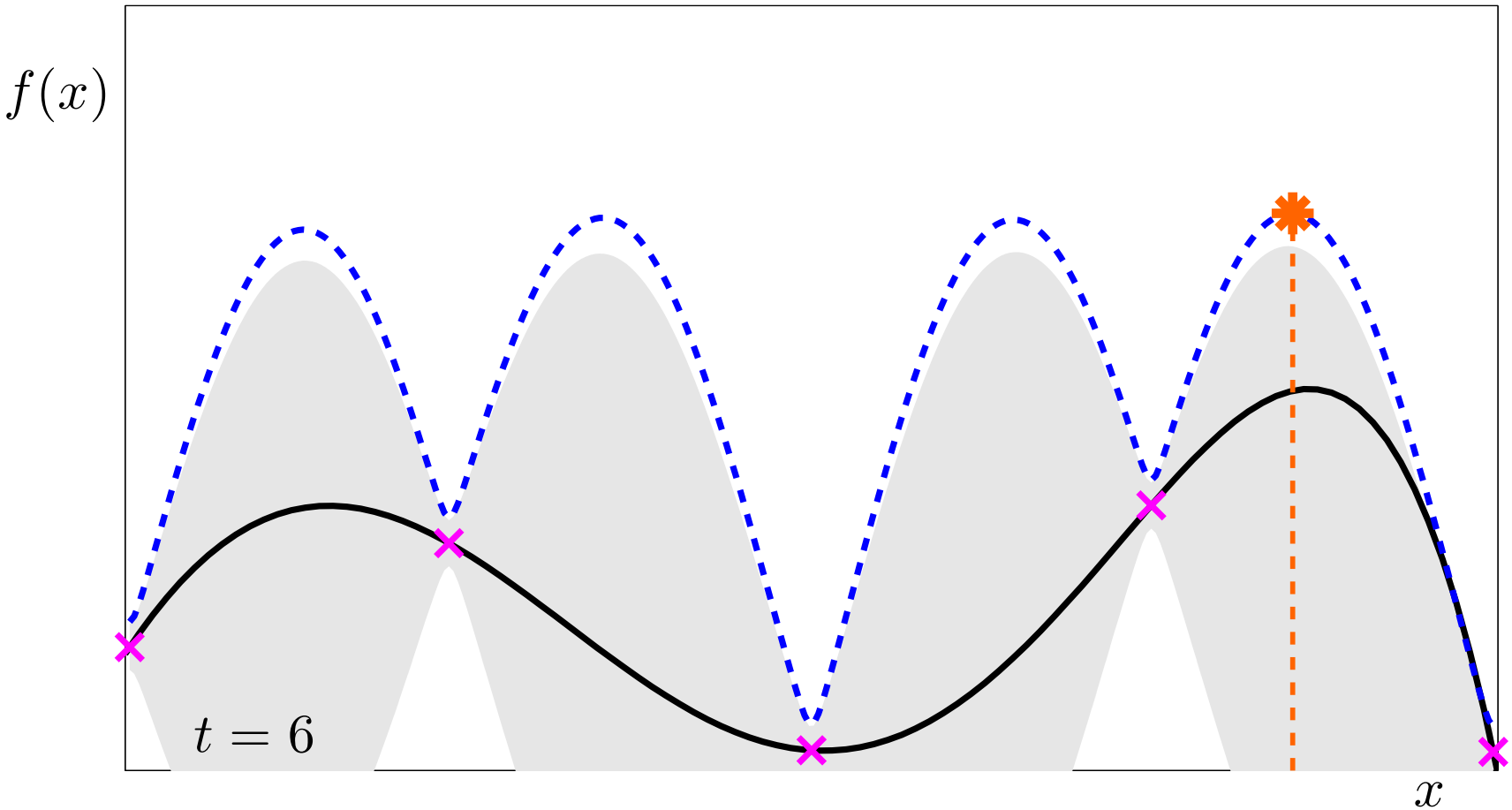
GP-UCB

(Srinivas et al. 2010)



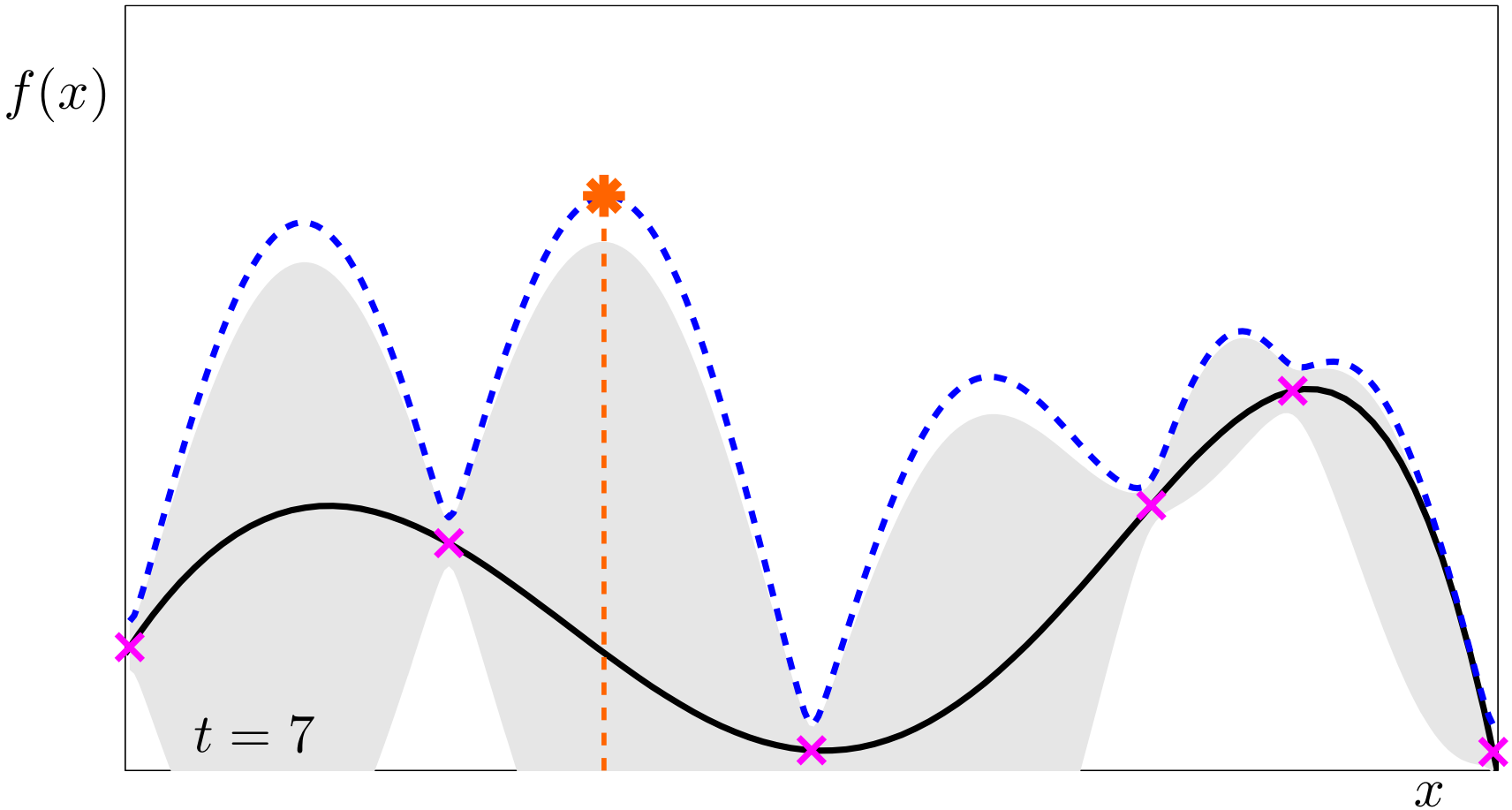
GP-UCB

(Srinivas et al. 2010)



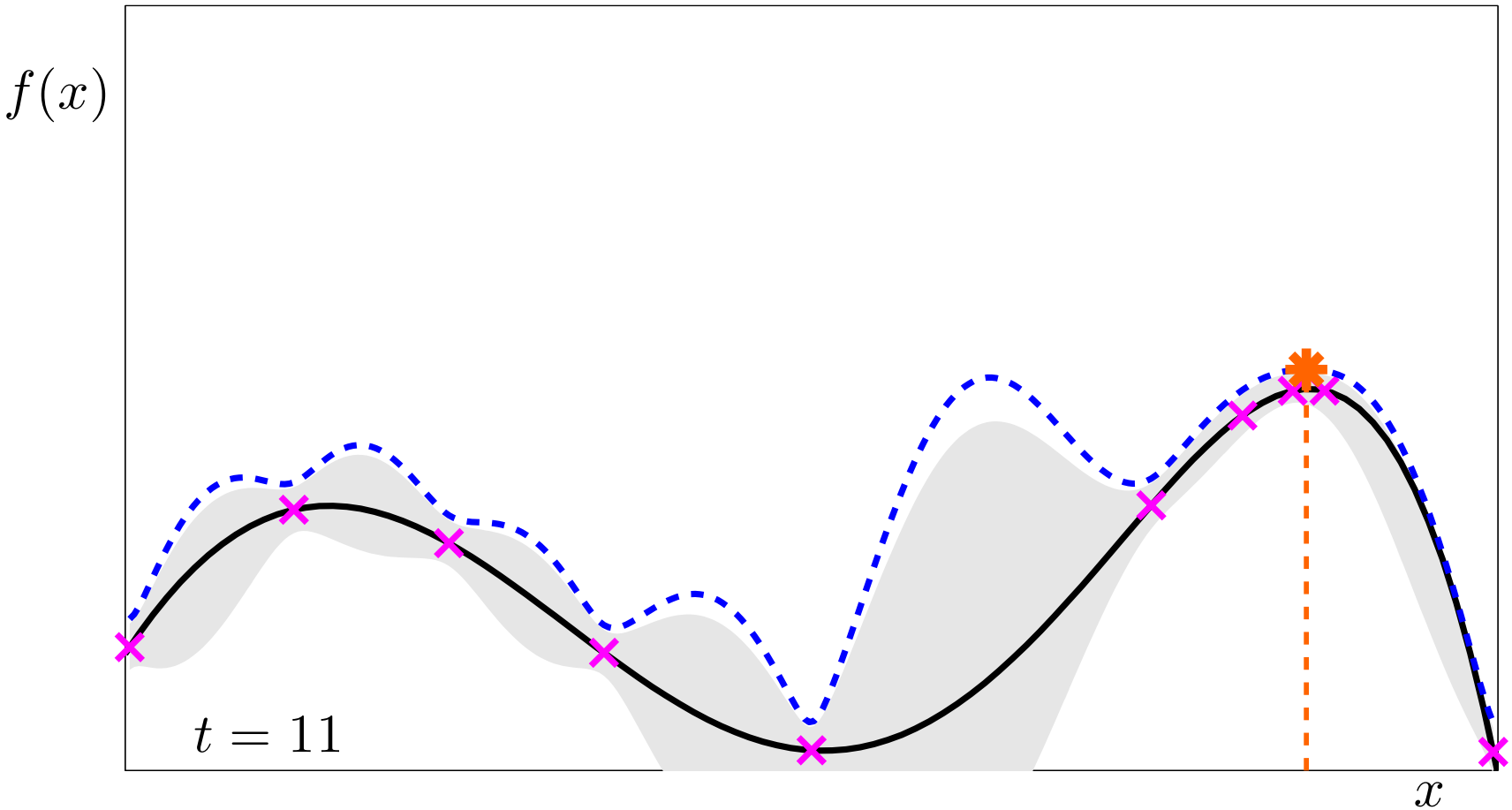
GP-UCB

(Srinivas et al. 2010)



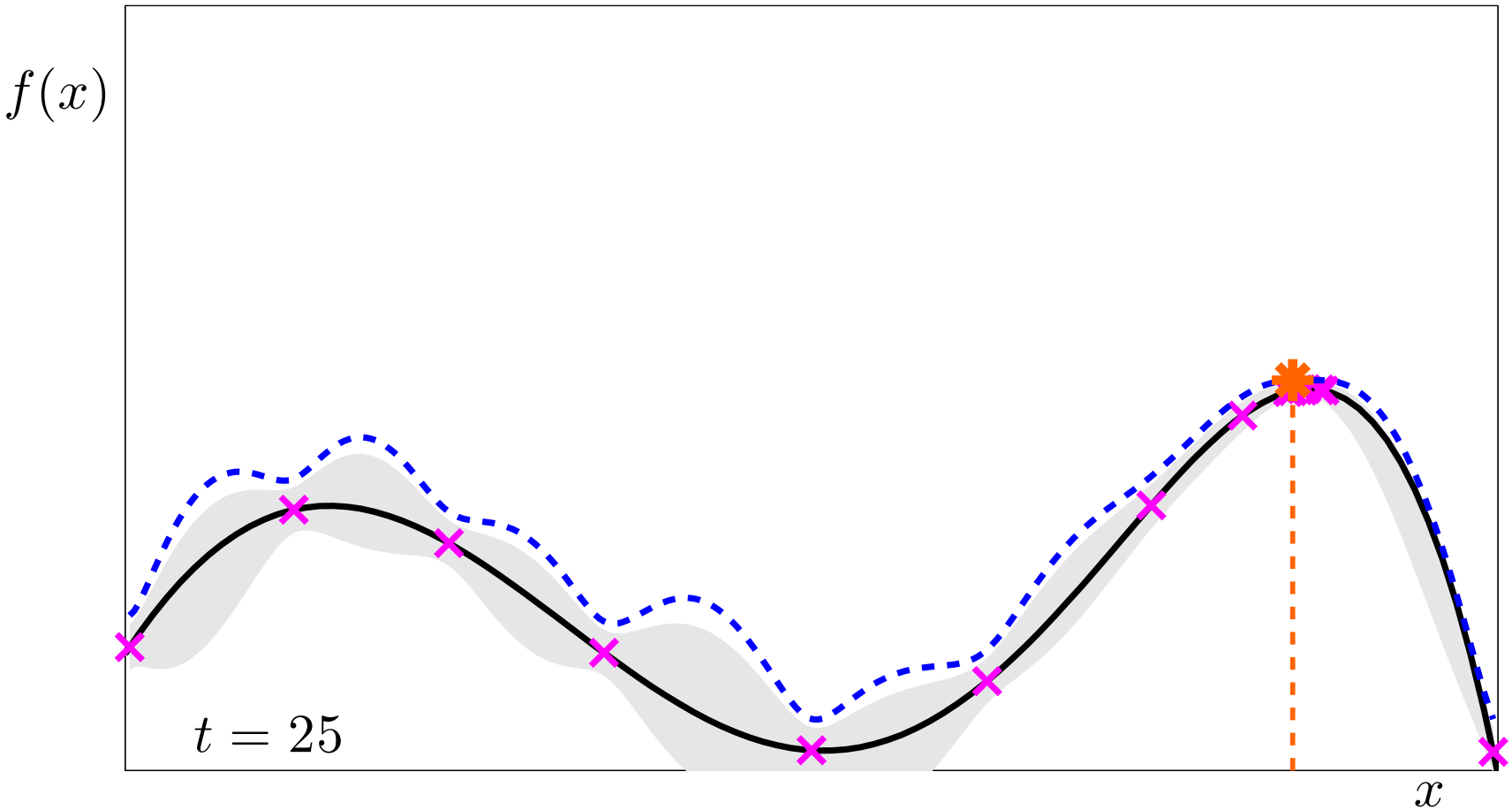
GP-UCB

(Srinivas et al. 2010)



GP-UCB

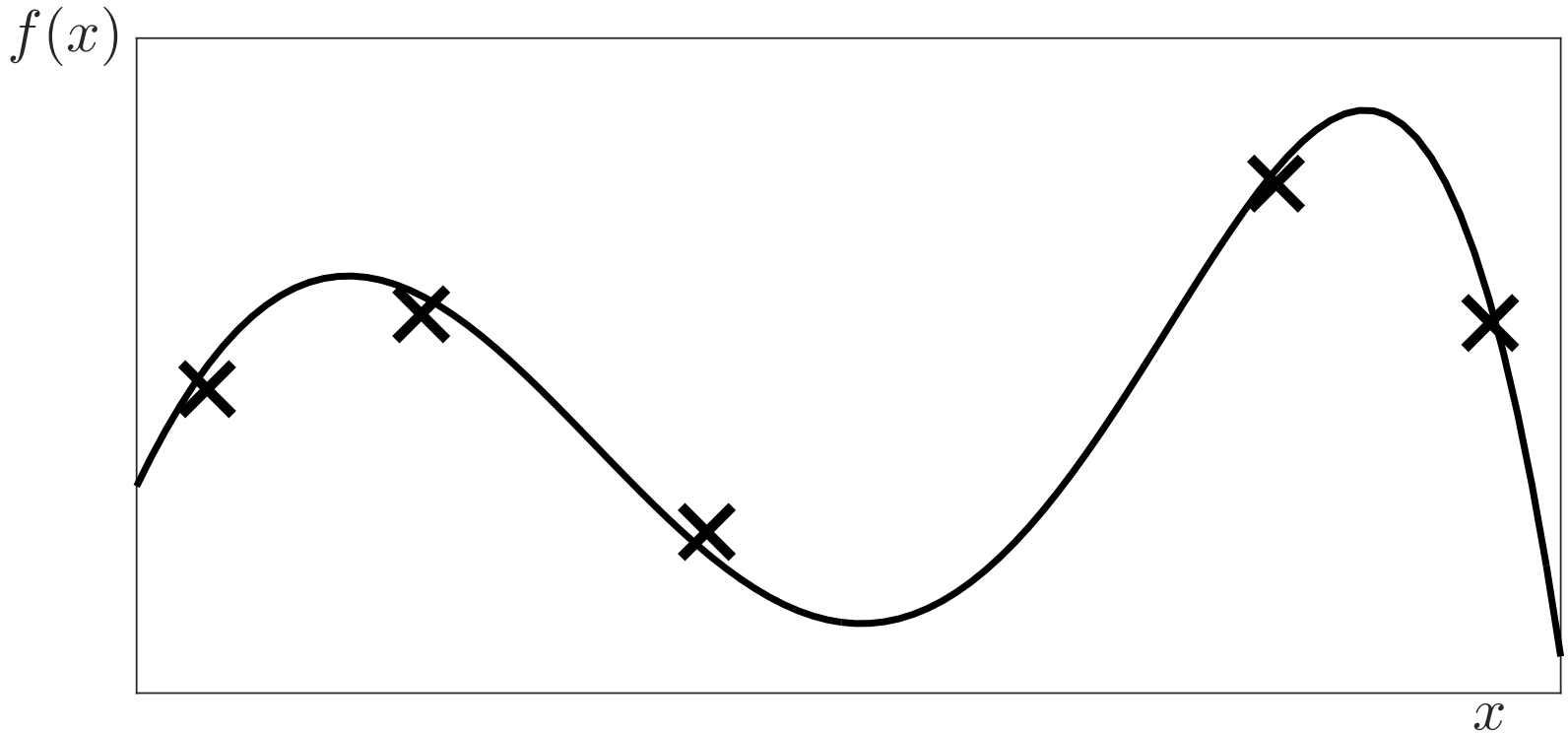
(Srinivas et al. 2010)



Algorithm 2: Thompson Sampling in GP Bandits

Model $f \sim \mathcal{GP}(\mathbf{0}, \kappa)$.

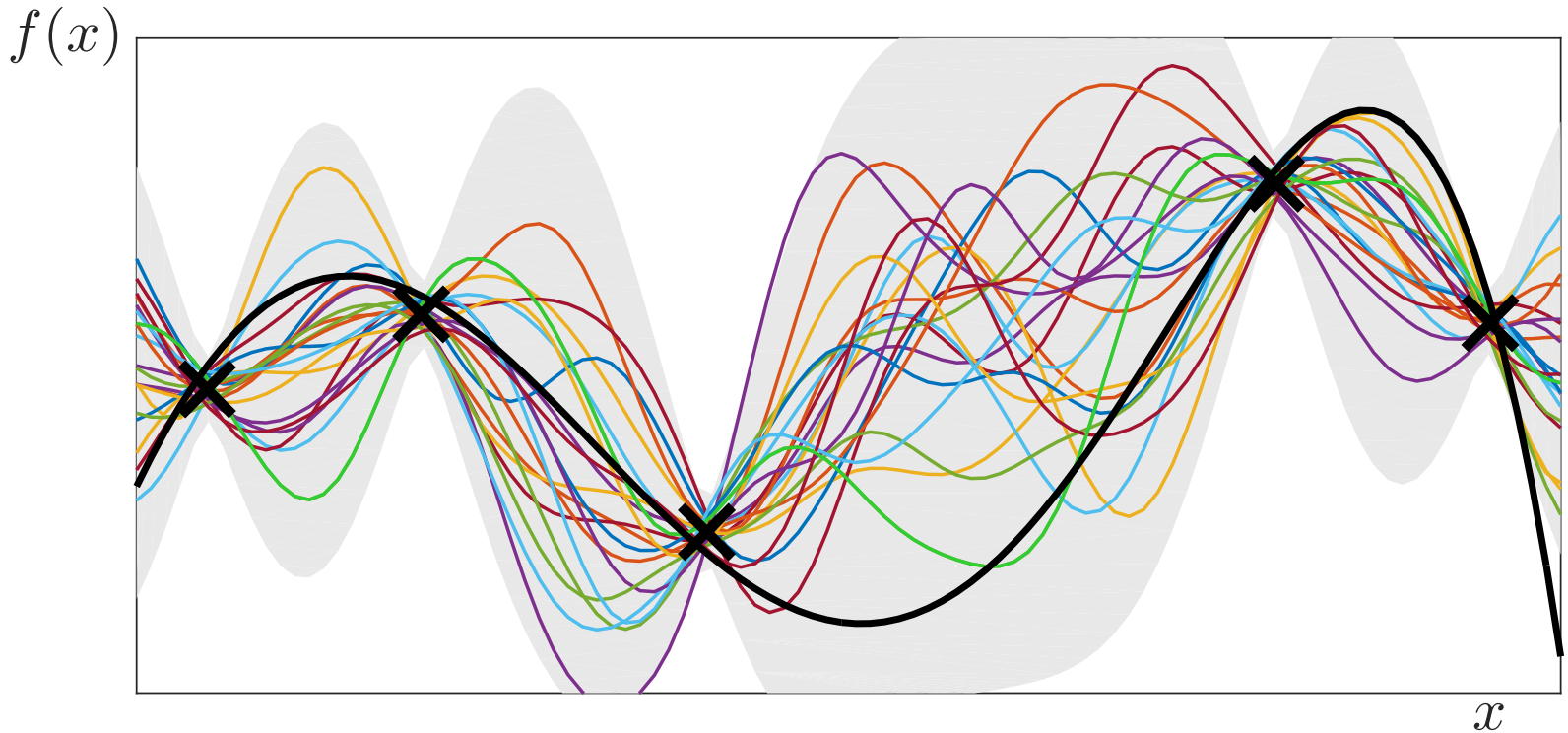
Thompson Sampling (TS) (Thompson, 1933).



Algorithm 2: Thompson Sampling in GP Bandits

Model $f \sim \mathcal{GP}(\mathbf{0}, \kappa)$.

Thompson Sampling (TS) (Thompson, 1933).

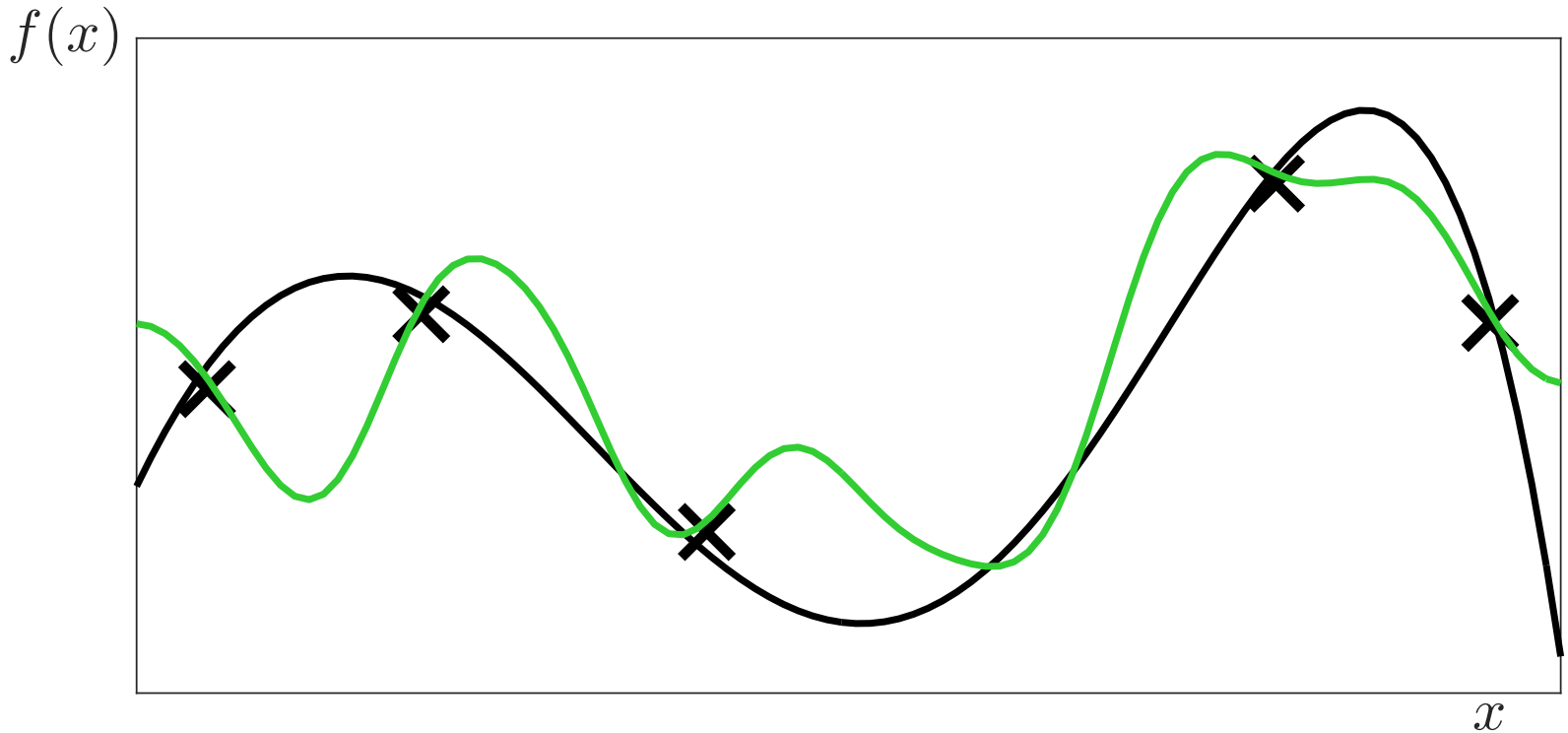


1) Construct posterior $\mathcal{GP}$.

Algorithm 2: Thompson Sampling in GP Bandits

Model $f \sim \mathcal{GP}(\mathbf{0}, \kappa)$.

Thompson Sampling (TS) (Thompson, 1933).

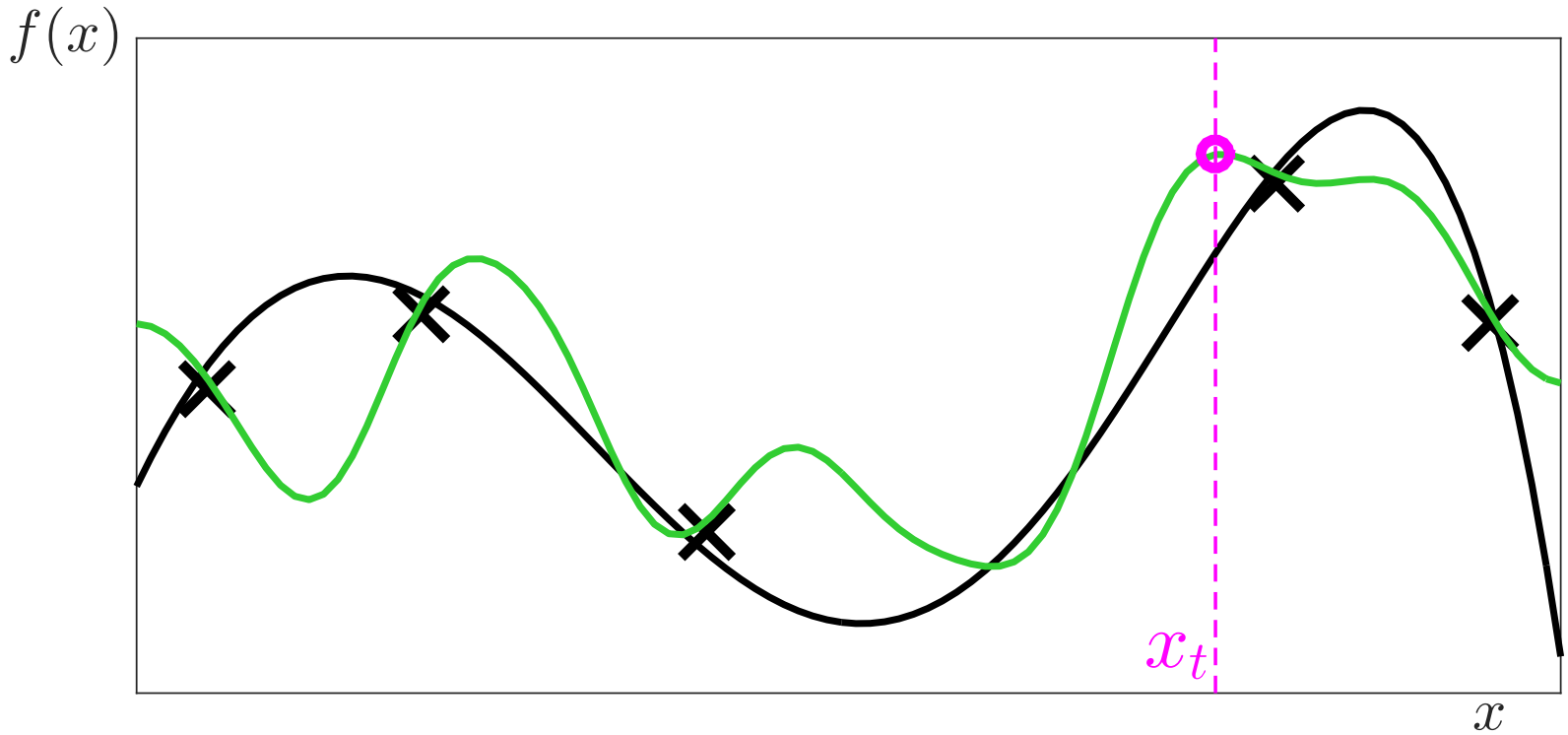


1) Construct posterior $\mathcal{GP}$. 2) Draw sample g from posterior.

Algorithm 2: Thompson Sampling in GP Bandits

Model $f \sim \mathcal{GP}(\mathbf{0}, \kappa)$.

Thompson Sampling (TS) (Thompson, 1933).

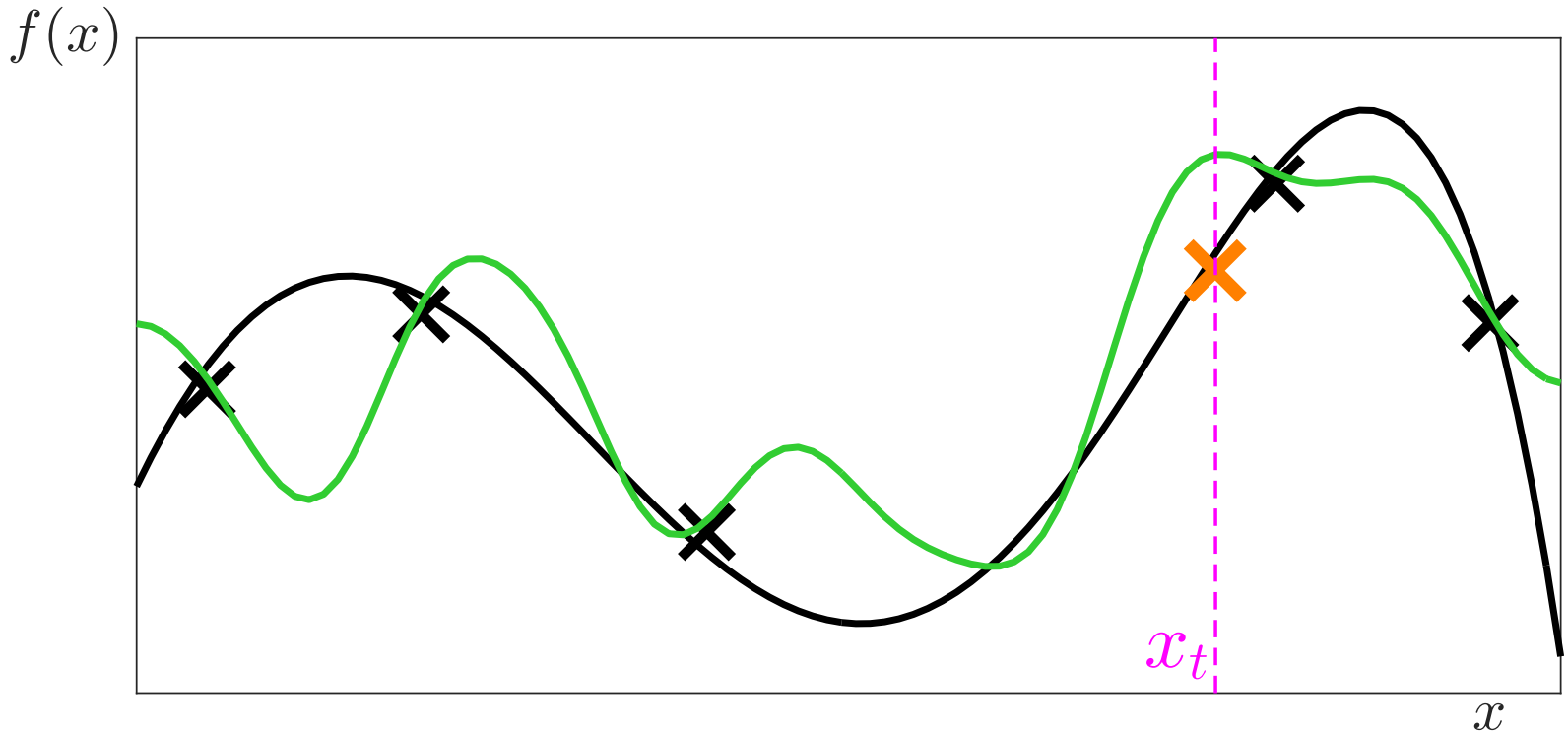


1) Construct posterior $\mathcal{GP}$. 2) Draw sample g from posterior.
3) Choose $x_t = \operatorname{argmax}_x g(x)$.

Algorithm 2: Thompson Sampling in GP Bandits

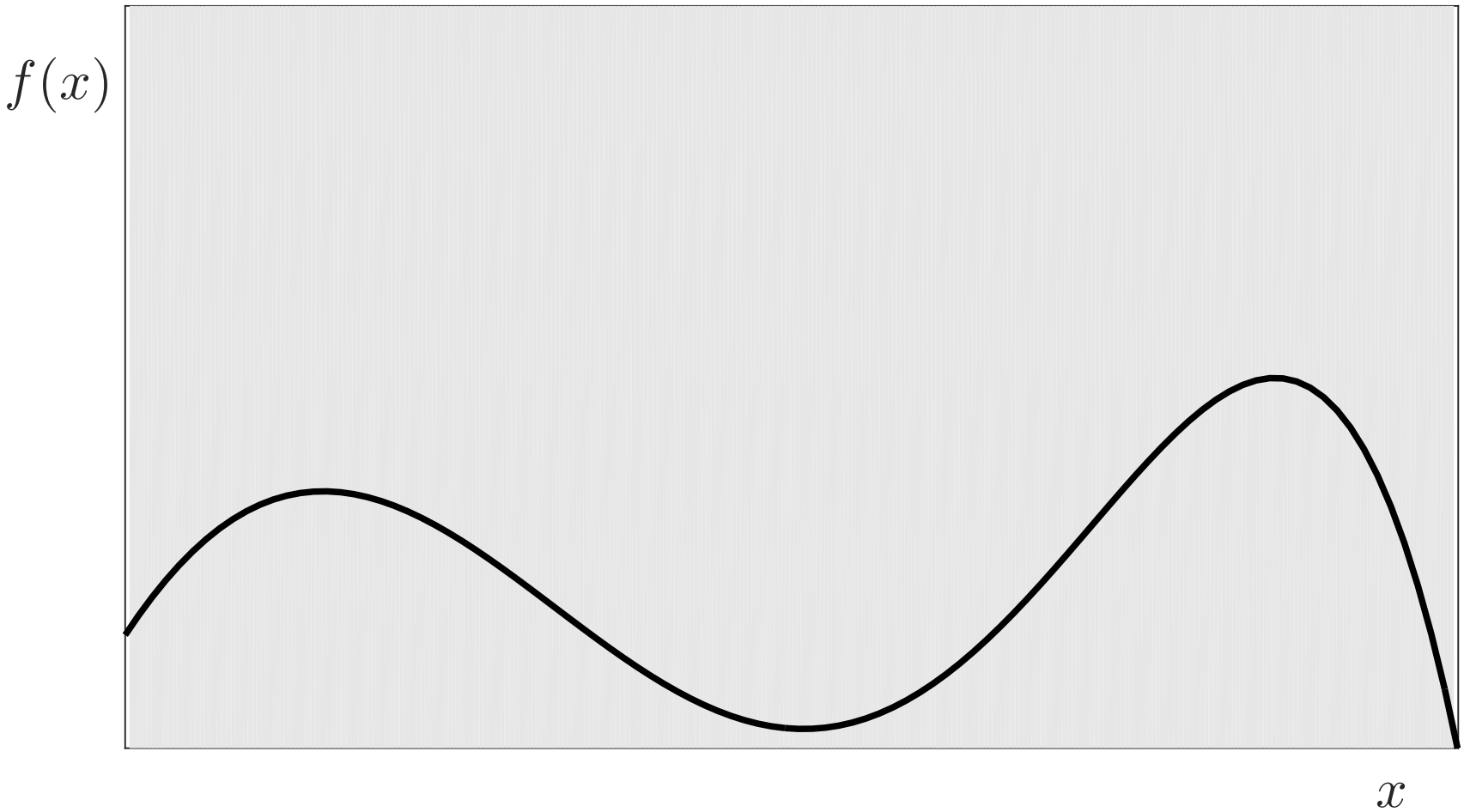
Model $f \sim \mathcal{GP}(\mathbf{0}, \kappa)$.

Thompson Sampling (TS) (Thompson, 1933).

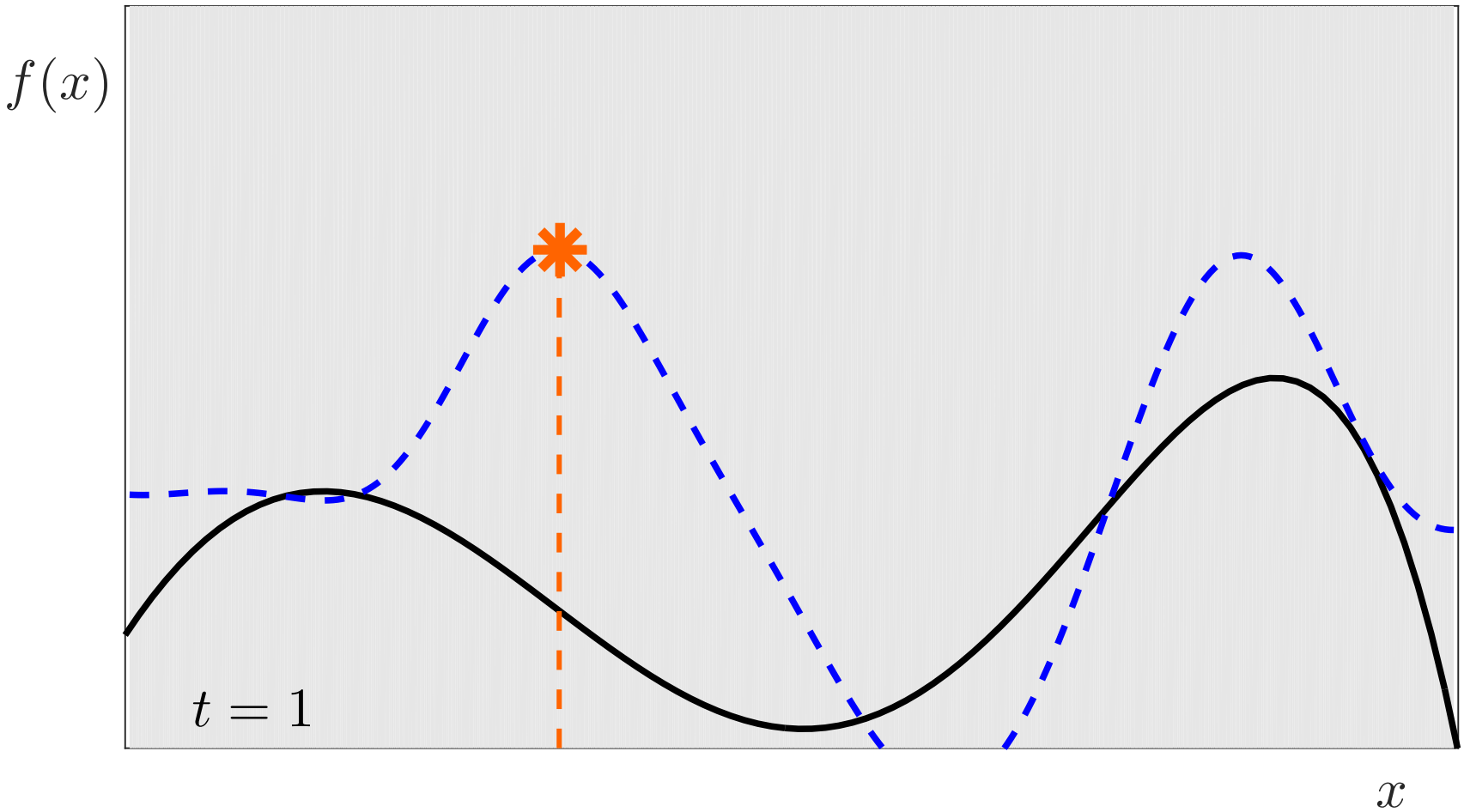


1) Construct posterior $\mathcal{GP}$. 2) Draw sample g from posterior.
3) Choose $x_t = \operatorname{argmax}_x g(x)$. 4) Evaluate f at x_t.

Thompson Sampling (TS) in GPs (Thompson, 1933)

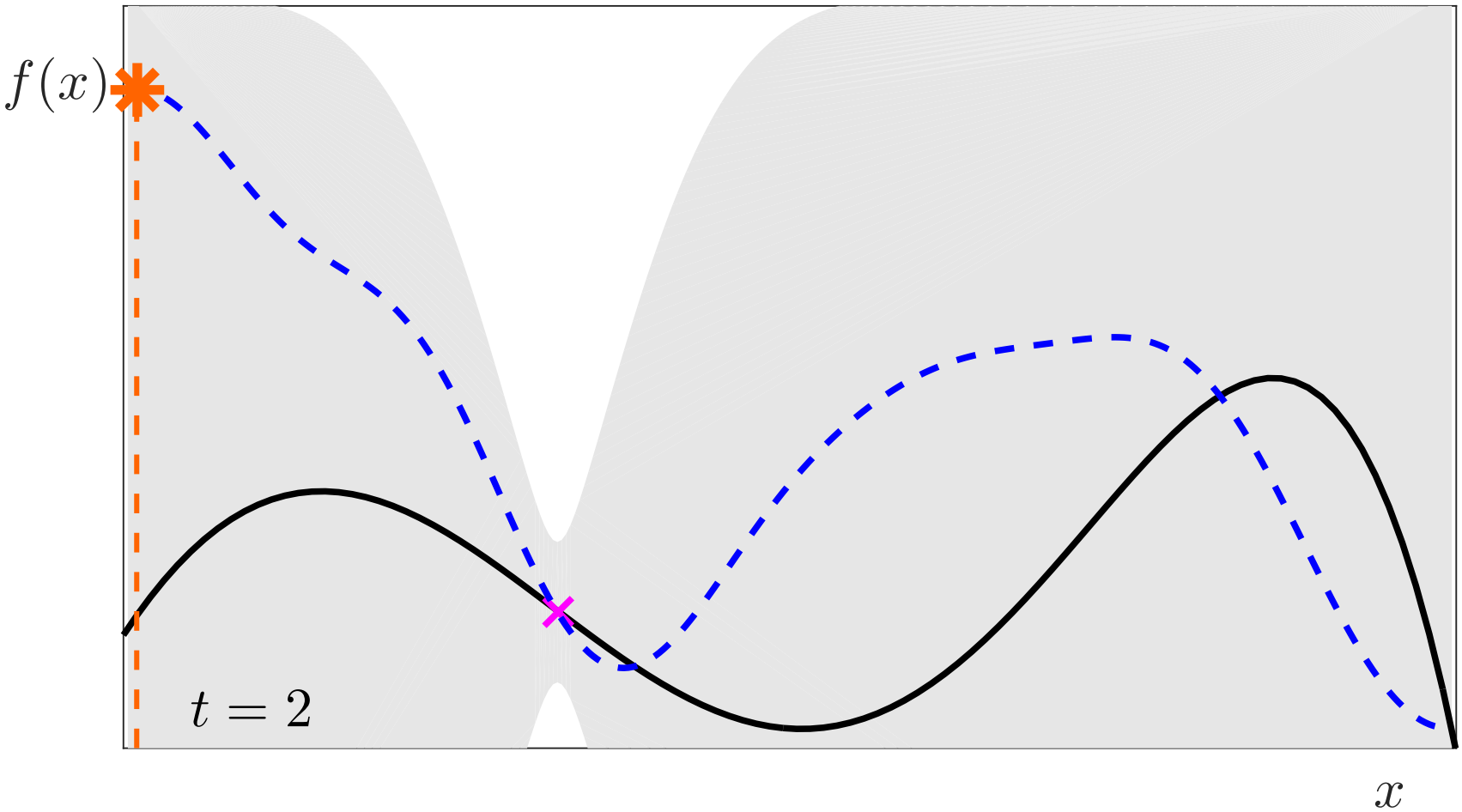


Thompson Sampling (TS) in GPs (Thompson, 1933)

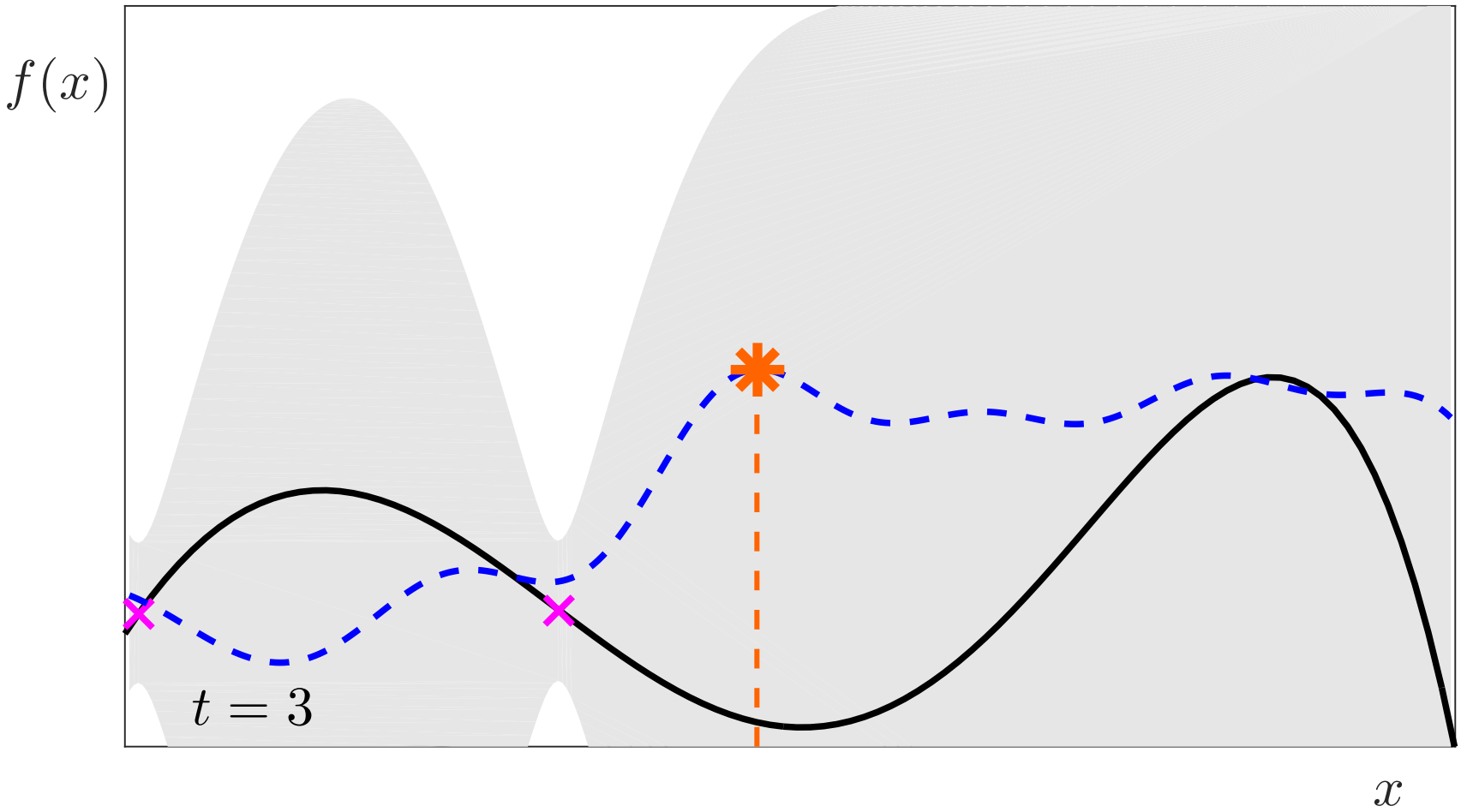


Thompson Sampling (TS) in GPs

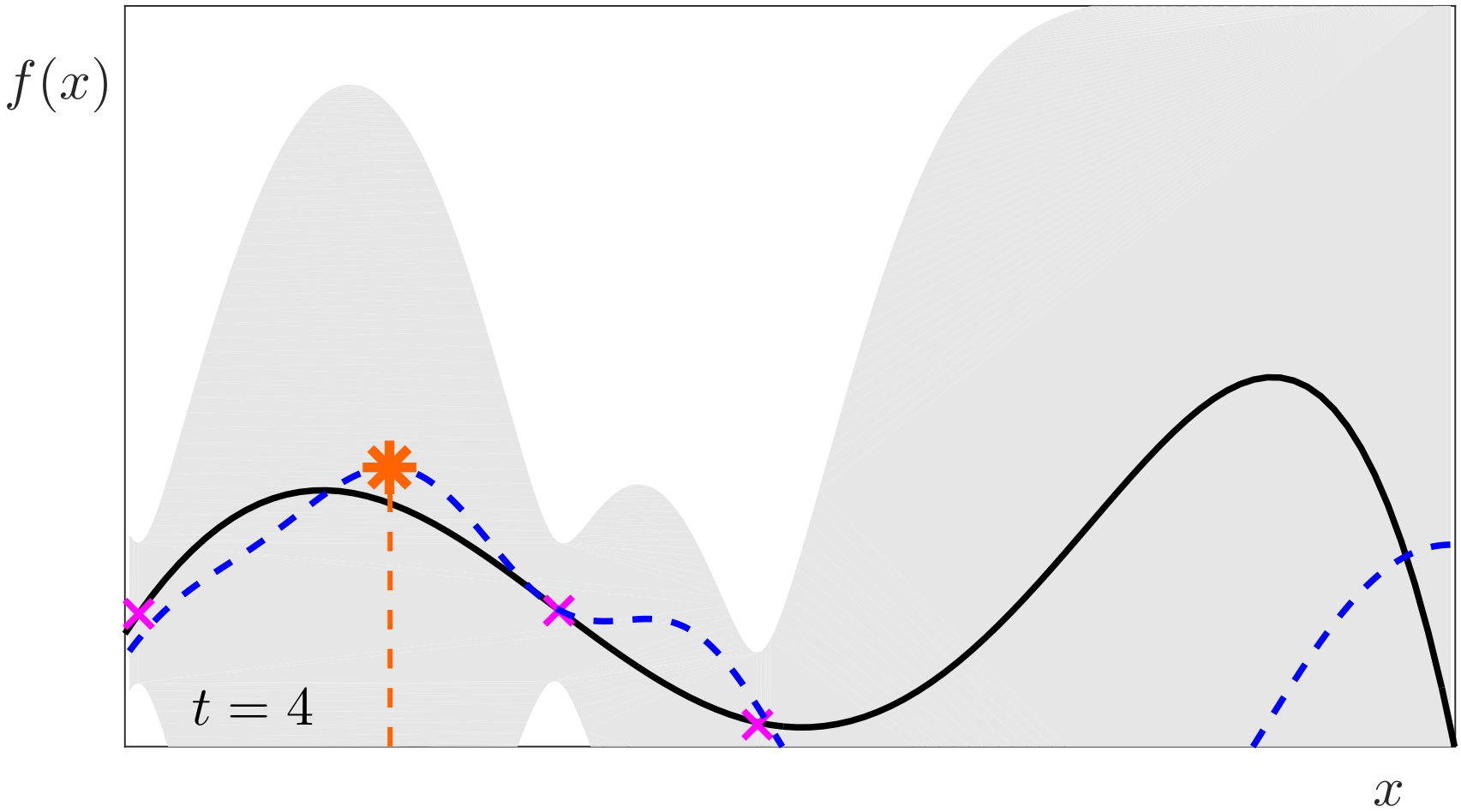
(Thompson, 1933)



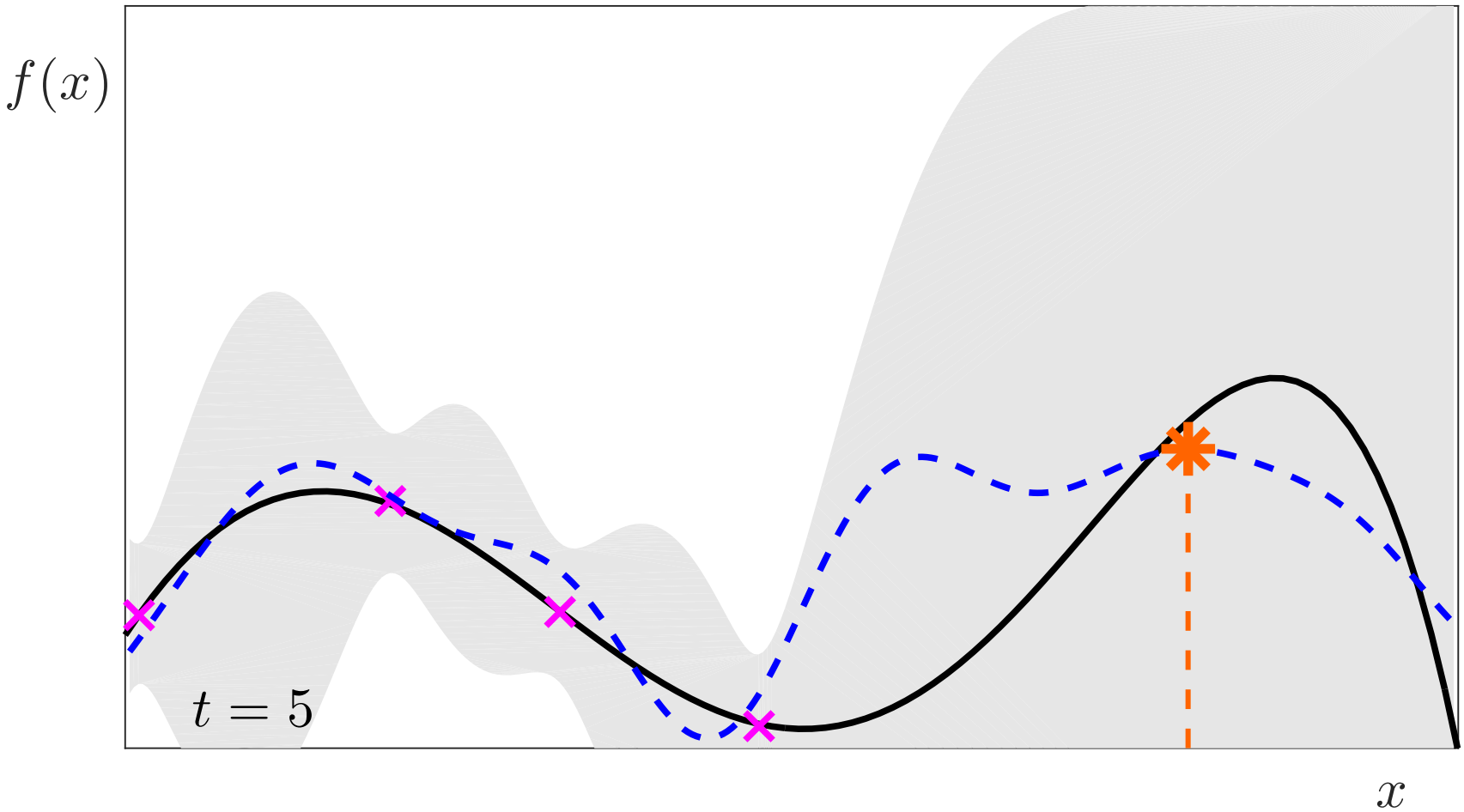
Thompson Sampling (TS) in GPs (Thompson, 1933)



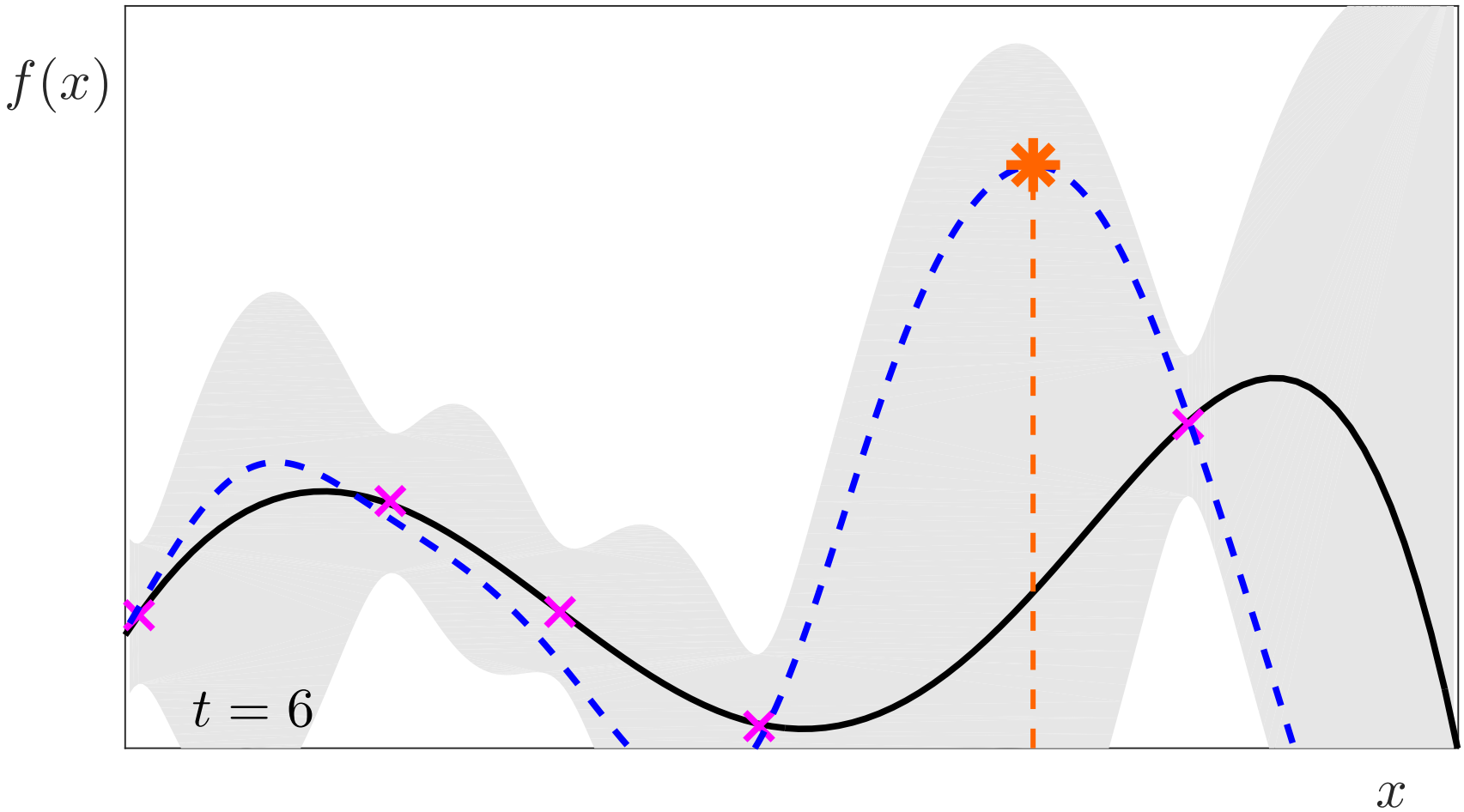
Thompson Sampling (TS) in GPs (Thompson, 1933)



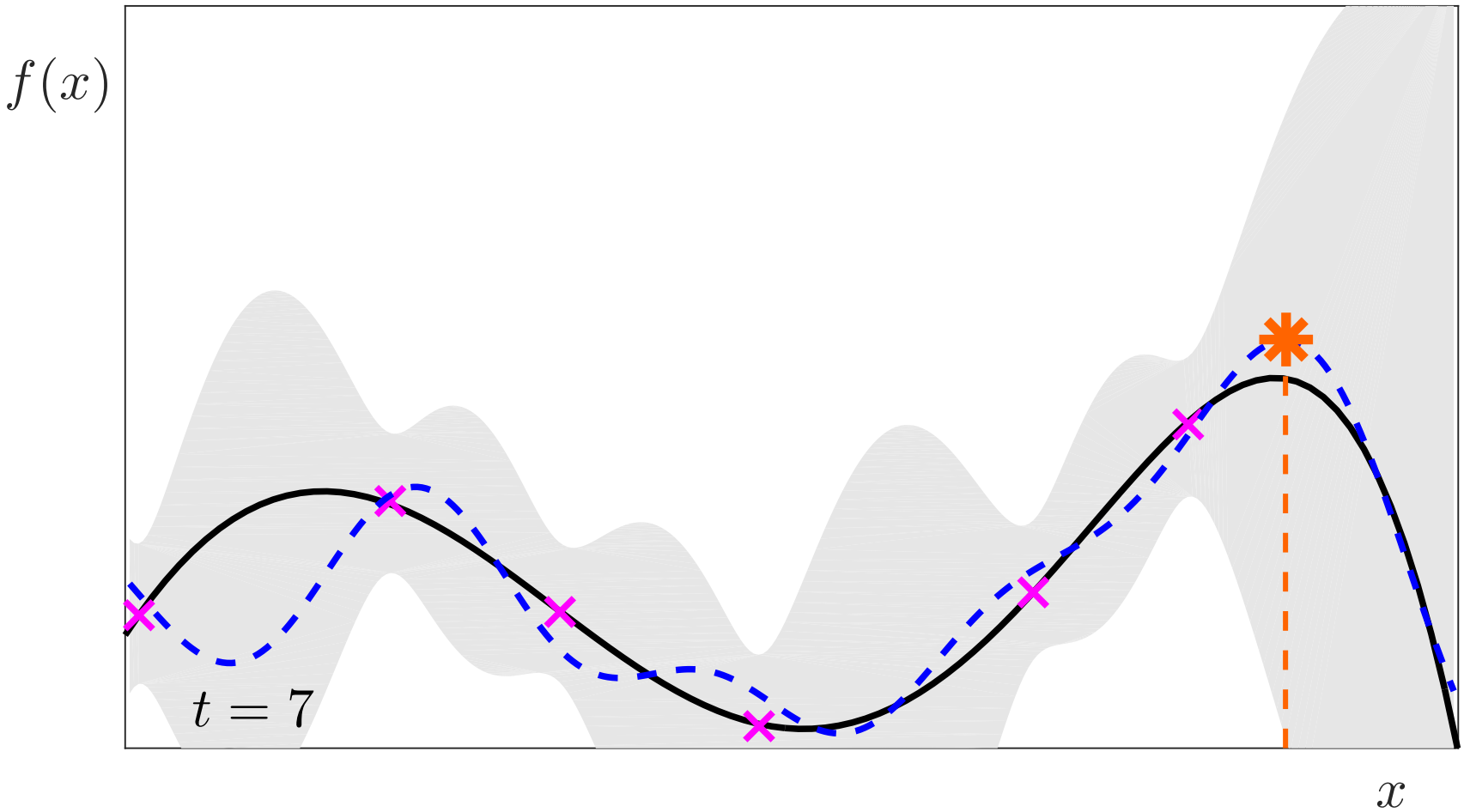
Thompson Sampling (TS) in GPs (Thompson, 1933)



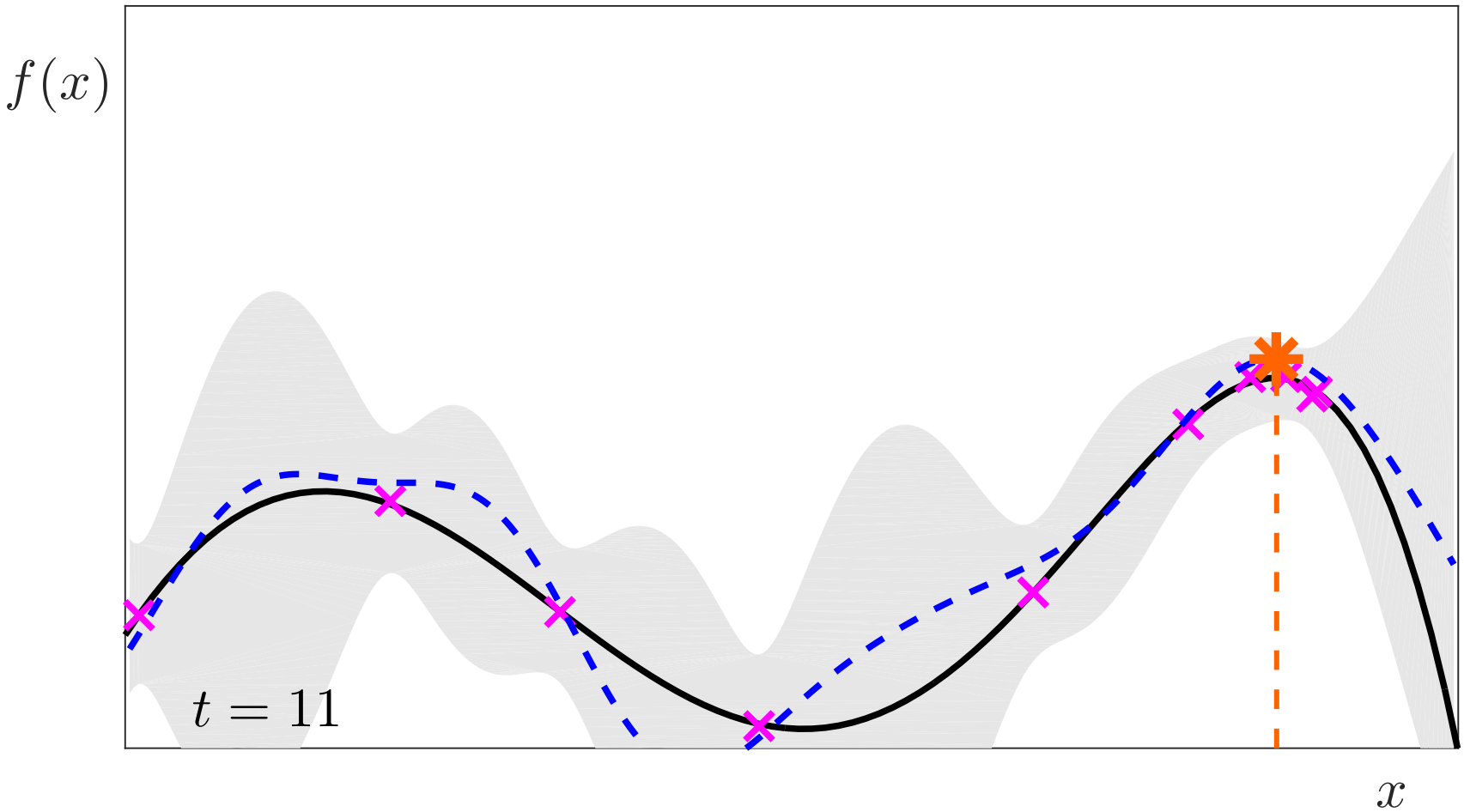
Thompson Sampling (TS) in GPs (Thompson, 1933)



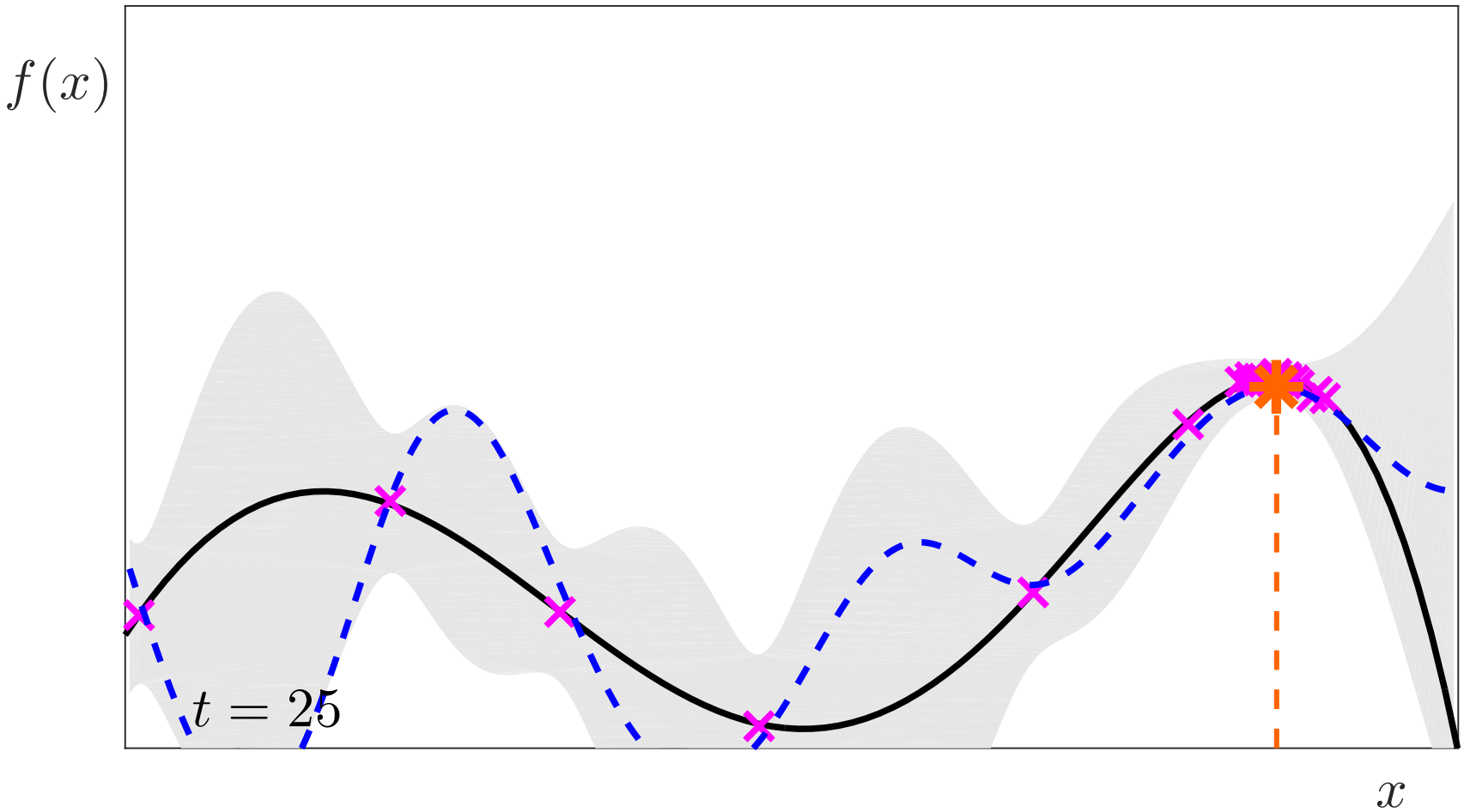
Thompson Sampling (TS) in GPs (Thompson, 1933)



Thompson Sampling (TS) in GPs (Thompson, 1933)



Thompson Sampling (TS) in GPs (Thompson, 1933)



Bandits in the Bayesian Paradigm

Theory: Both UCB and TS will eventually find the optimum under appropriate smoothness assumptions of f. That is,

$$S_n = f(x_\star) - \max_{t=1,\ldots,n} f(x_t) \quad \to 0, \quad \text{as} \quad n \to \infty$$

Bandits in the Bayesian Paradigm

Theory: Both UCB and TS will eventually find the optimum under appropriate smoothness assumptions of f. That is,

$$S_n = f(x_\star) - \max_{t=1,\ldots,n} f(x_t) \quad \to 0, \quad \text{as} \quad n \to \infty$$

Other criteria for selecting x_t:

- Expected improvement (Jones et al. 1998)
- Probability of improvement (Kushner et al. 1964)
- Predictive entropy search (Hernández-Lobato et al. 2014)
- . . . and a few more.

Bandits in the Bayesian Paradigm

Theory: Both UCB and TS will eventually find the optimum under appropriate smoothness assumptions of f. That is,

$$S_n = f(x_\star) - \max_{t=1,\ldots,n} f(x_t) \quad \to 0, \quad \text{as} \quad n \to \infty$$

Other criteria for selecting x_t:

- Expected improvement (Jones et al. 1998)
- Probability of improvement (Kushner et al. 1964)
- Predictive entropy search (Hernández-Lobato et al. 2014)
- . . . and a few more.

Other Bayesian models for f:

- Neural networks (Snoek et al. 2015)
- Random Forests (Hutter 2009)

Outline

- Part I: Bandits in the Bayesian Paradigm
  1. Gaussian processes
  2. Algorithms: Upper Confidence Bound (UCB) & Thompson Sampling (TS)
- Part II: Scaling up Bandits
  1. Multi-fidelity bandit: cheap approximations to an expensive experiment
  2. Parallelising function evaluations
  3. High dimensional input spaces

Outline

- Part I: Bandits in the Bayesian Paradigm
  1. Gaussian processes
  2. Algorithms: Upper Confidence Bound (UCB) & Thompson Sampling (TS)
- Part II: Scaling up Bandits
  1. Multi-fidelity bandit: cheap approximations to an expensive experiment
  2. Parallelising function evaluations
  3. High dimensional input spaces

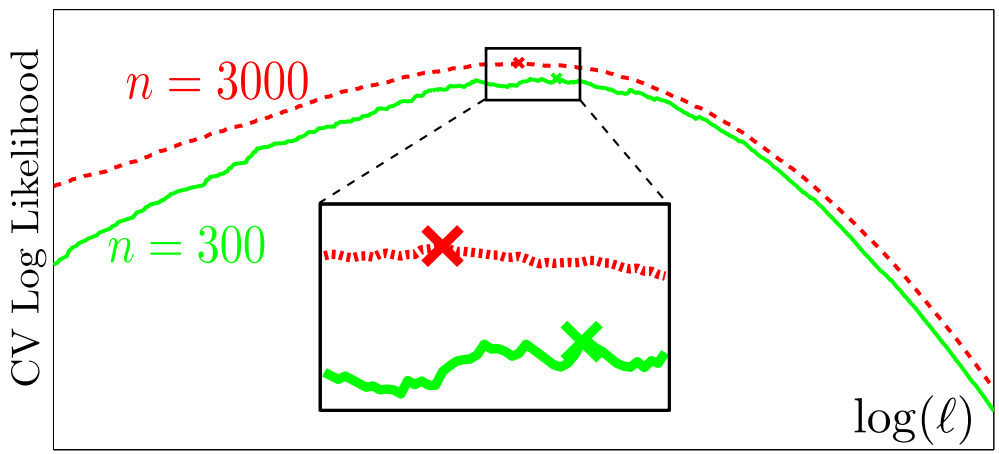
(N.B: Part II is a shameless plug for my research.)

Part 2.1: Multi-fidelity Bandits

Motivating question:
What if we have cheap approximations to f?

1. Hyper-parameter tuning: Train & validate with a subset of the data, and/or early stopping before convergence.

   E.g. Bandwidth (ℓ) selection in kernel density estimation.

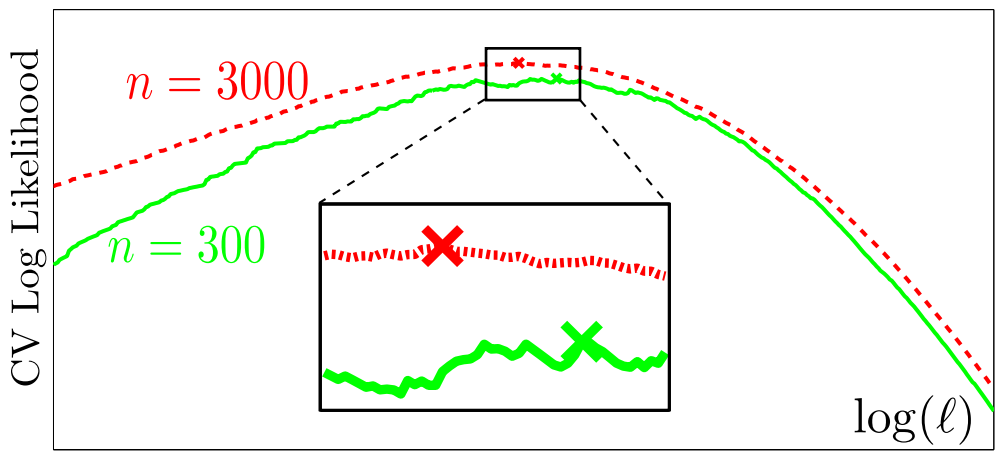


Part 2.1: Multi-fidelity Bandits

Motivating question:
What if we have cheap approximations to f?

1. Hyper-parameter tuning: Train & validate with a subset of the data, and/or early stopping before convergence.

   E.g. Bandwidth (ℓ) selection in kernel density estimation.



2. Computational astrophysics: cosmological simulations and numerical computations with less granularity.

3. Autonomous driving: simulation vs real world experiment.

Multi-fidelity Methods

For specific applications,

- Industrial design (Forrester et al. 2007)
- Hyper-parameter tuning (Agarwal et al. 2011, Klein et al. 2015, Li et al. 2016)
- Active learning (Zhang & Chaudhuri 2015)
- Robotics (Cutler et al. 2014)

Multi-fidelity bandits & optimisation (Huang et al. 2006, Forrester et al. 2007, March & Wilcox 2012, Poloczek et al. 2016)

Multi-fidelity Methods

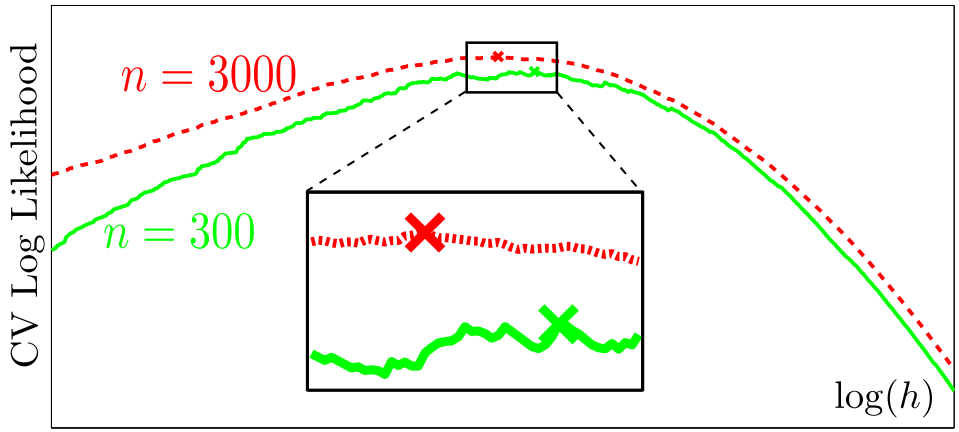
For specific applications,

- Industrial design (Forrester et al. 2007)
- Hyper-parameter tuning (Agarwal et al. 2011, Klein et al. 2015, Li et al. 2016)
- Active learning (Zhang & Chaudhuri 2015)
- Robotics (Cutler et al. 2014)

Multi-fidelity bandits & optimisation (Huang et al. 2006, Forrester et al. 2007, March & Wilcox 2012, Poloczek et al. 2016)

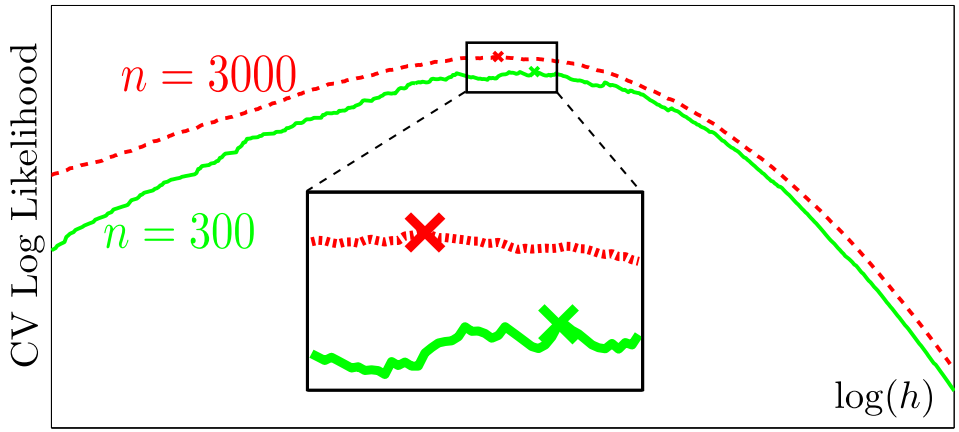
. . . with theoretical guarantees (Kandasamy et al. NIPS 2016a&b, Kandasamy et al. ICML 2017)

Multi-fidelity Bandits for Hyper-parameter tuning



- Use an arbitrary amount of data?
- Iterative algorithms: use arbitrary number of iterations?

Multi-fidelity Bandits for Hyper-parameter tuning

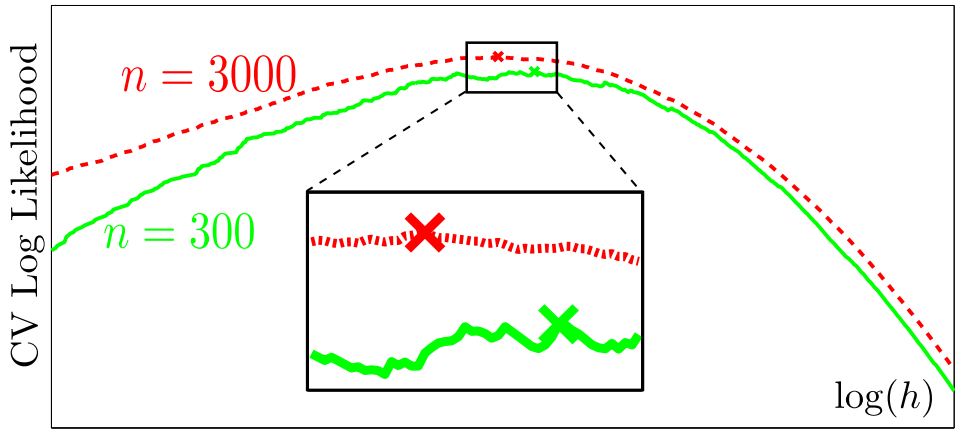


- Use an arbitrary amount of data?
- Iterative algorithms: use arbitrary number of iterations?

E.g. Train an ML model with $N_\bullet$ data and $T_\bullet$ iterations.

- But use $N < N_\bullet$ data and $T < T_\bullet$ iterations to approximate cross validation performance at $(N_\bullet, T_\bullet)$.

Multi-fidelity Bandits for Hyper-parameter tuning



- Use an arbitrary amount of data?
- Iterative algorithms: use arbitrary number of iterations?

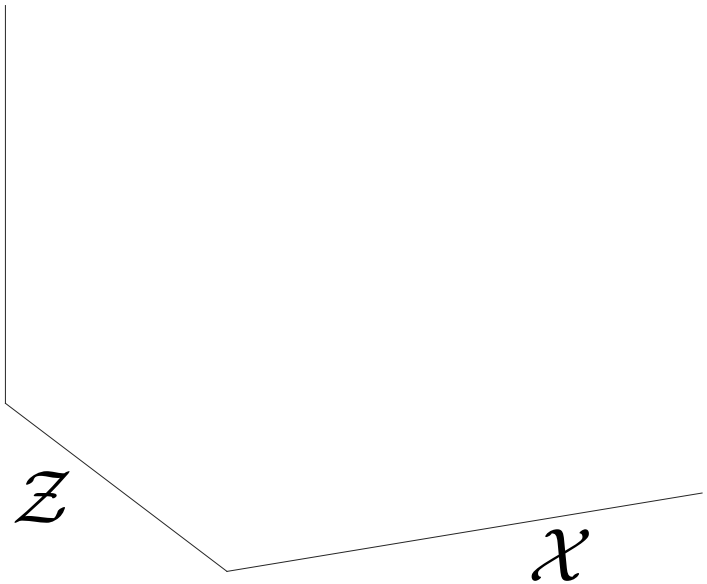
E.g. Train an ML model with $N_\bullet$ data and $T_\bullet$ iterations.

- But use $N < N_\bullet$ data and $T < T_\bullet$ iterations to approximate cross validation performance at $(N_\bullet, T_\bullet)$.

Approximations from a *continuous* 2D "fidelity space" (N, T).

Multi-fidelity Bandits

(Kandasamy et al. ICML 2017)



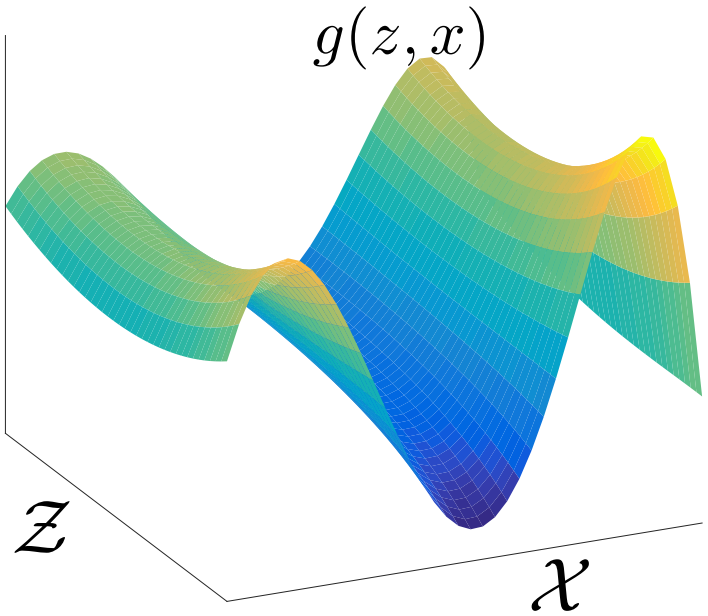
A fidelity space $\mathcal{Z}$ and domain $\mathcal{X}$

$\mathcal{Z} \leftarrow$ all (N, T) values.

$\mathcal{X} \leftarrow$ all hyper-parameter values.

Multi-fidelity Bandits

(Kandasamy et al. ICML 2017)



A fidelity space $\mathcal{Z}$ and domain $\mathcal{X}$

$\mathcal{Z} \leftarrow$ all (N, T) values.

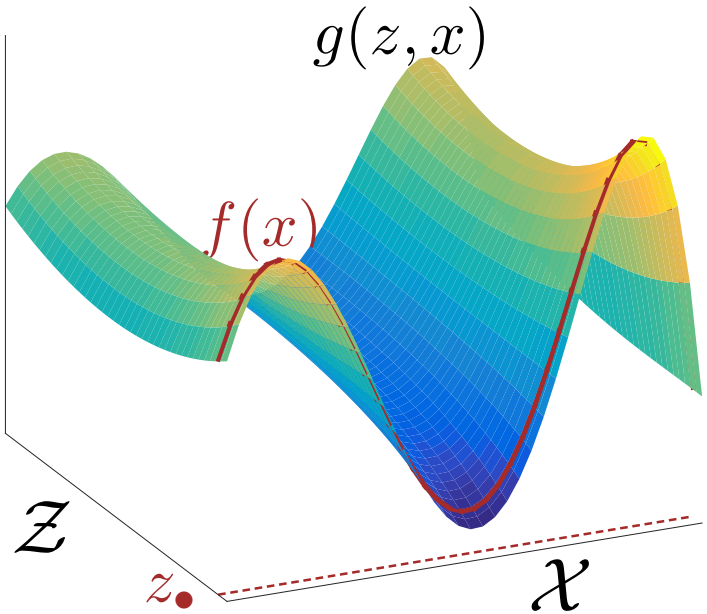
$\mathcal{X} \leftarrow$ all hyper-parameter values.

$g : \mathcal{Z} \times \mathcal{X} \rightarrow \mathbb{R}$.

$g([N, T], x) \leftarrow$ cv accuracy when training with N data for T iterations at hyper-parameter x.

Multi-fidelity Bandits

(Kandasamy et al. ICML 2017)



A fidelity space $\mathcal{Z}$ and domain $\mathcal{X}$

$\mathcal{Z} \leftarrow$ all (N, T) values.

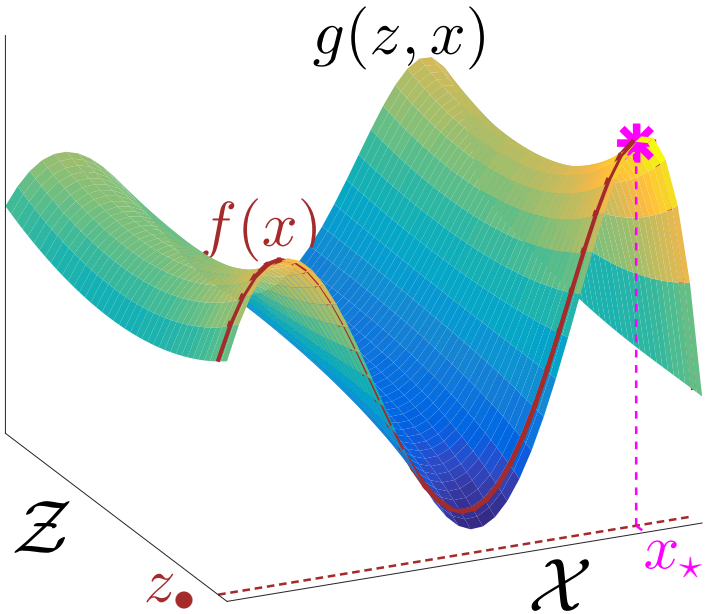
$\mathcal{X} \leftarrow$ all hyper-parameter values.

$g : \mathcal{Z} \times \mathcal{X} \rightarrow \mathbb{R}$.

$g([N, T], x) \leftarrow$ cv accuracy when training with N data for T iterations at hyper-parameter x.

Denote $f(x) = g(z_\bullet, x)$ where $z_\bullet \in \mathcal{Z}$. $z_\bullet = [N_\bullet, T_\bullet]$.

Multi-fidelity Bandits (Kandasamy et al. ICML 2017)



A fidelity space $\mathcal{Z}$ and domain $\mathcal{X}$

$\mathcal{Z} \leftarrow$ all (N, T) values.

$\mathcal{X} \leftarrow$ all hyper-parameter values.

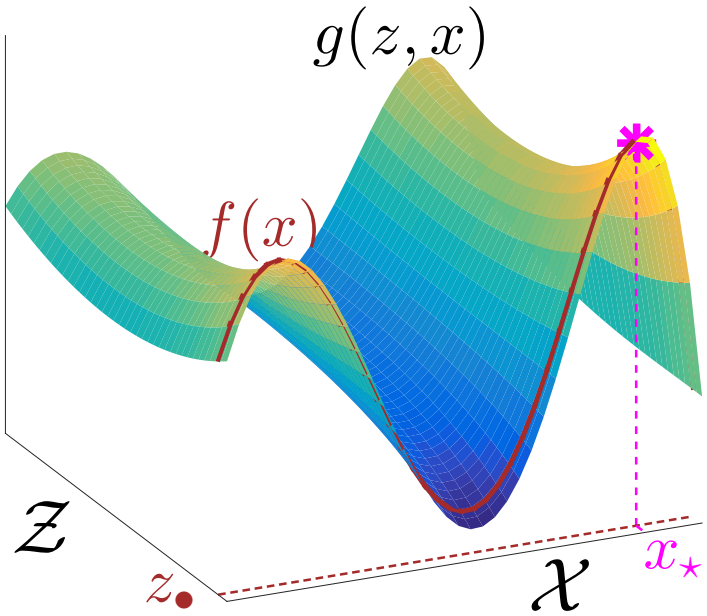
$g : \mathcal{Z} \times \mathcal{X} \rightarrow \mathbb{R}$.

$g([N, T], x) \leftarrow$ cv accuracy when training with N data for T iterations at hyper-parameter x.

Denote $f(x) = g(z_\bullet, x)$ where $z_\bullet \in \mathcal{Z}$. $z_\bullet = [N_\bullet, T_\bullet]$.

End Goal: Find $x_\star = \text{argmax}_x f(x)$.

Multi-fidelity Bandits (Kandasamy et al. ICML 2017)



A fidelity space $\mathcal{Z}$ and domain $\mathcal{X}$

$\mathcal{Z} \leftarrow$ all (N, T) values.

$\mathcal{X} \leftarrow$ all hyper-parameter values.

$g : \mathcal{Z} \times \mathcal{X} \to \mathbb{R}$.

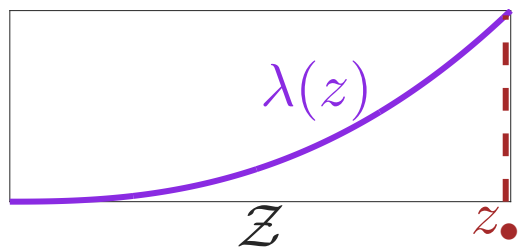
$g([N, T], x) \leftarrow$ cv accuracy when training with N data for T iterations at hyper-parameter x.

Denote $f(x) = g(z_\bullet, x)$ where $z_\bullet \in \mathcal{Z}$. $z_\bullet = [N_\bullet, T_\bullet]$.

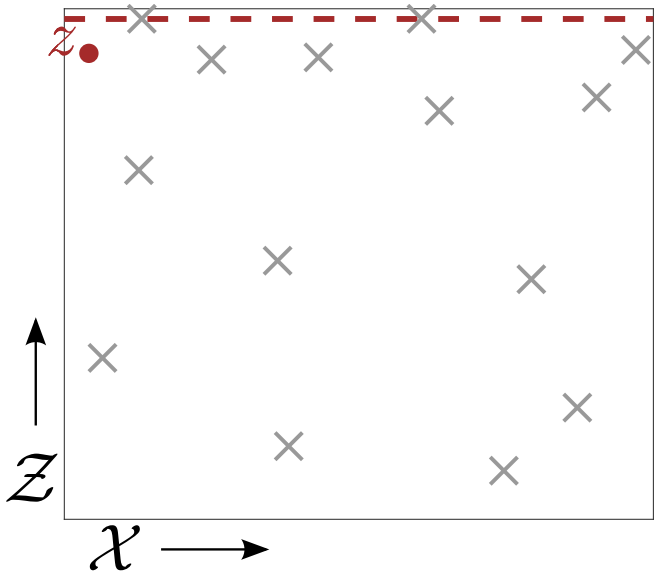
End Goal: Find $x_\star = \text{argmax}_x f(x)$.

A cost function, $\lambda : \mathcal{Z} \to \mathbb{R}_+$.

$\lambda(z) = \lambda(N, T) = \mathcal{O}(N^2 T)$ (say).

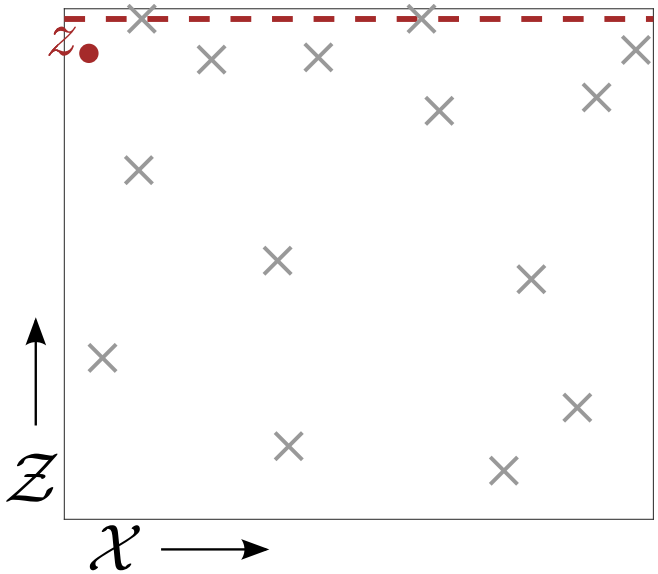


Algorithm: BOCA (Kandasamy et al. ICML 2017)



Algorithm: BOCA

(Kandasamy et al. ICML 2017)



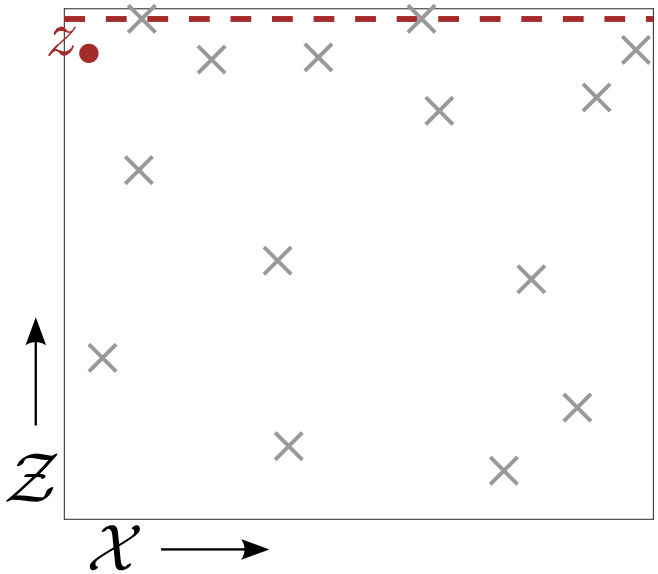
Model $g \sim \mathcal{GP}(0, \kappa)$ and compute posterior $\mathcal{GP}$:

mean $\quad \mu_{t-1} : \mathcal{Z} \times \mathcal{X} \to \mathbb{R}$

std-dev $\quad \sigma_{t-1} : \mathcal{Z} \times \mathcal{X} \to \mathbb{R}_+$

Algorithm: BOCA

(Kandasamy et al. ICML 2017)



Model $g \sim \mathcal{GP}(0, \kappa)$ and compute posterior $\mathcal{GP}$:

mean $\quad \mu_{t-1} : \mathcal{Z} \times \mathcal{X} \to \mathbb{R}$

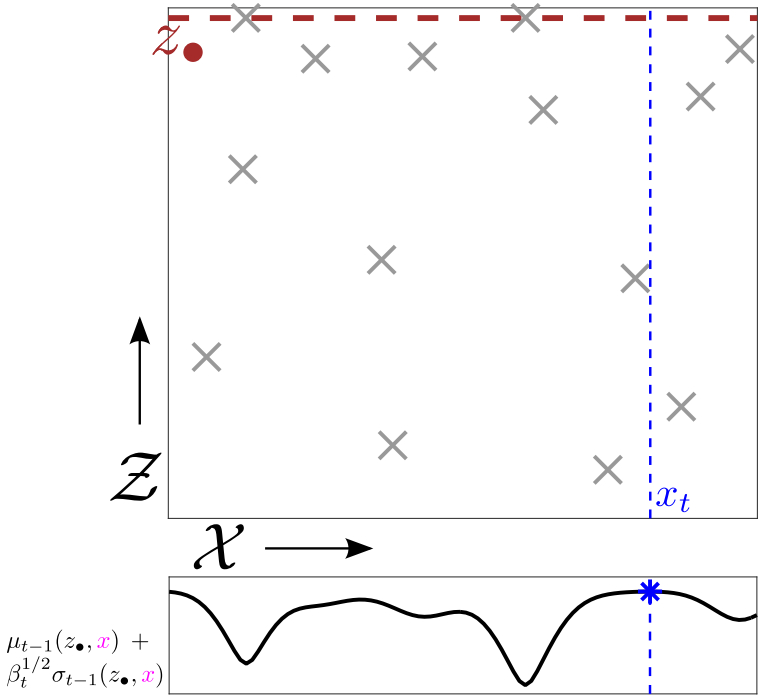
std-dev $\quad \sigma_{t-1} : \mathcal{Z} \times \mathcal{X} \to \mathbb{R}_+$

(1) $x_t \leftarrow$ maximise upper confidence bound for $f(x) = g(z_\bullet, x)$.

$$x_t = \operatorname*{argmax}_{x \in \mathcal{X}} \mu_{t-1}(z_\bullet, x) + \beta_t^{1/2} \sigma_{t-1}(z_\bullet, x)$$

Algorithm: BOCA

(Kandasamy et al. ICML 2017)



Model $g \sim \mathcal{GP}(0, \kappa)$ and compute posterior $\mathcal{GP}$:

mean $\quad \mu_{t-1} : \mathcal{Z} \times \mathcal{X} \to \mathbb{R}$

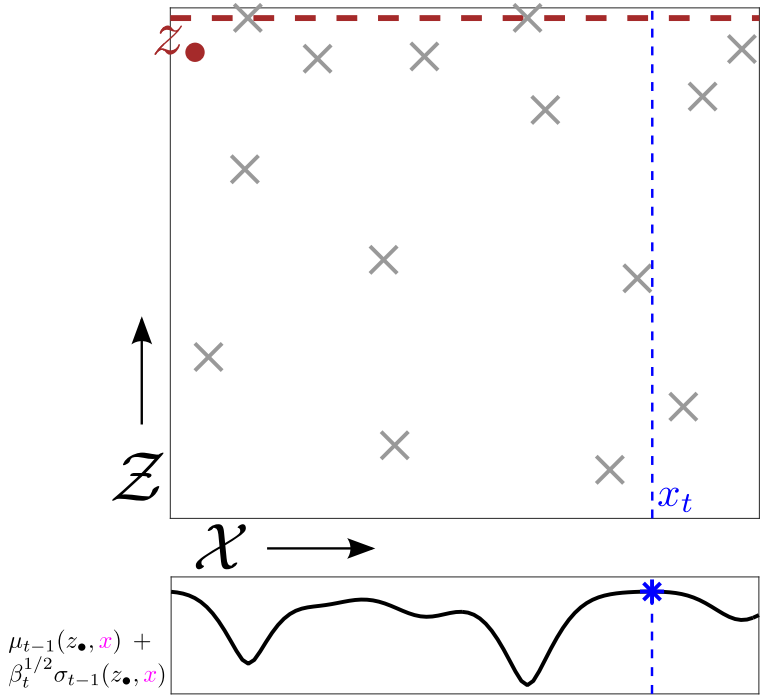
std-dev $\quad \sigma_{t-1} : \mathcal{Z} \times \mathcal{X} \to \mathbb{R}_+$

(1) $x_t \leftarrow$ maximise upper confidence bound for $f(x) = g(z_\bullet, x)$.

$$x_t = \underset{x \in \mathcal{X}}{\operatorname{argmax}} \ \mu_{t-1}(z_\bullet, x) + \beta_t^{1/2}\sigma_{t-1}(z_\bullet, x)$$

Algorithm: BOCA

(Kandasamy et al. ICML 2017)



Model $g \sim \mathcal{GP}(0, \kappa)$ and compute posterior $\mathcal{GP}$:

mean $\quad \mu_{t-1} : \mathcal{Z} \times \mathcal{X} \to \mathbb{R}$

std-dev $\quad \sigma_{t-1} : \mathcal{Z} \times \mathcal{X} \to \mathbb{R}_+$

(1) $x_t \leftarrow$ maximise upper confidence bound for $f(x) = g(z_\bullet, x)$.

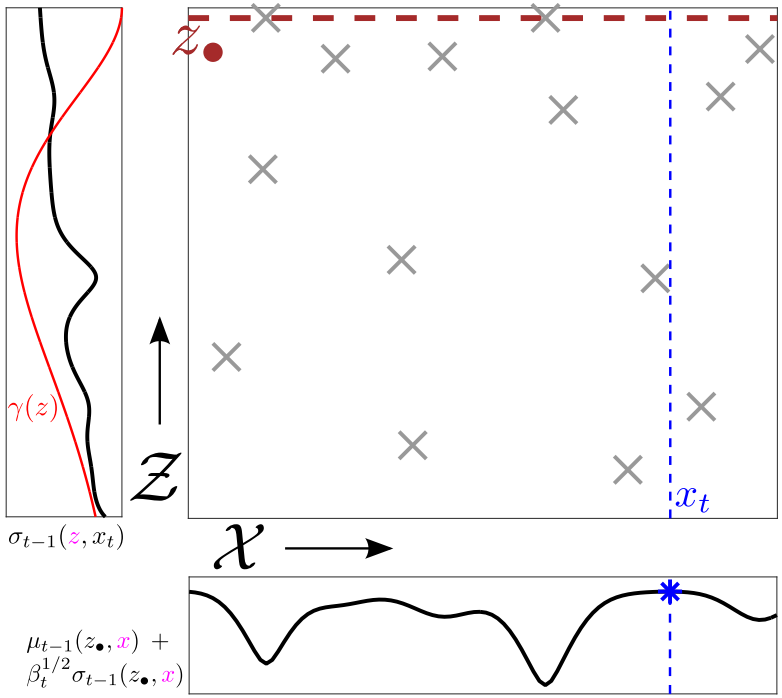
$$x_t = \underset{x \in \mathcal{X}}{\operatorname{argmax}} \ \mu_{t-1}(z_\bullet, x) + \beta_t^{1/2}\sigma_{t-1}(z_\bullet, x)$$

(2) $$\mathcal{Z}_t \approx \{z_\bullet\} \cup \left\{ z \ : \ \sigma_{t-1}(z, x_t) \geq \gamma(z) \right\}$$

(3) $$z_t = \underset{z \in \mathcal{Z}_t}{\operatorname{argmin}} \ \lambda(z) \qquad \text{(cheapest } z \text{ in } \mathcal{Z}_t\text{)}$$

Algorithm: BOCA

(Kandasamy et al. ICML 2017)



Model $g \sim \mathcal{GP}(0, \kappa)$ and compute posterior $\mathcal{GP}$:

mean $\quad \mu_{t-1} : \mathcal{Z} \times \mathcal{X} \to \mathbb{R}$

std-dev $\quad \sigma_{t-1} : \mathcal{Z} \times \mathcal{X} \to \mathbb{R}_+$

(1) $x_t \leftarrow$ maximise upper confidence bound for $f(x) = g(z_\bullet, x)$.

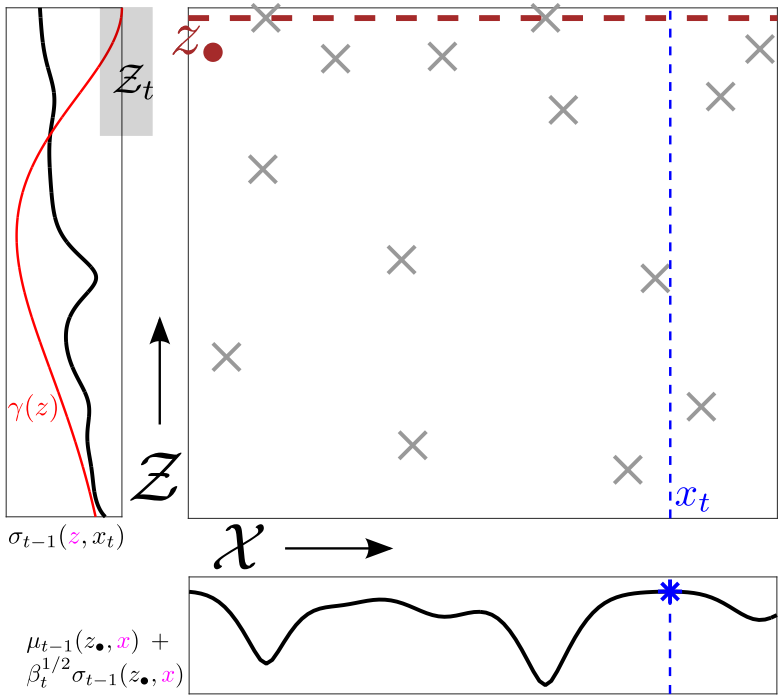
$$x_t = \operatorname*{argmax}_{x \in \mathcal{X}} \mu_{t-1}(z_\bullet, x) + \beta_t^{1/2}\sigma_{t-1}(z_\bullet, x)$$

(2) $$\mathcal{Z}_t \approx \{z_\bullet\} \cup \left\{ z : \sigma_{t-1}(z, x_t) \geq \gamma(z) \right\}$$

(3) $$z_t = \operatorname*{argmin}_{z \in \mathcal{Z}_t} \lambda(z) \qquad \text{(cheapest } z \text{ in } \mathcal{Z}_t\text{)}$$

Algorithm: BOCA

(Kandasamy et al. ICML 2017)



Model $g \sim \mathcal{GP}(0, \kappa)$ and compute posterior $\mathcal{GP}$:

mean $\quad \mu_{t-1} : \mathcal{Z} \times \mathcal{X} \to \mathbb{R}$

std-dev $\quad \sigma_{t-1} : \mathcal{Z} \times \mathcal{X} \to \mathbb{R}_+$

(1) $x_t \leftarrow$ maximise upper confidence bound for $f(x) = g(z_\bullet, x)$.

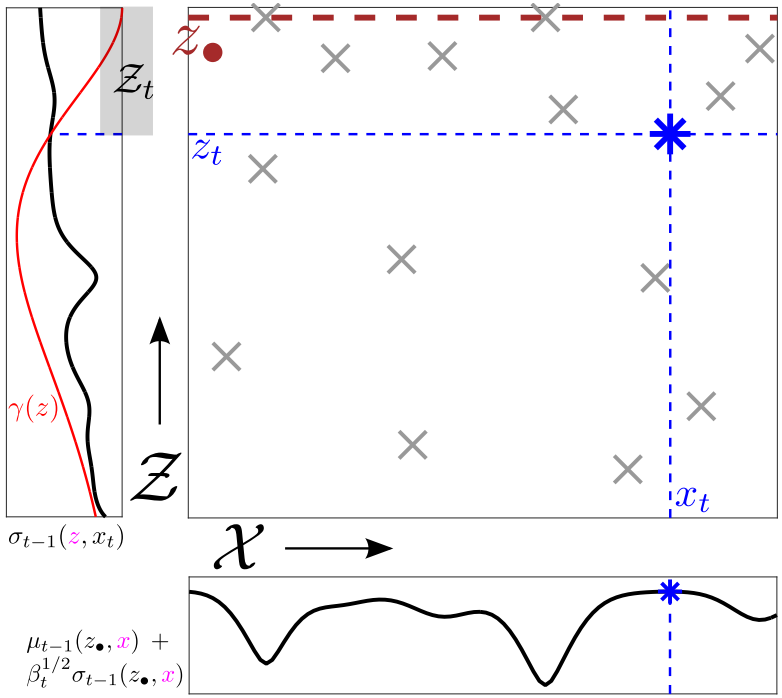
$$x_t = \underset{x \in \mathcal{X}}{\operatorname{argmax}} \ \mu_{t-1}(z_\bullet, x) + \beta_t^{1/2}\sigma_{t-1}(z_\bullet, x)$$

(2) $$\mathcal{Z}_t \approx \{z_\bullet\} \cup \left\{ z \ : \ \sigma_{t-1}(z, x_t) \geq \gamma(z) \right\}$$

(3) $$z_t = \underset{z \in \mathcal{Z}_t}{\operatorname{argmin}} \ \lambda(z) \qquad \text{(cheapest } z \text{ in } \mathcal{Z}_t\text{)}$$

Algorithm: BOCA

(Kandasamy et al. ICML 2017)



Model $g \sim \mathcal{GP}(0, \kappa)$ and compute posterior $\mathcal{GP}$:

mean $\quad \mu_{t-1} : \mathcal{Z} \times \mathcal{X} \to \mathbb{R}$

std-dev $\quad \sigma_{t-1} : \mathcal{Z} \times \mathcal{X} \to \mathbb{R}_+$

(1) $x_t \leftarrow$ maximise upper confidence bound for $f(x) = g(z_\bullet, x)$.

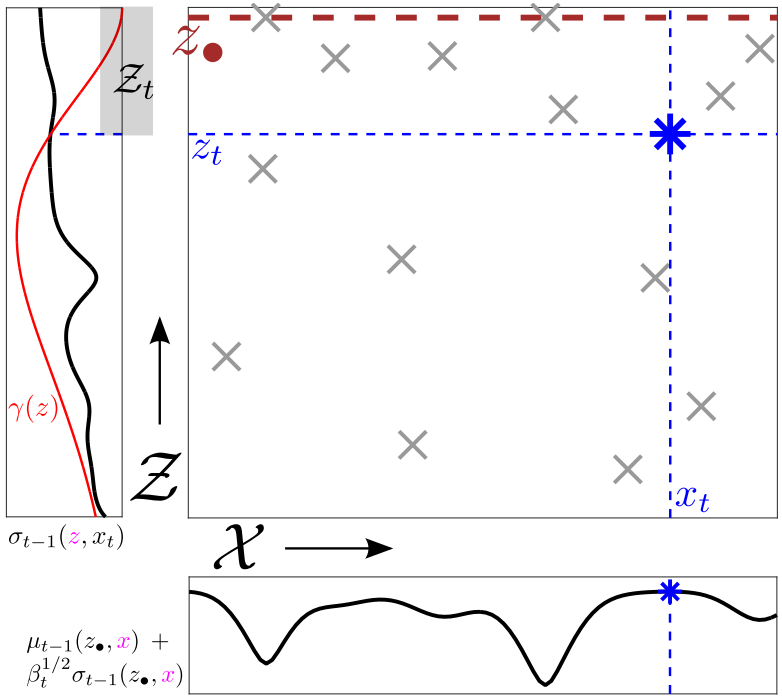
$$x_t = \underset{x \in \mathcal{X}}{\operatorname{argmax}} \ \mu_{t-1}(z_\bullet, x) + \beta_t^{1/2}\sigma_{t-1}(z_\bullet, x)$$

(2) $$\mathcal{Z}_t \approx \{z_\bullet\} \cup \left\{ z \ : \ \sigma_{t-1}(z, x_t) \geq \gamma(z) \right\}$$

(3) $$z_t = \underset{z \in \mathcal{Z}_t}{\operatorname{argmin}} \ \lambda(z) \qquad \text{(cheapest } z \text{ in } \mathcal{Z}_t\text{)}$$

Algorithm: BOCA

(Kandasamy et al. ICML 2017)



Model $g \sim \mathcal{GP}(0, \kappa)$ and compute posterior $\mathcal{GP}$:

mean $\quad \mu_{t-1} : \mathcal{Z} \times \mathcal{X} \to \mathbb{R}$

std-dev $\quad \sigma_{t-1} : \mathcal{Z} \times \mathcal{X} \to \mathbb{R}_+$

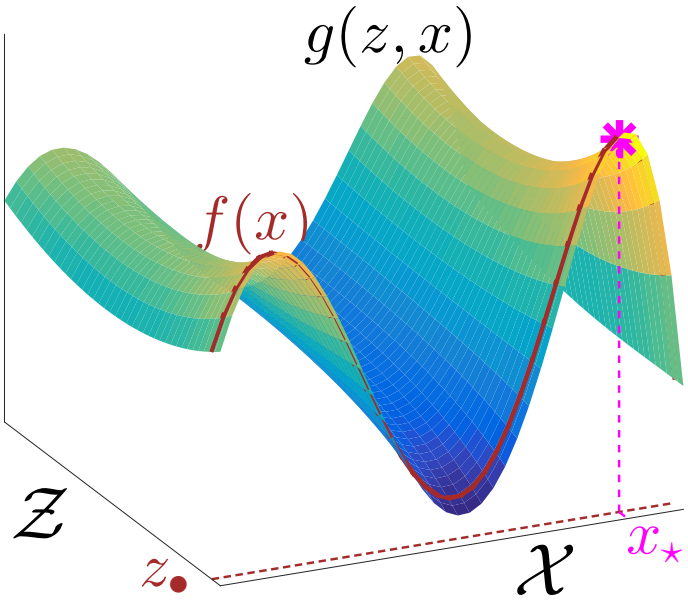
(1) $x_t \leftarrow$ maximise upper confidence bound for $f(x) = g(z_\bullet, x)$.

$$x_t = \operatorname*{argmax}_{x \in \mathcal{X}} \ \mu_{t-1}(z_\bullet, x) + \beta_t^{1/2}\sigma_{t-1}(z_\bullet, x)$$

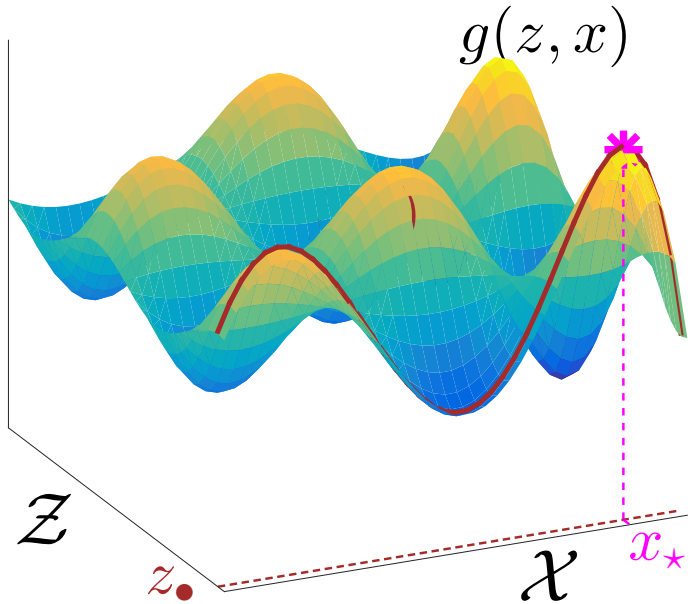
(2) $$\mathcal{Z}_t \approx \{z_\bullet\} \cup \left\{ z \,:\, \sigma_{t-1}(z, x_t) \geq \gamma(z) = \left(\frac{\lambda(z)}{\lambda(z_\bullet)}\right)^q \xi(z) \right\}$$

(3) $$z_t = \operatorname*{argmin}_{z \in \mathcal{Z}_t} \lambda(z) \qquad \text{(cheapest } z \text{ in } \mathcal{Z}_t\text{)}$$

Theoretical Results for BOCA

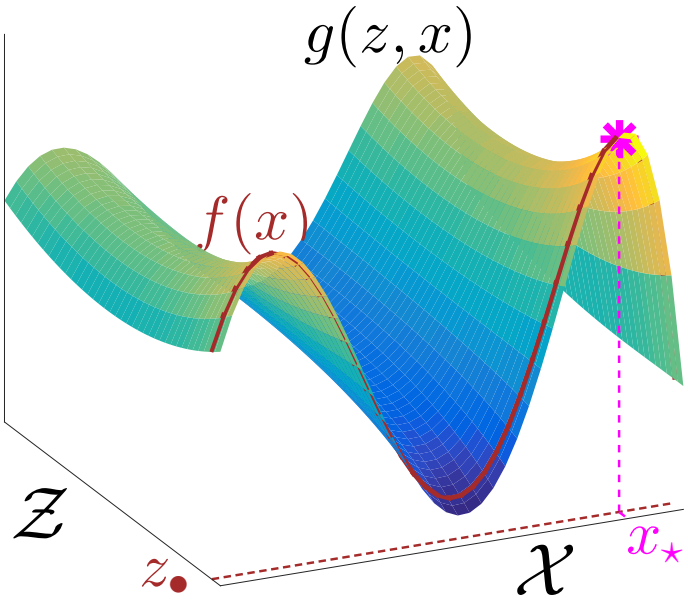


"good"

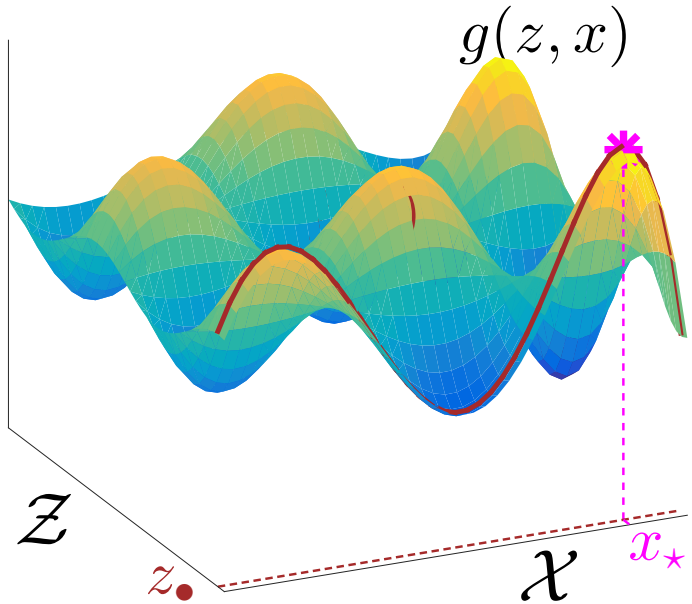


"bad"

Theoretical Results for BOCA



"good"

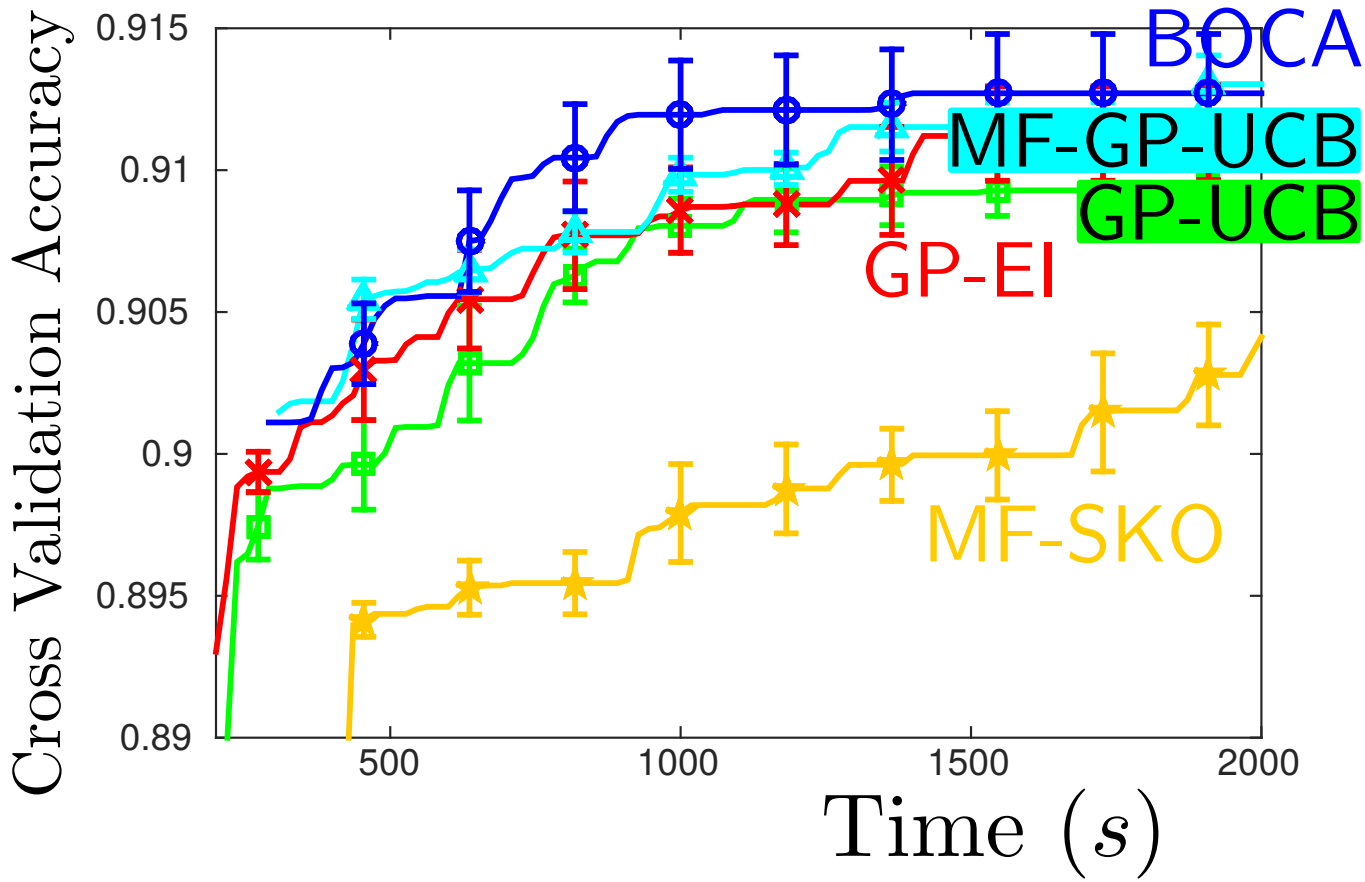


"bad"

Theorem: (Informal)
BOCA does better, i.e. achieves better Simple regret, than GP-UCB. The improvements are better in the "good" setting when compared to the "bad" setting.

Experiment: SVM with 20 News Groups

Tune two hyper-parameters for the SVM.
Dataset has $N_\bullet = 15K$ data and use $T_\bullet = 100$ iterations.
But can choose $N \in [5K, 15K]$ or $T \in [20, 100]$ (2D fidelity space).

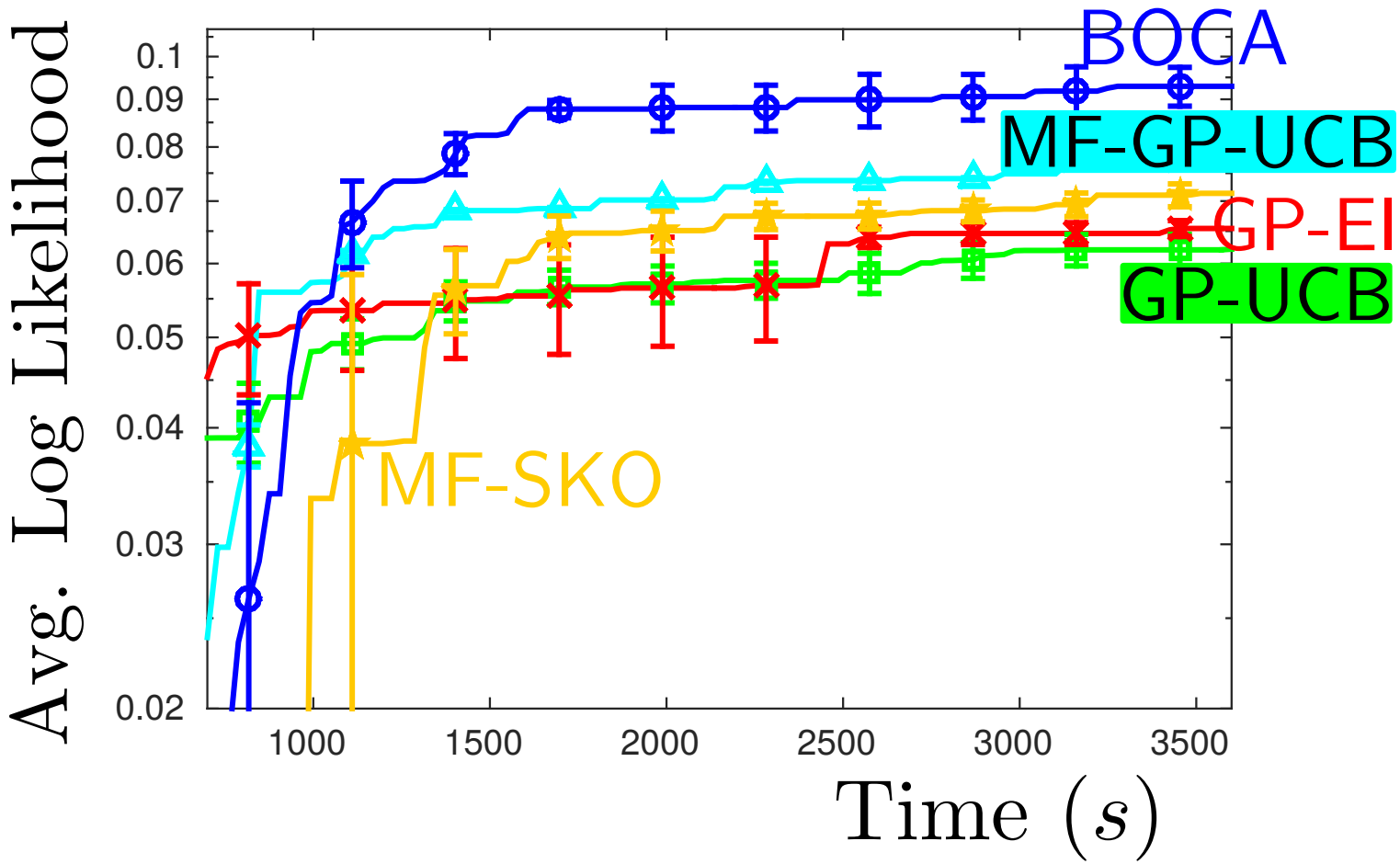


Experiment: Cosmological inference on Type-1a supernovae data

Estimate Hubble constant, dark matter fraction & dark energy fraction by maximising likelihood on $N_\bullet = 192$ data.
Requires numerical integration on a grid of size $G_\bullet = 10^6$.
Approximate with $N \in [50, 192]$ or $G \in [10^2, 10^6]$ (2D fidelity space).

Experiment: Cosmological inference on Type-1a supernovae data

Estimate Hubble constant, dark matter fraction & dark energy fraction by maximising likelihood on $N_\bullet = 192$ data.
Requires numerical integration on a grid of size $G_\bullet = 10^6$.
Approximate with $N \in [50, 192]$ or $G \in [10^2, 10^6]$ (2D fidelity space).



Hyper-band: A multi-fidelity method with incremental resource allocation (Li et al. 2016)

E.g: Training a neural network with gradient descent for several iterations.

Hyper-band: A multi-fidelity method with incremental resource allocation (Li et al. 2016)

E.g: Training a neural network with gradient descent for several iterations. If the CV error is bad after early iterations, then it will likely be bad at the end.

Hyper-band: A multi-fidelity method with incremental resource allocation (Li et al. 2016)

E.g: Training a neural network with gradient descent for several iterations. If the CV error is bad after early iterations, then it will likely be bad at the end.

Successive Halving (with finite $\mathcal{X}$):

1. Allocate a small resource R to each $x \in \mathcal{X}$.
   e.g. Train all hyper-parameters for 100 iterations.
2. Drop half of the x's that are performing worst.
3. Repeat steps 1 & 2 until one arm is left.

Hyper-band: A multi-fidelity method with incremental resource allocation (Li et al. 2016)

E.g: Training a neural network with gradient descent for several iterations. If the CV error is bad after early iterations, then it will likely be bad at the end.

Successive Halving (with finite $\mathcal{X}$):

1. Allocate a small resource R to each $x \in \mathcal{X}$.
   e.g. Train all hyper-parameters for 100 iterations.
2. Drop half of the x's that are performing worst.
3. Repeat steps 1 & 2 until one arm is left.

Can be extended to infinite $\mathcal{X}$.

Hyper-band: A multi-fidelity method with incremental resource allocation (Li et al. 2016)

E.g: Training a neural network with gradient descent for several iterations. If the CV error is bad after early iterations, then it will likely be bad at the end.

Successive Halving (with finite $\mathcal{X}$):

1. Allocate a small resource R to each $x \in \mathcal{X}$.
   e.g. Train all hyper-parameters for 100 iterations.
2. Drop half of the x's that are performing worst.
3. Repeat steps 1 & 2 until one arm is left.

Can be extended to infinite $\mathcal{X}$.

Does not fall within the GP/Bayesian framework.

Hyper-band (cont'd)

When compared to Bayesian methods,

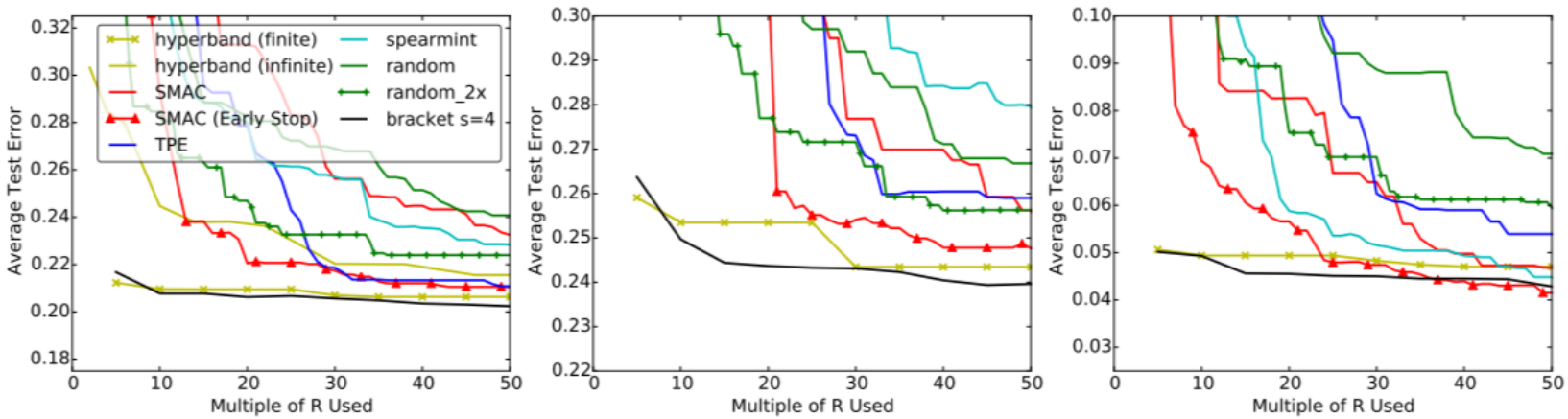
- **Pro:** Incremental resource allocation (do not need to retrain all models from the beginning).
- **Con:** Cannot use correlation between arms (e.g. if x_1 has large CV accuracy, then x_2 close to x_1 is also likely to do well).

Hyper-band (cont'd)

When compared to Bayesian methods,

- **Pro:** Incremental resource allocation (do not need to retrain all models from the beginning).
- **Con:** Cannot use correlation between arms (e.g. if x_1 has large CV accuracy, then x_2 close to x_1 is also likely to do well).

Experiments:



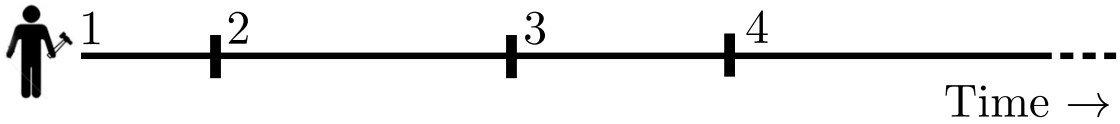
(a) CIFAR-10 (b) MRBI (c) SVHN

Outline

- Part I: Bandits in the Bayesian Paradigm
  1. Gaussian processes
  2. Algorithms: Upper Confidence Bound (UCB) & Thompson Sampling (TS)
- Part II: Scaling up Bandits
  1. Multi-fidelity bandit: cheap approximations to an expensive experiment
  2. Parallelising function evaluations
  3. High dimensional input spaces

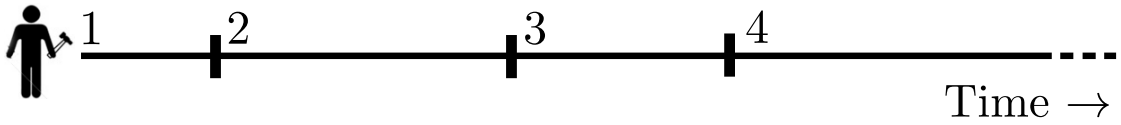
Part 2.2: Parallelising arm pulls

Sequential evaluations with one worker

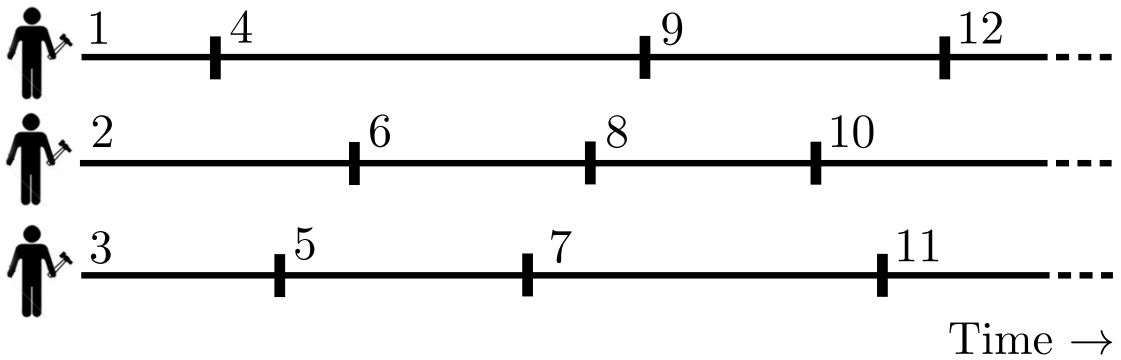


Part 2.2: Parallelising arm pulls

Sequential evaluations with one worker

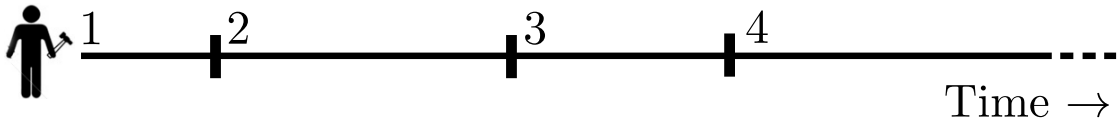


Parallel evaluations with M workers (Asynchronous)

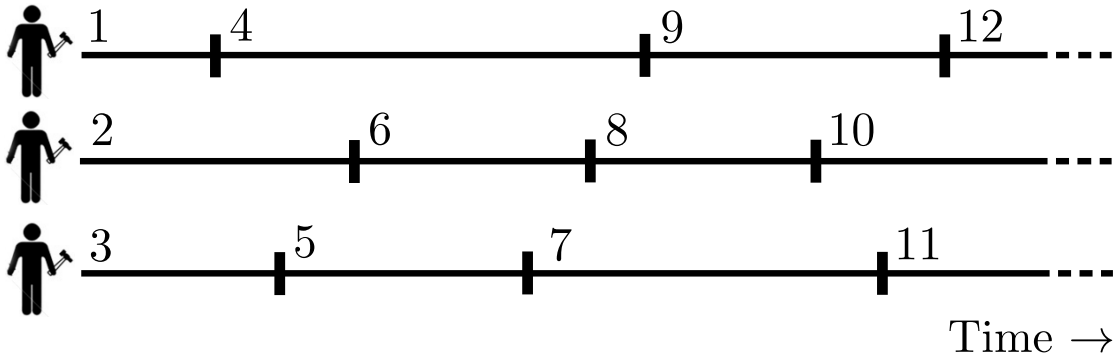


Part 2.2: Parallelising arm pulls

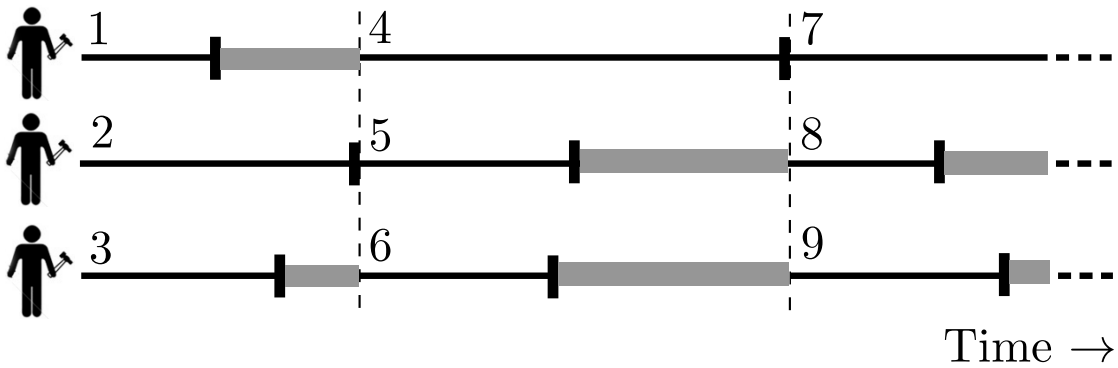
Sequential evaluations with one worker



Parallel evaluations with M workers (Asynchronous)



Parallel evaluations with M workers (Synchronous)



Why parallelisation?

- **Computational experiments:** infrastructure with 100-1000's CPUs or GPUs.

Why parallelisation?

- **Computational experiments:** infrastructure with 100-1000's CPUs or GPUs.

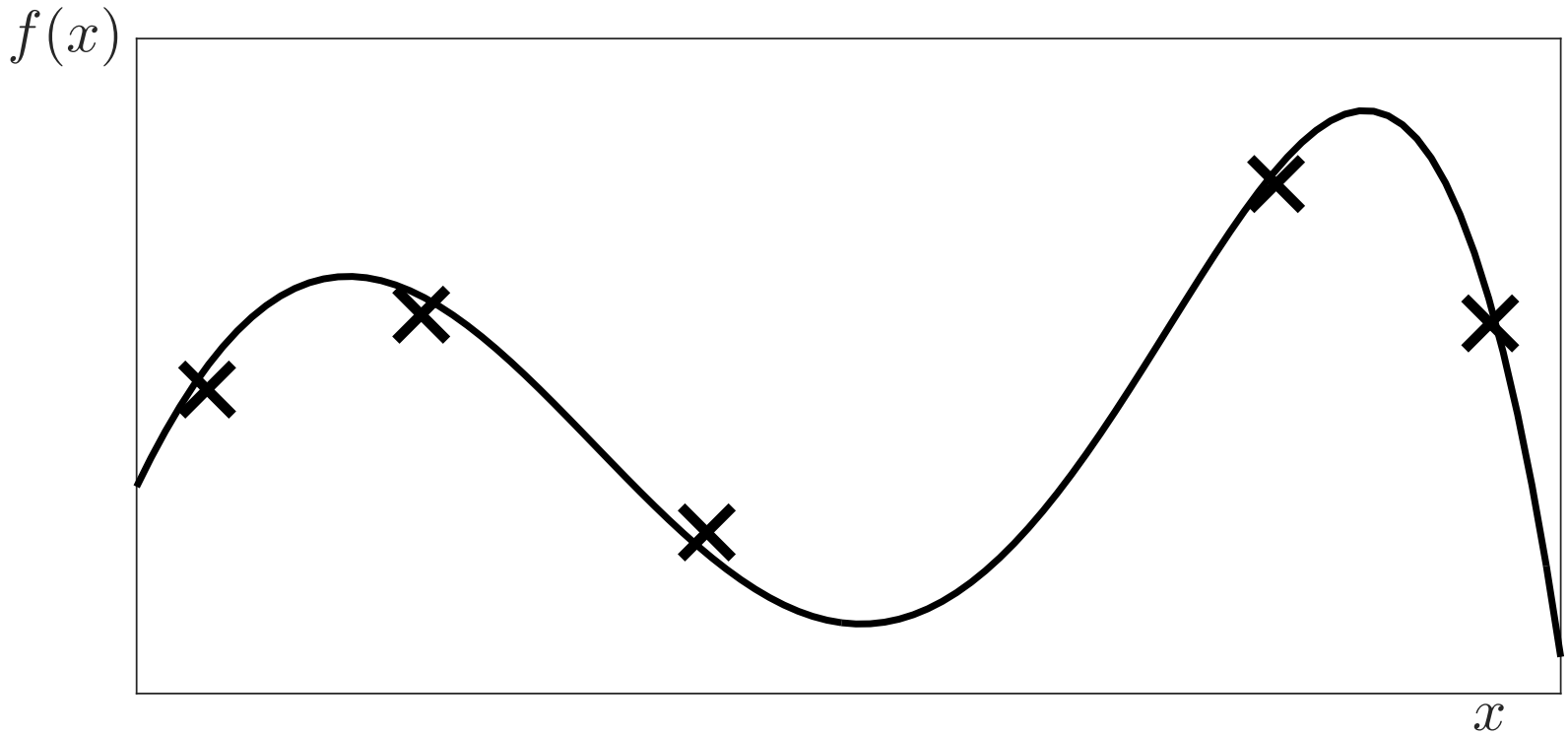
Prior work: (Ginsbourger et al. 2011, Janusevskis et al. 2012, Wang et al. 2016, González et al. 2015, Desautels et al. 2014, Contal et al. 2013, Shah and Ghahramani 2015, Kathuria et al. 2016, Wang et al. 2017, Wu and Frazier 2016, Hernandez-Lobato et al. 2017)

Shortcomings

- **Asynchronicity**
- **Theoretical guarantees**
- **Computationally & conceptually simple**

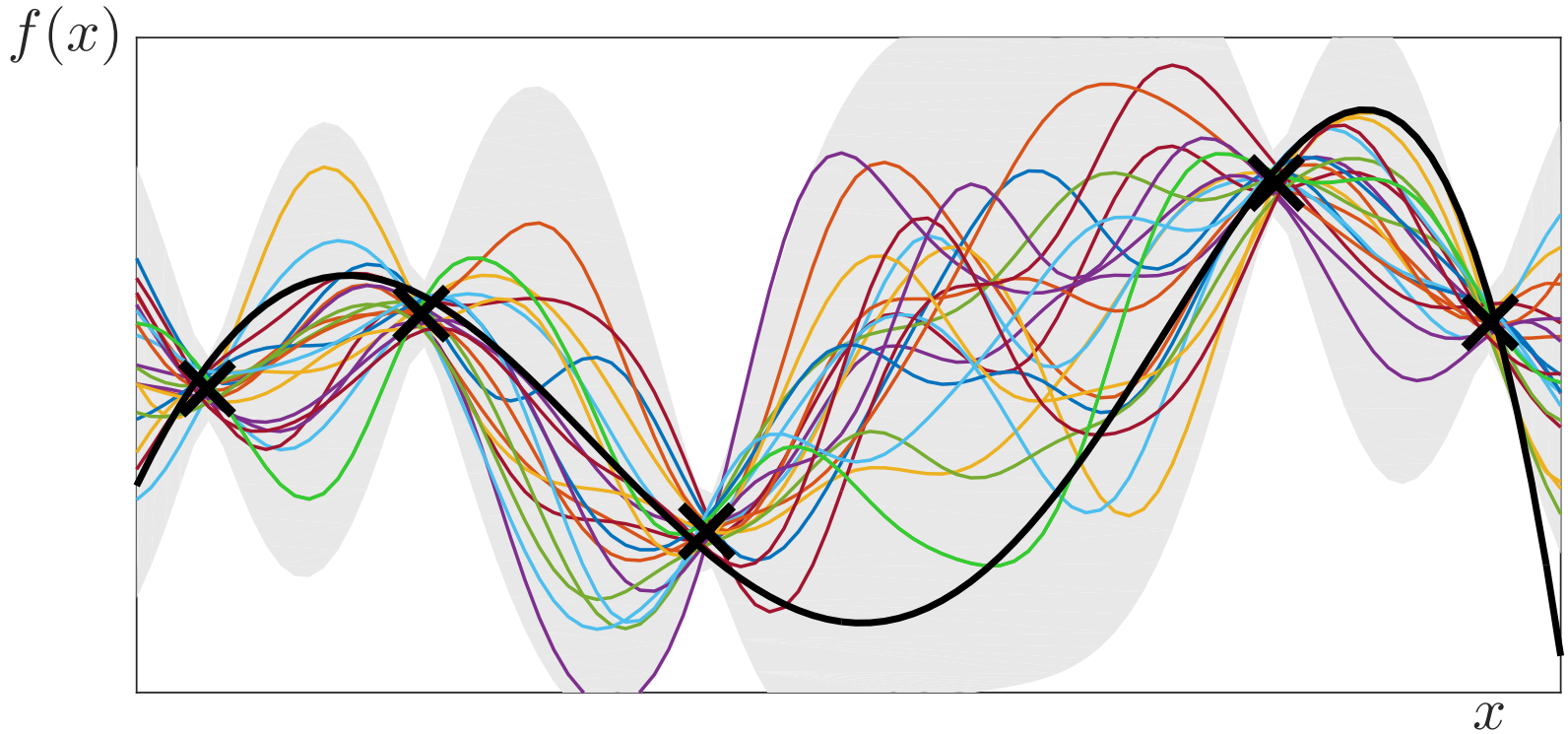
Review: Sequential Thompson Sampling in GP Bandits

Thompson Sampling (TS) (Thompson, 1933).



Review: Sequential Thompson Sampling in GP Bandits

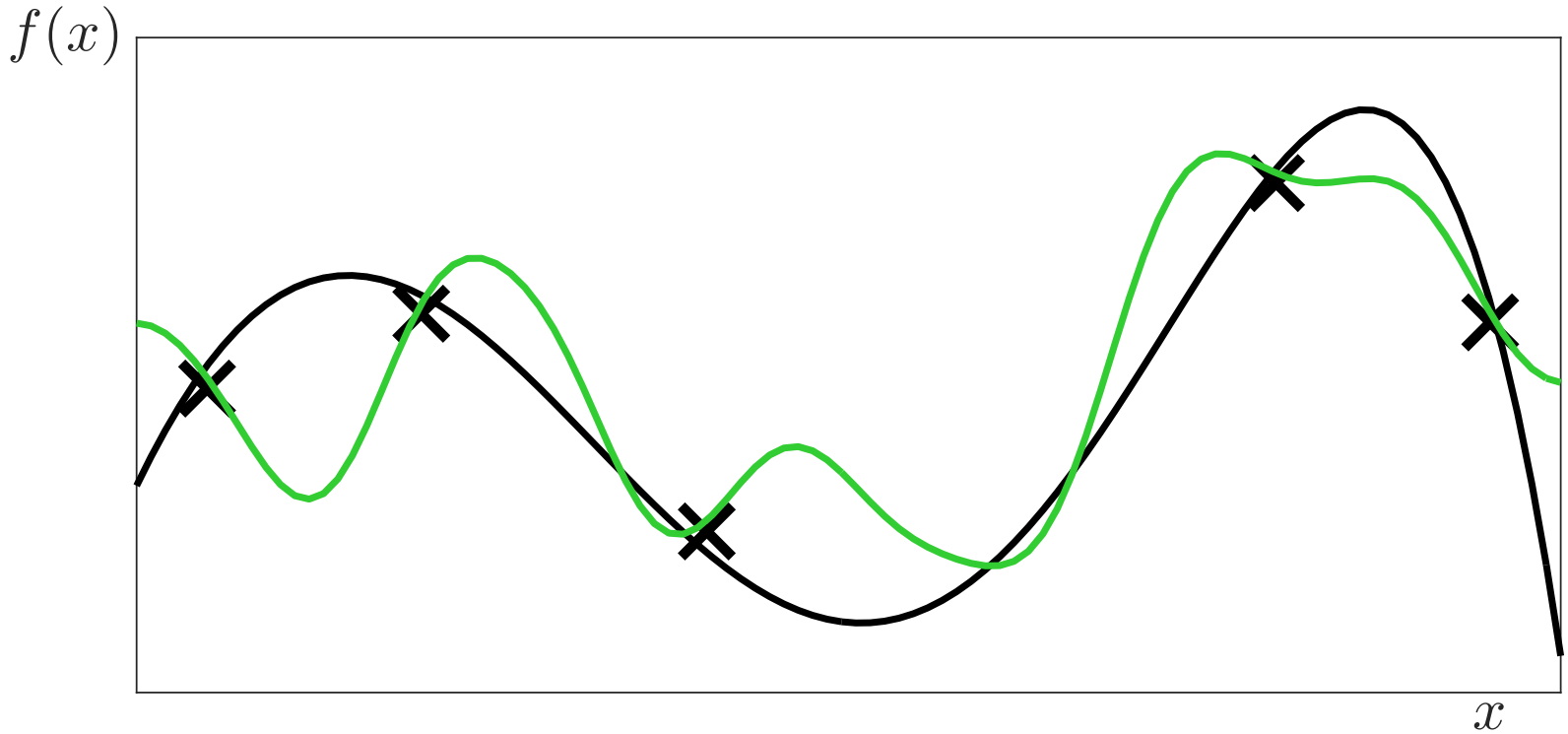
Thompson Sampling (TS) (Thompson, 1933).



1) Construct posterior $\mathcal{GP}$.

Review: Sequential Thompson Sampling in GP Bandits

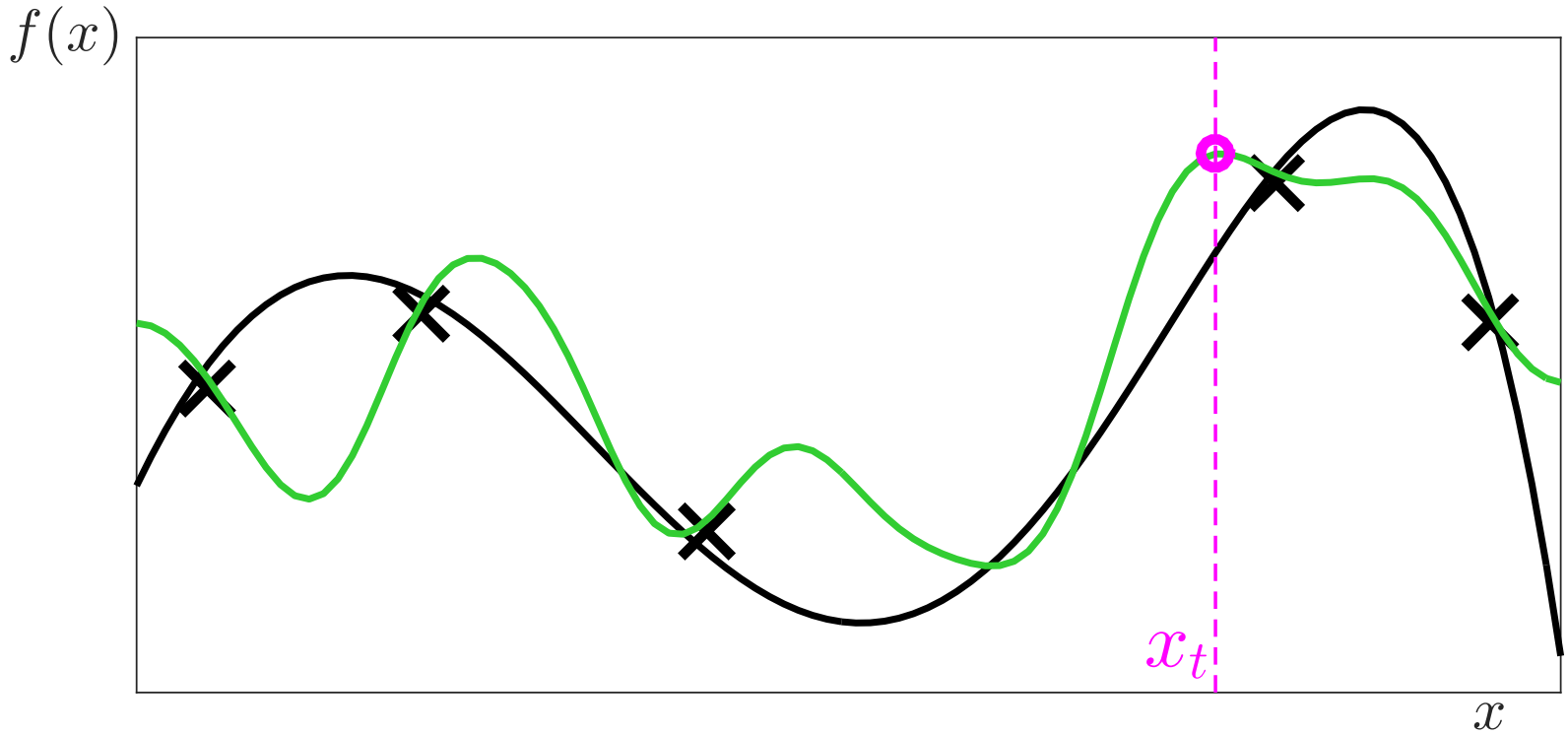
Thompson Sampling (TS) (Thompson, 1933).



1) Construct posterior $\mathcal{GP}$. 2) Draw sample g from posterior.

Review: Sequential Thompson Sampling in GP Bandits

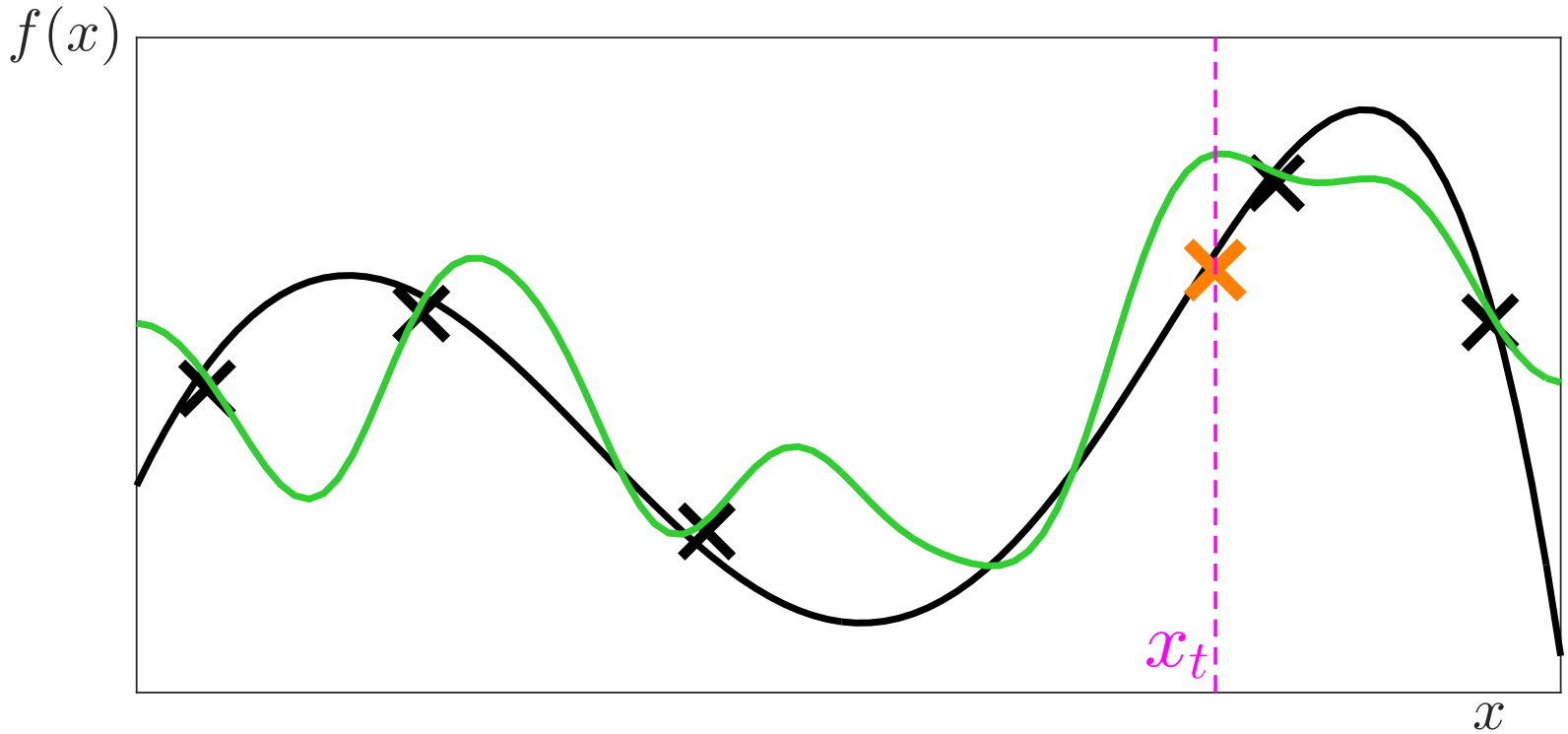
Thompson Sampling (TS) (Thompson, 1933).



1) Construct posterior $\mathcal{GP}$. 2) Draw sample g from posterior.
3) Choose $x_t = \mathrm{argmax}_x\, g(x)$.

Review: Sequential Thompson Sampling in GP Bandits

Thompson Sampling (TS) (Thompson, 1933).



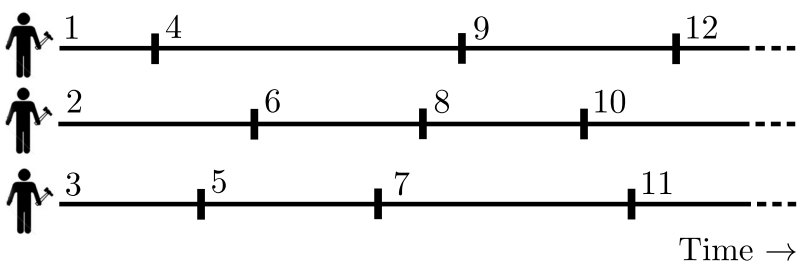
1) Construct posterior $\mathcal{GP}$. 2) Draw sample g from posterior.
3) Choose $x_t = \mathrm{argmax}_x\, g(x)$. 4) Evaluate f at x_t.

Parallelised Thompson Sampling (Kandasamy et al. Arxiv 2017)

Asynchronous: asyTS

At any given time,

1. $(x', y') \leftarrow$ Wait for a worker to finish.
2. Compute posterior $\mathcal{GP}$.
3. Draw a sample $g \sim \mathcal{GP}$.
4. Re-deploy worker at $\operatorname{argmax} g$.

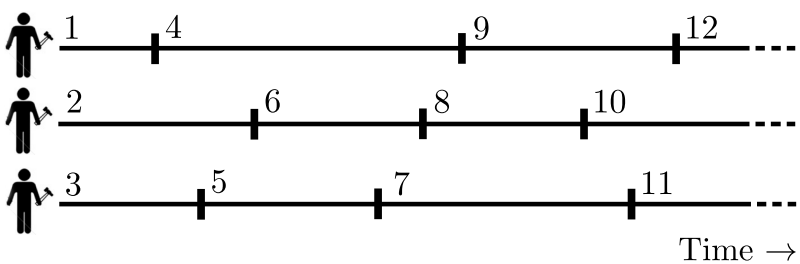


Parallelised Thompson Sampling (Kandasamy et al. Arxiv 2017)

Asynchronous: asyTS

At any given time,

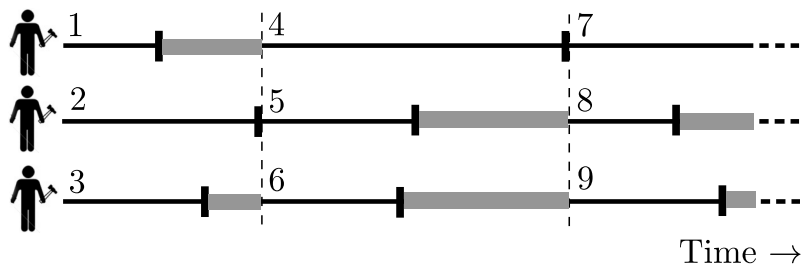
1. $(x', y') \leftarrow$ Wait for a worker to finish.
2. Compute posterior $\mathcal{GP}$.
3. Draw a sample $g \sim \mathcal{GP}$.
4. Re-deploy worker at $\operatorname{argmax} g$.



Synchronous: synTS

At any given time,

1. $\{(x'_m, y'_m)\}_{m=1}^{M} \leftarrow$ Wait for all workers to finish.
2. Compute posterior $\mathcal{GP}$.
3. Draw M samples $g_m \sim \mathcal{GP}$, $\forall m$.
4. Re-deploy worker m at $\operatorname{argmax} g_m$, $\forall m$.



Parallelised Thompson Sampling (Kandasamy et al. Arxiv 2017)

Asynchronous: asyTS

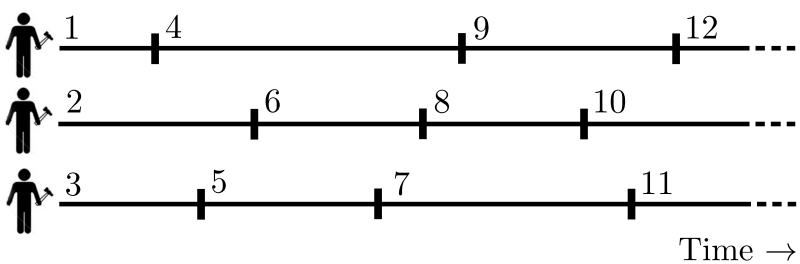
At any given time,

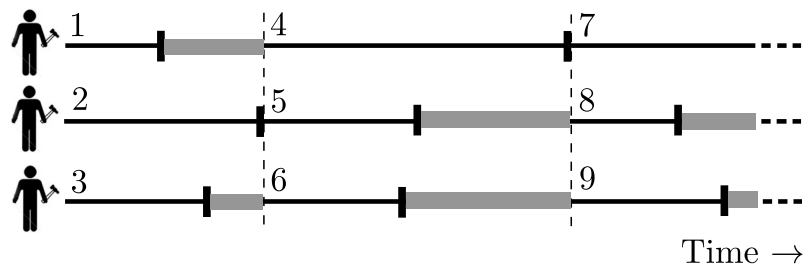
1. $(x', y') \leftarrow$ Wait for a worker to finish.
2. Compute posterior $\mathcal{GP}$.
3. Draw a sample $g \sim \mathcal{GP}$.
4. Re-deploy worker at $\operatorname{argmax} g$.

Synchronous: synTS

At any given time,

1. $\{(x'_m, y'_m)\}_{m=1}^{M} \leftarrow$ Wait for all workers to finish.
2. Compute posterior $\mathcal{GP}$.
3. Draw M samples $g_m \sim \mathcal{GP}$, $\forall m$.
4. Re-deploy worker m at $\operatorname{argmax} g_m$, $\forall m$.





Variants in prior work: (Osband et al. 2016, Israelsen et al. 2016, Hernandez-Lobato et al. 2017)

Theoretical Results for TS: number of evaluations

Sequential TS, SE Kernel (Russo & van Roy 2014)

$$\mathbb{E}[S_n] \lesssim \sqrt{\frac{\mathrm{vol}(\mathcal{X})\log(n)}{n}}$$

Theoretical Results for TS: number of evaluations

Sequential TS, SE Kernel (Russo & van Roy 2014)

$$\mathbb{E}[S_n] \lesssim \sqrt{\frac{\mathrm{vol}(\mathcal{X})\log(n)}{n}}$$

Theorem: synTS & asyTS, SE Kernel (Kandasamy et al. Arxiv 2017)

$$\mathbb{E}[S_n] \lesssim \frac{M\sqrt{\log(M)}}{n} + \sqrt{\frac{\mathrm{vol}(\mathcal{X})\log(n+M)}{n}}$$

$n \leftarrow \#$ completed arm pulls by all workers.

Theoretical Results for TS: number of evaluations

Sequential TS, SE Kernel (Russo & van Roy 2014)

$$\mathbb{E}[S_n] \lesssim \sqrt{\frac{\mathrm{vol}(\mathcal{X})\log(n)}{n}}$$

Theorem: synTS & asyTS, SE Kernel (Kandasamy et al. Arxiv 2017)

$$\mathbb{E}[S_n] \lesssim \frac{M\sqrt{\log(M)}}{n} + \sqrt{\frac{\mathrm{vol}(\mathcal{X})\log(n+M)}{n}}$$

$n \leftarrow \#$ completed arm pulls by all workers.

Why is this interesting?

- A sequential algorithm can make use of information from all previous rounds to determine where to evaluate next.
- A parallel algorithm could be missing up to $M-1$ results at any given time.

Theoretical Results for TS: number of evaluations

Sequential TS, SE Kernel (Russo & van Roy 2014)

$$\mathbb{E}[S_n] \lesssim \sqrt{\frac{\mathrm{vol}(\mathcal{X})\log(n)}{n}}$$

Theorem: synTS & asyTS, SE Kernel (Kandasamy et al. Arxiv 2017)

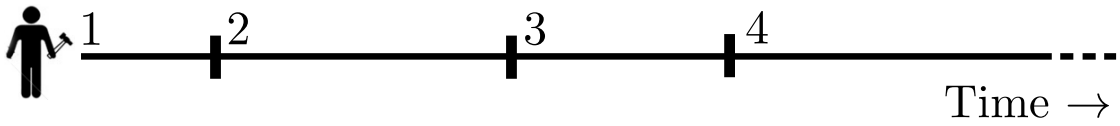
$$\mathbb{E}[S_n] \lesssim \frac{M\sqrt{\log(M)}}{n} + \sqrt{\frac{\mathrm{vol}(\mathcal{X})\log(n+M)}{n}}$$

$n \leftarrow \#$ completed arm pulls by all workers.

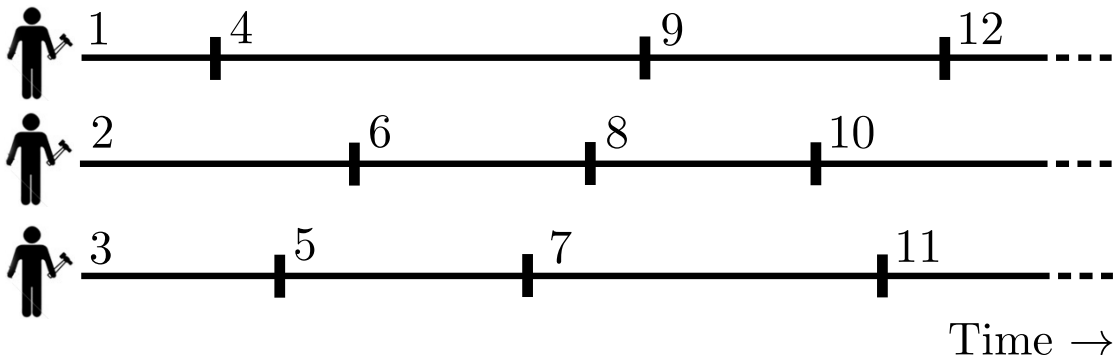
Why is this interesting?

- A sequential algorithm can make use of information from all previous rounds to determine where to evaluate next.
- A parallel algorithm could be missing up to $M-1$ results at any given time. But randomisation helps!

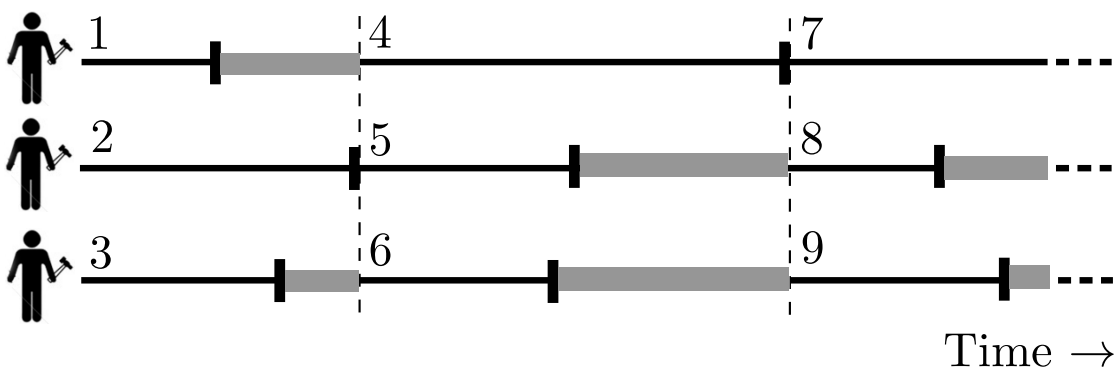
Sequential evaluations with one worker



Parallel evaluations with M workers (Asynchronous)

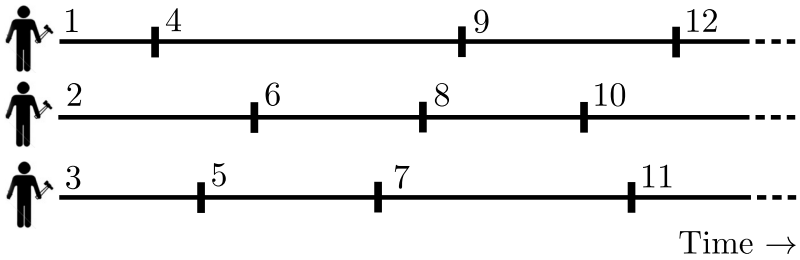


Parallel evaluations with M workers (Synchronous)

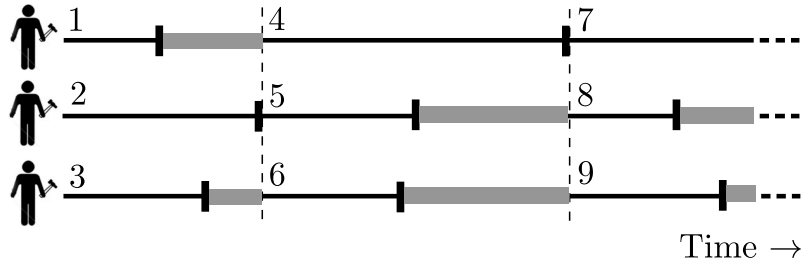


Theoretical Results: Simple regret with time

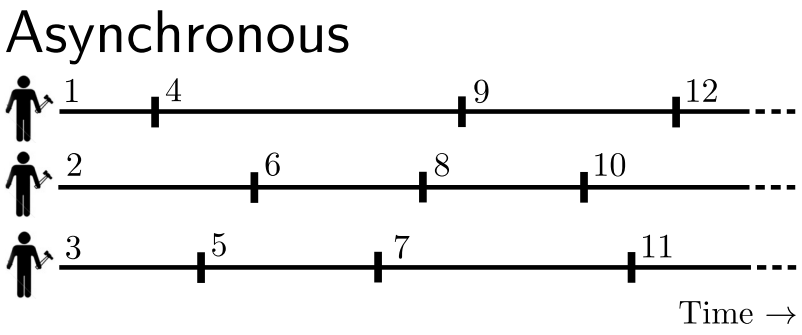
Asynchronous

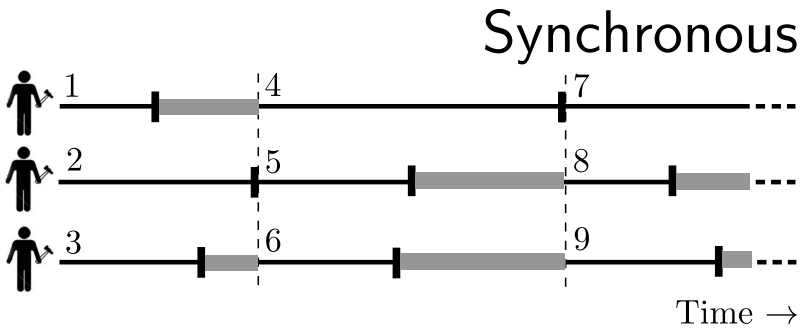


Synchronous



Theoretical Results: Simple regret with time



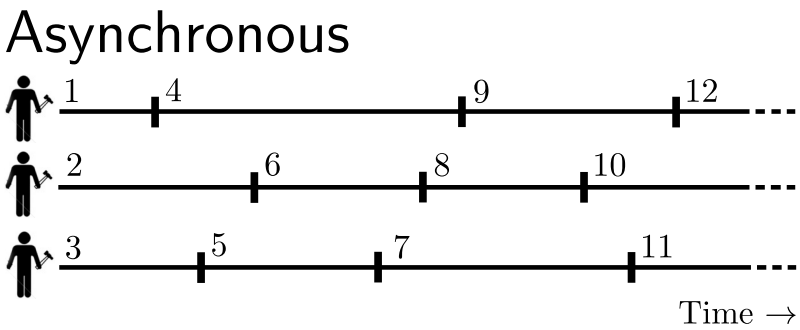


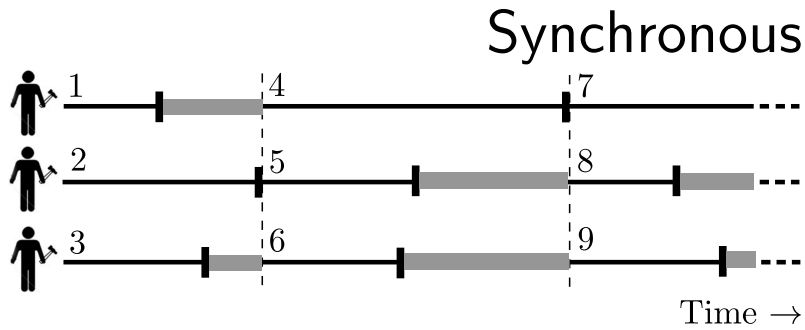
Theorem (Informal) (Kandasamy et al. Arxiv 2017)

If evaluation times are the same, asyTS $\approx$ synTS.

Otherwise, bounds for asyTS is better than synTS. More the variability in evaluation times, the bigger the difference.

Theoretical Results: Simple regret with time





Theorem (Informal) (Kandasamy et al. Arxiv 2017)

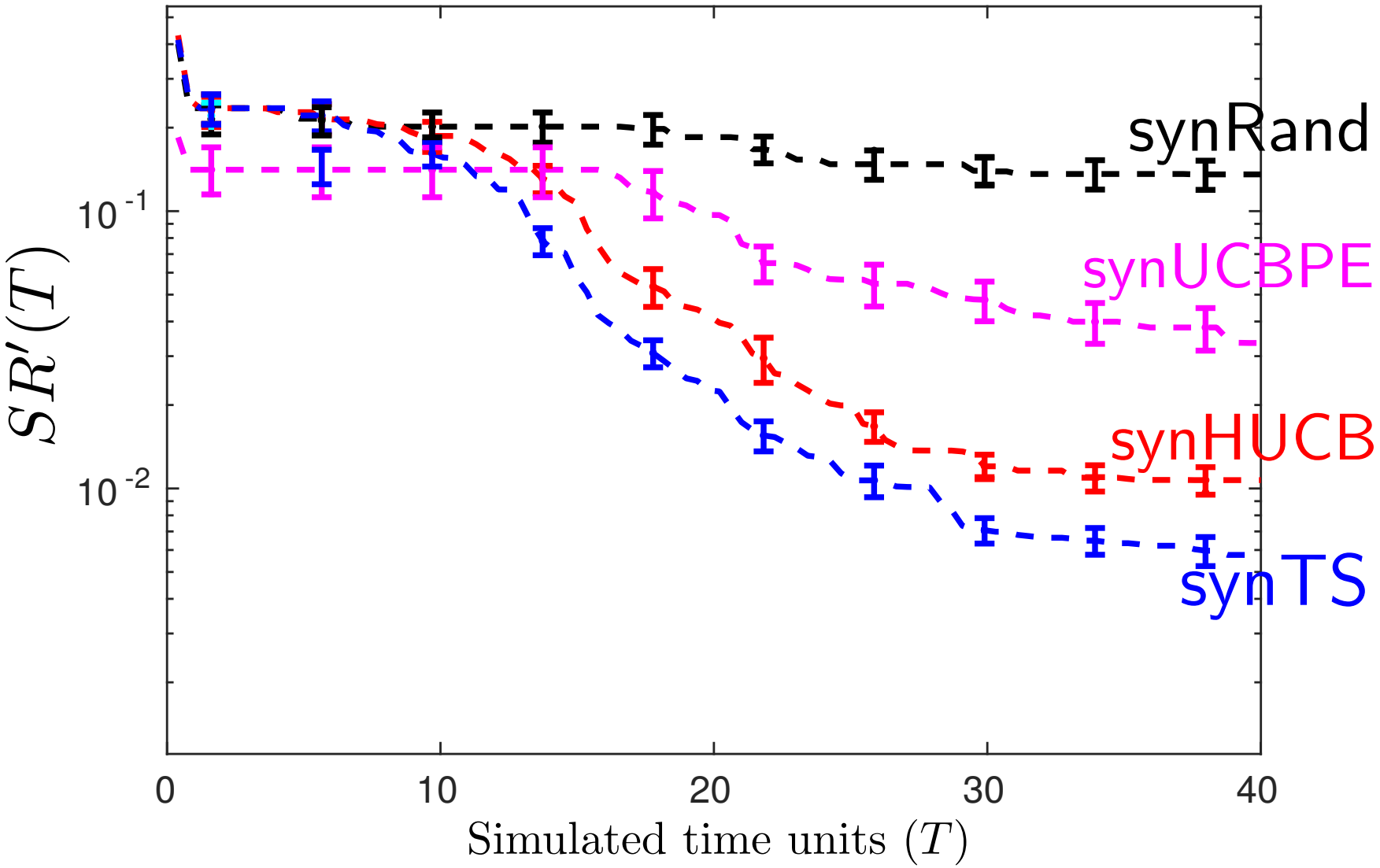
If evaluation times are the same, asyTS $\approx$ synTS.

Otherwise, bounds for asyTS is better than synTS. More the variability in evaluation times, the bigger the difference.

- Bounded tail decay: constant factor
- Sub-gaussian tail decay: $\sqrt{\log(M)}$ factor
- Sub-exponential tail decay: $\log(M)$ factor

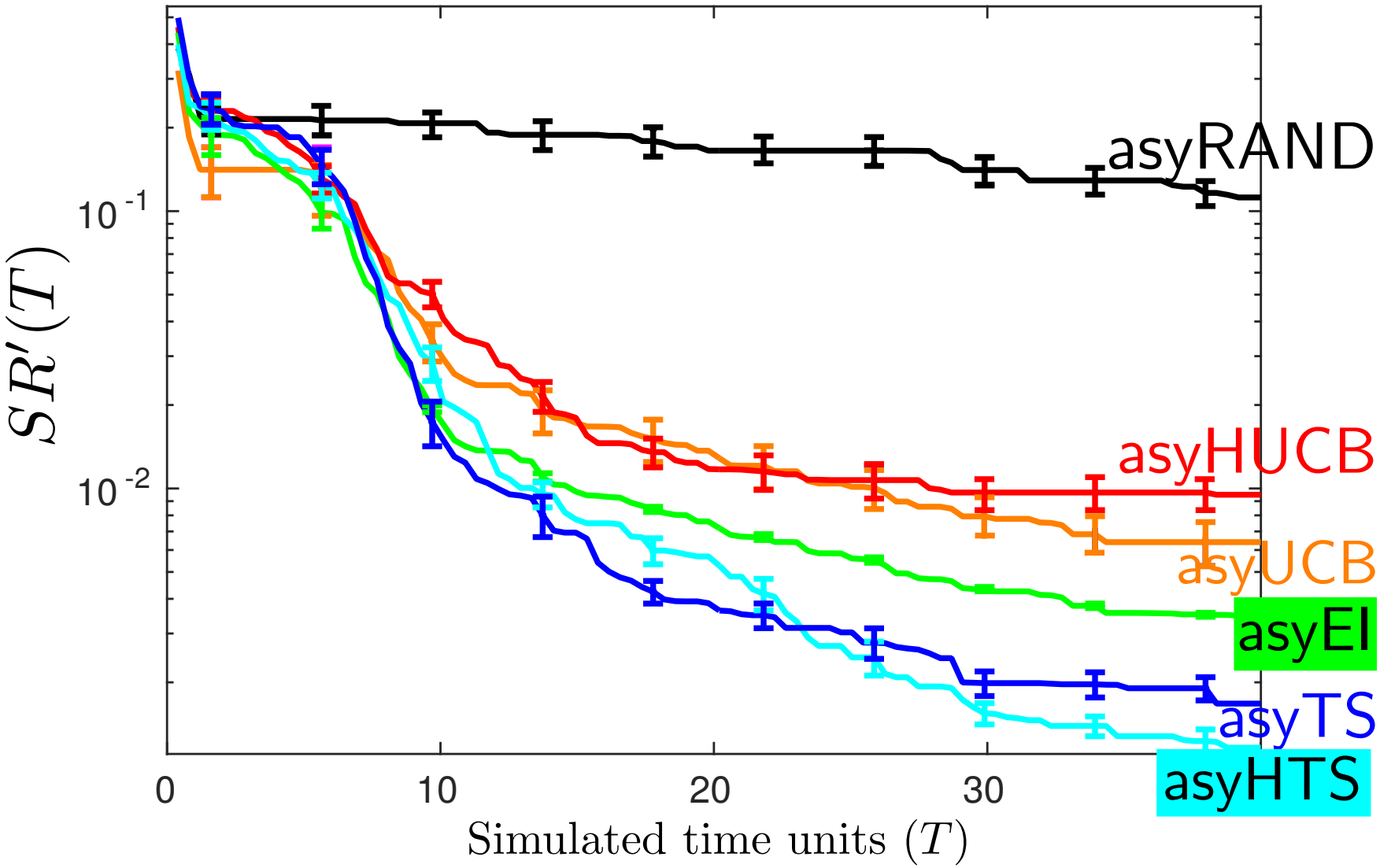
Experiment: Branin-2D $M = 4$

Evaluation time sampled from a uniform distribution



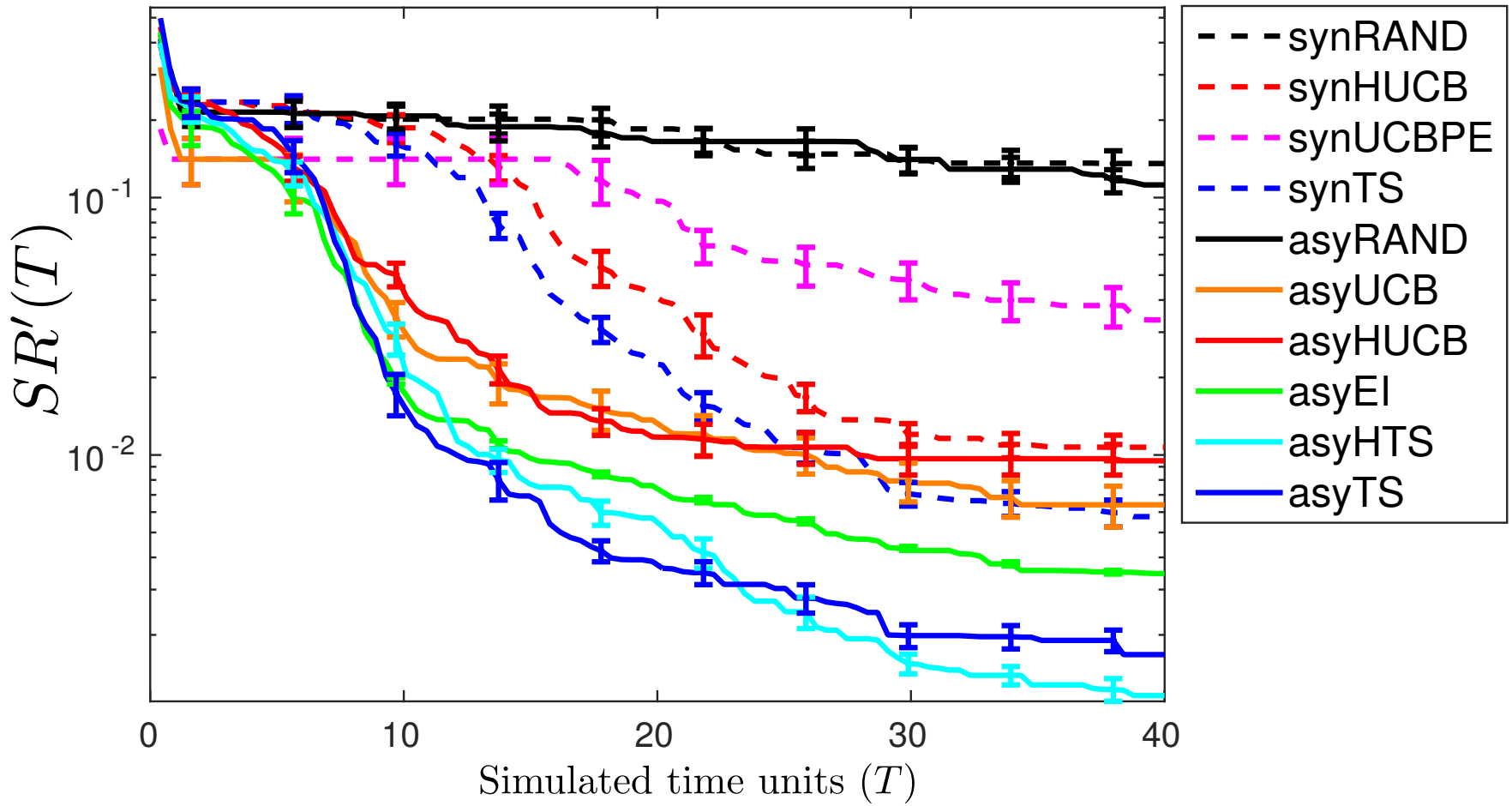
Experiment: Branin-2D $M = 4$

Evaluation time sampled from a uniform distribution



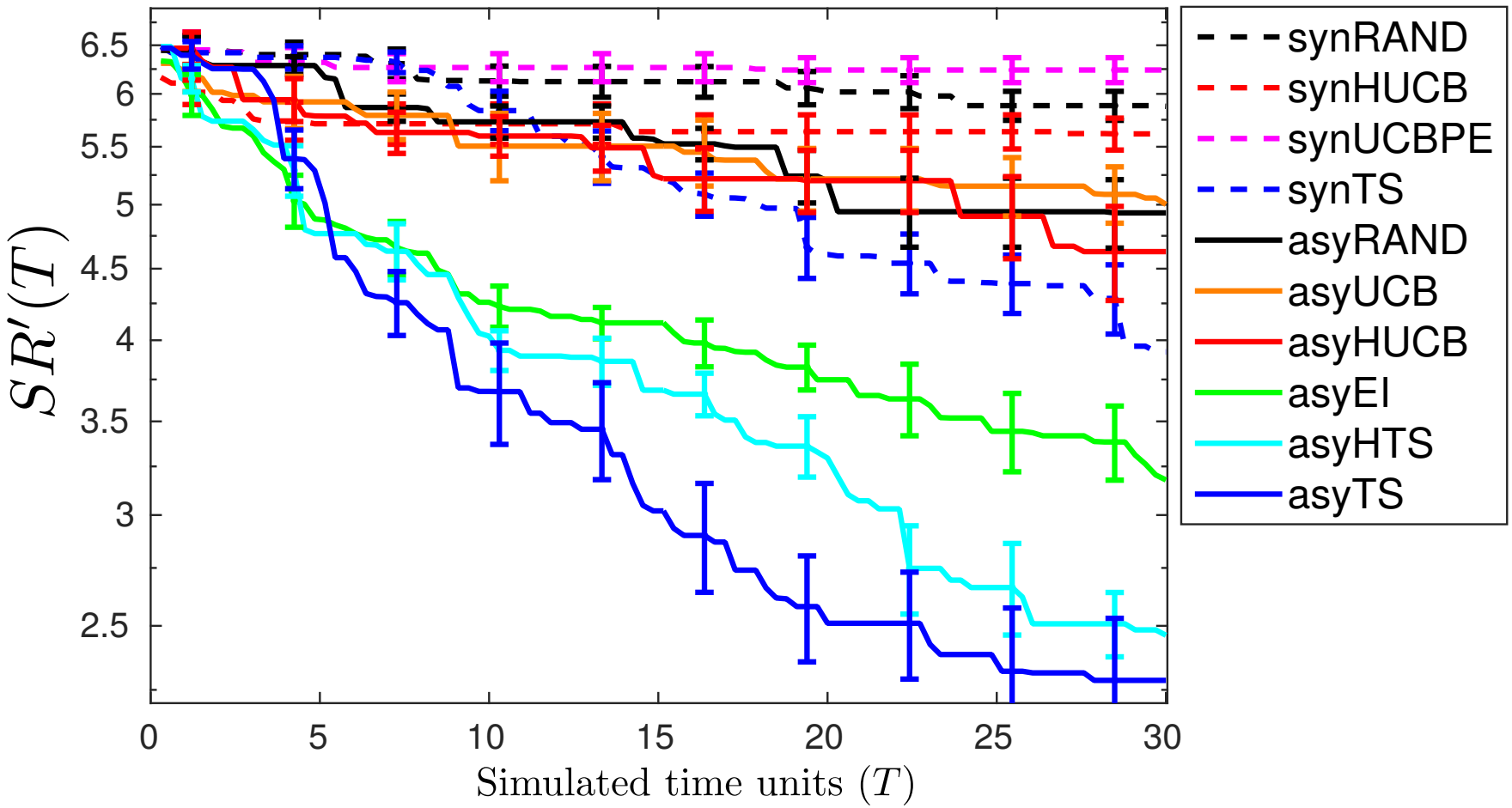
Experiment: Branin-2D $M = 4$

Evaluation time sampled from a uniform distribution



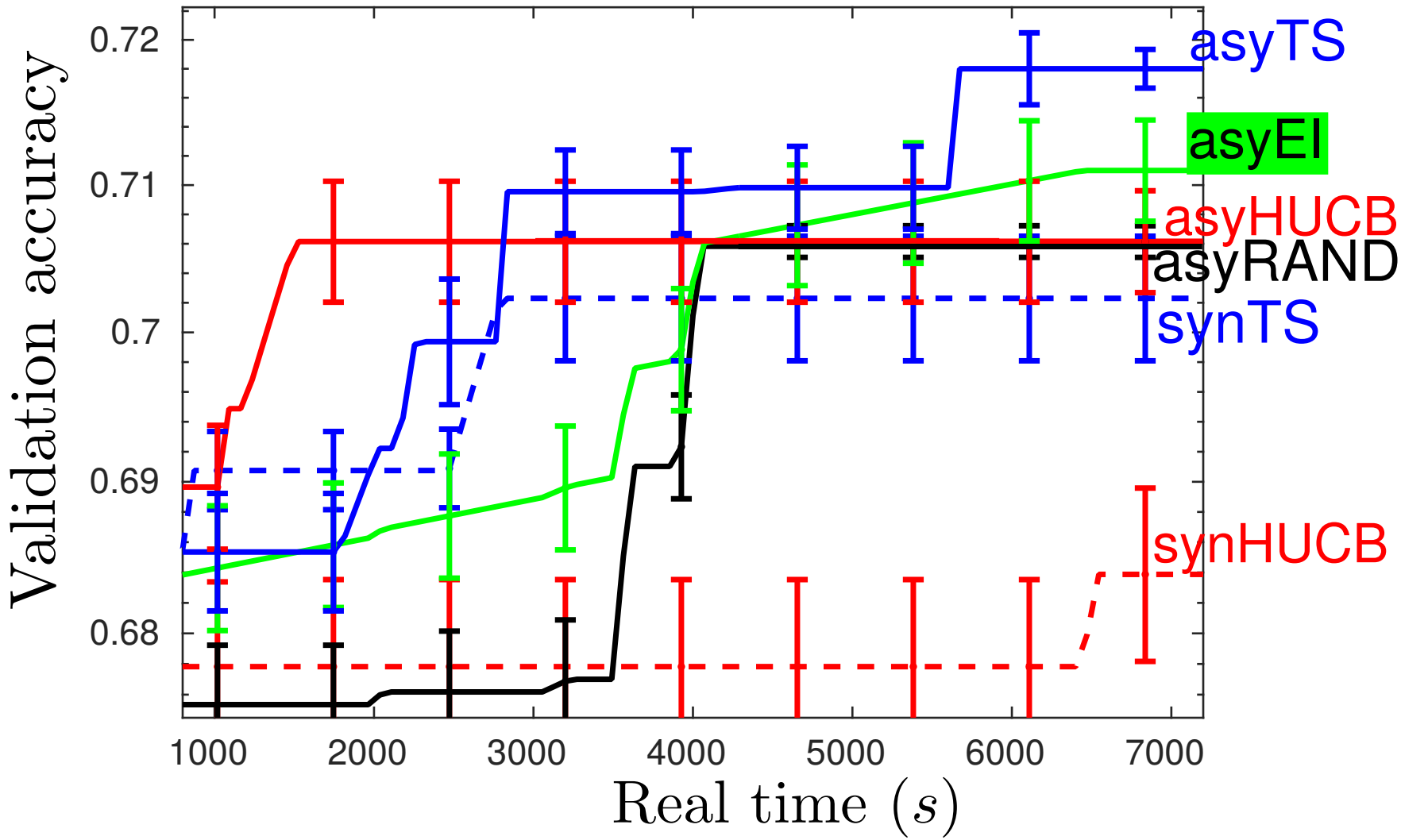
Experiment: Hartmann-18D $M = 25$

Evaluation time sampled from an exponential distribution



Experiment: Model Selection in Cifar10 $M = 4$

Tune # filters in in range $(32, 256)$ for each layer in a 6 layer CNN.
Time taken for an evaluation: 4 - 16 minutes.



Parallelised Thompson Sampling in Neural Networks

(Hernandez-Lobato et al. 2017)

Algorithm 2 Parallel and distributed Thompson sampling

Input: initial data $\mathcal{D}_{\mathcal{I}(1)} = \{\mathbf{x}_i, y_i\}_{i \in \mathcal{I}(1)}$, batch size S

for $t = 1$ **to** T **do**

  Compute current posterior $p(\boldsymbol{\theta}|\mathcal{D}_{\mathcal{I}(t)})$

  for $s = 1$ **to** S **do**

    Sample $\boldsymbol{\theta}$ from $p(\boldsymbol{\theta}|\mathcal{D}_{\mathcal{I}(t)})$

    Select $k(s) \leftarrow \text{argmax}_{j \notin \mathcal{I}(t)} \mathbf{E}[y_j|\mathbf{x}_j, \boldsymbol{\theta}]$

    Collect $y_{k(s)}$ by evaluating f at $\mathbf{x}_{k(s)}$

  Executed in parallel in node s

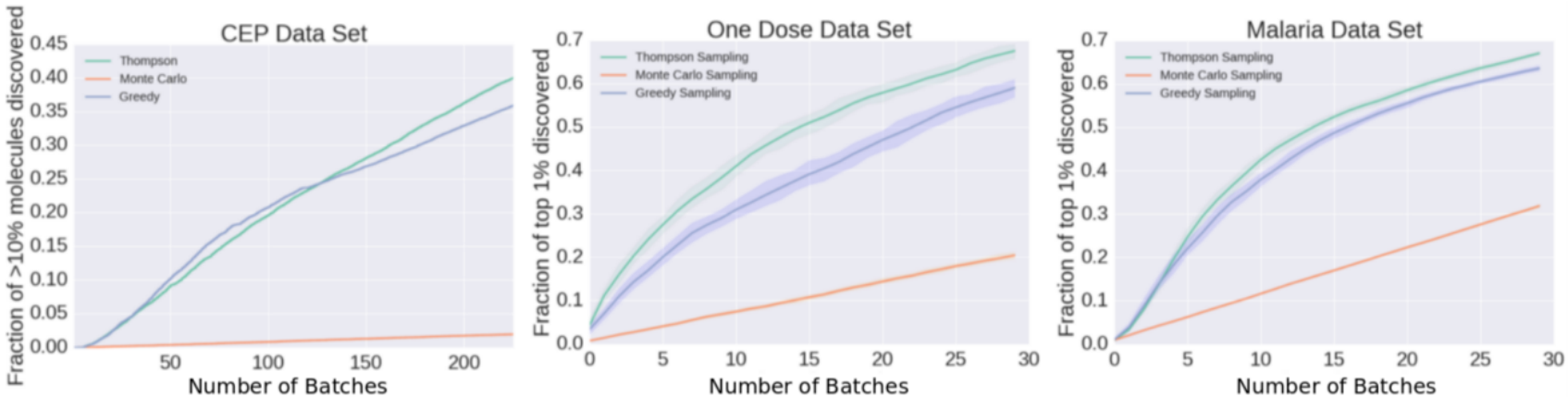
  end for

  $\mathcal{D}_{\mathcal{I}(t+1)} = \mathcal{D}_{\mathcal{I}(t)} \cup \{\mathbf{x}_{k(s)}, y_{k(s)}\}_{s=1}^{S}$

end for

Parallelised Thompson Sampling in Neural Networks

(Hernandez-Lobato et al. 2017)



Outline

- Part I: Bandits in the Bayesian Paradigm
  1. Gaussian processes
  2. Algorithms: Upper Confidence Bound (UCB) & Thompson Sampling (TS)
- Part II: Scaling up Bandits
  1. Multi-fidelity bandit: cheap approximations to an expensive experiment
  2. Parallelising function evaluations
  3. High dimensional input spaces

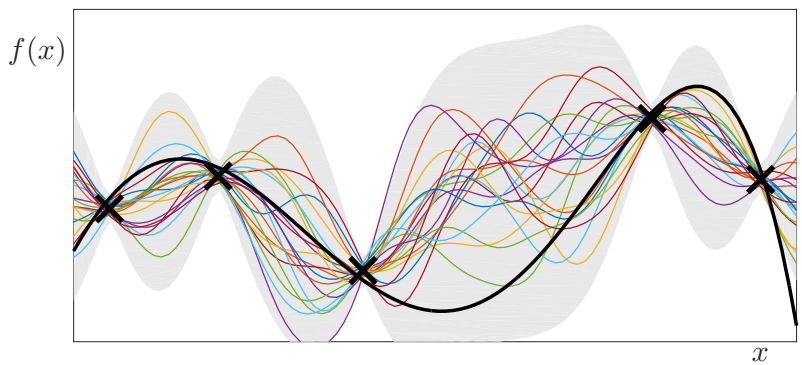
Part 2.3: Optimisation in High Dimensional Input Spaces

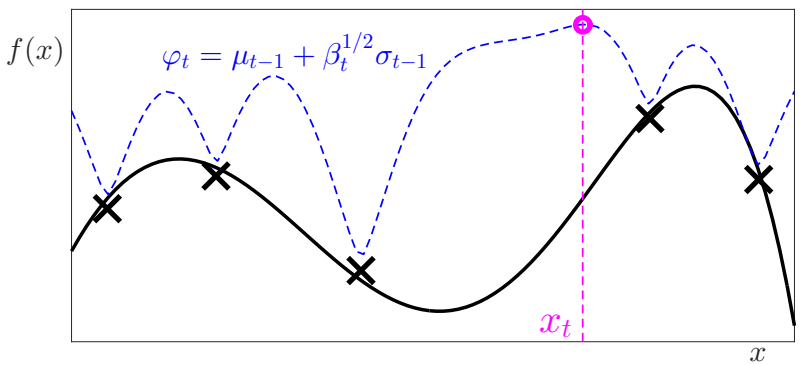
E.g. Tuning a machine learning model with several hyper-parameters

Part 2.3: Optimisation in High Dimensional Input Spaces

E.g. Tuning a machine learning model with several hyper-parameters

At each time step

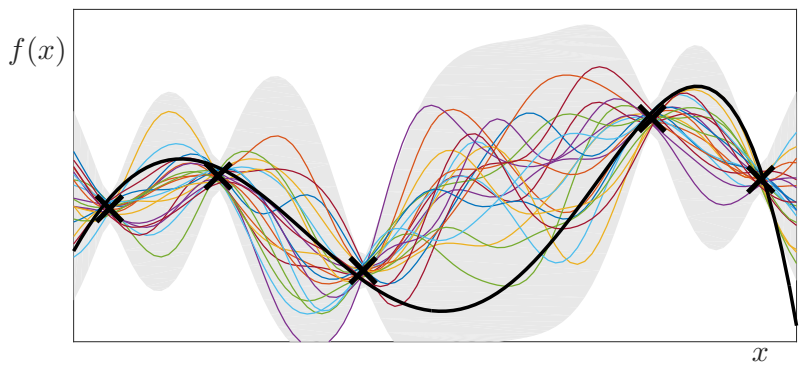


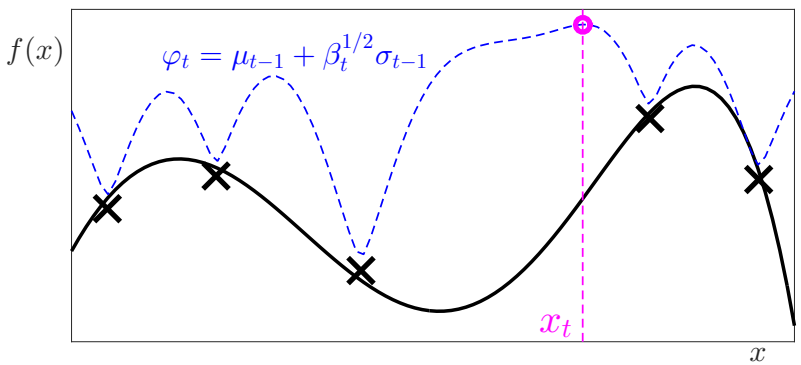


Part 2.3: Optimisation in High Dimensional Input Spaces

E.g. Tuning a machine learning model with several hyper-parameters

At each time step





1. **Statistical Difficulty:** estimating a high dimensional GP.
2. **Computational Difficulty:** maximising a high dimensional acquisition (e.g. upper confidence bound) φ_t.

Additive Models for High Dimensional BO

(Kandasamy et al. ICML 2015)

E.g. $f(x_{\{1,\ldots,10\}}) = f^{(1)}(x_{\{1,3,9\}}) + f^{(2)}(x_{\{2,4,8\}}) + f^{(3)}(x_{\{5,6,10\}})$.

Additive Models for High Dimensional BO

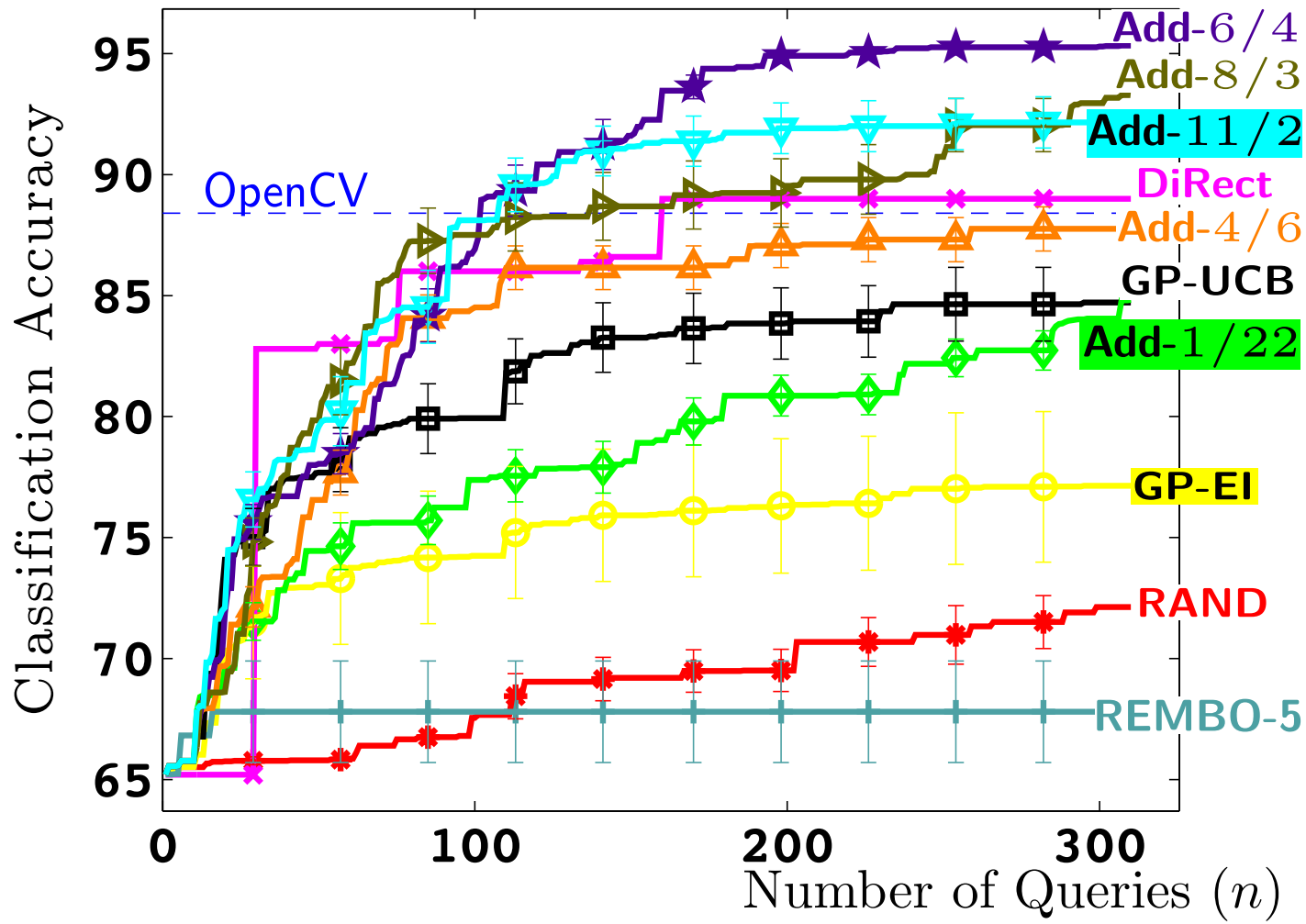
(Kandasamy et al. ICML 2015)

E.g. $f(x_{\{1,\ldots,10\}}) = f^{(1)}(x_{\{1,3,9\}}) + f^{(2)}(x_{\{2,4,8\}}) + f^{(3)}(x_{\{5,6,10\}})$.

- Better statistical properties: sample complexity improves from exponential in d to linear in d.
- **Add-GP-UCB** algorithm: computationally tractable even for large d.
- Better bias variance trade-off in practice: algorithm does well even if f is not additive.

Experiment: Viola & Jones Cascade classifier

Tune 22 hyper-parameters in the V&J classifier.



Summary

- Bandits are a framework for studying exploration vs exploitation trade-offs when optimising black-box functions.
- Several applications: Hyper-parameter Tuning, materials synthesis, scientific experiments etc.
- Several algorithms: UCB, TS, EI etc.

Summary

- Bandits are a framework for studying exploration vs exploitation trade-offs when optimising black-box functions.
- Several applications: Hyper-parameter Tuning, materials synthesis, scientific experiments etc.
- Several algorithms: UCB, TS, EI etc.

- **Multi-fidelity Bandits:** Use cheap approximations to a an expensive experiment to speed up optimisation.
- **Parallelised TS:** Simple and intuitive way to deal with multiple workers.
- **High dimensional optimisation:** Additive models have favourable statistical and computational properties.

Akshay

Barnabás

Gautam

Jeff

Junier

Thank You

Slides: `www.cs.cmu.edu/~kkandasa/talks/pitt-hptune-slides.pdf`