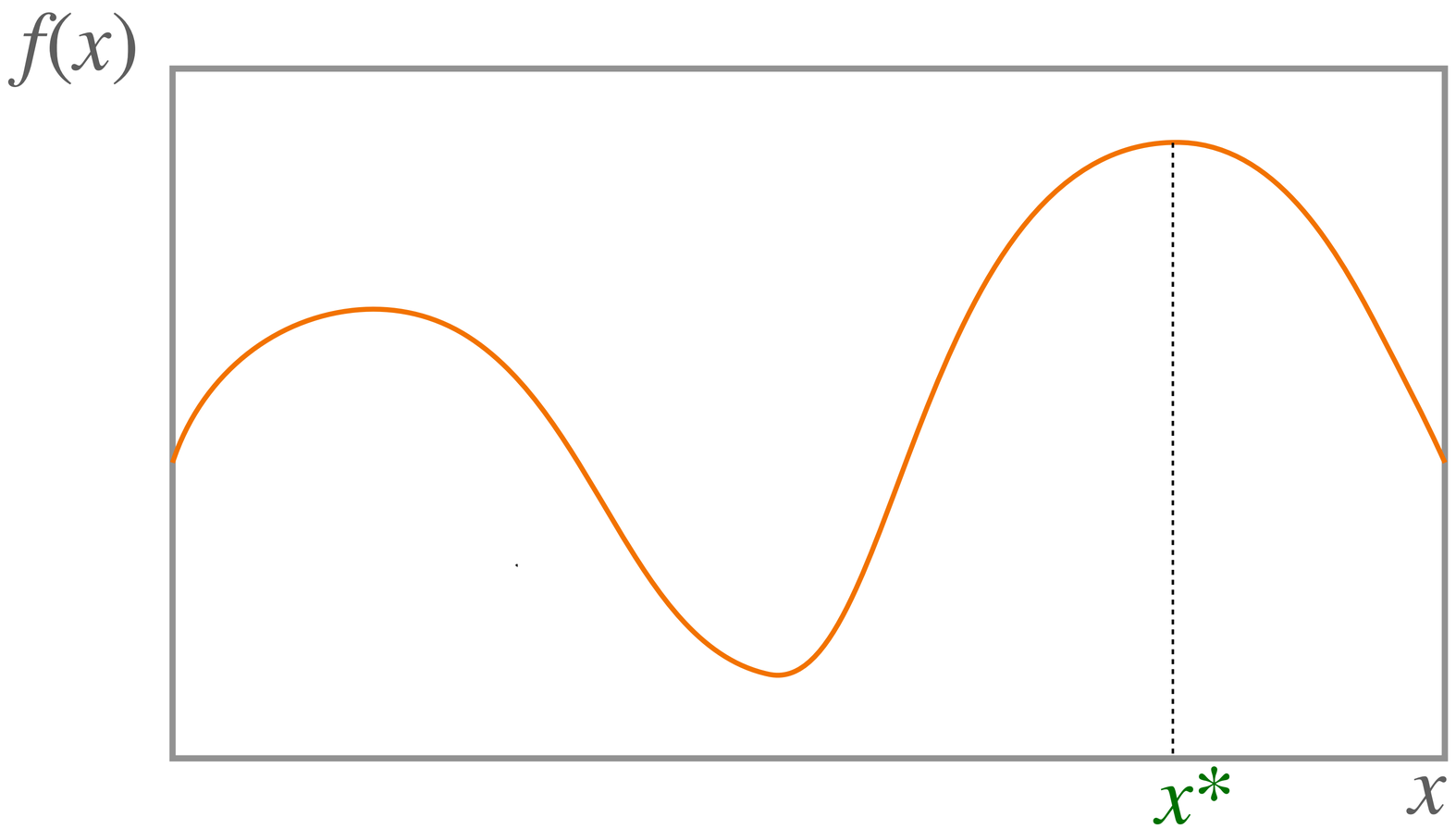
(Private) Kernelized Bandits with Distributed Biased Feedback

Fengjiao Li[1] Xingyu Zhou[2] Bo Ji[1]

[1]Virginia Tech [2]Wayne State University

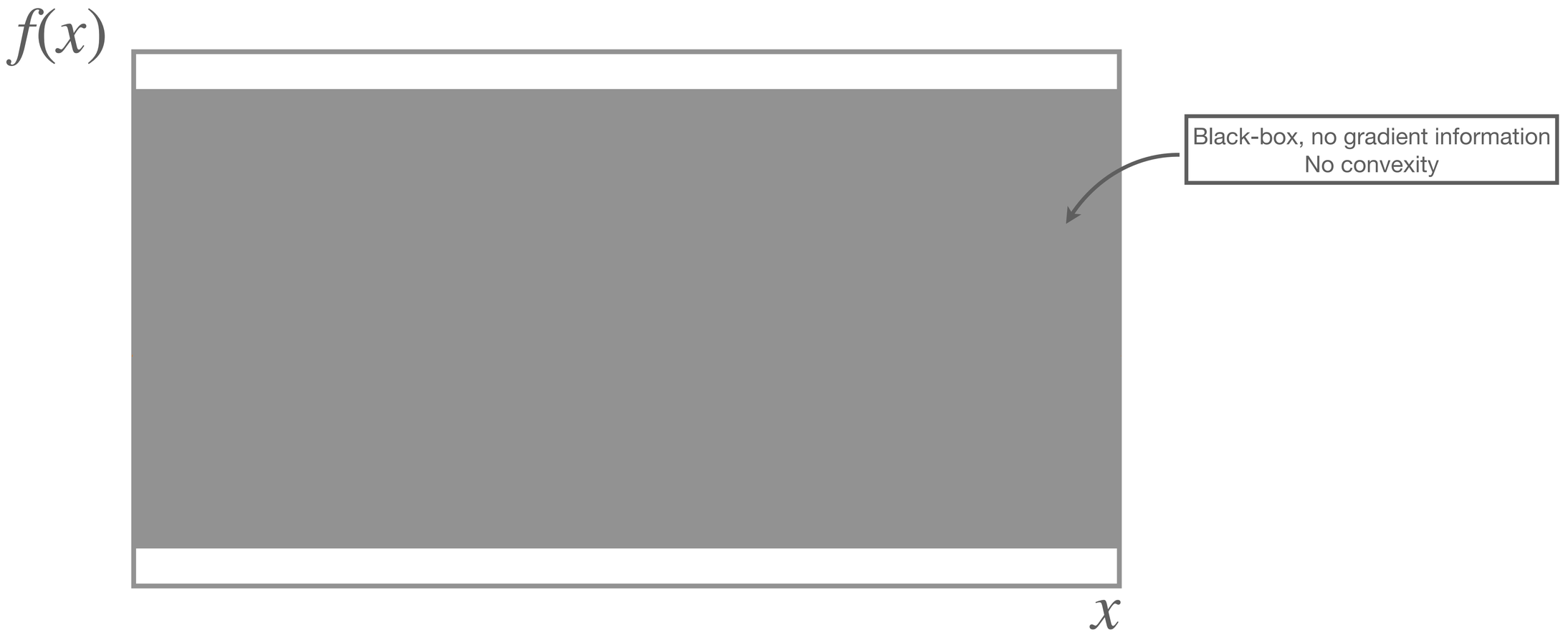
Sequential Decision Making

Black-box optimization



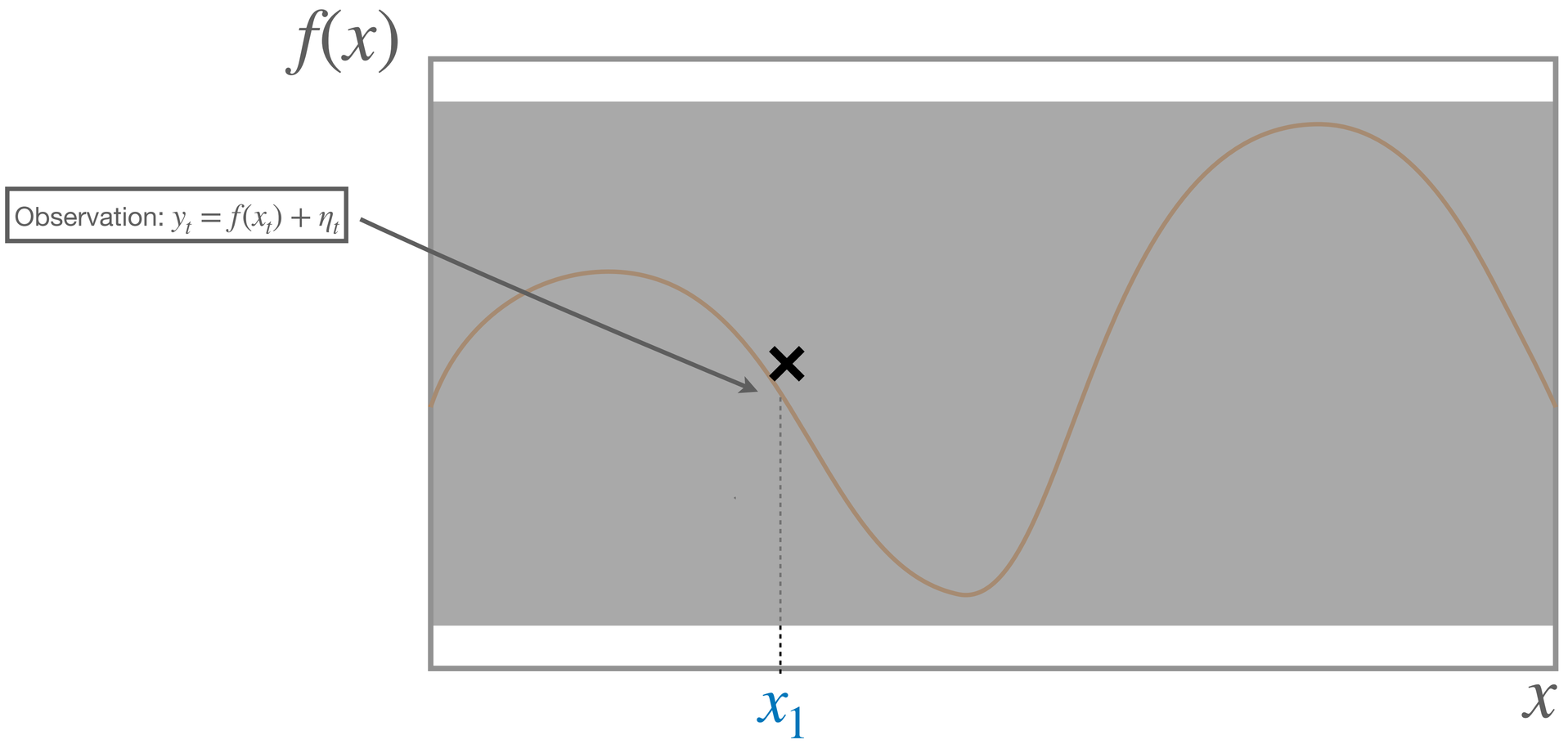
Sequential Decision Making

Black-box optimization



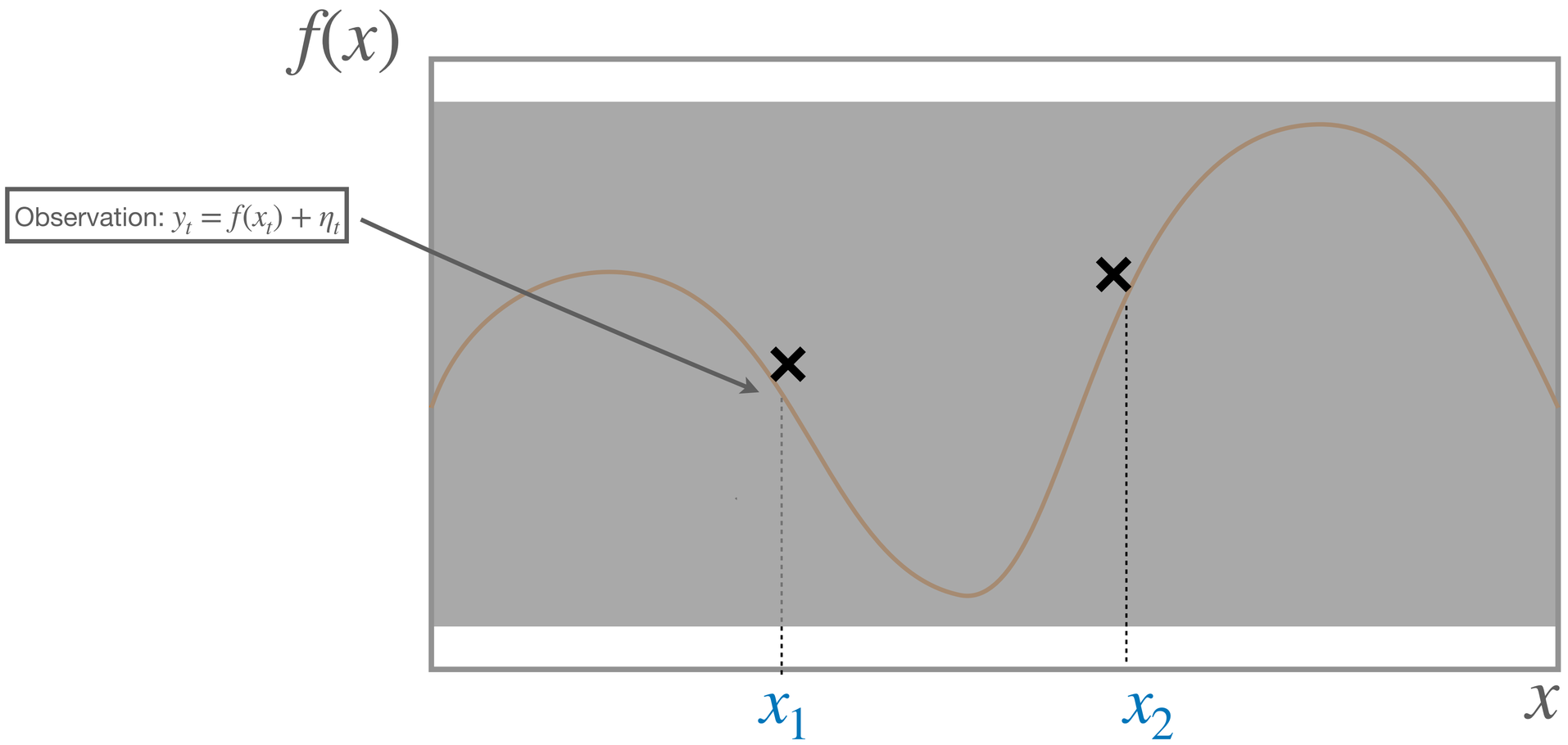
Sequential Decision Making

Black-box optimization



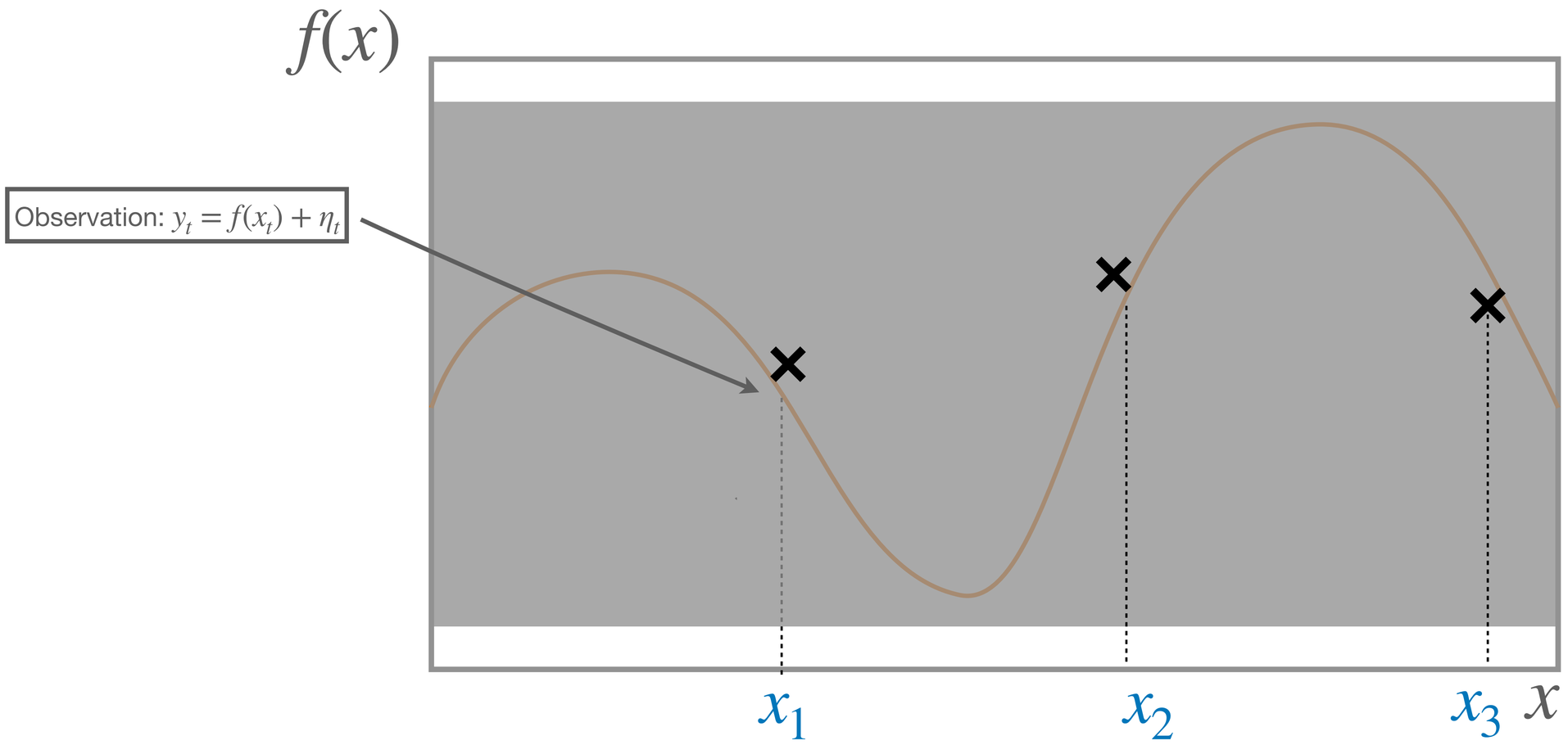
Sequential Decision Making

Black-box optimization



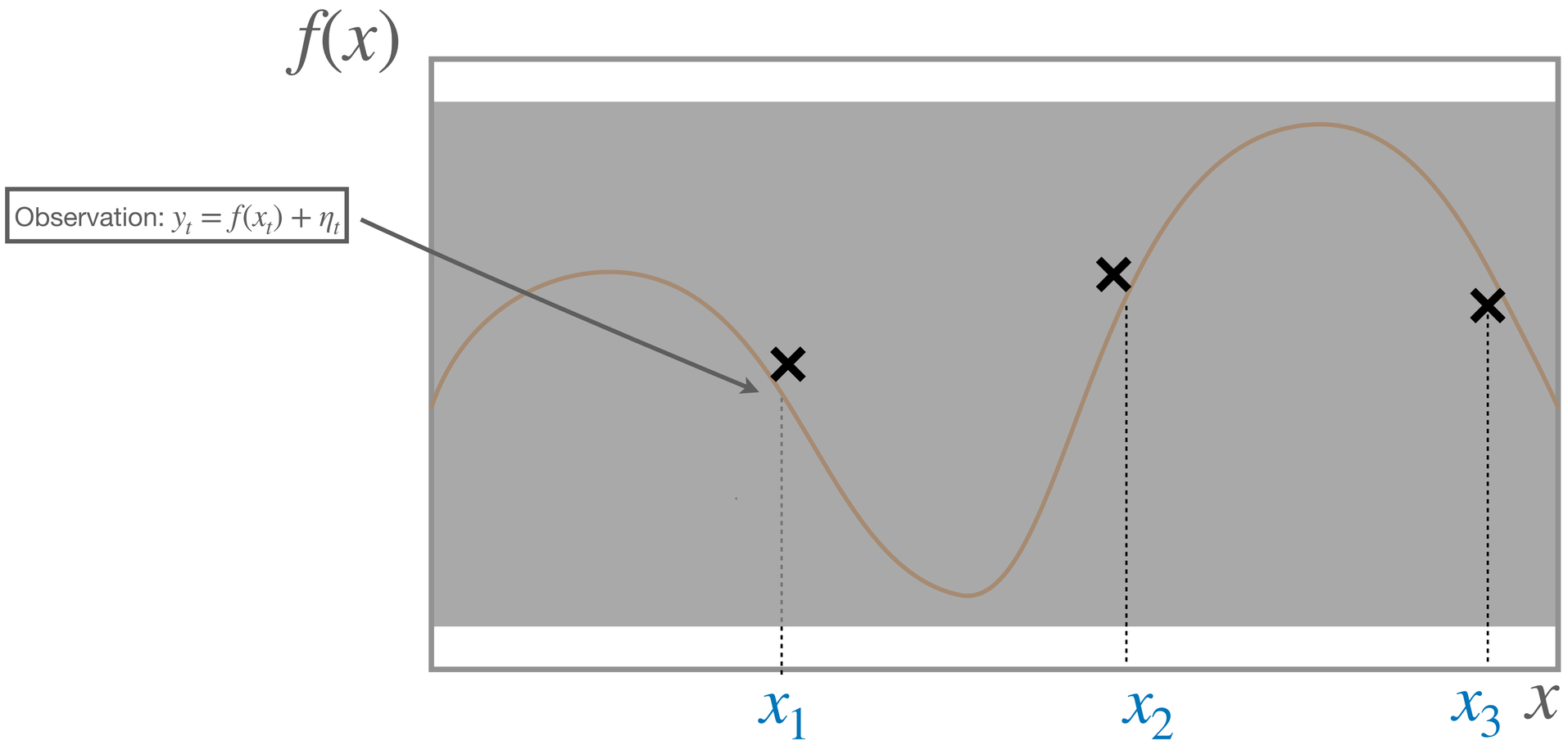
Sequential Decision Making

Black-box optimization



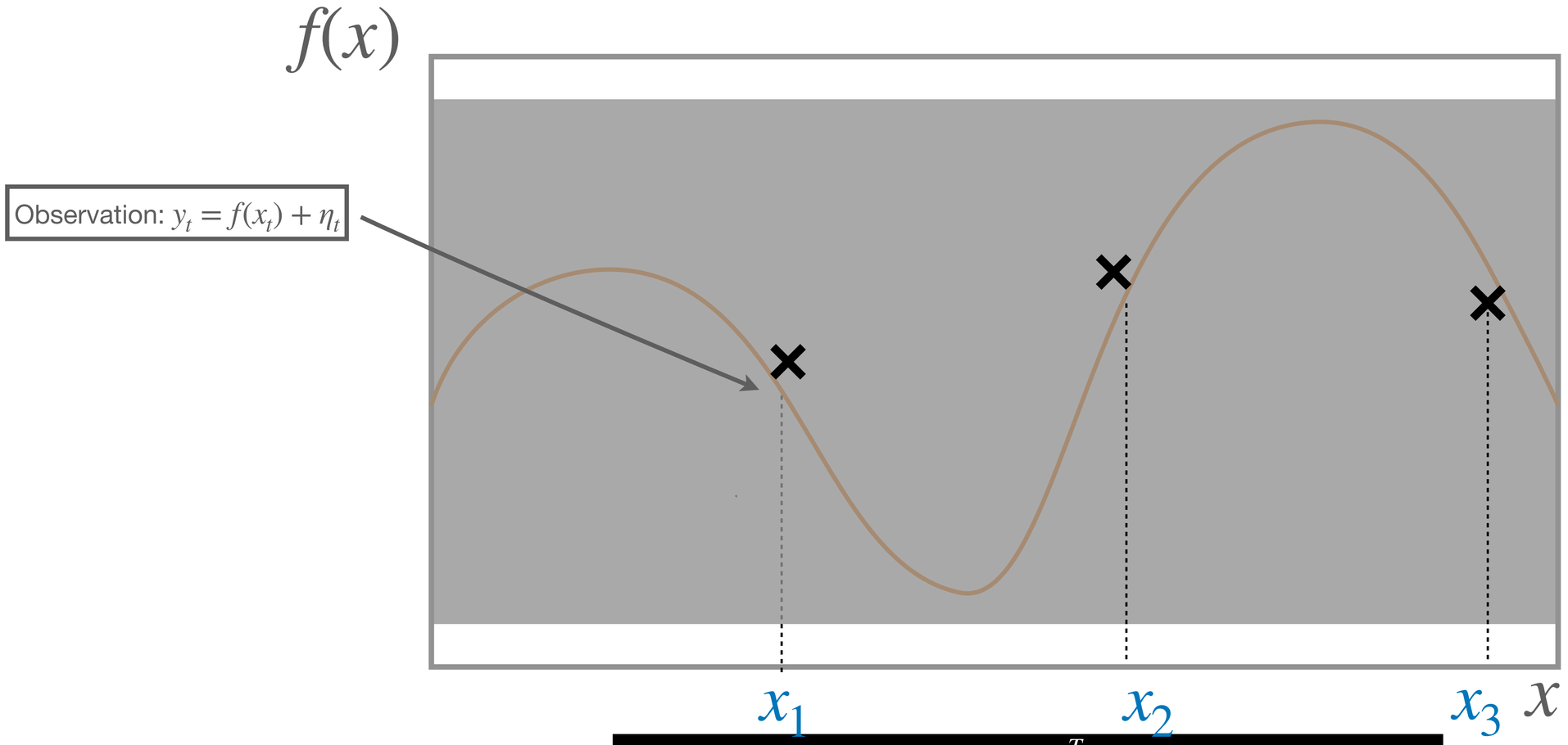
Sequential Decision Making

Black-box optimization



Sequential Decision Making

Black-box optimization



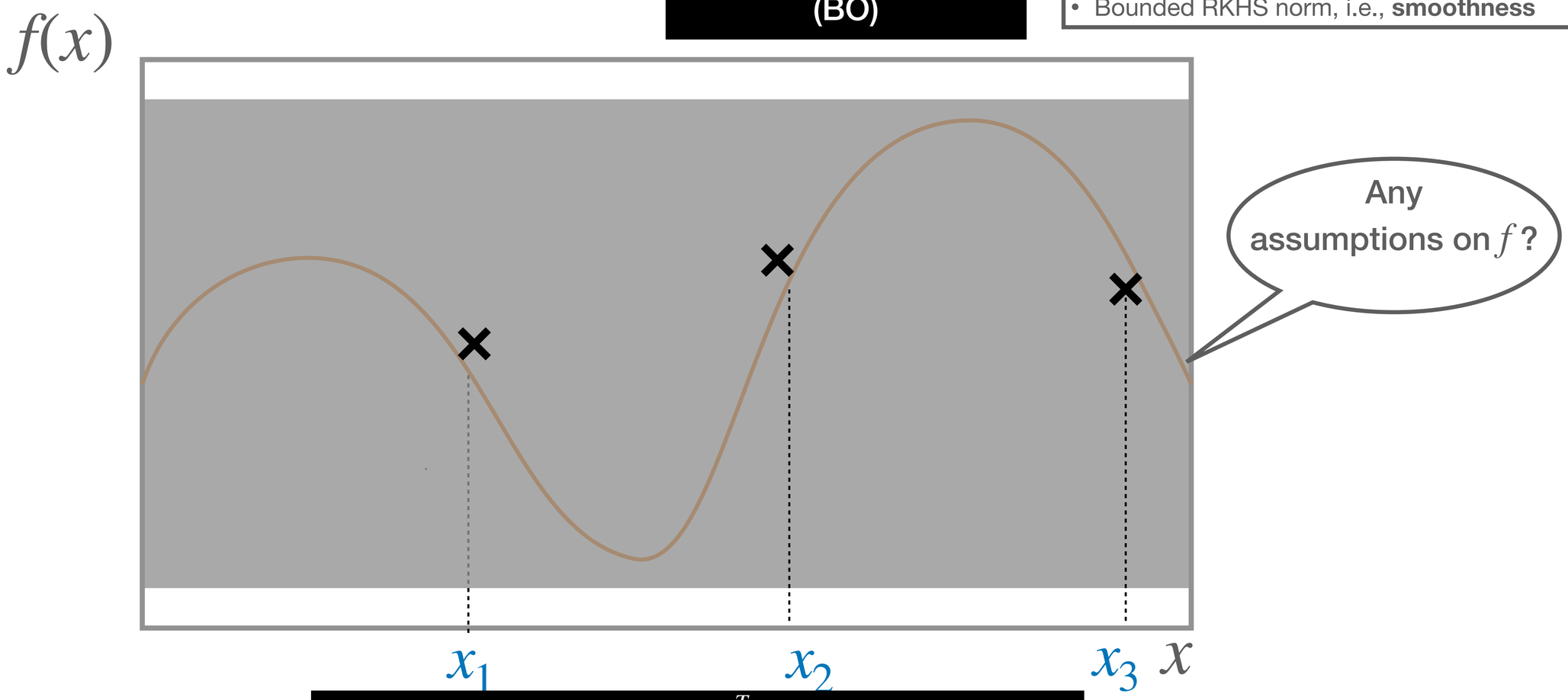
Minimize $R(T) = \sum_{t=1}^{T} f(x^*) - f(x_t)$

Sequential Decision Making

Black-box optimization

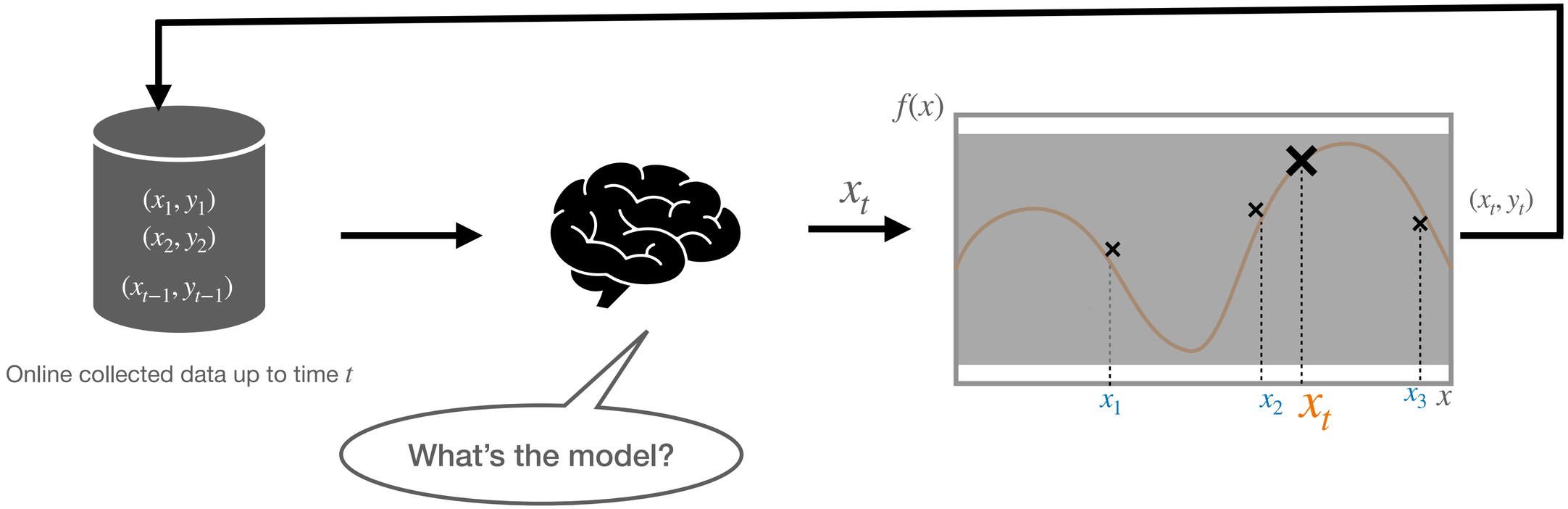
Bayesian Optimization (BO)

- f lies in Reproducing Kernel Hilbert Space (**RKHS**)
- RKHS approximates **any** continuous function
- Include **linear function** as a special case
- Bounded RKHS norm, i.e., **smoothness**



Minimize $R(T) = \sum_{t=1}^{T} f(x^*) - f(x_t)$

General Algorithm Design



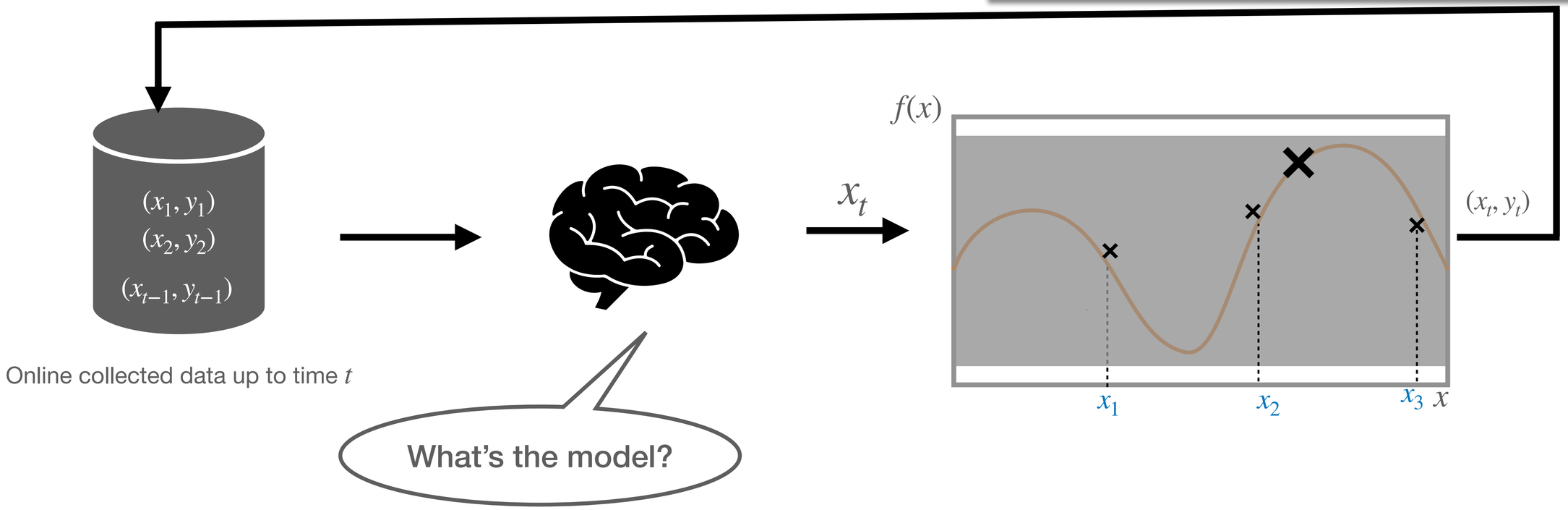
General Algorithm Design

Dual form of least-square regression

Dose not depend on y!

Intuition&Fact Box

- $Y_t = [y_1, y_2, \ldots, y_T]^\top$ - reward col-vector
- $K_t = [k(u,v)]_{u,v \in D_t}$ — $t \times t$ matrix
- $k_t(x) = [k(x_1, x), \ldots, k(x_t, x)]^\top$
- $\mu_t(x) = k_t(x)^\top (K_t + \lambda I)^{-1} Y_t$ — **updated mean**
- $k_t(x, x') = k(x, x') - k_t(x)^\top (K_t + \lambda I)^{-1} k_t(x')$ - **updated cov**



Gaussian Process Surrogate Model

$GP(0, k(x, x'))$

$\mathcal{N}(0, \lambda)$

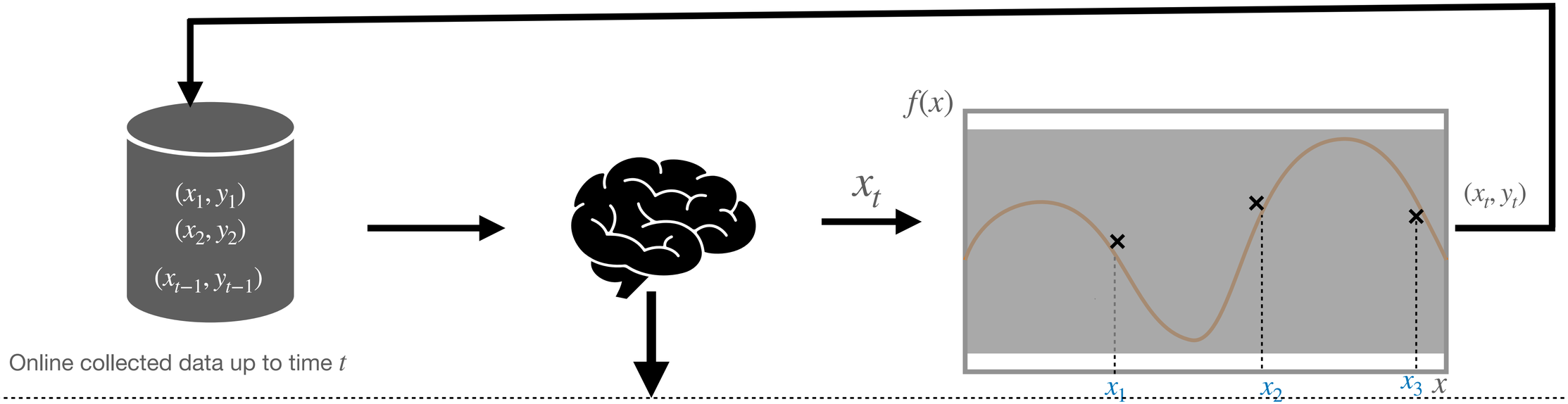
D_t

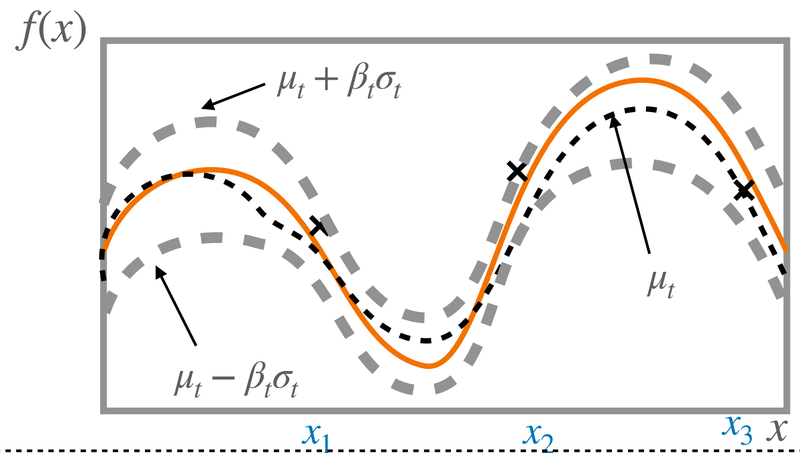
$GP(\mu_t, k_t(x, x'))$

Only used for algorithm design!

Closed-form posterior update

General Algorithm Design

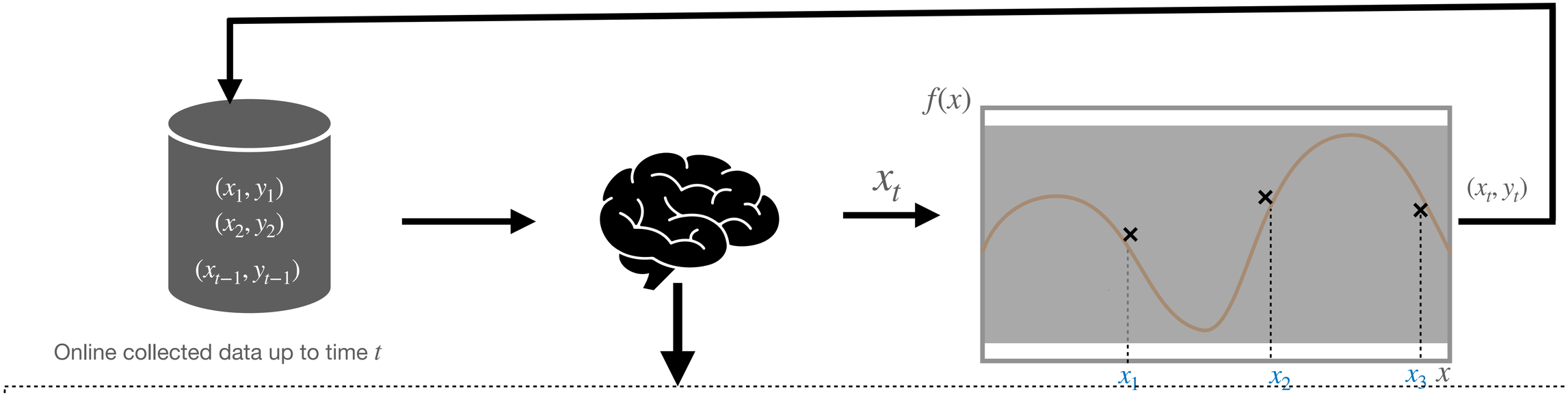


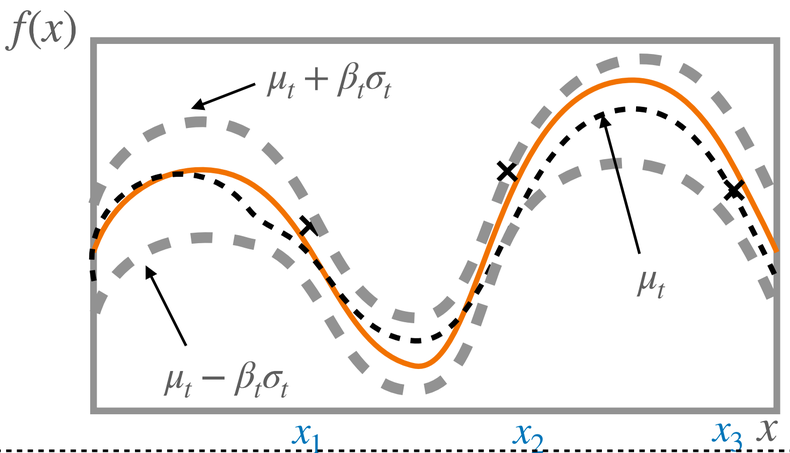


Gaussian Process Surrogate Model
$GP(\mu_t, k_t(x, x'))$

w.h.p. $|f(x) - \mu_t(x)| \leq \beta_t\sigma_t(x), \forall x, t$ [CG'17]

General Algorithm Design





Gaussian Process Surrogate Model
$GP(\mu_t, k_t(x, x'))$

GP-UCB
$f_t = \mu_t + \beta_t\sigma_t$

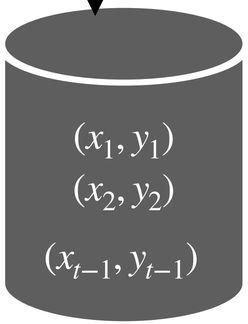
GP-TS
$f_t \sim GP(\mu_t, k_t)$

$x_t = \arg\max f_t(x)$

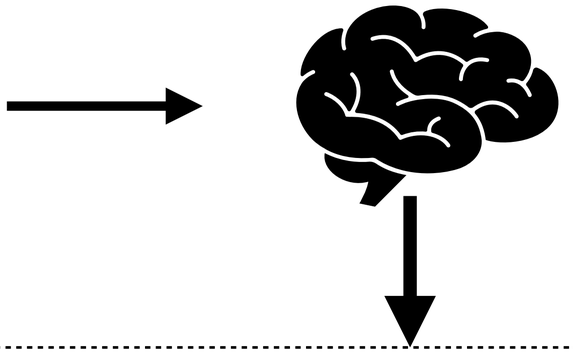
General Algorithm Design

Intuition&Fact Box

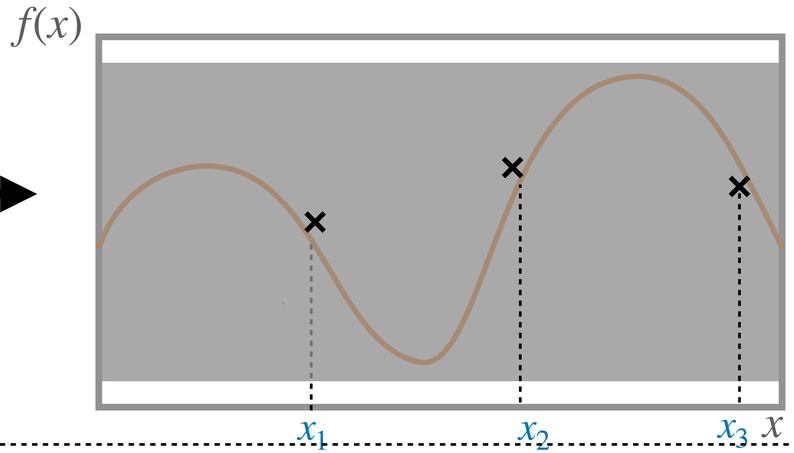
- Achieve $\tilde{O}(\gamma_T\sqrt{T})$ regret bound
- γ_T — kernel-dependent, intrinsic dimensionality
  - Linear kernel: $O(d \log T)$, recover linear bandits
  - Gaussian kernel (SE): $\log^{d+1}(T)$
- Time complexity $O(T^3)$
  - Each GP update $O(t^2)$ using rank-1 update

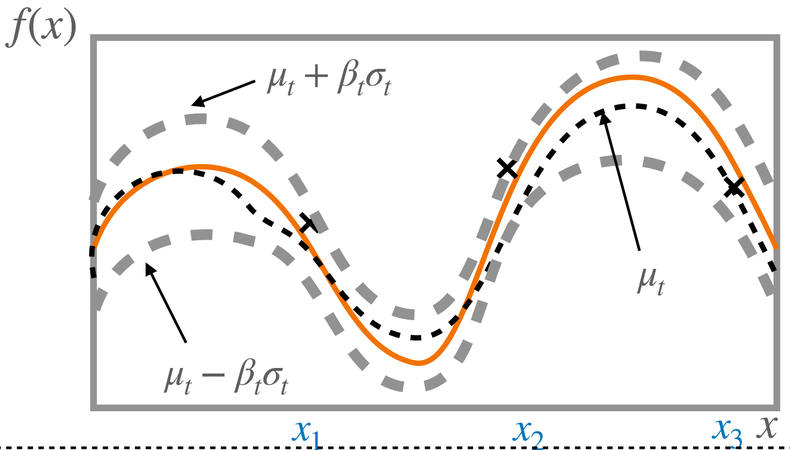


Online collected data up to time t



x_t





Gaussian Process Surrogate Model
$GP(\mu_t, k_t(x, x'))$

GP-UCB
$f_t = \mu_t + \beta_t\sigma_t$

GP-TS
$f_t \sim GP(\mu_t, k_t)$

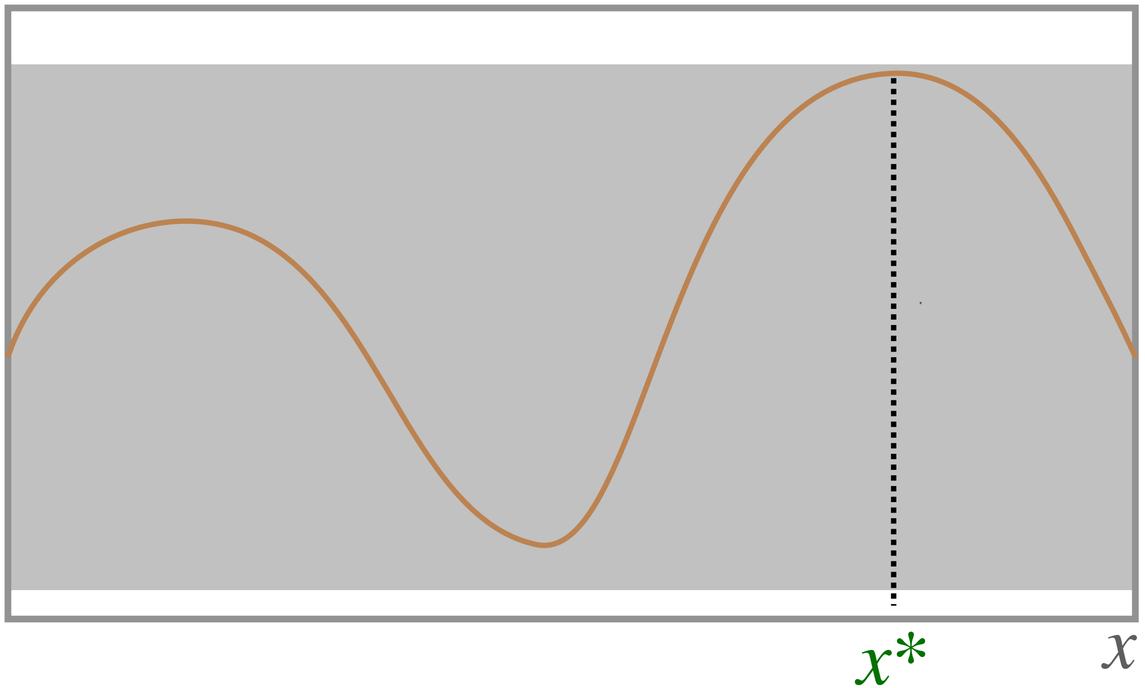
$x_t = \arg\max f_t(x)$

What if…

f is an expectation

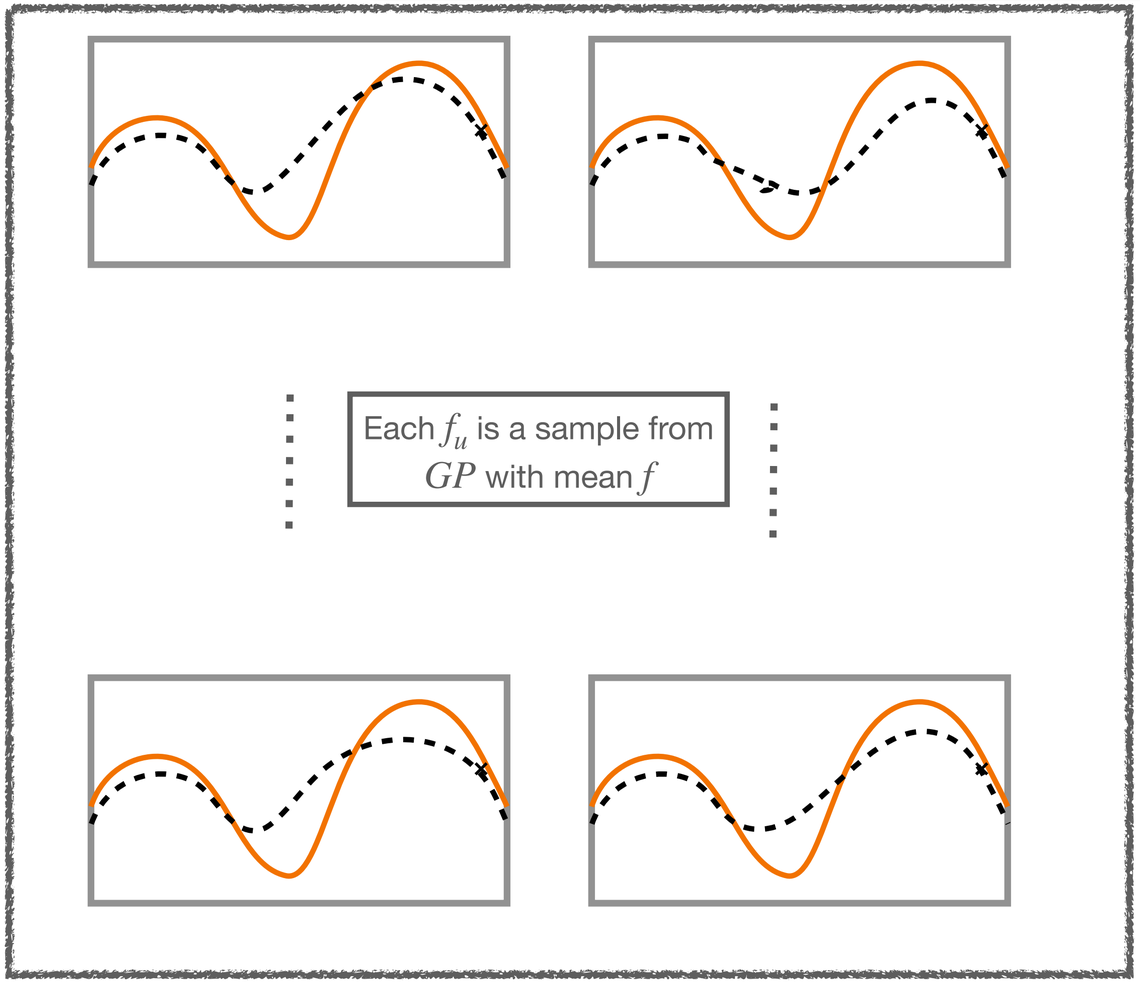
Stochastic optimization
$f_u(x) := F(x, u)$

$$f(x) = \mathbb{E}_u F(x, u)$$



Related Area

- Zero-order non-convex **stochastic** opt.
- Different assumptions and metrics
  - Without Lipschitz
  - Total regret vs. convergence rate



Examples

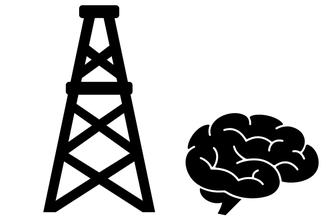
Policy making, cellular config and more…

How to apply BO (kernel bandits) in this case?

Policy maximizes expected user satisfaction

$$f(x) = \mathbb{E}_u F(x, u)$$

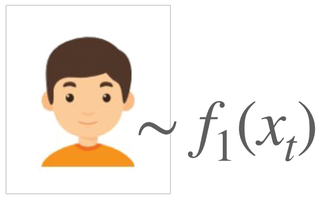
$$x_t$$



Base station config optimizes expected performance


$\sim f_5(x_t)$


$\sim f_3(x_t)$

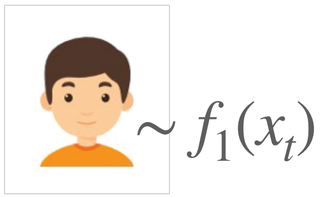

$\sim f_1(x_t)$


$\sim f_2(x_t)$

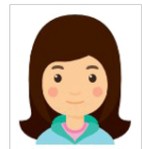

$\sim f_4(x_t)$


$\sim f_5(x_t)$


$\sim f_3(x_t)$


$\sim f_1(x_t)$


$\sim f_2(x_t)$


$\sim f_4(x_t)$

Option I: Aggregate All

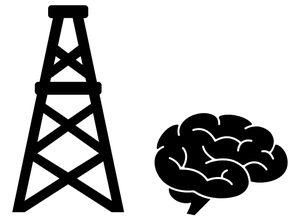
Impractical…

How to apply BO (kernel bandits) in this case?

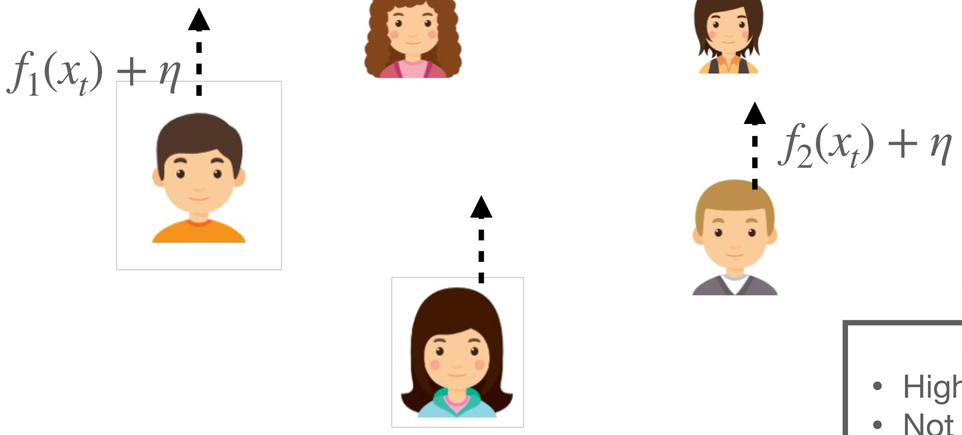
Policy maximizes expected user satisfaction

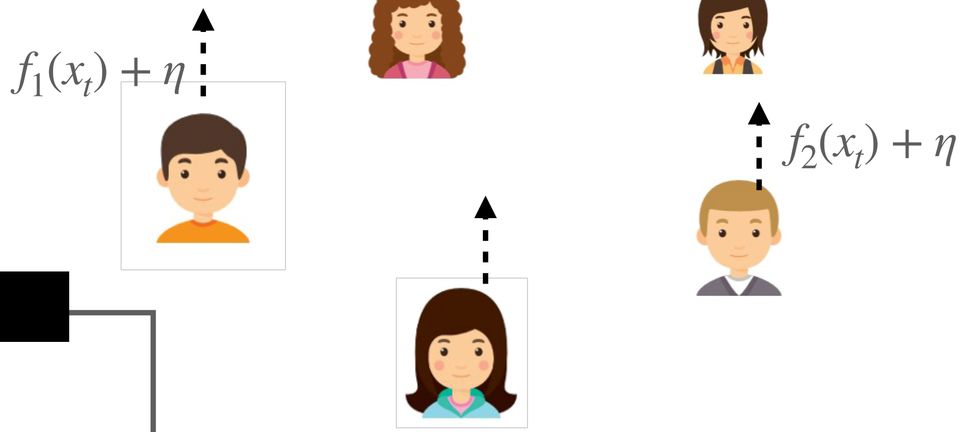
$$f(x) = \mathbb{E}_u F(x, u)$$

$$x_t$$



Base station config optimizes expected performance





Large amount of comm.

- High communication cost
- Not everyone will share their feedback

Option II: Sample One

GP-UCB/TS with a larger noise

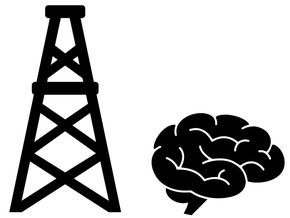
Intuition&Fact Box

- This is similar to SGD for Stochastic Optimization
  - One-pass over the dataset to have unbiased sample

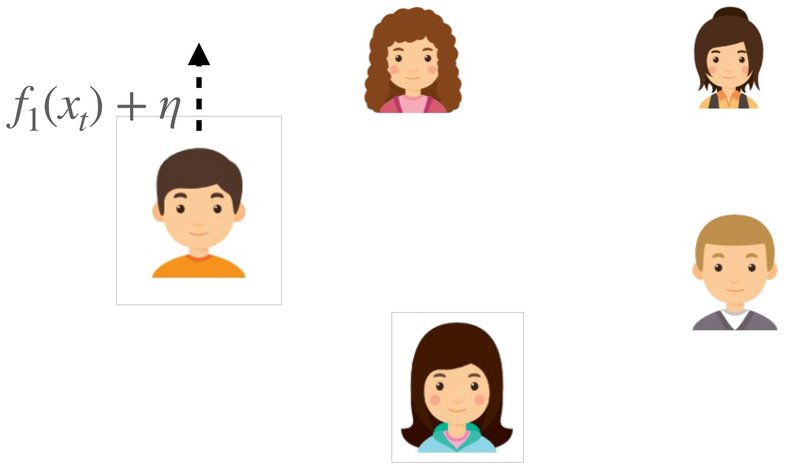
Policy maximizes expected user satisfaction

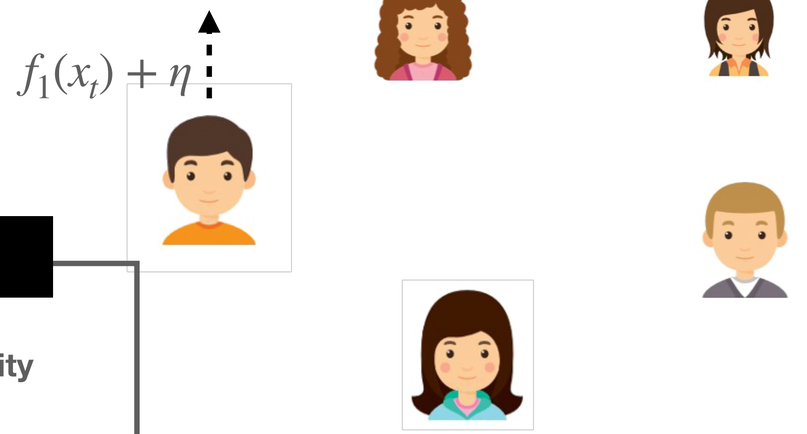
$$f(x) = \mathbb{E}_u F(x, u)$$

$$x_t$$



Base station config optimizes expected performance





Three Limitations

- Need $O(T)$ **fresh users**
- Poor scalability due to $O(T^3)$ **time complexity**
- User's **privacy** needs to be protected

Option II: Sample One

GP-UCB/TS with larger noise

Intuition&Fact Box

- This is similar to SGD for Stoc. Optimization
  - One-pass over the dataset to have unbiased sample

$$f(x) = \mathbb{E}_u F(x, u)$$

Is there any **computation/communication efficient** algorithm that interacts with **only** $o(T)$ users while still guaranteeing **sub-linear regret** and **privacy?**

Policy ma

ormance

x_t

$f_1(x_t) + \eta$

$f_1(x_t) + \eta$

Three Limitations

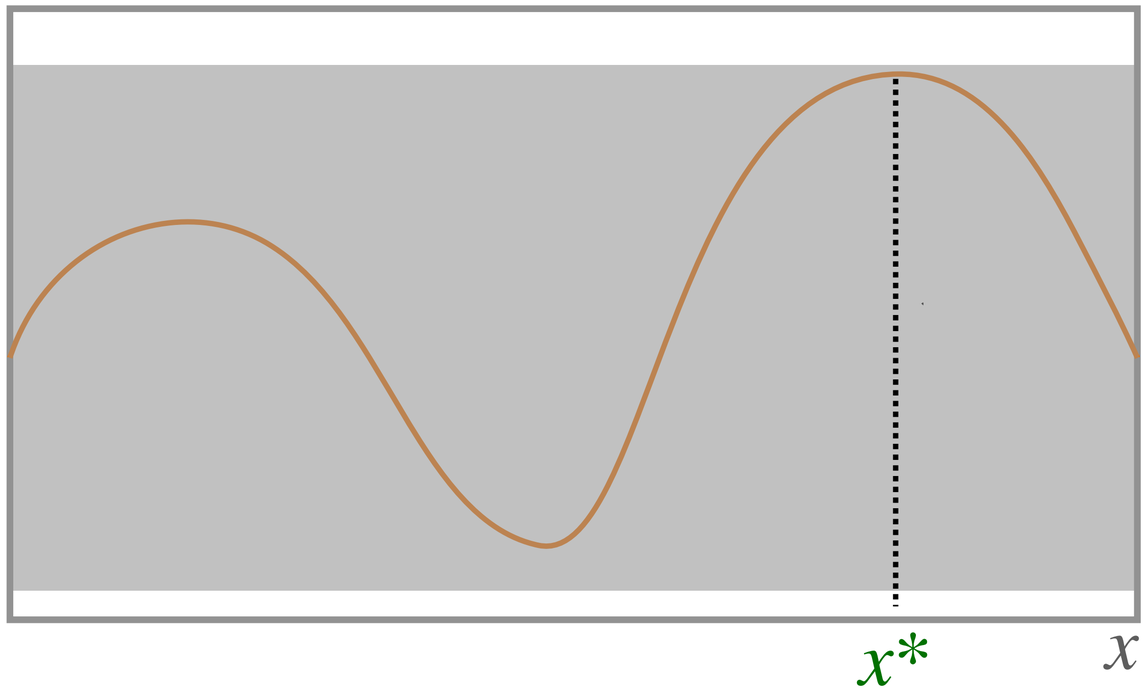
- Need $O(T)$ fresh users
- Poor scalability due to $O(T^3)$ time complexity
- User's privacy needs to be protected

Contribution

Main Results

Policy maximizes expected user satisfaction

$$f(x) = \mathbb{E}_u F(x, u)$$



1. Propose a distributed **phase-elimination-type** BO algorithm with **user sampling** and **batching**
2. Involve with only $O(T^\alpha)$ unique users, $\alpha \in (0,1]$ —user sampling parameter
3. Incur (sub-linear) regret $\tilde{O}(T^{1-\alpha/2} + \sqrt{\gamma_T T})$, computation $O(\gamma_T^4 \log T + \gamma_T T^\alpha)$, communication $O(\gamma_T T^\alpha)$
4. Also establish regret performance under different Differential Privacy (DP) trust models — privacy “for free” in certain models

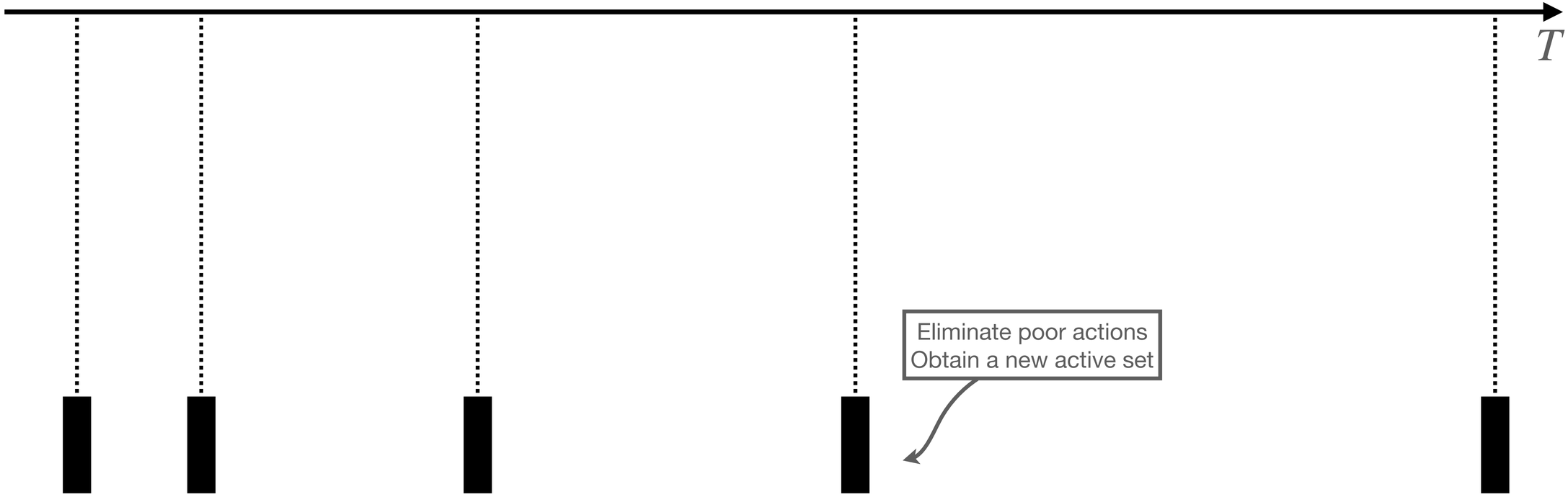
Reminder: log order for SE kernel

Our Approach

Policy maximizes expected user satisfaction

Phase-elimination

- Doubling phase length
- Eliminate poor actions/arms at end of each phase
- Can be easily extended to infinite domain setting



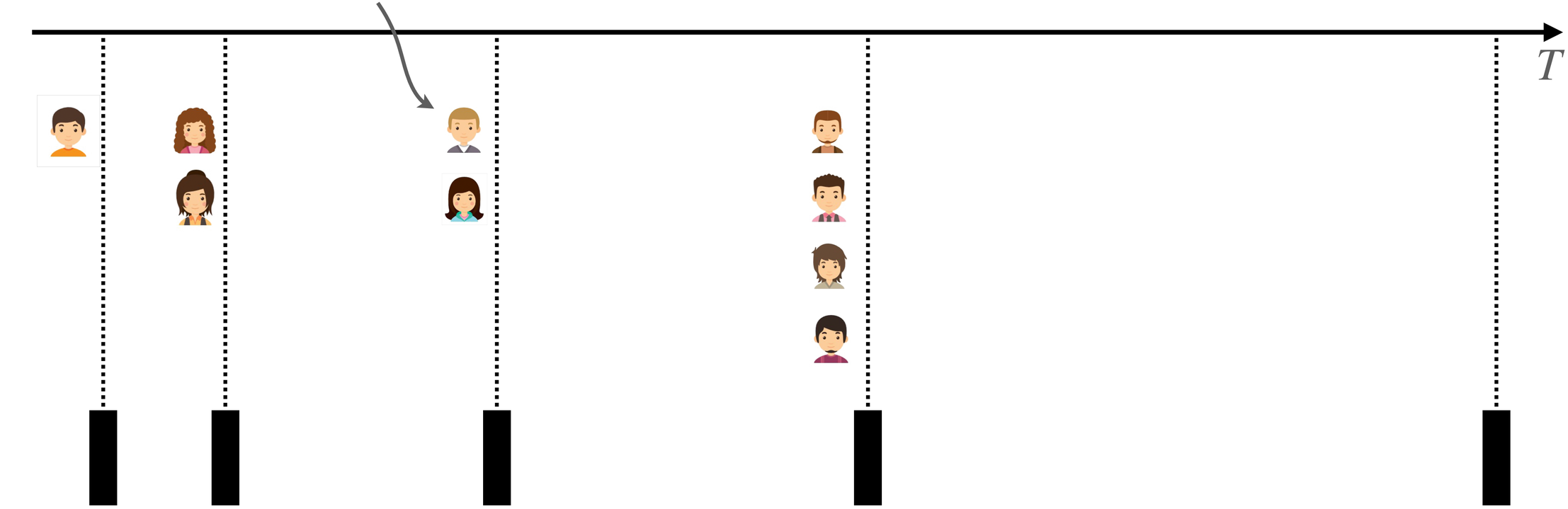
Our Approach

Policy maximizes expected user satisfaction

User Sampling

- Each phase samples a new set of users
- Those users are fixed during the phase
- Each phase l, $|U_l| = 2^{\alpha l}$, $\alpha \in (0,1]$

Need new users, otherwise the feedback is purely biased

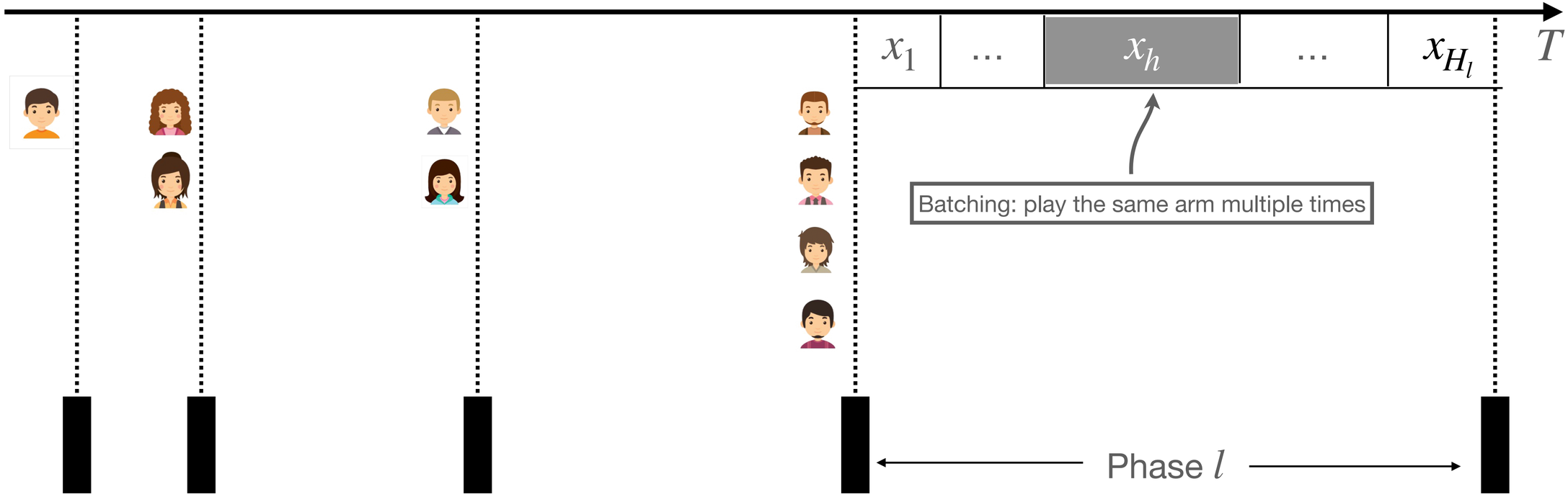


Our Approach

Policy maximizes expected user satisfaction

Action Selection

- Rarely switching via batching

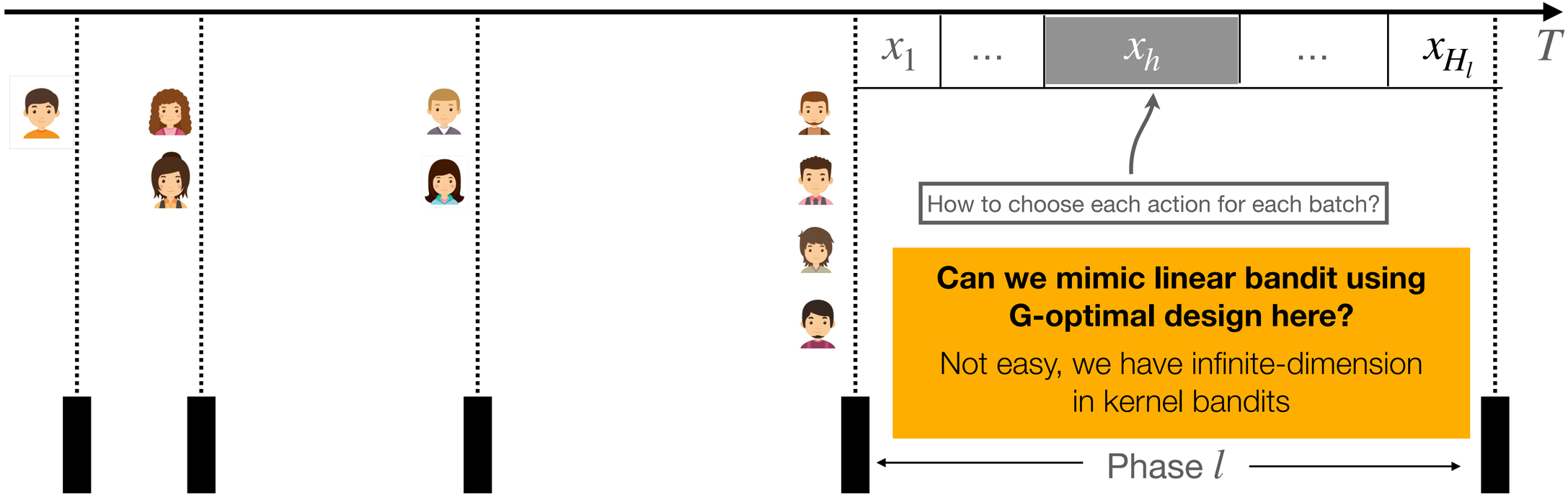


Our Approach

Policy maximizes expected user satisfaction

Action Selection

- Rarely switching via batching
- **Maximum variance reduction**

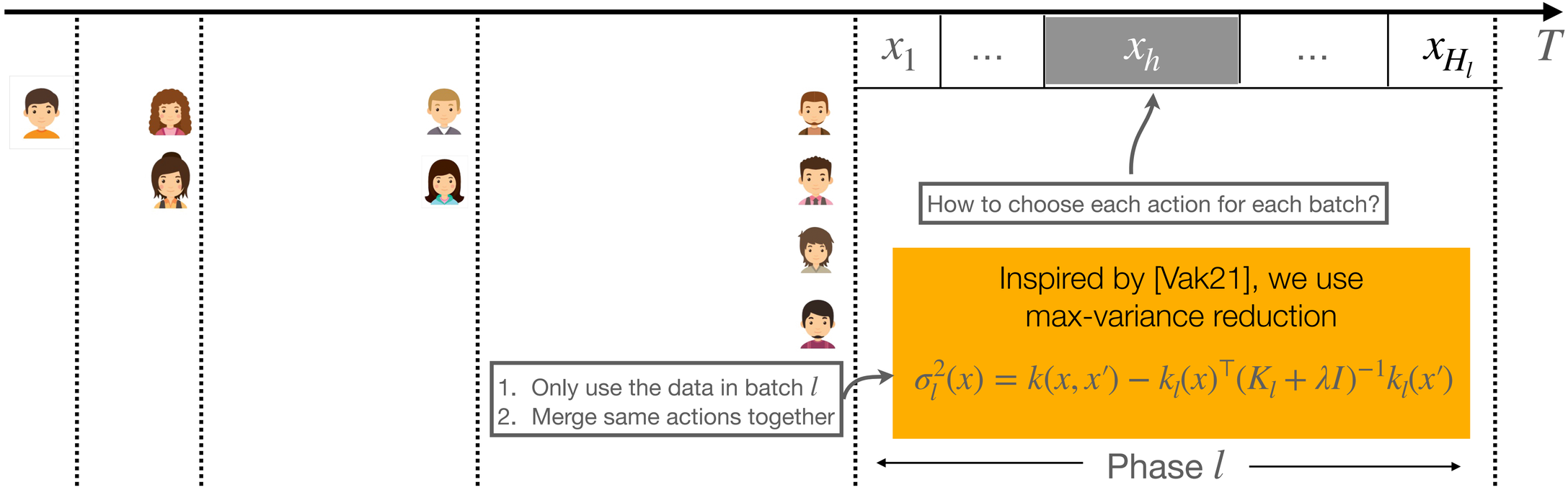


Our Approach

Policy maximizes expected user satisfaction

Action Selection

- Rarely switching via batching
- Max-variance reduction principle

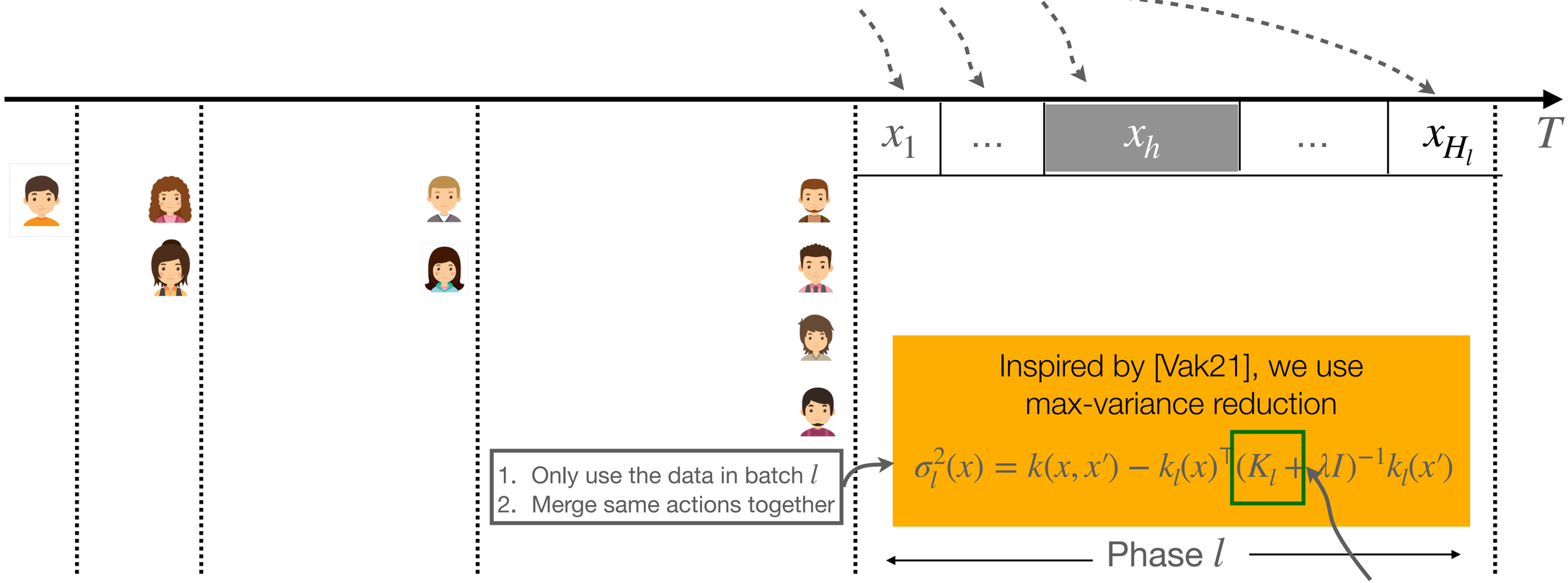


Our Approach

Action Selection

- Rarely switching via batching
- Max-variance reduction principle

Policy maximizes expected user satisfaction



1. Only use the data in batch l
2. Merge same actions together

Inspired by [Vak21], we use max-variance reduction

$$\sigma_l^2(x) = k(x, x') - k_l(x)^\top (K_l + \lambda I)^{-1} k_l(x')$$

Phase l

Reduce dim to H_l, which is $O(\gamma_T)$

Our Approach

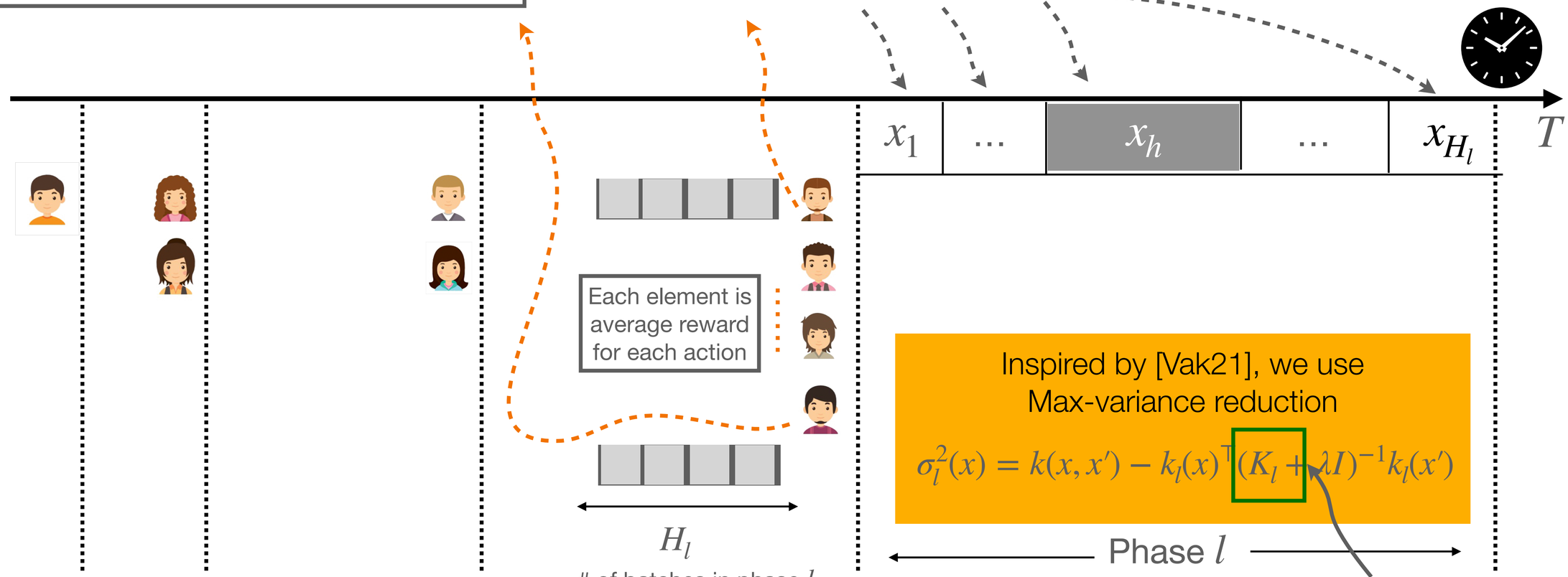
One Stone, Two Birds

- Reduce computation complexity
- Control the communication cost

Action Selection

- Rarely switching via batching
- Max-variance reduction principle

Policy maximizes expected user satisfaction

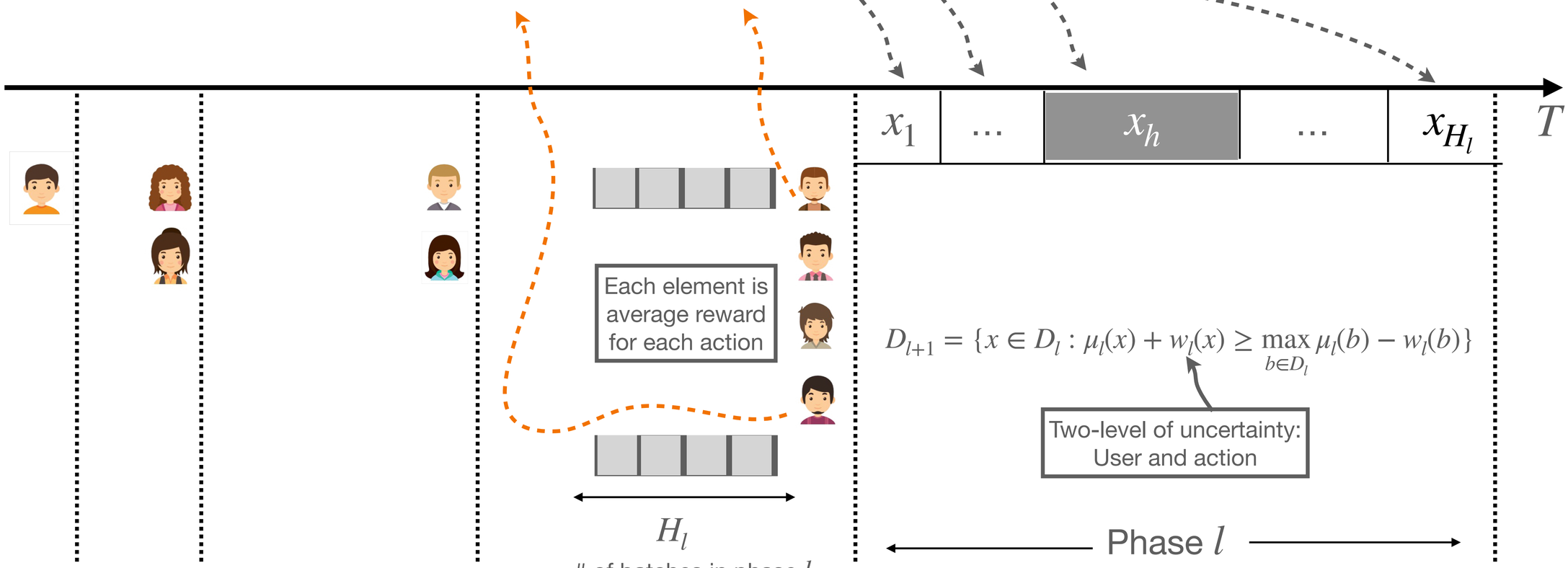


Our Approach

Arm Elimination

- Standard UCB and LCB comparison
- With proper confidence bonus

Policy maximizes expected user satisfaction



Theoretical Results

Performance Guarantees

Non-private version

Intuition&Fact Box

- GP-UCB achieves $\tilde{O}(\gamma_T\sqrt{T})$ regret bound
- γ_T — kernel-dependent, intrinsic dimensionality
  - Linear kernel: $O(d\log T)$, recover linear bandits
  - Gaussian kernel (SE): $\log^{d+1}(T)$
- Time complexity $O(T^3)$ for GP-UCB

Theorem 1

There exists proper parameter choices of our algorithm such that it enjoys

1. **Regret** — $\tilde{O}(T^{1-\alpha/2}+\sqrt{\gamma_T T})$ — thanks to phase-elimination
2. **Total number of users** — $O(T^\alpha)$ — thanks to user reuse
3. **Computation complexity** — $O(\gamma_T T^\alpha)$ — thanks to batching and merging
4. **Communication cost** — $O(\gamma_T T^\alpha)$ — thanks to batching and merging

Remark: comparisons with GP-UCB

1. **vs.** $\tilde{O}(\gamma_T\sqrt{T})$ **— sub-optimal**
2. **vs.** $O(T)$
3. **vs.** $O(T^3)$ **— poor scalability**
4. **vs.** $O(T)$

Compare with prior work

1. Most related work is our prior work on linear bandit [LZJ'22] — our results include it as a special case
2. Experimental design for action selection in kernel bandits [CJK21] — our selection is much simpler without additional estimator
3. Zero-order non-convex stochastic opt.[BG18] — our metric is total regret rather than convergence rate

Privacy Protection

Differential Privacy

Differential Privacy 101

Definition. If for any two neighboring datasets D and D', and any outcome E

$$\mathbb{P}(M(D) \in E) \leq e^{\epsilon}\mathbb{P}(M(D') \in E) + \delta$$

Then, M satisfies (ϵ, δ)-DP

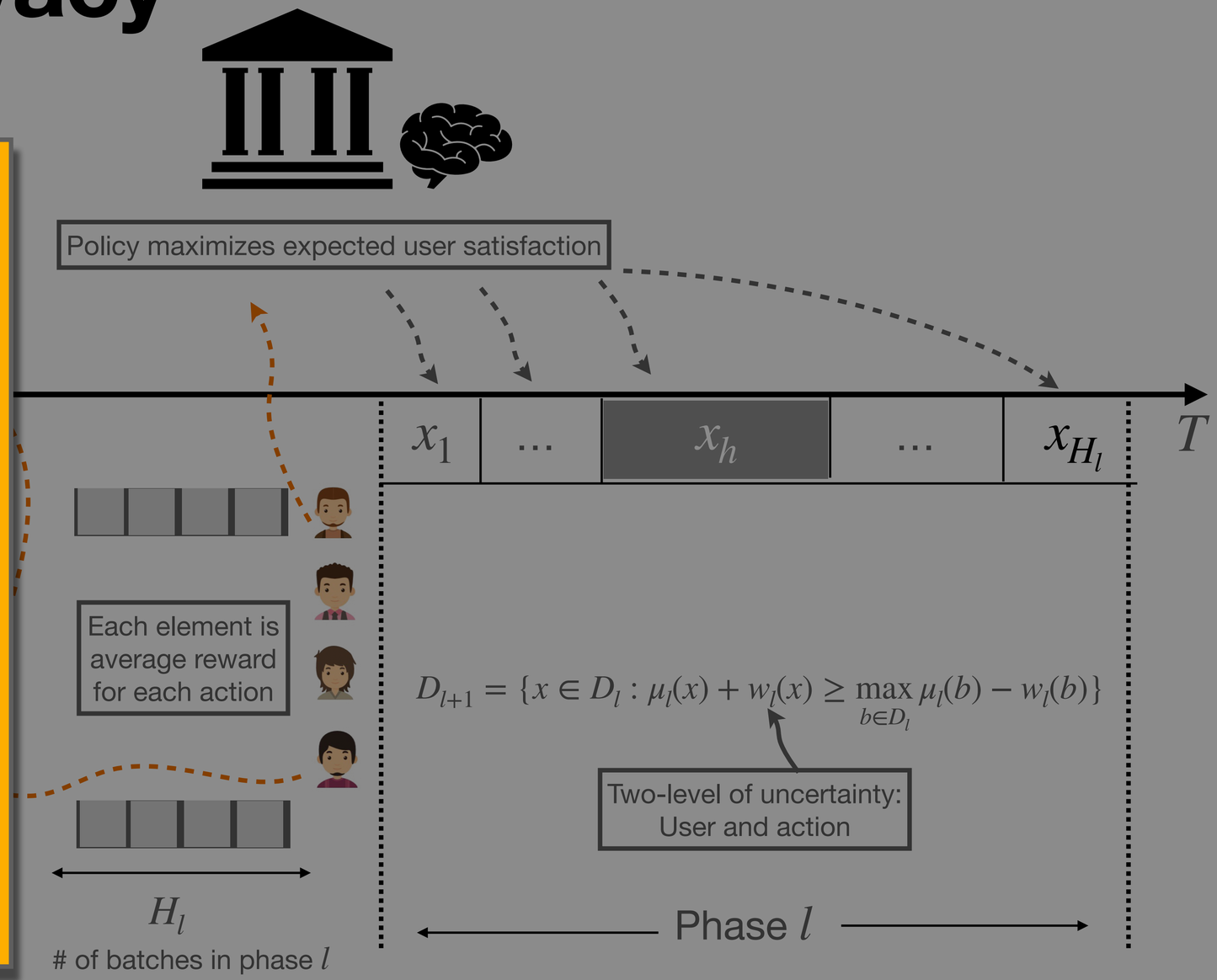
— DP means that outputs are "close" in probability on two neighboring datasets

Key components:

1. *What are the neighboring datasets?*
   — the identity for protection
2. *What are the outputs?*
   — the view of adversary

Key properties:

1. *Composition*, privacy loss adds up
2. *Post-processing,* immune to further processing if data is not touched



Differential Privacy

Central model

Differential Privacy 201

Gaussian mechanism for private sum of l_2 bounded vectors

i.e., $\widetilde{s}$ is the private sum of $\sum_{i=1}^{n} \gamma_i$ under (ϵ, δ)-DP

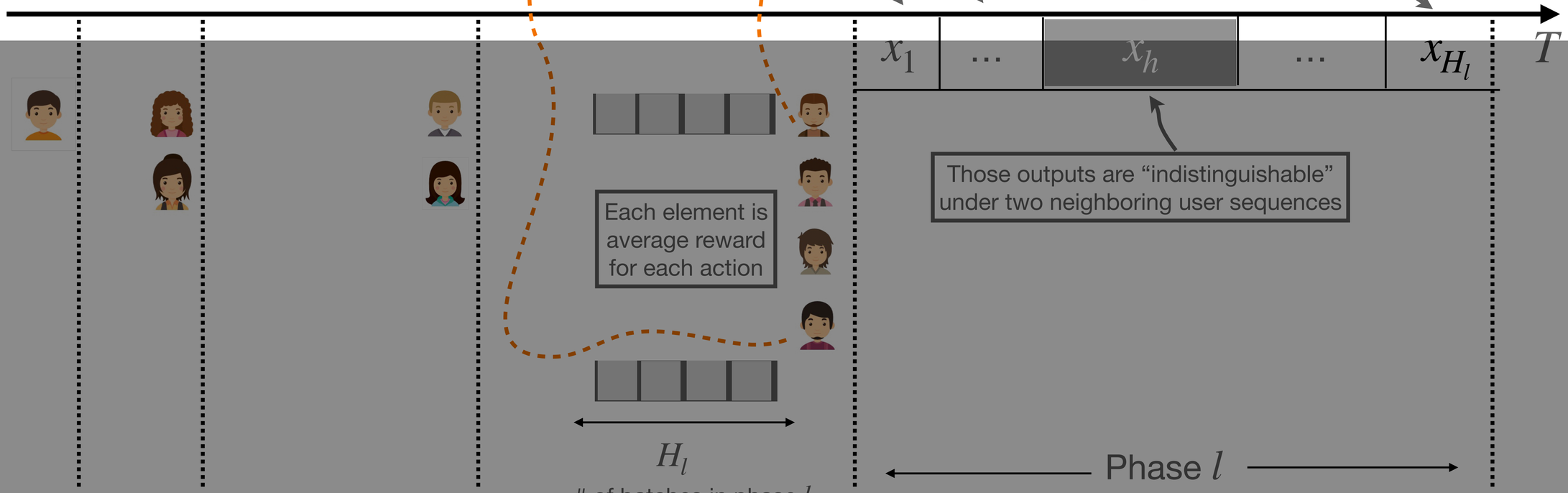
$$\widetilde{s} = \sum_{i=1}^{n} \gamma_i + \mathcal{N}(0, \sigma^2 I),\ \sigma^2 \approx \frac{L^2 \log(1/\delta)}{\epsilon^2}$$

Trusted

Neighboring Dataset & Output

- Two user sequences differing in one
- All action outputs of the central agent

Policy maximizes expected user satisfaction



Differential Privacy

Central model

Differential Privacy 201

Gaussian mechanism for private sum of l_2 bounded vectors

i.e., $\widetilde{s}$ is the private sum of $\sum_{i=1}^{n} \gamma_i$ under (ϵ, δ)-DP

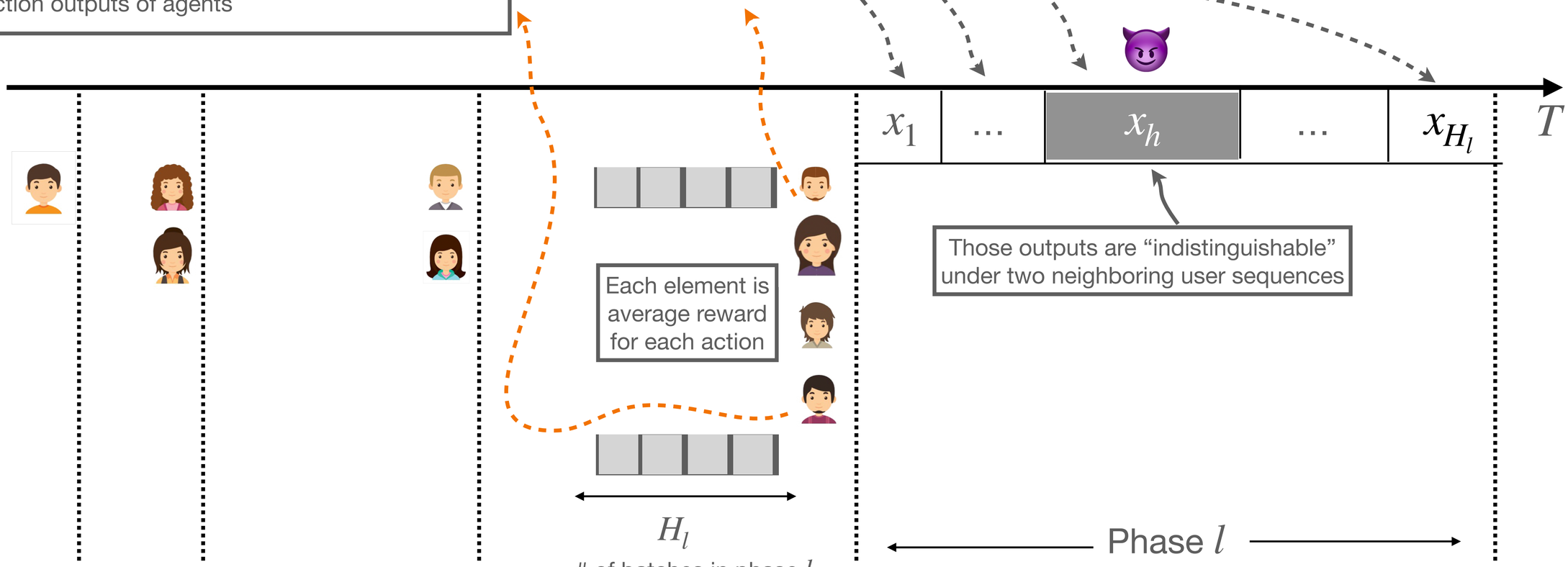
$$\widetilde{s} = \sum_{i=1}^{n} \gamma_i + \mathcal{N}(0, \sigma^2 I),\ \sigma^2 \approx \frac{L^2 \log(1/\delta)}{\epsilon^2}$$

Trusted

Neighboring Dataset & Output

- Two user sequences differing in one
- All action outputs of agents

Policy maximizes expected user satisfaction



Differential Privacy

Local model

Differential Privacy 101

Post-processing, immune to further processing if data is not touched

— local implies central model

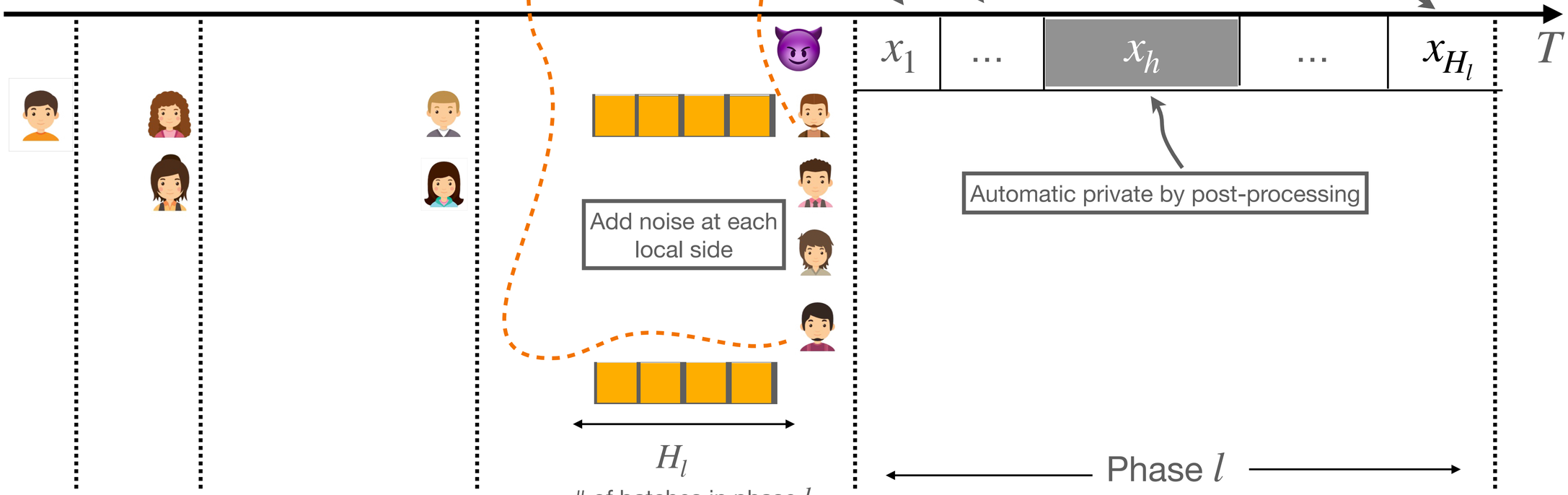
Untrusted

Noise will be too much :(

Neighboring Dataset & Output

- Each user's own output neighboring vector
- Local randomizer's output

Policy maximizes expected user satisfaction



Differential Privacy

Shuffle model

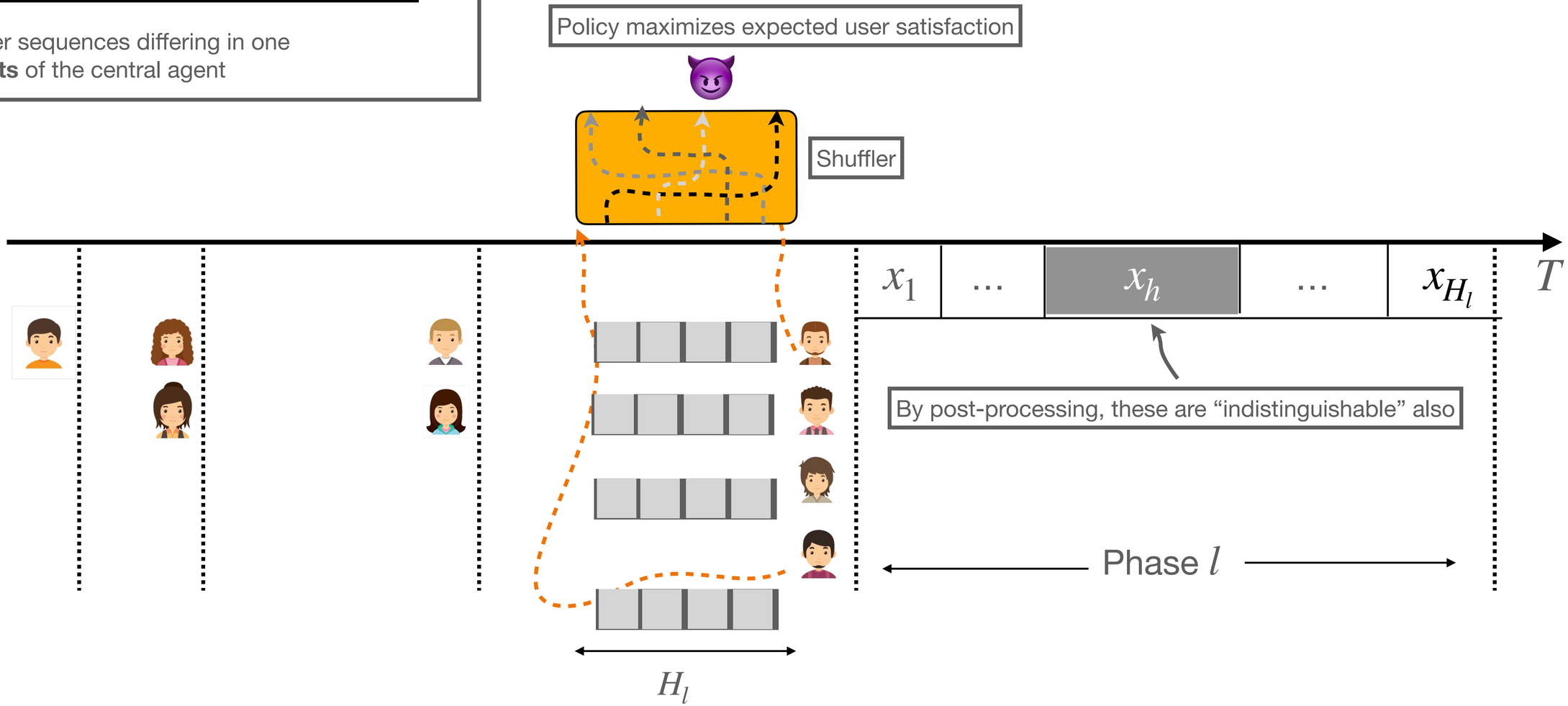
Differential Privacy 301

- Shuffle DP leverages additional randomness in the shuffler to amplify privacy
- Same untrusted agent as local model
- Huge improvement of utility

Untrusted

Neighboring Dataset & Output

- Two user sequences differing in one
- All **inputs** of the central agent



Performance Guarantees

private version

Recall

- Our non-private regret is $\tilde{O}(T^{1-\alpha/2} + \sqrt{\gamma_T T})$
- One stone, two birds
  - Computation
  - Communication
- **Gaussian noise depends on the vector dimension**

Theorem

There exists proper parameter choices of our algorithm such that it is (ϵ, δ)-DP with regret

1. **Central model** — $\tilde{O}\left(T^{1-\alpha/2} + \frac{\log(1/\delta)\gamma_T T^{1-\alpha}}{\epsilon}\right)$
2. **Local model** — $\tilde{O}\left(T^{1-\alpha/2} + \frac{\log(1/\delta)\gamma_T T^{1-\alpha/2}}{\epsilon}\right)$
3. **Shuffle model** — $\tilde{O}\left(T^{1-\alpha/2} + \frac{\log^{3/2}(1/\delta)\gamma_T T^{1-\alpha}}{\epsilon}\right)$

Privacy "for-free"

- The additional privacy cost is lower order term
- Thanks to our **batching** so that dim is controlled
- This shares similar spirit as **DP-SCO**
  - Central and shuffler model: $\tilde{O}\left(\frac{1}{\sqrt{n}} + \frac{1}{n\epsilon}\right)$

One Stone, Three Birds

Our batching (lazy update) and merging enable us to

1. Achieve the state-of-the-art time complexity —even for standard BO setting
2. Keep our communication cost in control
3. Keep our privacy noise in control

Thank you!