UniGe | MaLGa

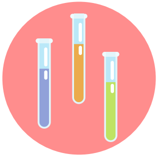
Near-linear Time Gaussian Process Optimization with Adaptive Batching and Resparsification

D. Calandriello* [1], **L. Carratino*** [2], A. Lazaric [3], M. Valko [1], L. Rosasco [2,4]

* equal contribution. [1] DeepMind, [2] MaLGa - UniGe, [3] Facebook, [4] MIT - IIT

Bayesian/Bandit Optimization

Set of candidates $\mathcal{A}$



Bayesian/Bandit Optimization

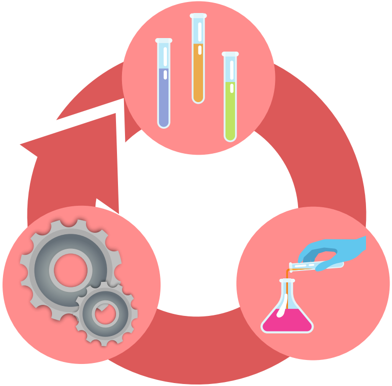
Set of candidates $\mathcal{A}$

$\texttt{for}\ t = 1, \ldots, T$:

(1) Select candidate

(2) Receive noisy feedback

(3) Update model



Bayesian/Bandit Optimization

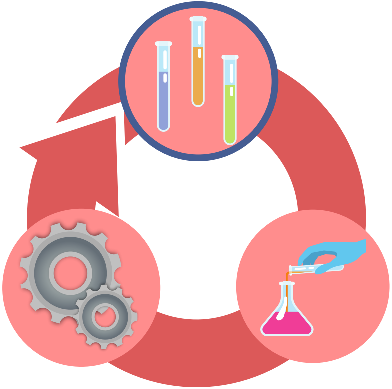
Set of candidates $\mathcal{A}$

`for` $t = 1, \ldots, T$:

(1) Select candidate

(2) Receive noisy feedback

(3) Update model



Bayesian/Bandit Optimization

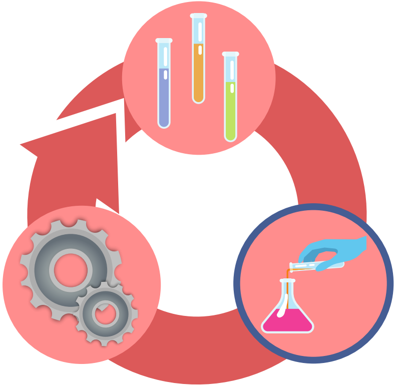
Set of candidates $\mathcal{A}$

$\texttt{for}$ $t = 1, \ldots, T$:

$\texttt{(1)}$ Select candidate

$\texttt{(2)}$ Receive noisy feedback

$\texttt{(3)}$ Update model



Bayesian/Bandit Optimization

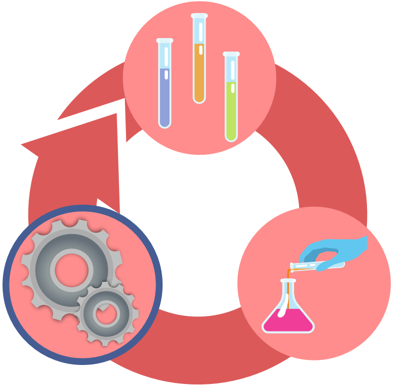
Set of candidates $\mathcal{A}$

$\texttt{for}$ $t = 1, \ldots, T$:

(1) Select candidate

(2) Receive noisy feedback

(3) Update model



UniGe | MaLGa

Bayesian/Bandit Optimization

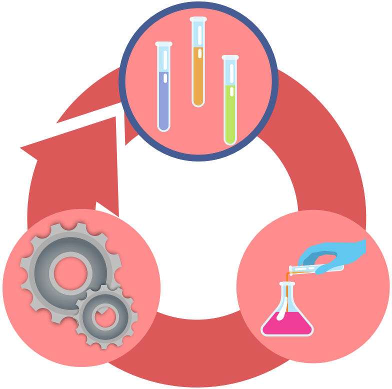
Set of candidates $\mathcal{A}$

`for` $t = 1, \ldots, T$:

(1) Select candidate

(2) Receive noisy feedback

(3) Update model



UniGe | MaLGa

Bayesian/Bandit Optimization

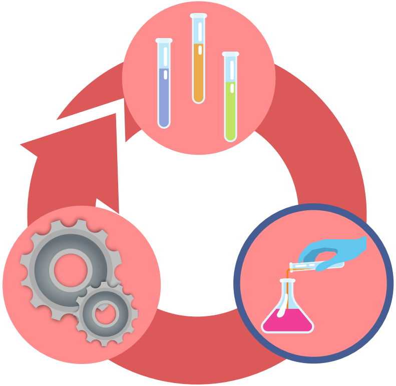
Set of candidates $\mathcal{A}$

$\texttt{for}\ t = 1, \ldots, T$:

(1) Select candidate

(2) Receive noisy feedback

(3) Update model



Bayesian/Bandit Optimization

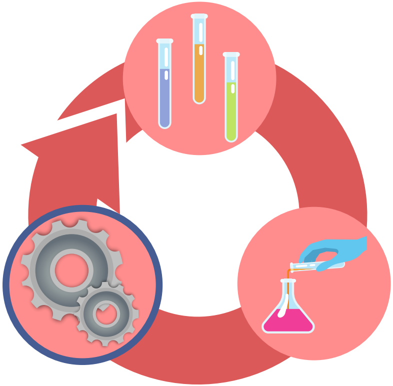
Set of candidates $\mathcal{A}$

$\texttt{for}\ t = 1, \ldots, T$:

(1) Select candidate

(2) Receive noisy feedback

(3) Update model



Bayesian/Bandit Optimization

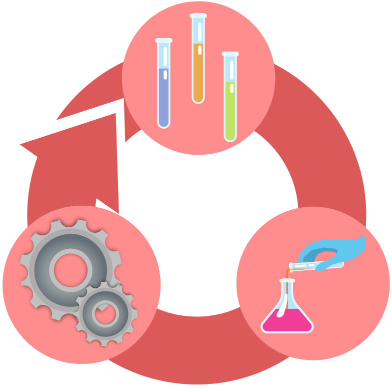
Set of candidates $\mathcal{A}$

`for` $t = 1, \ldots, T$:

(1) Select candidate

(2) Receive noisy feedback

(3) Update model



UniGe | MaLGa

Bayesian/Bandit Optimization

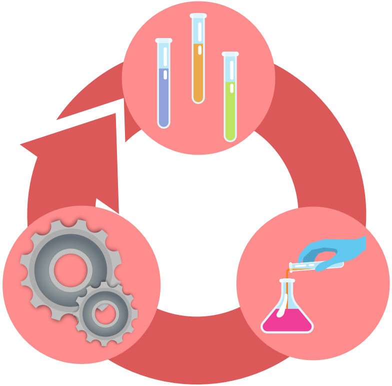
Set of candidates $\mathcal{A} = \{x_1, \ldots, x_A\} \subset \mathbb{R}^d$,

$\texttt{for}\ t = 1, \ldots, T$:

(1) Select candidate

(2) Receive noisy feedback

(3) Update model



Bayesian/Bandit Optimization

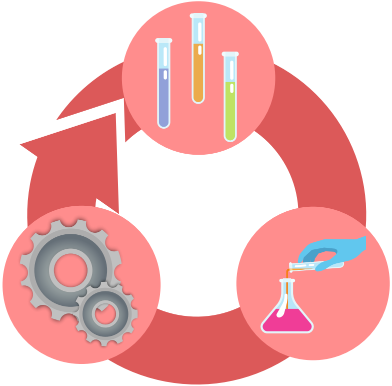
Set of candidates $\mathcal{A} = \{x_1, \ldots, x_A\} \subset \mathbb{R}^d$,
unknown reward function $f : \mathcal{A} \to \mathbb{R}$

`for` $t = 1, \ldots, T$:

(1) Select candidate

(2) Receive noisy feedback

(3) Update model

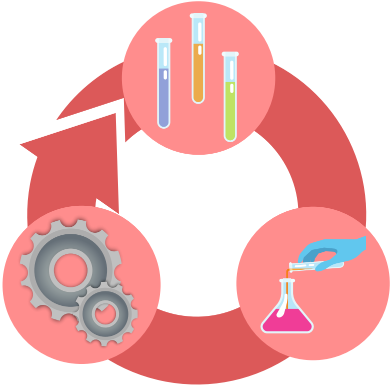


Bayesian/Bandit Optimization

Set of candidates $\mathcal{A} = \{x_1, \ldots, x_A\} \subset \mathbb{R}^d$,
unknown reward function $f : \mathcal{A} \to \mathbb{R}$

`for` $t = 1, \ldots, T$:

(1) Select candidate x_t using model u_t (ideally $u_t \approx f$)
(2) Receive noisy feedback
(3) Update model



UniGe | MaLGa

Bayesian/Bandit Optimization

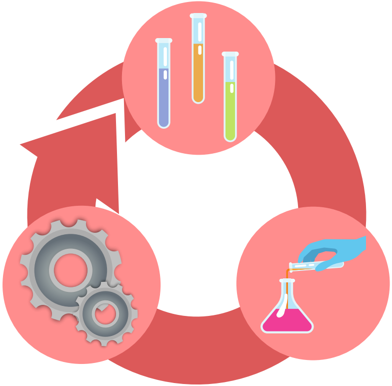
Set of candidates $\mathcal{A} = \{x_1, \ldots, x_A\} \subset \mathbb{R}^d$,
unknown reward function $f : \mathcal{A} \to \mathbb{R}$

$\texttt{for}$ $t = 1, \ldots, T$:

(1) Select candidate x_t using model u_t (ideally $u_t \approx f$)

(2) Receive noisy feedback $y_t = f(x_t) + \eta_t$

(3) Update model



UniGe | MaLGa

Bayesian/Bandit Optimization

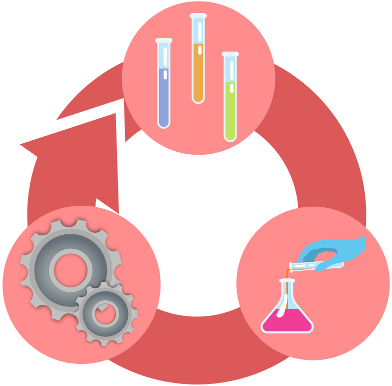
Set of candidates $\mathcal{A} = \{x_1, \dots, x_A\} \subset \mathbb{R}^d$,
unknown reward function $f : \mathcal{A} \to \mathbb{R}$

for $t = 1, \dots, T$:

(1) Select candidate x_t using model u_t (ideally $u_t \approx f$)

(2) Receive noisy feedback $y_t = f(x_t) + \eta_t$

(3) Update model u_t



Bayesian/Bandit Optimization

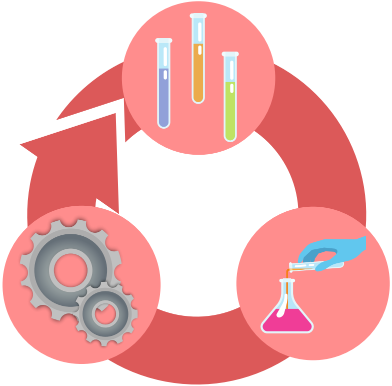
Set of candidates $\mathcal{A} = \{x_1, \ldots, x_A\} \subset \mathbb{R}^d$,
unknown reward function $f : \mathcal{A} \to \mathbb{R}$

for $t = 1, \ldots, T$:

(1) Select candidate x_t using model u_t (ideally $u_t \approx f$)

(2) Receive noisy feedback $y_t = f(x_t) + \eta_t$

(3) Update model u_t



Performance measure: cumulative regret w.r.t. best x_*

$$R_T = \sum_{t=1}^{T} f(x_*) - f(x_t).$$

Bayesian/Bandit Optimization

Set of candidates $\mathcal{A} = \{x_1, \ldots, x_A\} \subset \mathbb{R}^d$,
unknown reward function $f : \mathcal{A} \to \mathbb{R}$

for $t = 1, \ldots, T$:

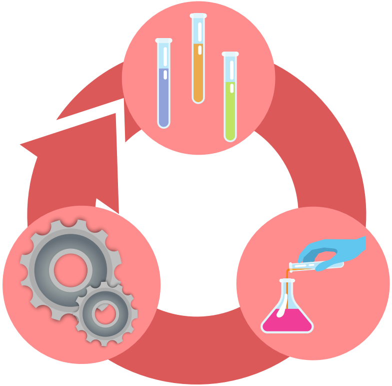
(1) Select candidate x_t using model u_t (ideally $u_t \approx f$)

(2) Receive noisy feedback $y_t = f(x_t) + \eta_t$

(3) Update model u_t

Performance measure: cumulative regret w.r.t. best x_*

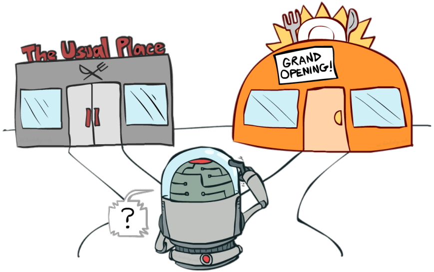
$$R_T = \sum_{t=1}^{T} f(x_*) - f(x_t).$$



💡 Use Gaussian process/kernelized Bandit to model f

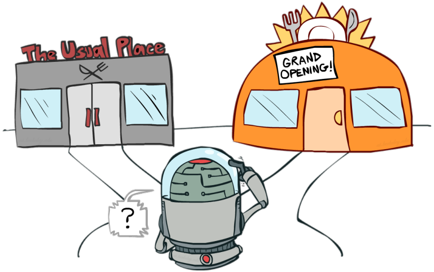
Gaussian Process Optimization

🙂 Well studied: exploration vs exploitation → no-regret (low error)🙂



Gaussian Process Optimization

🙂 Well studied: exploration vs exploitation → no-regret (low error)🙂

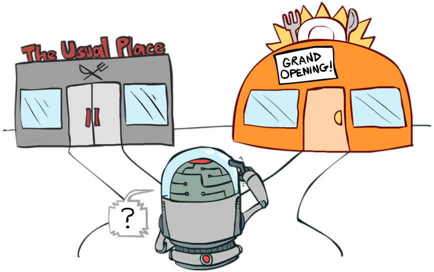


🤔 performance vs scalability ? 🤔

Image from Berkeley's CS 188

Gaussian Process Optimization

🙂 Well studied: exploration vs exploitation → no-regret (low error) 🙂



🤔 performance vs scalability ? 🤔

🙂 Batch BKB: no-regret and scalable 🙂

UniGe | MaLGa

Image from Berkeley's CS 188

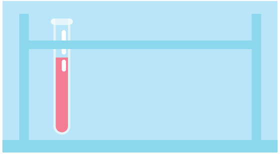
Why Scalable GP Optimization is Hard

Experimental
scalability

Computational
scalability

Why Scalable GP Optimization is Hard

Experimental scalability



sequential

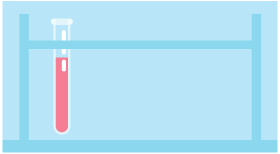
vs

batch

Computational scalability

Why Scalable GP Optimization is Hard

Experimental scalability



sequential

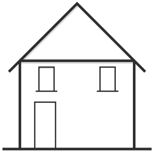
vs

batch

Computational scalability

exact GP

vs



approximate GP

UniGe | MaLGa

Why Scalable GP Optimization is Hard

Experimental scalability

sequential

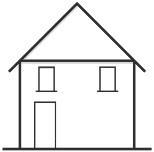
vs

batch

Computational scalability

exact GP

vs

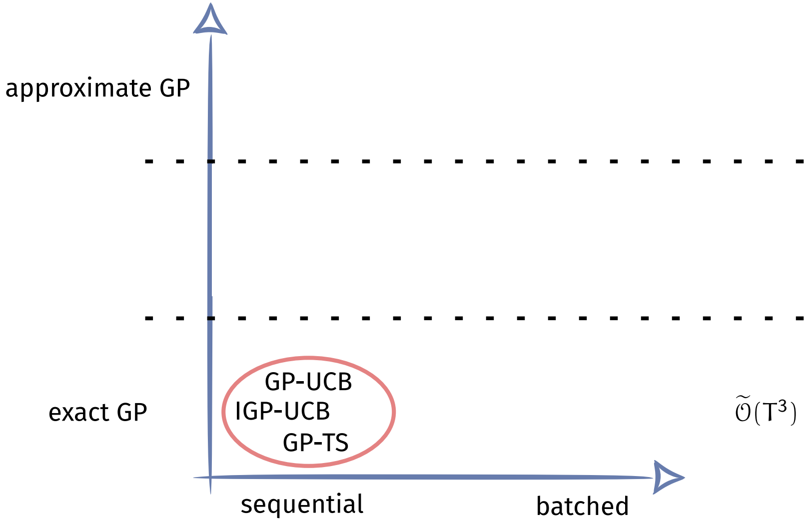


approximate GP

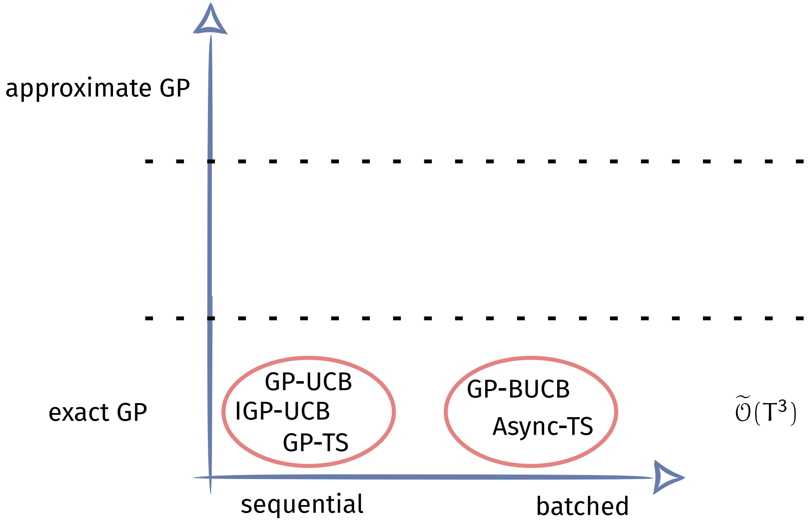
🙁 Batching and approximations increase regret 🙁

UniGe | MaLGa

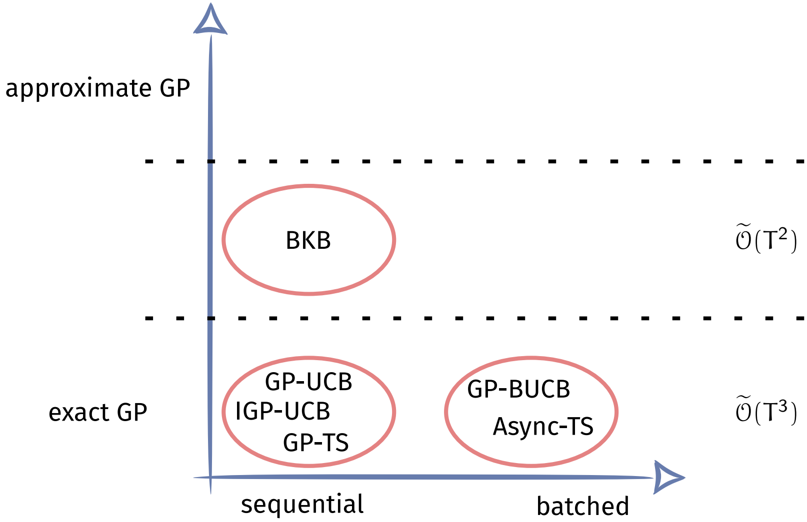
Landscape of No-Regret GP Optimization



Landscape of No-Regret GP Optimization

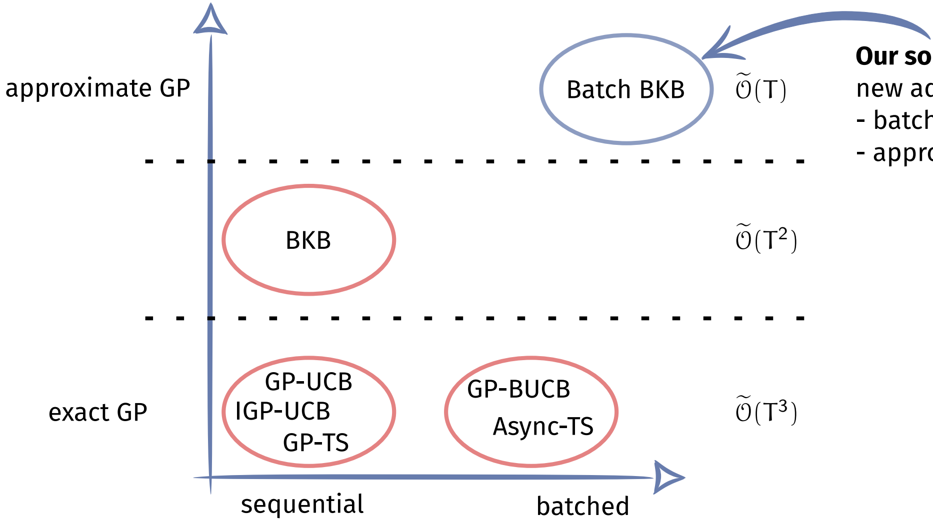


Landscape of No-Regret GP Optimization



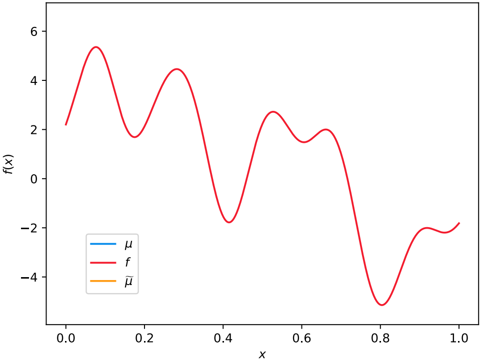
UniGe | MaLGa

Landscape of No-Regret GP Optimization



Our solution:
new adaptive schedule for
- batch-size
- approximation updates

Choosing good candidates with GP-UCB



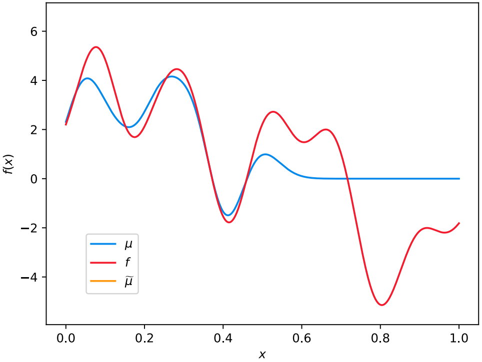
UniGe | MaLGa

Choosing good candidates with GP-UCB

$$X_t = \{x_1, \dots, x_t\}, Y_t = \{y_1, \dots, y_t\}$$

Exact GP-UCB:

$$u_t(\cdot) = \mu(\cdot \mid X_t, Y_t)$$

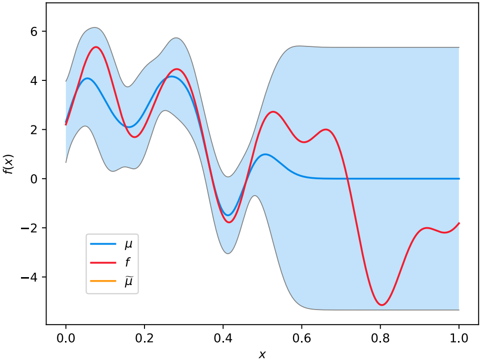


Choosing good candidates with GP-UCB

$$X_t = \{x_1, \ldots, x_t\}, Y_t = \{y_1, \ldots, y_t\}$$

Exact GP-UCB:

$$u_t(\cdot) = \mu(\cdot \mid X_t, Y_t) + \beta_t \sigma(\cdot \mid X_t)$$



UniGe | MaLGa

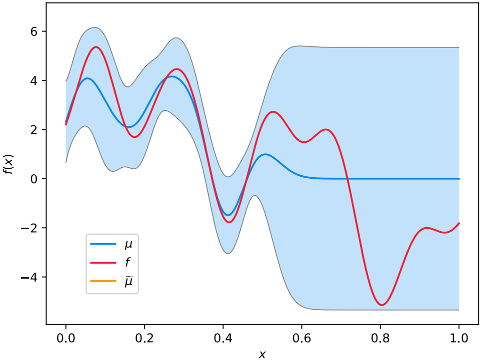
Choosing good candidates with GP-UCB

$$X_t = \{x_1, \dots, x_t\}, Y_t = \{y_1, \dots, y_t\}$$

Exact GP-UCB:

$$u_t(\cdot) = \mu(\cdot \,|\, X_t, Y_t) + \beta_t \sigma(\cdot \,|\, X_t)$$

[Sri+10]: u_t valid UCB.



Choosing good candidates with GP-UCB

$X_t = \{x_1, \dots, x_t\}, Y_t = \{y_1, \dots, y_t\}$

Exact GP-UCB:

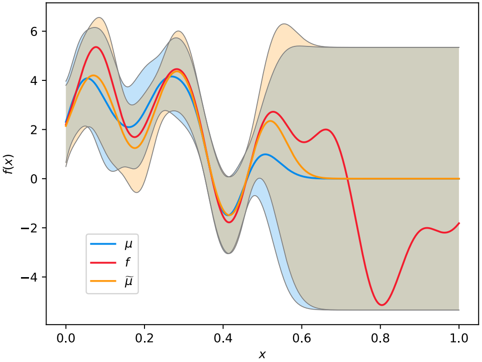
$$u_t(\cdot) = \mu(\cdot \mid X_t, Y_t) + \beta_t \sigma(\cdot \mid X_t)$$

[Sri+10]: u_t valid UCB.

Sparse GP-UCB:

$$\widetilde{u}_t(\cdot) = \widetilde{\mu}(\cdot \mid X_t, Y_t, \mathcal{D}_t) + \widetilde{\beta}_t \widetilde{\sigma}(\cdot \mid X_t, \mathcal{D}_t)$$

with $\mathcal{D}_t \subset X_t$ inducing points



UniGe | MaLGa

Choosing good candidates with GP-UCB

$X_t = \{x_1, \dots, x_t\}, Y_t = \{y_1, \dots, y_t\}$

Exact GP-UCB:

$$u_t(\cdot) = \mu(\cdot \mid X_t, Y_t) + \beta_t \sigma(\cdot \mid X_t)$$

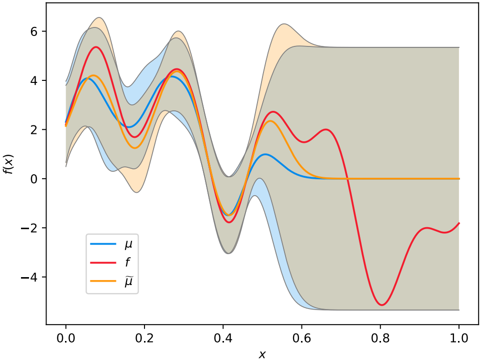
[Sri+10]: u_t valid UCB.

Sparse GP-UCB:

$$\widetilde{u}_t(\cdot) = \widetilde{\mu}(\cdot \mid X_t, Y_t, \mathcal{D}_t) + \widetilde{\beta}_t \widetilde{\sigma}(\cdot \mid X_t, \mathcal{D}_t)$$

with $\mathcal{D}_t \subset X_t$ inducing points

[Cal+19]: $\widetilde{u}_t$ valid UCB if $\mathcal{D}_t$ updated at every t.



Performance vs Scalability

Better performance: collect more feedback, update inducing points (resparsify)

$$\widetilde{u}_t(\,\cdot\,) = \widetilde{\mu}(\,\cdot\,|\,X_t, Y_t, \mathcal{D}_t) + \widetilde{\beta}_t \widetilde{\sigma}\,(\,\cdot\,|\,X_t, \mathcal{D}_t)$$

UniGe | MaLGa

Performance vs Scalability

Better performance: collect more feedback, update inducing points (resparsify)

$$\widetilde{u}_t(\cdot) = \widetilde{\mu}(\cdot \,|\, X_t, Y_t, \mathcal{D}_t) + \widetilde{\beta}_t \widetilde{\sigma}(\cdot \,|\, X_t, \mathcal{D}_t)$$

Worse scalability: experimental cost, resparsification cost

Performance vs Scalability

Better performance: collect more feedback, update inducing points (resparsify)

$$\widetilde{u}_t(\cdot) = \widetilde{\mu}(\cdot|X_t, Y_t, \mathcal{D}_t) + \widetilde{\beta}_t\widetilde{\sigma}(\cdot|X_t, \mathcal{D}_t)$$

Worse scalability: experimental cost, resparsification cost

Improve scalability: batching feedback (GP-BUCB), batching resparsification ?

UniGe | MaLGa

Delayed Resparsification

New adaptive batching rule

$$\texttt{no-resparsify until} \sum_{i \in \text{Batch}} \widetilde{\sigma}^2(x_i) \geqslant 1$$

UniGe | MaLGa

Delayed Resparsification

New adaptive batching rule

$$\texttt{no-resparsify until} \sum_{i \in \text{Batch}} \widetilde{\sigma}^2(x_i) \geqslant 1$$

"Not too big" Lemma: valid UCB

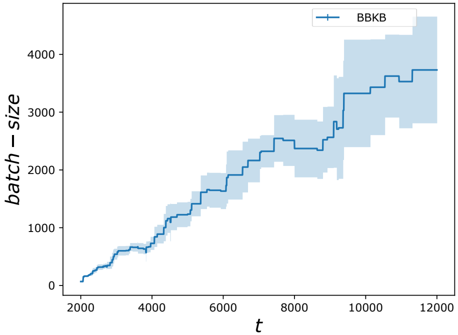
Delayed Resparsification

New adaptive batching rule

$$\texttt{no-resparsify until} \sum_{i \in \text{Batch}} \widetilde{\sigma}^2(x_i) \geqslant 1$$

“Not too big” Lemma: valid UCB

“Not too small” Lemma: batch-size $= \Omega(t)$



Batch-BKB

Theorem

*With high probability **Batch-BKB** achieves no-regret with time complexity $O(Td_{eff}^2)$, where $d_{eff} \ll T$ is the effective dimension / degrees of freedom of the GP.*

Batch-BKB

Theorem

*With high probability **Batch-BKB** achieves no-regret with time complexity $O(Td_{eff}^2)$, where $d_{eff} \ll T$ is the effective dimension / degrees of freedom of the GP.*

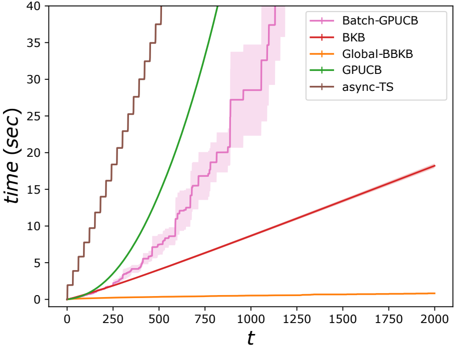
Comparisons:

- 🙂 Same regret of GP-UCB/IGP-UCB and better scalability (form $O(T^3)$ to $O(Td_{eff}^2)$)
- 🙂 Larger batches than GP-BUCB
- 🙂 Better regret and better scalability than async-TS

In practice: Scalability

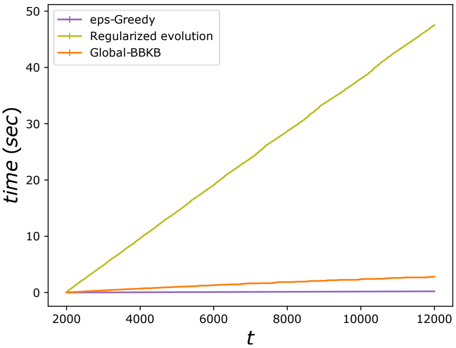
Cadata

$A = 20640, d = 8, T = 2000$



NAS-bench-101

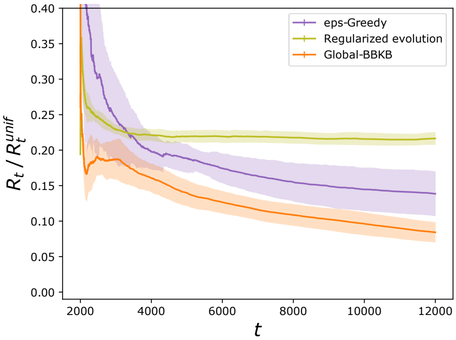
$A = 12416, d = 19, T = 12000$



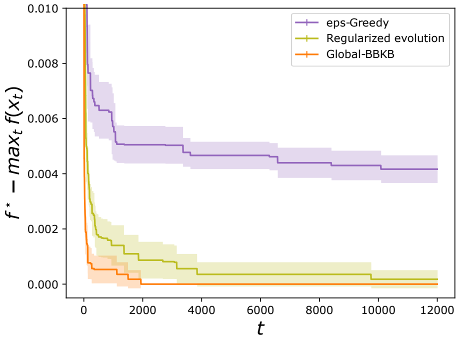
UniGe | MaLGa

In practice: Performance

NAS-bench-101
Regret / Regret uniform



NAS-bench-101
Simple Regret



UniGe | MaLGa

Thank you

UniGe | MaLGa