Scaling Gaussian Process Optimization by Evaluating a Few Unique Candidates Multiple Times

DeepMind

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

[1]DeepMind, [2]MalGa University of Genova, [3]Meta AI, [4]MIT/IIT

Our contribution in a nutshell

Traditional GP-Opt approaches achieve good explore/exploit balance, but suffer from high computational costs and poor use of parallel evaluations.

We propose a new GP-Opt approach that repeatedly evaluates the same candidate before switching. By carefully choosing the switching time convergence rate is provably preserved, and we can also guarantee

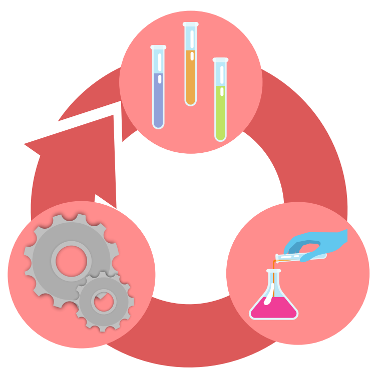
- computational savings, through non-trivial and exact GP compression,
- high evaluation parallellism, through very rare switchings.

As long as repeated evaluations are allowed by the task, our approach can be directly applied to most GP-Opt algorithms (e.g., GP-UCB, GP-EI) and maybe can help scale your favourite GP-Opt approach!

Gaussian Process Optimization (GP-Opt)

Given unknown noisy function f and a decision set $\mathcal{A}$ (e.g., $\mathcal{A} \subseteq \mathbb{R}^d$) at each step t the learner:

1. optimizes acquisition function u_t as a surrogate of f to selects candidate $\mathbf{x}_t = \arg\max_{\mathbf{x}\in\mathcal{A}} u_t(\mathbf{x})$;
2. evaluates $\mathbf{x}_t$ and receives feedback $y_t \triangleq f(\mathbf{x}_t) + \eta_t$;
3. improves u_t's approximation of f.

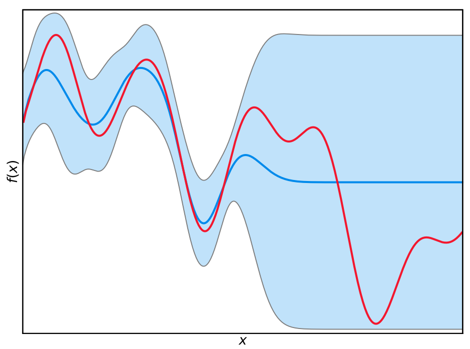


We measure convergence using cumulative regret $R_t = \sum_{s=1}^{t} \max_{\mathcal{A}} f(\mathbf{x}) - f(\mathbf{x}_s)$.

The posterior of the GP conditioned on evaluations $\mathbf{X}_t, \mathbf{y}_t$ is formulated as

$$\mu_t(\mathbf{x}_i) = \mathrm{k}(\mathbf{x}_i, \mathbf{X}_t)(\mathbf{K}_t + \lambda \mathbf{I})^{-1}\mathbf{K}_t\mathbf{y}_t,$$
$$\sigma_t^2(\mathbf{x}_i) = \mathrm{k}(\mathbf{x}_i, \mathbf{x}_i) - \mathrm{k}(\mathbf{x}_i, \mathbf{X}_t)(\mathbf{K}_t + \lambda \mathbf{I})^{-1}\mathrm{k}(\mathbf{X}_t, \mathbf{x}_i),$$

with $\mathbf{K}_t = \mathrm{k}(\mathbf{X}_t, \mathbf{X}_t)$. Using μ_t and σ_t we construct acquistion functions



$$u_t^{\text{GP-UCB}}(\mathbf{x}) = \mu_t(\mathbf{x}) + \beta_t^{\text{GP-UCB}}\sigma_t(\mathbf{x}),$$
$$u_t^{\text{GP-EI}}(\mathbf{x}) = \left(\beta_t^{\text{GP-EI}}\sigma_t(\mathbf{x}) \cdot \left[\left(\tfrac{z}{\beta_t^{\text{GP-EI}}}\right)\mathbf{CDF}_{\mathcal{N}}\left(\tfrac{z}{\beta_t^{\text{GP-EI}}}\right) + \mathbf{PDF}_{\mathcal{N}}\left(\tfrac{z}{\beta_t^{\text{GP-EI}}}\right)\right]\right),$$

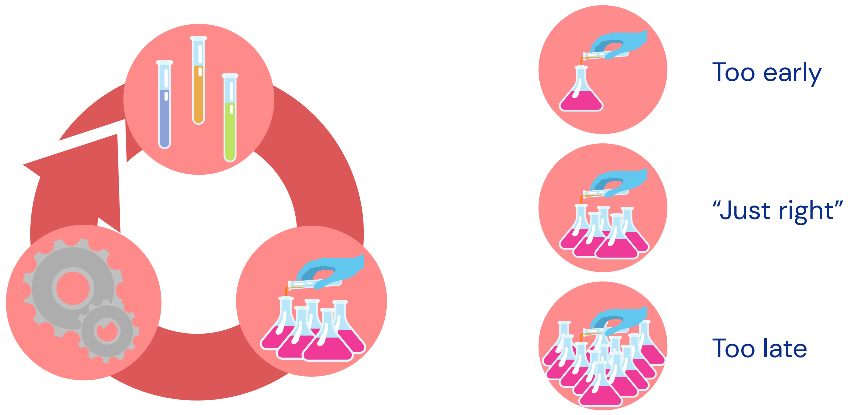
with $z = \frac{\mu_t(\mathbf{x}) - \max_{\mathbf{x}'} \mu_t(\mathbf{x}')}{\sigma_t(\mathbf{x})}$ and β_t is proportional to the GP information gain

$$\gamma(\mathbf{X}, \mathbf{y}) = \tfrac{1}{2}\log\det(\mathbf{I} + \lambda^{-2}\mathrm{k}(\mathbf{X}, \mathbf{X})).$$

The maximum information gain at step t is $\gamma_t = \max_{\mathbf{X}:|\mathbf{X}|=t} \gamma(\mathbf{X}, \mathbf{y})$.

One easy trick to scale GP-Optimization

Keep choosing and evaluating the same candidate but **not for too long!**



Mini-META rule to minimize switches

Keep choosing $\mathbf{x}_{t+1}$, and switch after $B_h = \lfloor (C^2 - 1)/\sigma_t^2(\mathbf{x}_{t+1}) \rfloor$ evaluations.

Strengths:
- Preserves convergence rate up to a C factor ("not too late" lemma)
- Switches at most $H \le \mathcal{O}((1 + 1/(C^2 - 1))\gamma_t)$ ("not too early" lemma)
  - Calls to u_t optimizer reduced from $\mathcal{O}(T)$ to $\mathcal{O}(H)$
  - GP posterior inference time reduced from $\mathcal{O}(T^2)$ to $\mathcal{O}(H^3)$
  - Overall computational complexity of $\mathcal{O}(T + H \cdot (|\mathcal{A}|H^2 + H^3))$
  - Unlocks experimental parallelism
- Easily applicable to popular GP-Opt algorithms (Mini-fied variants)
  - Mini-GP-UCB from GP-UCB [6], Mini-GP-EI from GP-EI [7]

Weaknesses:
- Not applicable when repeated choices are not possible
- Not very useful for noiseless scenarios

GP compression with few unique candidates

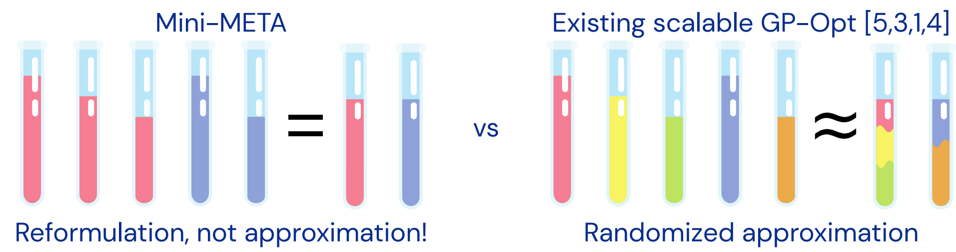
Low number of switches + selecting same candidate

⇓

few unique candidates in the GP

⇓

identical candidates can be aggregated for a lossless GP compression!



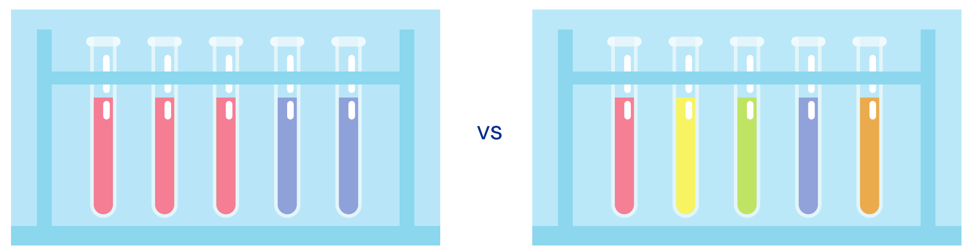
Simple to implement using diagonal $\mathbf{W}_H \in \mathbb{R}^{H\times H} = \mathrm{Diag}(\{B_h\})$

$$\mu_{t_H}(\mathbf{x}) = \mathrm{k}(\mathbf{x}, \mathbf{X}_H)(\mathbf{K}_H + \lambda \mathbf{W}_H^{-1})^{-1}\mathbf{y}_H,$$
$$\sigma_{t_H}^2(\mathbf{x}) = \mathrm{k}(\mathbf{x}, \mathbf{x}) - \mathrm{k}(\mathbf{x}, \mathbf{X}_H)(\mathbf{K}_H + \lambda \mathbf{W}_H^{-1})^{-1}\mathrm{k}(\mathbf{X}_H, \mathbf{x})$$

where $\mathbf{K}_H = \mathrm{k}(\mathbf{X}_H, \mathbf{X}_H) \in \mathbb{R}^{H\times H}$, and $\mathbf{y}_H \in \mathbb{R}^H$ is such that $[\mathbf{y}_H]_h = \sum_{s=t_h}^{t_h+B_h} y_s$.

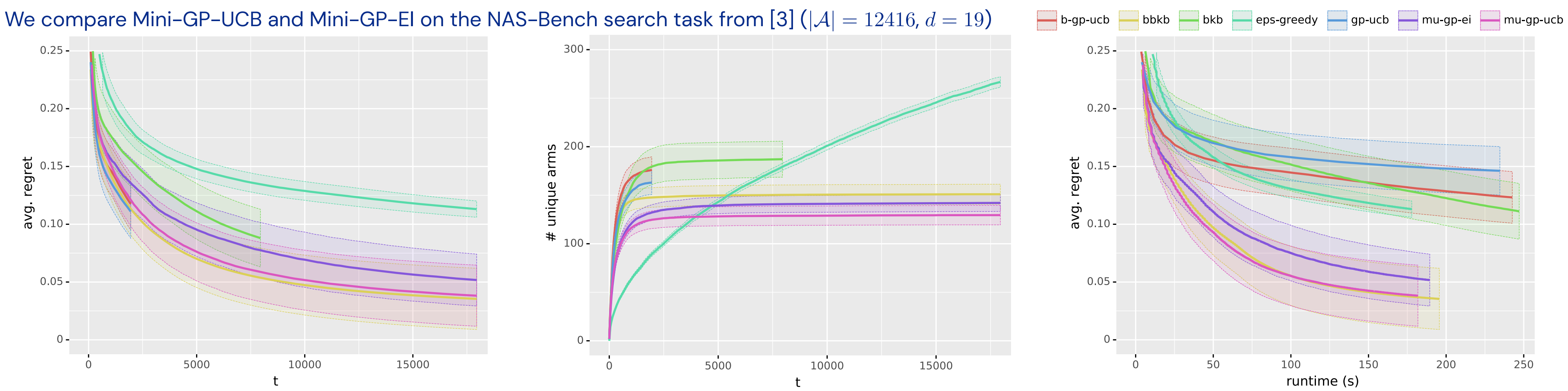
Special case of stratified sampling [2], inapplicable to GPs with i.i.d. samples

Reduced switching costs



Experiments

We compare Mini-GP-UCB and Mini-GP-EI on the NAS-Bench search task from [3] ($|\mathcal{A}| = 12416$, $d = 19$)



R_t/t against step t — # unique candidates against step t — R_T against wall-clock time (s)

[1] Maximilian Balandat et al. "Botorch: Programmable bayesian optimization in pytorch". arXiv:1910.06403 (2019) [2] Mickaël Binois et al. "Replication or exploration? Sequential design for stochastic simulation experiments". Technometrics 61.1 (2019) [3] Daniele Calandriello et al. "Near-linear Time Gaussian Process Optimization with Adaptive Batching and Resparsification". ICML (2020) [4] Jacob R Gardner et al. "Gpytorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration". NeurIPS (2018) [5] Mojmir Mutny et al. "Efficient High Dimensional Bayesian Optimization with Additivity and Quadrature Fourier Features". NeurIPS (2018) [6] Niranjan Srinivas et al. "Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design". ICML (2010) [7] Ziyu Wang et al. "Bayesian multi-scale optimistic optimization". Artificial Intelligence and Statistics (2014)