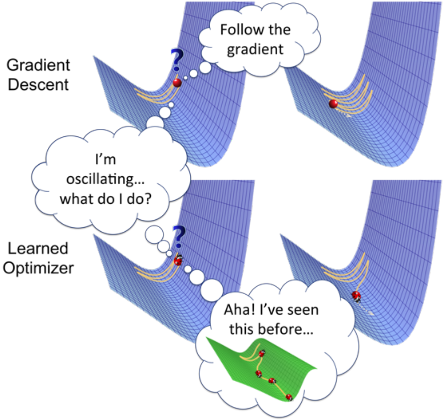
AutoML: Dynamic Configuration & Learning

Learning to Learn: Reinforcement Learning

Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer Joaquin Vanschoren

Learning to Optimize via Reinforcement Learning [Li and Malik. 2017]



Source: `https://bair.berkeley.edu/blog/2017/09/12/learning-to-optimize-with-rl/`

Learning to Optimize via Reinforcement Learning [Li and Malik. 2017]

Reinforcement Learning for Learning to Optimize

State current location, objective values and gradients evaluated at the current and past locations

Learning to Optimize via Reinforcement Learning [Li and Malik. 2017]

Reinforcement Learning for Learning to Optimize

State current location, objective values and gradients evaluated at the current and past locations

Action Step update $\Delta \mathbf{x}$

Learning to Optimize via Reinforcement Learning [Li and Malik. 2017]

Reinforcement Learning for Learning to Optimize

State current location, objective values and gradients evaluated at the current and past locations

Action Step update $\Delta \mathbf{x}$

Transition $\mathbf{x}^{(t)} \leftarrow \mathbf{x}^{(t-1)} + \Delta \mathbf{x}$

Learning to Optimize via Reinforcement Learning [Li and Malik. 2017]

Reinforcement Learning for Learning to Optimize

State current location, objective values and gradients evaluated at the current and past locations

Action Step update $\Delta\mathbf{x}$

Transition $\mathbf{x}^{(t)} \leftarrow \mathbf{x}^{(t-1)} + \Delta\mathbf{x}$

Cost/Reward Objective value at the current location

- Since the RL agent will optimize the cumulative cost, this is equivalent to L_{sum} [Chen et al. 2017] $(\gamma = 0)$
- encourages the policy to reach the minimum of the objective function as quickly as possible

Learning to Optimize via Reinforcement Learning [Li and Malik. 2017]

Reinforcement Learning for Learning to Optimize

State current location, objective values and gradients evaluated at the current and past locations

Action Step update $\Delta\mathbf{x}$

Transition $\mathbf{x}^{(t)} \leftarrow \mathbf{x}^{(t-1)} + \Delta\mathbf{x}$

Cost/Reward Objective value at the current location

- Since the RL agent will optimize the cumulative cost, this is equivalent to L_{sum} [Chen et al. 2017] $(\gamma = 0)$
- encourages the policy to reach the minimum of the objective function as quickly as possible

Policy DNN predicting μ_d of Gaussian (with constant variance σ^2) for dimension d; sample $\Delta\mathbf{x}_d \sim \mathcal{N}(\mu_d, \sigma^2)$

Learning to Optimize via Reinforcement Learning [Li and Malik. 2017]

Reinforcement Learning for Learning to Optimize

State current location, objective values and gradients evaluated at the current and past locations

Action Step update $\Delta \mathbf{x}$

Transition $\mathbf{x}^{(t)} \leftarrow \mathbf{x}^{(t-1)} + \Delta \mathbf{x}$

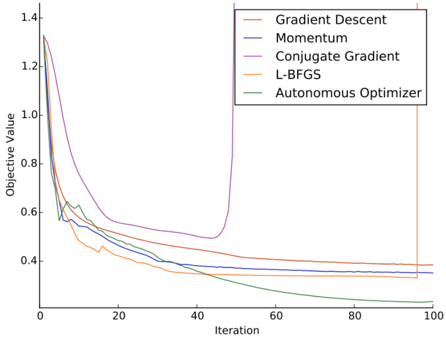
Cost/Reward Objective value at the current location

- Since the RL agent will optimize the cumulative cost, this is equivalent to L_{sum} [Chen et al. 2017] $(\gamma = 0)$
- encourages the policy to reach the minimum of the objective function as quickly as possible

Policy DNN predicting μ_d of Gaussian (with constant variance σ^2) for dimension d; sample $\Delta \mathbf{x}_d \sim \mathcal{N}(\mu_d, \sigma^2)$

Training Set randomly generated objective functions

Learning to Optimize via Reinforcement Learning Results [Li and Malik. 2017]



- 2-layer DNN with ReLUs
- Training datasets for training RL agent:
  four multivariate Gaussians and sampling 25 points from each
  - $\rightsquigarrow$ hard toy problem

Learning Acquisition Functions [Volpp et al. 2019]

- Instead of learning everything, it might be sufficient to learn hand-design heuristics

Learning Acquisition Functions [Volpp et al. 2019]

- Instead of learning everything, it might be sufficient to learn hand-design heuristics
- In Bayesian Optimization (BO), the most critical hand-design heuristic is the acquisition function
  - trade-off between exploitation and exploration
  - Depending on the problem at hand, you might need a different acquisition function

Learning Acquisition Functions [Volpp et al. 2019]

- Instead of learning everything, it might be sufficient to learn hand-design heuristics
- In Bayesian Optimization (BO), the most critical hand-design heuristic is the acquisition function
  - trade-off between exploitation and exploration
  - Depending on the problem at hand, you might need a different acquisition function
  - Choices:
    - probability of improvement (PI)
    - expected improvement (EI)
    - upper confidence bounds (UCB)
    - entropy search (ES) – quite expensive!
    - knowledge gradient (KG)
    - ...

Learning Acquisition Functions [Volpp et al. 2019]

- Instead of learning everything, it might be sufficient to learn hand-design heuristics
- In Bayesian Optimization (BO), the most critical hand-design heuristic is the acquisition function
  - trade-off between exploitation and exploration
  - Depending on the problem at hand, you might need a different acquisition function
  - Choices:
    - probability of improvement (PI)
    - expected improvement (EI)
    - upper confidence bounds (UCB)
    - entropy search (ES) – quite expensive!
    - knowledge gradient (KG)
    - ...
- Idea: Learn a *neural acquisition function* from data

$\rightsquigarrow$ Replace acquisition function

Bayesian Optimization: Algorithm

Algorithm 1 Bayesian Optimization (BO)

Input : Search Space $\mathcal{X}$, black box function f, acquisition function α, maximal number of function evaluations T

$\mathcal{D}^{(0)} \leftarrow$ initial_design($\mathcal{X}$);
for $t = 1, 2, \ldots T - |D_0|$ **do**
  $\hat{c} : \mathbf{x} \mapsto c(\mathbf{x}) \leftarrow$ fit predictive model on $\mathcal{D}^{(t-1)}$;
  select $\mathbf{x}^{(t)}$ by optimizing $\mathbf{x}^{(t)} \in \arg\max_{\mathbf{x} \in \mathcal{X}} \alpha(\mathbf{x}; \mathcal{D}^{(t-1)}, \hat{c})$;
  Query $y^{(t)} := f(\mathbf{x}^{(t)})$;
  Add observation to data $D^{(t)} := D^{(t-1)} \cup \{\langle \mathbf{x}^{(t)}, y^{(t)} \rangle\}$;
return *Best* $\mathbf{x}$ *according to* D *or* $\hat{c}$

Neural Acquisition Function [Volpp et al. 2019]

Although the acquisition function α depends on the history $\mathcal{D}^{(t-1)}$ and the predictive model $\hat{c}$, α mainly makes use of the predictive mean μ and variance σ^2.

Neural Acquisition Function [Volpp et al. 2019]

Although the acquisition function α depends on the history $\mathcal{D}^{(t-1)}$ and the predictive model $\hat{c}$, α mainly makes use of the predictive mean μ and variance σ^2.

Neural acquisition function (AF):

$$\alpha_\theta(\mathbf{x}) = \alpha_\theta(\mu^{(t)}(\mathbf{x}), \sigma^{(t)}(\mathbf{x}), \mathbf{x}, t, T)$$

where θ are the parameters of a neural network, and μ, σ, $\mathbf{x}$, t, T are its inputs.

RL to train Neural AF [Volpp et al. 2019]

Policy π_θ: Neural acquisition function α_θ

RL to train Neural AF [Volpp et al. 2019]

Policy π_θ: Neural acquisition function α_θ

Episode: run of π on $f \in \mathcal{F}'$

- $\mathcal{F}$ is a set of functions we can sample functions from

RL to train Neural AF [Volpp et al. 2019]

Policy π_θ: Neural acquisition function α_θ

Episode: run of π on $f \in \mathcal{F}'$

- $\mathcal{F}$ is a set of functions we can sample functions from

State $s^{(t)}$: $\mu^{(t)}$ and $\sigma^{(t)}$ on a set of points $\xi^{(t)}$, and t and T

RL to train Neural AF [Volpp et al. 2019]

Policy π_θ: Neural acquisition function α_θ

Episode: run of π on $f \in \mathcal{F}'$

- $\mathcal{F}$ is a set of functions we can sample functions from

State $s^{(t)}$: $\mu^{(t)}$ and $\sigma^{(t)}$ on a set of points $\xi^{(t)}$, and t and T

Action $a^{(t)}$: Sampled point $\mathbf{x}^{(t)} \in \xi^{(t)}$

RL to train Neural AF [Volpp et al. 2019]

Policy π_θ: Neural acquisition function α_θ

Episode: run of π on $f \in \mathcal{F}'$

- $\mathcal{F}$ is a set of functions we can sample functions from

State $s^{(t)}$: $\mu^{(t)}$ and $\sigma^{(t)}$ on a set of points $\xi^{(t)}$, and t and T

Action $a^{(t)}$: Sampled point $\mathbf{x}^{(t)} \in \xi^{(t)}$

Reward $r^{(t)}$: negative simple regret: $r^{(t)} = f(\mathbf{x}^*) - f(\hat{\mathbf{x}})$

- assumes that we can estimate the optimal $\mathbf{x}^*$ for *training* functions

RL to train Neural AF [Volpp et al. 2019]

Policy π_θ: Neural acquisition function α_θ

Episode: run of π on $f \in \mathcal{F}'$

- $\mathcal{F}$ is a set of functions we can sample functions from

State $s^{(t)}$: $\mu^{(t)}$ and $\sigma^{(t)}$ on a set of points $\xi^{(t)}$, and t and T

Action $a^{(t)}$: Sampled point $\mathbf{x}^{(t)} \in \xi^{(t)}$

Reward $r^{(t)}$: negative simple regret: $r^{(t)} = f(\mathbf{x}^*) - f(\hat{\mathbf{x}})$

- assumes that we can estimate the optimal $\mathbf{x}^*$ for *training* functions

Transition probability : Noisy evaluation of f and the predictive model update

Policy [Volpp et al. 2019]

- The state is described by a discrete set of points $\xi^{(t)} = \{\xi_n\}_{n=1}^{N}$

Policy [Volpp et al. 2019]

- The state is described by a discrete set of points $\xi^{(t)} = \{\xi_n\}_{n=1}^{N}$
- We feed these points through the predictive model and the neural AF to obtain $\alpha_\theta(\xi_n) = \alpha_\theta(\mu^{(t)}(\xi_n), \sigma^{(t)}(\xi_n), \xi_n, t, T)$,

Policy [Volpp et al. 2019]

- The state is described by a discrete set of points $\xi^{(t)} = \{\xi_n\}_{n=1}^N$
- We feed these points through the predictive model and the neural AF to obtain $\alpha_\theta(\xi_n) = \alpha_\theta(\mu^{(t)}(\xi_n), \sigma^{(t)}(\xi_n), \xi_n, t, T)$,
- $\alpha_\theta(\xi_i)$ are interpreted as the logits of categorical distribution, s.t.

$$\pi_\alpha(\cdot \mid s^{(t)}) = \mathrm{Cat}\left[\alpha_\theta(\xi_1), \ldots, \alpha_\theta(\xi_N)\right]$$

Policy [Volpp et al. 2019]

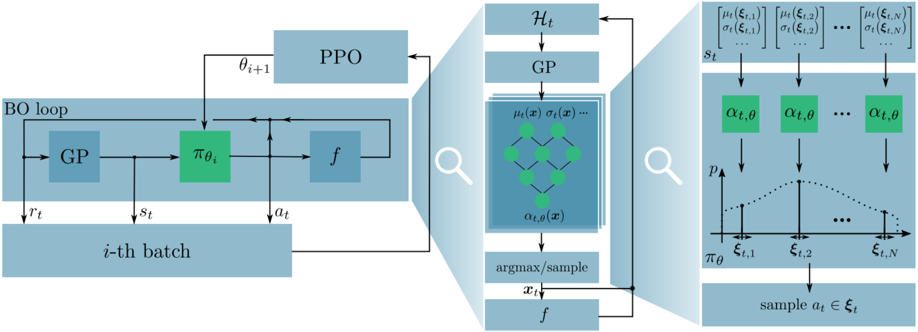
- The state is described by a discrete set of points $\xi^{(t)} = \{\xi_n\}_{n=1}^{N}$
- We feed these points through the predictive model and the neural AF to obtain $\alpha_\theta(\xi_n) = \alpha_\theta(\mu^{(t)}(\xi_n), \sigma^{(t)}(\xi_n), \xi_n, t, T)$,
- $\alpha_\theta(\xi_i)$ are interpreted as the logits of categorical distribution, s.t.

$$\pi_\alpha(\cdot \mid s^{(t)}) = \mathrm{Cat}\left[\alpha_\theta(\xi_1), \ldots, \alpha_\theta(\xi_N)\right]$$

- Due to curse of dimensionality, we need a two step approach for $\xi^{(t)}$
  1. sample ξ_{global} using a coarse Sobol grid
  2. sample ξ_{local} using local optimization starting from the best samples in ξ_{global}

$\rightsquigarrow$ $\xi^{(t)} = \xi_{\text{global}} \cup \xi_{\text{local}}$

Learning Acquisition Functions: Overview [Volpp et al. 2019]

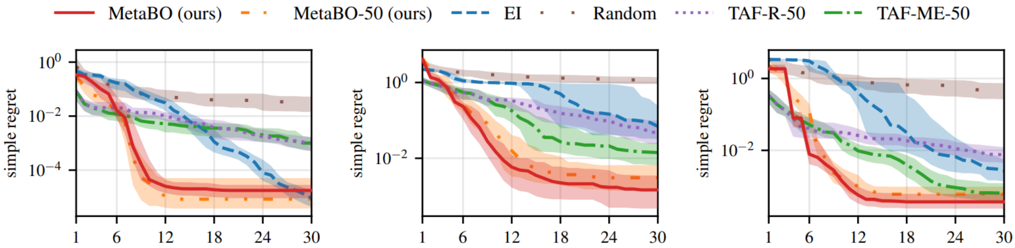


MetaBO Training Loop

Neural AF in the BO loop

Policy architecture

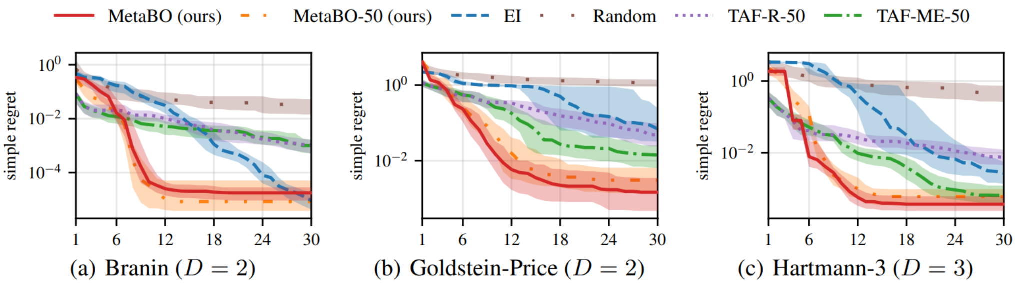
Results on Artificial Functions [Volpp et al. 2019]



(a) Branin ($D = 2$) (b) Goldstein-Price ($D = 2$) (c) Hartmann-3 ($D = 3$)

- Approach by [Volpp et al. 2019] called MetaBO
- MetaBO performs better than other acquisition functions (EI, GP-UCB, PI) and other baselines (Random, TAF)

Results on Artificial Functions [Volpp et al. 2019]



- Approach by [Volpp et al. 2019] called MetaBO
- MetaBO performs better than other acquisition functions (EI, GP-UCB, PI) and other baselines (Random, TAF)

Assumption: You have a family of functions at hand that resembles your target function.