Hyperparameters Optimization in Large Scale ML

Шугаепов Ильнур

VK.com
Performance Advertising Team

Higher School of Economics, 2020

Why we need HPO?

Examples of hyperparameters

- Number of trees in GBDT
- Regularization params in LogReg
- ...

Why we need HPO?

Examples of hyperparameters

- Number of trees in GBDT
- Regularization params in LogReg
- ...

Goals of HPO

- Improve the performance of machine learning algorithms

Why we need HPO?

Examples of hyperparameters

- Number of trees in GBDT
- Regularization params in LogReg
- ...

Goals of HPO

- Improve the performance of machine learning algorithms
- Reduce the human effort necessary for applying machine learning (`AutoML`)

Problem Statement [7]

- $\mathcal{A}$ — algorithm with N hyperparameters
- Λ_n — domain of n-th hyperparameter
- $\mathbf{\Lambda} = \Lambda_1 \times \Lambda_2 \times \ldots \times \Lambda_N$ — configuration space
- $\mathcal{A}_{\boldsymbol{\lambda}}$ — algorithm with hyperparameters $\boldsymbol{\lambda} \in \mathbf{\Lambda}$

Problem Statement [7]

- $\mathcal{A}$ — algorithm with N hyperparameters
- Λ_n — domain of n-th hyperparameter
- $\boldsymbol{\Lambda} = \Lambda_1 \times \Lambda_2 \times \ldots \times \Lambda_N$ — configuration space
- $\mathcal{A}_{\boldsymbol{\lambda}}$ — algorithm with hyperparameters $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$

HPO Problem

Diven a dataset $\mathcal{D}$, our goal is to find

$$\boldsymbol{\lambda}^* = \arg\min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} \mathbb{E}_{(\mathcal{D}_{train}, \mathcal{D}_{valid}) \sim \mathcal{D}} \boldsymbol{V}(\mathcal{L}, \mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid}),$$

where $\boldsymbol{V}(\mathcal{L}, \mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$ measure the loss of a model generated by $\mathcal{A}_{\boldsymbol{\lambda}}$ on training data $\mathcal{D}_{train}$ and evaluated on validation data $\mathcal{D}_{valid}$.

Algorithm

1. For each Λ_i, $i = 1, \ldots, N$ choose $L_i \subseteq \Lambda_i$ - set of values
2. $S = \prod_{i=1}^{N} L_i \subseteq \mathbf{\Lambda}$ - set of trial points
3. Evaluate $\boldsymbol{V}(\mathcal{L}, \mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$ for each $\boldsymbol{\lambda} \in S$ tiral point
4. Report best result

Algorithm

1. For each Λ_i, $i = 1, \ldots, N$ choose $L_i \subseteq \Lambda_i$ - set of values
2. $S = \prod_{i=1}^{N} L_i \subseteq \mathbf{\Lambda}$ - set of trial points
3. Evaluate $\boldsymbol{V}(\mathcal{L}, \mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$ for each $\boldsymbol{\lambda} \in S$ tiral point
4. Report best result

Q: How to compare different HPO algorithms

Algorithm

1. For each Λ_i, $i = 1, \ldots, N$ choose $L_i \subseteq \Lambda_i$ - set of values
2. $S = \prod_{i=1}^{N} L_i \subseteq \mathbf{\Lambda}$ - set of trial points
3. Evaluate $\boldsymbol{V}(\mathcal{L}, \mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$ for each $\boldsymbol{\lambda} \in S$ tiral point
4. Report best result

Q: How to compare different HPO algorithms **A:** In terms of number of evaluations/wall-clock time

Challanges of HPO

- Function evaluations can be extremely expensive

Challanges of HPO

- Function evaluations can be extremely expensive
- The configuration space is often complex and high-dimensional

Challanges of HPO

- Function evaluations can be extremely expensive
- The configuration space is often complex and high-dimensional
- The configuration space can contain *conditionality*

Challanges of HPO

- Function evaluations can be extremely expensive
- The configuration space is often complex and high-dimensional
- The configuration space can contain *conditionality*
- We usually don’t have access to a gradient of the loss function with respect to the hyperparameters

Challanges of HPO

Example

- XGBoost training takes on average about 10 hours

Challanges of HPO

Example

- XGBoost training takes on average about 10 hours
- Configuration space

```
space = {
    'num_round' : choice(range(200, 1500, 100)),  # 14
    'eta' : quniform(0.05, 0.55, 0.05),  # 11
    'max_depth' : choice(range(4, 10)),  # 6
    'min_child_weight' : quniform(1, 100, 10),  # 10
    'subsample' : quniform(0.5, 1, 0.05),  # 11
    'gamma' : quniform(0, 1, 0.1),  # 10
    'colsample_bytree' : quniform(0.5, 1, 0.1),  # 5
    'alpha': choice(range(0, 100, 10)),  # 11
    'grow_policy' : choice(['depthwise', 'lossguide']),  # 2
}
```

Challanges of HPO

Example

- XGBoost training takes on average about 10 hours
- Configuration space

```
space = {
    'num_round' : choice(range(200, 1500, 100)),  # 14
    'eta' : quniform(0.05, 0.55, 0.05),  # 11
    'max_depth' : choice(range(4, 10)),  # 6
    'min_child_weight' : quniform(1, 100, 10),  # 10
    'subsample' : quniform(0.5, 1, 0.05),  # 11
    'gamma' : quniform(0, 1, 0.1),  # 10
    'colsample_bytree' : quniform(0.5, 1, 0.1),  # 5
    'alpha': choice(range(0, 100, 10)),  # 11
    'grow_policy' : choice(['depthwise', 'lossguide']),  # 2
}
```

- Number of evaluations required

$$14 \times 6 \times 10 \times 11 \times 10 \times 5 \times 11 \times 2 = 111804000$$

curse of dimensionality

Validation protocol

Reducing the evaluation time

- $\boldsymbol{V}(\mathcal{L}, \mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$ — holdout and cross-validation error
- Strategies for reducing the evaluation time
  - test algorithms on a subset of folds
  - only on a subset of data
  - for a small amount of iterations
  - auxiliary tasks

Table of Contents

Algorithm

1. Set of trials $S \sim_{\texttt{iid}} \mathbf{\Lambda}$
2. Evaluate $\boldsymbol{V}(\mathcal{L}, \mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$ for each $\boldsymbol{\lambda} \in S$ tiral point
3. Report best result

Algorithm

1. Set of trials $S \sim_{\texttt{iid}} \boldsymbol{\Lambda}$
2. Evaluate $\boldsymbol{V}(\mathcal{L}, \mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$ for each $\boldsymbol{\lambda} \in S$ tiral point
3. Report best result

- In the simplest scenario $\boldsymbol{\Lambda} = U(\Lambda_1) \times U(\Lambda_2) \times \ldots \times U(\Lambda_N)$
- User can set distribution over Λ_i

Random vs Grid Search

- Given fixed budget B - number of evaluations
- The number of different values grid search can afford to evaluate for each of the N hyperparameters is only $B^{1/N}$, whereas random search will explore B different values for each

Random vs Grid Search

- Given fixed budget B - number of evaluations
- The number of different values grid search can afford to evaluate for each of the N hyperparameters is only $B^{1/N}$, whereas random search will explore B different values for each

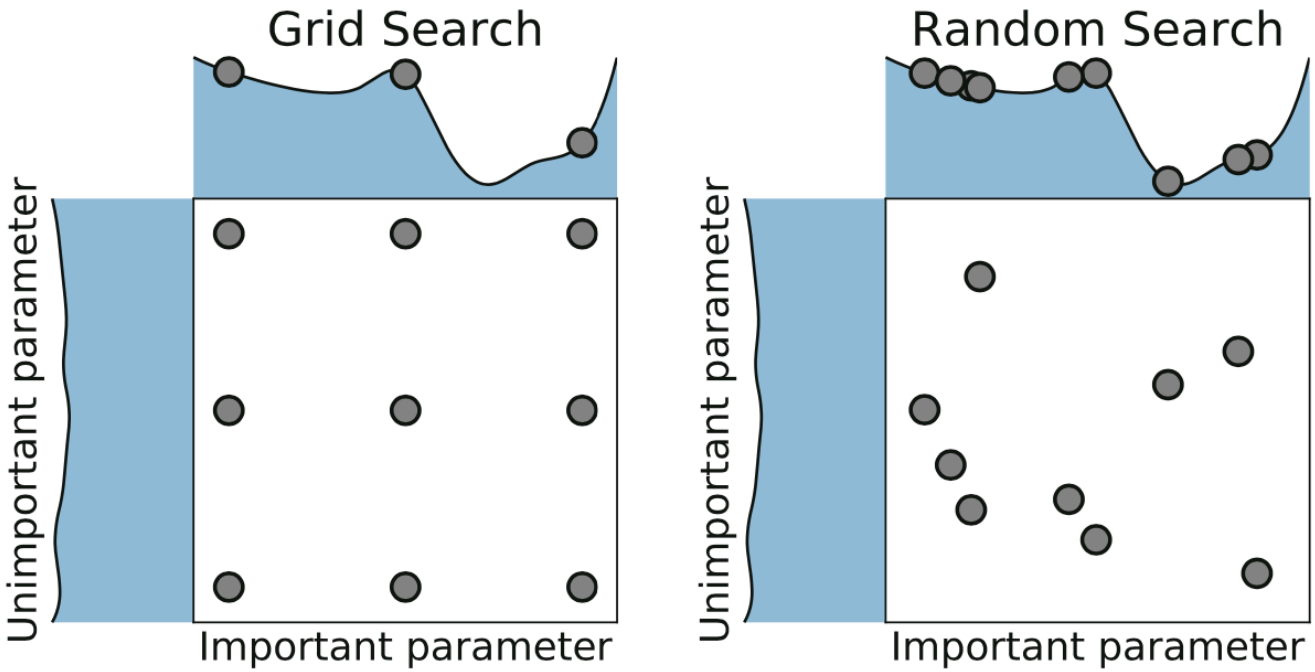


Рис.: Comparison of grid search and random search for minimizing a function with one important and one unimportant parameter. $f(x, y) = g(x) + h(y) \approx g(x)$ with *low effective dimensionality*. Above each square $g(x)$, left of each square $h(y)$

Experimental Results

Grid Search is usually worse because:

- a small fraction of hyperparameters matter for any one data set, but
- different hyperparameters matter on different data sets

Remark

Failure of grid search is the rule rather than the exception in high dimensional hyperparameter optimization

How to prove that only a few hyperparameters matter? See [2]

Table of Contents

Sequential Model-based Global Optimization [4]

Bayesian Optimization

- $$\max_{\mathbf{x} \in \Lambda \subset \mathbb{R}^N} f(\mathbf{x})$$
- $f \colon \Lambda \to \mathbb{R}$ — black-box true objective is costly to evaluate

- $$\max_{\mathbf{x} \in \Lambda \subset \mathbb{R}^N} f(\mathbf{x})$$
- $f \colon \Lambda \to \mathbb{R}$ — black-box true objective is costly to evaluate

Algorithm 2.2 SMBO(f, M_0, T, S)

$\mathcal{D}_0 \leftarrow \emptyset$ ▷ Observations
for $t = 1, \ldots, T$ **do**
  $\mathbf{x}^* \leftarrow \arg\min_x S(x, M_{t-1})$
  Evaluate $f(\mathbf{x}^*)$ ▷ Expensive step
  $\mathcal{D}_{1:t} \leftarrow \mathcal{D}_{1:t-1} \cup \{(\mathbf{x}^*, f(\mathbf{x}^*))\}$
  Fit a new model M_t to $\mathcal{D}_{1:t}$
return $\mathcal{D}_{1:t}$ ▷ Observation history

- $$\max_{\mathbf{x} \in \Lambda \subset \mathbb{R}^N} f(\mathbf{x})$$
- $f \colon \Lambda \to \mathbb{R}$ — black-box true objective is costly to evaluate

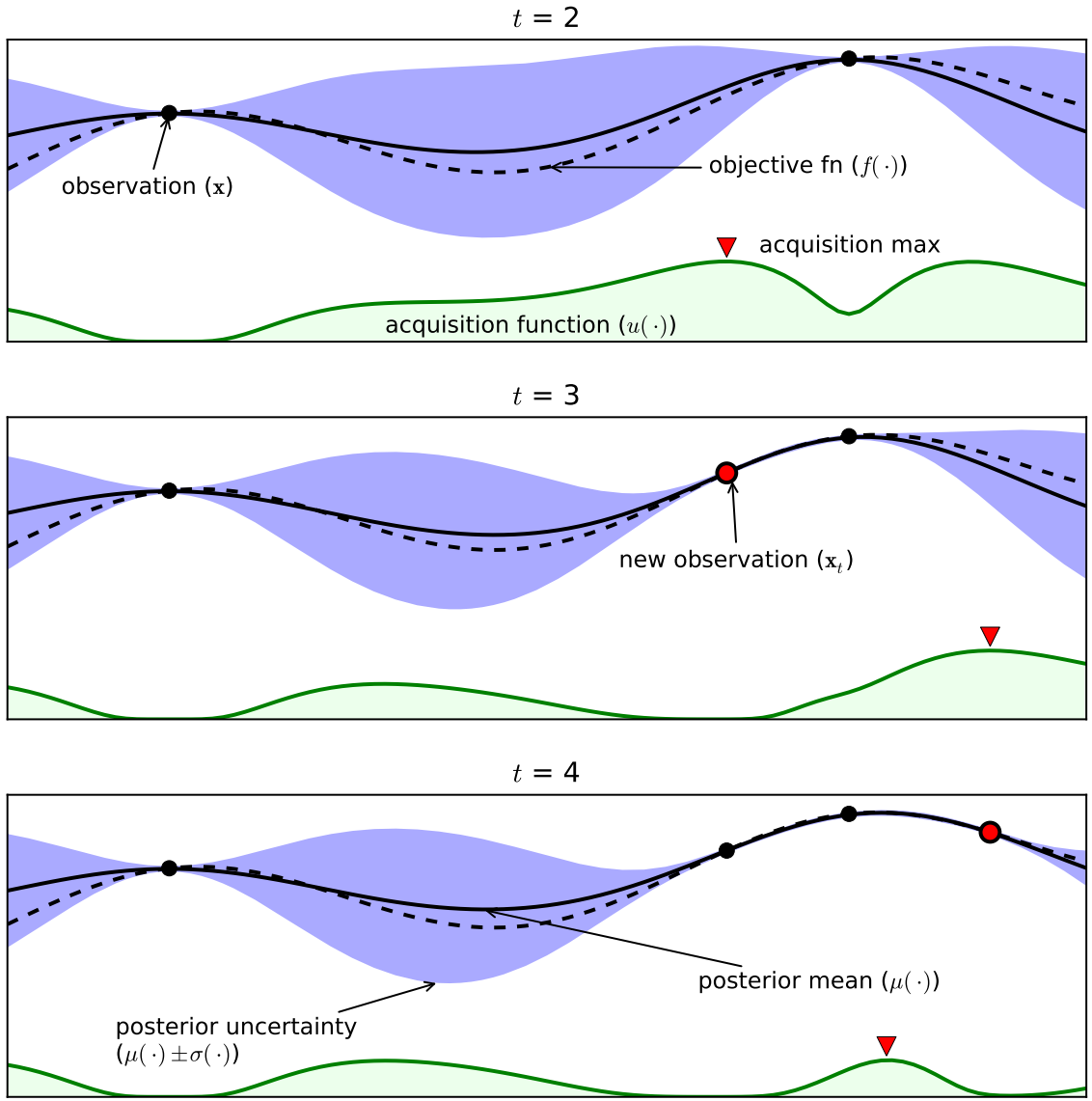
Algorithm 2.3 SMBO(f, M_0, T, S)

$\mathcal{D}_0 \leftarrow \emptyset$ ▷ Observations
for $t = 1, \ldots, T$ **do**
  $\mathbf{x}^* \leftarrow \arg\min_x S(x, M_{t-1})$
  Evaluate $f(\mathbf{x}^*)$ ▷ Expensive step
  $\mathcal{D}_{1:t} \leftarrow \mathcal{D}_{1:t-1} \cup \{(\mathbf{x}^*, f(\mathbf{x}^*))\}$
  Fit a new model M_t to $\mathcal{D}_{1:t}$
return $\mathcal{D}_{1:t}$ ▷ Observation history

Key ingredients: a probabilistic *surrogate model* (M) and an *acquisition function* (S) to decide which point to evaluate next

Bayesian Optimization [5]

- $P(f \mid \mathcal{D}_{1:t}) \propto P(\mathcal{D}_{1:t} \mid f)P(f)$
- *surrogate model* — Gaussian process (GP)



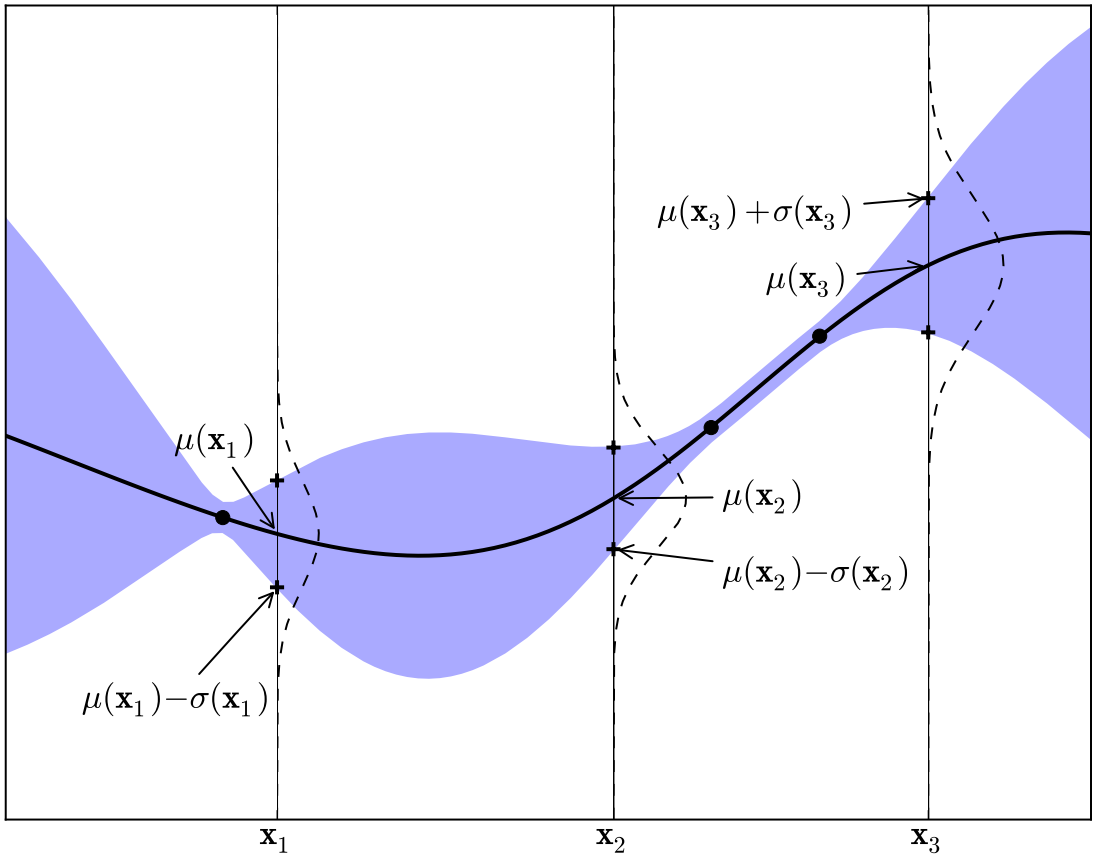


Рис.: Simple 1D Gaussian process with three observations

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$$

$m(\mathbf{x})$ — mean function
$k(\mathbf{x}, \mathbf{x}')$ — covariance function

$$k\left(\mathbf{x}_i, \mathbf{x}_j\right) = \exp\left(-\frac{1}{2}\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2\right)$$

Surrogate Models

Gaussian process [10, 5]

- What we want

$$P(f \mid \mathcal{D}_{1:t}) \propto P(\mathcal{D}_{1:t} \mid f)P(f)$$

- What we want

$$P(f \mid \mathcal{D}_{1:t}) \propto P(\mathcal{D}_{1:t} \mid f)P(f)$$

- For GP holds the following (assume that $m(\mathbf{x}) = 0$)

$$\begin{bmatrix} \mathbf{f}_{1:t} \\ f_{t+1} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} & \mathbf{k} \\ \mathbf{k}^T & k\left(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}\right) \end{bmatrix}\right),$$

where $\mathbf{k} = \left[k\left(\mathbf{x}_{t+1}, \mathbf{x}_1\right), k\left(\mathbf{x}_{t+1}, \mathbf{x}_2\right), \ldots, k\left(\mathbf{x}_{t+1}, \mathbf{x}_t\right)\right]$ and $\mathbf{K} = \left(k(\mathbf{x}_i, \mathbf{x}_j)\right)$ - kernel matrix.

- What we want

$$P(f \mid \mathcal{D}_{1:t}) \propto P(\mathcal{D}_{1:t} \mid f) P(f)$$

- For GP holds the following (assume that $m(\mathbf{x}) = 0$)

$$\begin{bmatrix} \mathbf{f}_{1:t} \\ f_{t+1} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} & \mathbf{k} \\ \mathbf{k}^T & k\left(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}\right) \end{bmatrix}\right),$$

where $\mathbf{k} = \left[k\left(\mathbf{x}_{t+1}, \mathbf{x}_1\right), k\left(\mathbf{x}_{t+1}, \mathbf{x}_2\right), \ldots, k\left(\mathbf{x}_{t+1}, \mathbf{x}_t\right)\right]$ and $\mathbf{K} = \left(k(\mathbf{x}_i, \mathbf{x}_j)\right)$ - kernel matrix.

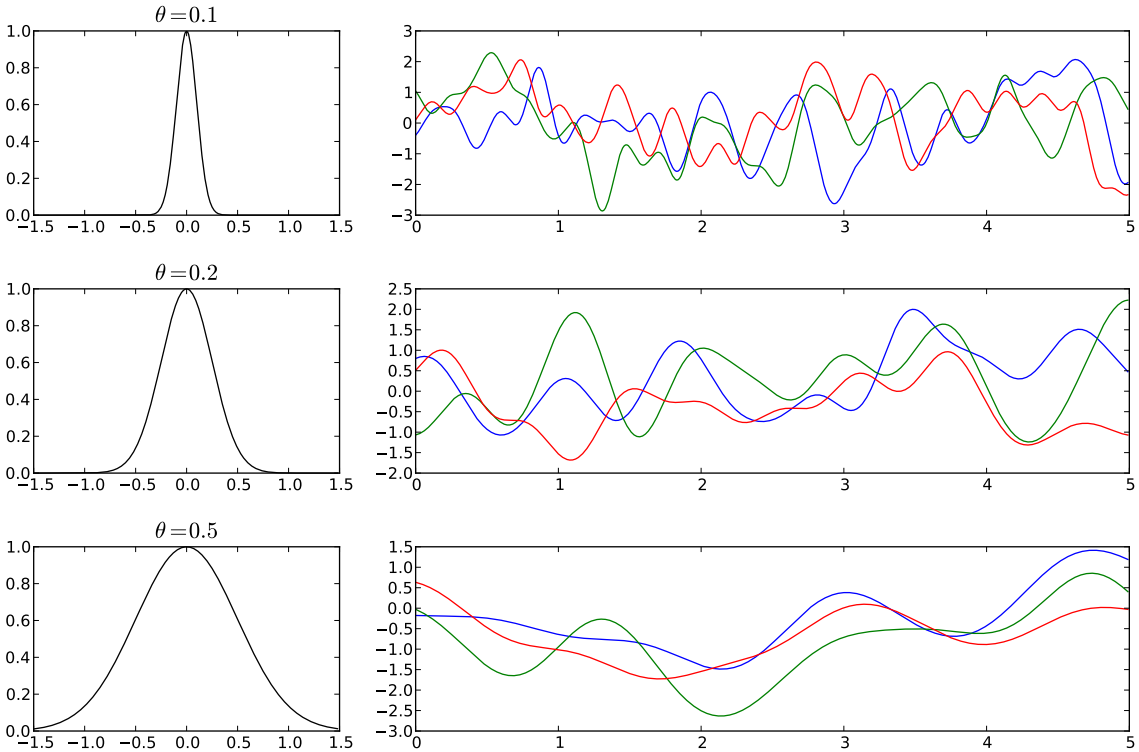
GP surrogate

$$P\left(f_{t+1} | \mathcal{D}_{1:t}, \mathbf{x}_{t+1}\right) = \mathcal{N}\left(\mu_t\left(\mathbf{x}_{t+1}\right), \sigma_t^2\left(\mathbf{x}_{t+1}\right)\right),$$

where

$$\mu_t\left(\mathbf{x}_{t+1}\right) = \mathbf{k}^T \mathbf{K}^{-1} \mathbf{f}_{1:t}$$
$$\sigma_t^2\left(\mathbf{x}_{t+1}\right) = k\left(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}\right) - \mathbf{k}^T \mathbf{K}^{-1} \mathbf{k}$$

Hyperparameters of GP



$$k\left(\mathbf{x}_i, \mathbf{x}_j\right) = \exp\left(-\frac{1}{2\theta^2}\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2\right),$$

where θ controls the width of the kernel

Q: How to find optimal values of θ and other possible hyperparameters?

Q: How to find optimal values of θ and other possible hyperparameters?

A: MLE

Q: How to find optimal values of θ and other possible hyperparameters?

A: MLE

- Observations

$$\mathbf{f}_{1:t} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma),$$

  where $\boldsymbol{\mu} = (m(\mathbf{x}_1), \ldots, m(\mathbf{x}_t))$

- Log Likelihood

$$L = \log p(\mathbf{f}_{1:t} \mid \mathbf{x}_{1:t}, \theta) = -\frac{1}{2}\log|\Sigma| - \frac{1}{2}(\mathbf{f}_{1:t} - \boldsymbol{\mu})^\top \Sigma^{-1}(\mathbf{f}_{1:t} - \boldsymbol{\mu}) - \frac{t}{2}\log(2\pi)$$

- Find $\frac{\partial L}{\partial \theta_m}$, $\frac{\partial L}{\partial \theta_k}$, where θ_m, θ_k — hyperparameters of the m and k functions

Acquisition Functions

Acquisition Function

The role of the acquisition function is to guide the search for the optimum

$$\mathbf{x}_{t+1} = \arg\max_{\mathbf{x}} u(\mathbf{x} \mid \mathcal{D}_{1:t}),$$

where u — acquisition function.
High acquisition corresponds to *potentially* high values of the objective function

Acquisition Function

The role of the acquisition function is to guide the search for the optimum

$$\mathbf{x}_{t+1} = \arg\max_{\mathbf{x}} u(\mathbf{x} \mid \mathcal{D}_{1:t}),$$

where u — acquisition function.
High acquisition corresponds to *potentially* high values of the objective function

Examples

- Probability of Improvement

$$\begin{aligned} \mathrm{PI}(\mathbf{x}) &= P\left(f(\mathbf{x}) \geq f\left(\mathbf{x}^+\right)\right) \\ &= \Phi\left(\frac{\mu(\mathbf{x}) - f\left(\mathbf{x}^+\right)}{\sigma(\mathbf{x})}\right), \end{aligned}$$

where $\mathbf{x}^+ = \arg\max_{\mathbf{x}_i \in \mathbf{x}_{1:t}} f(\mathbf{x}_i)$

Acquisition Functions

Unlike the original unknown objective function, $u(\cdot)$ can be cheaply sampled.

Noise

Instead of observing $f(\mathbf{x})$, we can often only observe a noisy transformation of $f(\mathbf{x})$.

Noise

Instead of observing $f(\mathbf{x})$, we can often only observe a noisy transformation of $f(\mathbf{x})$.

- Assume that $f(\mathbf{x})$ is corrupted with Gaussian noise $\varepsilon \sim \mathcal{N}(0, \sigma^2_{noise})$
- $y_i = f(\mathbf{x}_i) + \varepsilon_i$
- Since mean of noise is zero $\boldsymbol{K} = \boldsymbol{K} + \sigma^2_{noise} I$

Noise

Instead of observing $f(\mathbf{x})$, we can often only observe a noisy transformation of $f(\mathbf{x})$.

- Assume that $f(\mathbf{x})$ is corrupted with Gaussian noise $\varepsilon \sim \mathcal{N}(0, \sigma^2_{noise})$
- $y_i = f(\mathbf{x}_i) + \varepsilon_i$
- Since mean of noise is zero $\boldsymbol{K} = \boldsymbol{K} + \sigma^2_{noise} I$

GP surrogate with noise

$$P\left(f_{t+1} | \mathcal{D}_{1:t}, \mathbf{x}_{t+1}\right) = \mathcal{N}\left(\mu_t\left(\mathbf{x}_{t+1}\right), \sigma_t^2\left(\mathbf{x}_{t+1}\right) + \sigma^2_{noise}\right),$$

where

$$\mu_t\left(\mathbf{x}_{t+1}\right) = \mathbf{k}^T \left[\mathbf{K} + \sigma^2_{noise} I\right]^{-1} \mathbf{f}_{1:t}$$

$$\sigma_t^2\left(\mathbf{x}_{t+1}\right) = k\left(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}\right) - \mathbf{k}^T \left[\mathbf{K} + \sigma^2_{noise} I\right]^{-1} \mathbf{k}$$

Noise

Instead of observing $f(\mathbf{x})$, we can often only observe a noisy transformation of $f(\mathbf{x})$.

- Assume that $f(\mathbf{x})$ is corrupted with Gaussian noise $\varepsilon \sim \mathcal{N}(0, \sigma^2_{noise})$
- $y_i = f(\mathbf{x}_i) + \varepsilon_i$
- Since mean of noise is zero $\boldsymbol{K} = \boldsymbol{K} + \sigma^2_{noise} I$

GP surrogate with noise

$$P\left(f_{t+1} | \mathcal{D}_{1:t}, \mathbf{x}_{t+1}\right) = \mathcal{N}\left(\mu_t\left(\mathbf{x}_{t+1}\right), \sigma_t^2\left(\mathbf{x}_{t+1}\right) + \sigma^2_{noise}\right),$$

where

$$\mu_t\left(\mathbf{x}_{t+1}\right) = \mathbf{k}^T \left[\mathbf{K} + \sigma^2_{noise} I\right]^{-1} \mathbf{f}_{1:t}$$

$$\sigma_t^2\left(\mathbf{x}_{t+1}\right) = k\left(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}\right) - \mathbf{k}^T \left[\mathbf{K} + \sigma^2_{noise} I\right]^{-1} \mathbf{k}$$

Remark

Parameter σ^2_{noise} of noise distribution could be learned while trainig GP

Table of Contents

Multi-fidelity Optimization

A common technique to speed up *manual* tuning

- probe an hyperparameter configuration on a small subset of $\mathcal{D}_{train}$
- training it only for a few iterations
- running it on a subset of features

Multi-fidelity Optimization

A common technique to speed up *manual* tuning

- probe an hyperparameter configuration on a small subset of $\mathcal{D}_{train}$
- training it only for a few iterations
- running it on a subset of features

Multi-fidelity Optimization

- low-fidelity (low cost) approximations of the actual loss function $\boldsymbol{V}(\mathcal{L}, \mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$
- tradeoff between optimization performance and runtime

Remark

MFO used to speedup discussed HPO algorithms

Predictive Termination [6]

Types of Learning Curves

1. Performance of an iterative algorithm as a function of its number of iterations
2. Performance of an algorithm as a function of the size of $\mathcal{D}_{train}$

Predictive Termination [6]

Approach

- When running SGD on DNNs we measure validation performance ($-\boldsymbol{V}(\mathcal{L}, \mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$) in regular intervals
- $\boldsymbol{f}_{1:t}$ — observed performance values for the first t intervals

Predictive Termination [6]

Approach

- When running SGD on DNNs we measure validation performance ($-\boldsymbol{V}(\mathcal{L}, \mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$) in regular intervals
- $\boldsymbol{f}_{1:t}$ — observed performance values for the first t intervals
- We observe $\boldsymbol{f}_{1:t}$ and aim to predict performance f_m after a large number of intervals $m \gg t$

Predictive Termination [6]

Approach

- When running SGD on DNNs we measure validation performance ($-\boldsymbol{V}(\mathcal{L}, \mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$) in regular intervals
- $\boldsymbol{f}_{1:t}$ — observed performance values for the first t intervals
- We observe $\boldsymbol{f}_{1:t}$ and aim to predict performance f_m after a large number of intervals $m \gg t$

Predictive Termination Criteria

Probability that the network, after training for m intervals, will exceed the performance y^+ (current best).

$$P(f_m \geq f^+ \mid \boldsymbol{f}_{1:t}) \geq \delta$$

If this probability is above a threshold δ then training continues

Predictive Termination [6]

Experimental results

Predictive termination speeds up current hyperparameter optimization methods for DNNs by roughly a factor of two

Hyperband [8]

Successive halving

Algorithm

1. For a given initial budget (amount of data), query all algorithms for that budget
2. Remove the half that performed worst
3. Double the budget
4. Repeat until only a single algorithm is left

Hyperband [8]

Successive halving

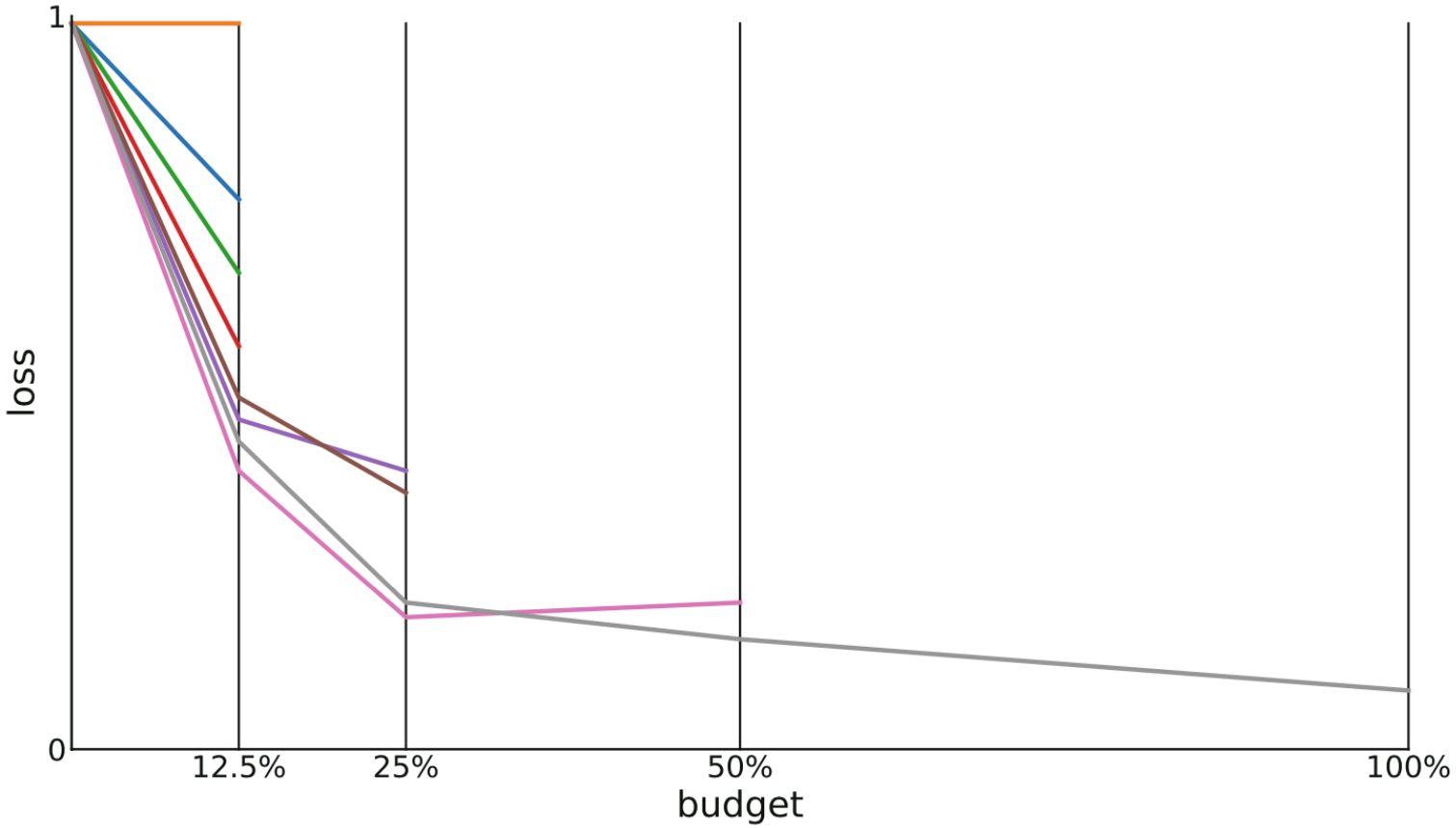


Рис.: Illustration of successive halving for eight algorithms/configurations. After evaluating all algorithms on $1/8$ of the total budget, half of them are dropped and the budget given to the remaining algorithms is doubled

User has to decide beforehand:

1. to try many configurations and only assign a small budget to each, or
2. to try only a few and assign them a larger budget.

Multi-task Bayesian Optimization [12]

Table of Contents

HPO Systems

- hyperopt[1] [3]
- Facebook/Ax[2]
- fmfn/BayesianOptimization[3]
- Optuna[4] [1]
- hyperband[5] [8]
- tune[6] [9]
- ...

[1] https://github.com/hyperopt/hyperopt
[2] https://github.com/facebook/Ax
[3] https://github.com/fmfn/BayesianOptimization
[4] https://github.com/optuna/optuna
[5] https://github.com/zygmuntz/hyperband
[6] https://github.com/ray-project/ray

Table of Contents

Resume

- Why Random Search is better than Grid Search (low effective dimensionality)
- How to compare different HPO algorithms
- SOTA Bayesian Optimization (surrogate, acquisition)
- How to speedup Random Search and BO
  - Predictive Termination
  - Successive halving
  - Multi-task Bayesian Optimization

References I

[1] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2623–2631, 2019.

[2] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(Feb):281–305, 2012.

[3] J. Bergstra, D. Yamins, and D. D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. 2013.

[4] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*, pages 2546–2554, 2011.

[5] E. Brochu, V. M. Cora, and N. De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

References II

[6] T. Domhan, J. T. Springenberg, and F. Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[7] F. Hutter, L. Kotthoff, and J. Vanschoren. *Automated Machine Learning*. Springer, 2019.

[8] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1):6765–6816, 2017.

[9] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018.

[10] C. E. Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

[11] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

References III

[12] K. Swersky, J. Snoek, and R. P. Adams. Multi-task bayesian optimization. In *Advances in neural information processing systems*, pages 2004–2012, 2013.