An Overview and Comparison on Bayesian Optimization Techniques

Das, R., Roy Chowdhury, A., Thompson, C.

A Tutorial on Bayesian Optimization by Peter Fraiser (2018)

October 3, 2022

Background

- Hyperparameter optimization methods are generally 4 types:
  - Manual Search
  - Grid Search
  - Random Search
  - Bayesian Optimization
- **Manual Search:** We choose some model hyperparameters based on our judgment/experience to train/test the model.
- **Grid Search:** We set up a grid of hyperparameters and train/test our model on each of the possible combinations. In case of a large number of parameters it is not time efficient.
- **Random Search:** We randomly chooses combinations of hyperparameters from all possible combinations. In practice random search is often more time efficient than grid search.

Bayesian Optimization Overview

Bayesian Optimization:

Bayesian optimization is a sequential design strategy for global optimization of black-box functions that does not assume any functional forms. It is usually used to optimize functions which are expensive to evaluate .

- It attempts to find the global optimum in a minimum number of steps.
- Bayesian optimization incorporates prior belief about objective function in f and updates the prior with samples drawn from f to get a posterior that better approximates f.

Bayesian Optimization Overview

- **Application:** Bayesian Optimization is mainly used when function evaluation is expensive and hence we have to minimize the number of samples drawn from it.
  - Tuning hyperparameters of a deep neural network
  - Computer graphics and visual design
  - Automatic machine learning toolboxes
  - Reinforcement learning
  - Robotics
  - Sensor networks
  - Experimental particle physics
  - Evaluation of effectiveness of a drug
  - Probe drilling for oil at given geographic coordinates

Bayesian Optimization: Comparison with other methods

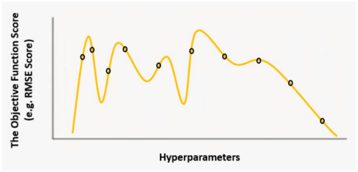
- Bayesian Optimization differs from Random Search and Grid Search because it improves the search speed using past performances, whereas the other two methods are uniform (or independent) of past evaluations.
- In this regard, Bayesian Optimization is similar to Manual Search. But it has a vast area of application compared to Manual Search.

Surrogate Functions

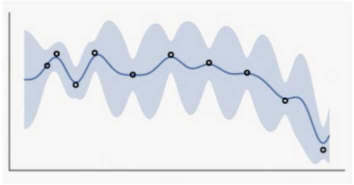
Definition

A surrogate model is the probability representation of the objective function, which maps hyperparameters to a probability of a score on the objective function.

$$p(\text{objective function score} \mid \text{hyperparameter})$$



(a) True Objective Function



(b) Surrogate Model

Acquisition functions

Definition

Acquisition function is a technique by which the posterior is used to select the next sample from the search space. Acquisition function is maximized to select the next choice of hyperparameter.

More formally, the objective function f will be sampled at

- $x_t = \text{argmax}_x u(x|D_{1:t-1})$, where u is the acquisition function
- $D_{1:t-1} = (x_1, y_1), \cdots, (x_{t-1}, y_{t-1})$ are the $t-1$ samples drawn from f.

Tree-Structured Parzen Estimators (TPE)

Tree-Structured Parzen Estimators

One approach which utilizes Bayes rule to model the probability distribution of hyper-parameters x given the score on the objective function y.

Instead of constructing the direct surrogate model $p(y|x)$, TPE uses:

$$p(y|x) = \frac{p(x|y) \cdot p(y)}{p(x)}$$

to model $p(x|y)$ and $p(y)$. In this way, the algorithm uses the (hyper-parameter, score) pairs to build probability models of the objective function that improve with each iteration.

TPE

TPE models observations in the following manner:

- The observations are split into two groups, where y^* is the splitting value.
- The chosen y^* is some quantile γ of the observed y values such that $p(y < y^*) = \gamma$.

Thus, TPE defines:

$$p(x|y) = \begin{cases} \ell(x), & \text{if } y < y^* \\ g(x), & \text{if } y \geq y^*, \end{cases}$$

Essentially, $\ell(x)$ is the density of the best observations, and $g(x)$ is the density of the poor observations. These densities are constructed using adaptive Parzen estimators (kernel density estimators), which are a simple average of kernels centered on existing data points.

Expected Improvement (EI) with TPE

Expected Improvement (EI) is a type of acquisition function that is built on top of a surrogate model. The formula for EI in combination with TPE is:

$$
\begin{aligned}
EI_{y^*}(x) &= \int_{-\infty}^{y^*} (y^* - y) p(y|x) dy \\
&= \int_{-\infty}^{y^*} (y^* - y) \frac{p(x|y)p(y)}{p(x)} dy
\end{aligned}
$$

Using what we know about TPE, we can rewrite this as:

$$
EI_{y^*}(x) \propto \left(\gamma + \frac{g(x)}{\ell(x)}(1 - \gamma) \right)^{-1} ,
$$

Thus, drawn values of the hyper-parameters will be more likely under the density of our best observations than the density of our poor observations.

TPE Algorithm

Initially sample the response surface by random search
For each iteration:

1. Split the observations into two groups
2. Model $\ell(x)$ and $g(x)$ based on the observations using adaptive Parzen estimators
3. Find the hyper-parameters that perform the best with EI
4. Apply these hyper-parameters to the true objective function

Repeat until max iterations/time limit exceeded

Random Forest (RF) Regression in Sequential Model-Based Algorithm Configuration (SMAC)

SMAC

A tool for optimizing algorithm parameters where the models are based on random forests.

Random forest

RFs are collections of regression trees that have real values (target algorithm performance values) at their leaves. They allow us to quantify our uncertainty in a given prediction.

The algorithm is as follows:

- Construct a RF model to predict performance based on a cost metric
- Use that model to select promising hyper-parameter configurations
- Compare each selected configuration against the best known

Repeat until max iterations/time limit

Regression Trees

Regression trees are a predictive modeling approach that go from observations about an item to conclusions about an item's target value. The two major steps are:

- Divide the predictor space into distinct regions
- For each observation in a region, we make the prediction (the mean of response in the training set in that particular region)

The splitting criterion is the set of left and right children L and R that minimizes

$$\ell(L, R) = \sum_{x_i \in L} (y_i - \mu_L)^2 + \sum_{x_i \in R} (y_i - \mu_R)^2$$

Instead of looking at every possible configuration, start at the top (where every observation is in one region) and successively split into further branches

EI with RF

SMAC uses the model to select a list of promising hyper-parameter configurations.

- The mean and variance of the predictive distribution of the RF is obtained for new configuration θ as the empirical mean and variance of its individual trees' predictions for θ
- The prediction is adapted to be the user-defined cost metric of that data
- To quantify how promising a configuration is, it uses the model's predictive distribution for θ to compute its EI over the best configuration seen so far

Identify configurations that maximize EI

Gaussian Process (GP) Regression

Gaussian Process

A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.

A Gaussian process is completely specified by its mean function and covariance function. We define mean function $m(x)$ and the covariance function mean function $k(x, x')$ of a real process $f(x)$ as

$$\begin{aligned} m(x) &= \mathrm{E}[f(x)] \\ k(x, x') &= \mathrm{E}[(f(x) - m(x))(f(x') - m(x'))] \end{aligned}$$

and will write the Gaussian process as

$$f(x) \sim GP(m(x), k(x, x'))$$

Gaussian Process (GP) Regression

In real life we observe, $y = f(x) + \epsilon$. Using a GP prior on f, we have,

$$f(x_{1:k}) \sim N(\mu_0(x_{1:k}), \Sigma_0(x_{1:k}, x_{1:k}))$$

where

$$\begin{aligned} f(x_{1:k}) &= [f(x_1), ..., f(x_k)], \\ \mu_0(x_{1:k}) &= [\mu_0(x1), \cdots, \mu_0(x_k)] \\ \Sigma_0(x_{1:k}, x_{1:k}) &= \begin{pmatrix} \Sigma_0(x_1, x_1) & \cdots & \Sigma_0(x_1, x_k) \\ \cdots & \cdots & \cdots \\ \Sigma_0(x_k, x_1) & \cdots & \Sigma_0(x_k, x_k) \end{pmatrix} \end{aligned}$$

Gaussian Process (GP) Regression

Suppose we observe $f(x_{1:n})$ without noise for some n and we wish to infer the value of $f(x)$ at some new point x. Then the conditional distribution of $f(x)$ (**key predictive equations for Gaussian process regression**) given these observations is given by:

$$
\begin{aligned}
& f(x)|f(x_{1:n}) \sim N(\mu_n(x), \sigma_n^2(x)), \qquad (1) \\
\text{where} \quad & \mu_n(x) = \Sigma_0(x, x_{1:k})\Sigma_0(x_{1:k}, x_{1:k})^{-1}(f(x_{1:k}) - \mu_0(x_{1:k})) + \mu_0(x) \\
& \sigma_n^2(x) = \Sigma_0(x, x) - \Sigma_0(x, x_{1:k})\Sigma_0(x_{1:k}, x_{1:k})^{-1}\Sigma_0(x_{1:k}, x)
\end{aligned}
$$

This conditional distribution is also called the *posterior probability distribution*.

Expected Improvement(EI) with GP

The expected improvement can be evaluated in closed form as:

$$EI_n(x) := [\Delta_n(x)]^+ + \sigma_n(x)\psi(\frac{\Delta_n(x)}{\sigma_n(x)}) - |\Delta_n(x)|\Phi(\frac{\Delta_n(x)}{\sigma_n(x)})$$

where $\Delta_n(x) := \mu_n(x) - f_n^*$ is the expected difference in quality between the proposed point x and the previous best.

The expected improvement algorithm then evaluates at the point with the largest expected improvement,

$$x_{n+1} = \text{argmax} \quad EI_n(x)$$

breaking ties arbitrarily.

GPR algorithm

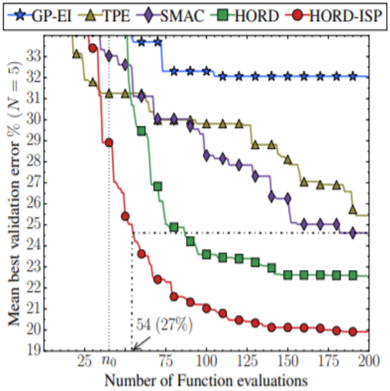
```
1. Place a Gaussian process prior on f
2. Observe $f$ at $n_0$ points according to an initial space-filling
   experimental design. Set $n = n_0$.
3. **while** $n \leq N$ **do**
4.     Update the posterior probability distribution on $f$ using all available
       data
5.     Let $x_n$ be a maximizer of the acquisition function (EI) over $x$,
       where the acquisition function is computed using the current
       posterior distribution.
6.     Observe $y_n = f(x_n)$.
7.     Increment n
8. **end while**
9. Return a solution: either the point evaluated with the largest $f(x)$, or
   the point with the largest posterior mean
```

Comparison

Source: Ilija Ilievski, Taimoor Akhtar, Jiashi Feng and Christine Annette Shoemaker (2017); Efficient Hyperparameter Optimization of Deep Learning Algorithms Using Deterministic RBF Surrogates

The authors compare Hyperparameter Optimization using RBFbased surrogate and DYCORS (HORD) against Gaussian processes with expected improvement (GP-EI), the Tree Parzen Estimator (TPE), and the Sequential Model based Algorithm Configuration (SMAC) algorithm on four DNN hyperparameter optimization problems with 6, 8, 15 and 19 hyperparameters.

Comparison



The figure plots the mean value of the best solution found so far as a function of the number of expensive evaluations of $f(x)$. The average is over five trials.

Exotic Bayesian Optimization: Examples

Definition

When one or more of underlying assumptions for Bayesian Optimization are broken, we call these **Exotic Bayesian Optimization problems**.

- Parallel Evaluations
- Noisy Evaluations
- Multi-Fidelity and Multi-Information Source Evaluations
- Random Environmental Conditions and Multi-Task Bayesian Optimization
- Derivative Observations

Exotic Bayesian Optimization: Parallel Evaluations

- Parallel evaluations is an appealing way to solve optimization problems in less time.
- **EI:**

$$\mathrm{EI}_n(x^{(1:q)}) = \mathrm{E}_n\left[[\max_{i=1,\cdots,q} f(x^{(i)}) - f_n^*]^+ \right]$$

  where $x^{(1:q)} = (x^{(1)}, \cdots, x^{(q)})$ are the set of points that jointly maximize this criteria.
- **Constant Liar approximation:** Here we choose $x^{(i)}$ sequentially by assuming that $f(x^{(j)})$ have been already observed for $j < i$, and have values equal to the expected value of $f(x^{(j)})$ under the posterior. This substantially speeds up computation.

Conclusion

We can look for some further research directions:

- So far multi-step optimal algorithms are extremely limited. Even in the asymptotic regime, we know very little about rates of convergence of Bayesian optimization algorithms.
- Also, we can look for new acquisition functions that may provide substantial better result in high dimensional set up.