A Tutorial on Bayesian Optimization

Peter I. Frazier

July 10, 2018

【算法】Bayesian Optimization

张楚珩
清华大学 交叉信息院博士在读

Bayesian optimization，即贝叶斯优化。

原文传送门

Frazier, Peter I. "A tutorial on bayesian optimization." arXiv preprint arXiv:1807.02811 (2018).

Introduction to Bayesian Optimization (slides)

特色

最近有做离子阱实验的同学涉及到一些实验参数的调参问题，其中主要需要用到贝叶斯优化。同时，我自己在想的一些问题也可能会用到贝叶斯优化。

过程

1. 问题设定

贝叶斯优化主要解决一个优化问题，即

> Consider a 'well behaved' function $f : \mathcal{X} \to \mathbb{R}$ where $\mathcal{X} \subseteq \mathbb{R}^D$ is a bounded domain.

$$x_M = \arg\min_{x \in \mathcal{X}} f(x).$$

该待优化函数 f 具有如下特点：

- Expensive to evaluate：每次给定一个 x 来获取 $f(x)$ 数值都有一定的成本，因此目标是尽可能少地去采样而找到一个好的解。
- Black box：它不具有一些特殊的结构性质，比如 convexity、linearity 等，因此有一些能够针对这些结构特性来加速优化的方法不能被使用；
- Derivative-free：无法获得它的导数，即不能使用一些基于导数来加速优化的方法；

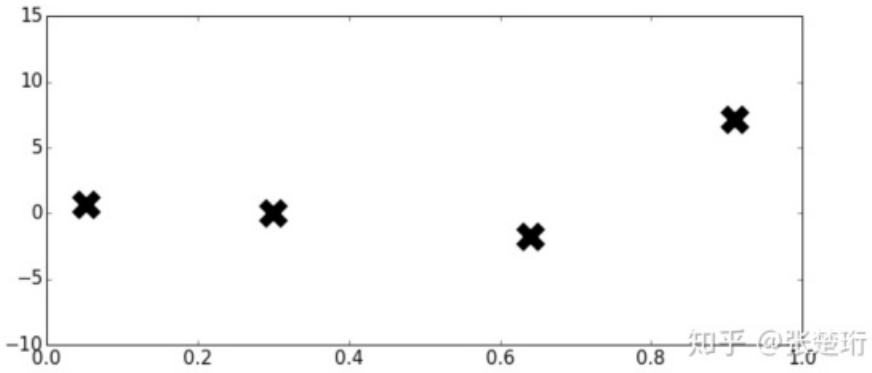
- (Maybe) noisy：对于函数 f 的一次采样得到的数值可能包含一个均值为零的噪声，因此即使在某一个点上也不能完全相信采样得到的数值；

该设定有着广泛的应用。比如在离子阱的实验中，需要调节激光器的各种参数，从而形成一个由电磁波产生的“陷阱”从而“困住”离子，而实验的目标是需要让激光形成一个足够好的“陷阱”使得被困住的离子的温度（能量）能够足够低。该系统中 x 就是相应的激光器参数，$f(x)$ 就是相应参数下形成离子阱的温度。每组实验测试需要花费几十分钟的时间（expensive to evaluate），因此不能测试很多参数；该过程涉及到非常多的物理过程而几乎无法解析地求解，因此只能把它当做一个黑盒子来优化（black box）。

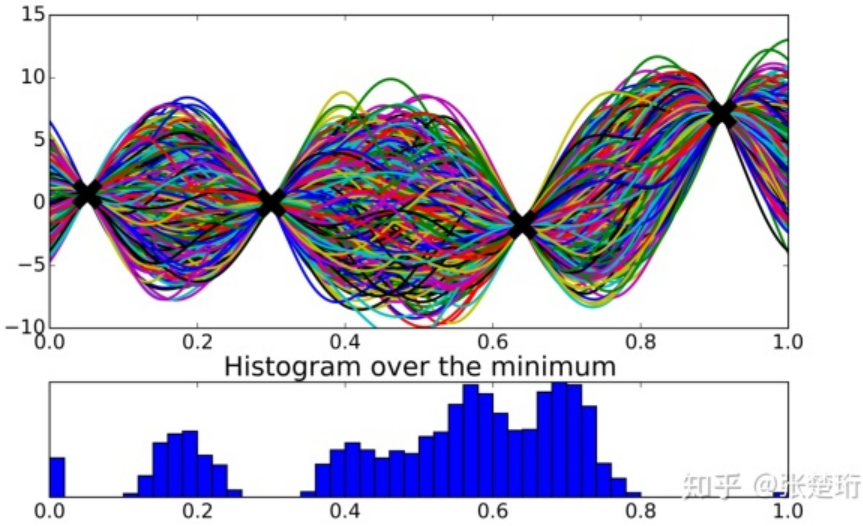
对于该类问题，有一些比较 naive 的解法，比如使用先验知识来确定参数（玄学）、使用 grid search（维度爆炸）、使用 random search（效率低）。因此，我们需要使用贝叶斯优化的方法来有效地优化这类问题。

2. 贝叶斯优化的基本成分

考虑如下一个例子，希望在 $[0,1]$ 区间上找到使得函数 $f(x)$ 取最小值的位置，函数 $f(x)$ 是 L-Lipschitz 的，并且是可导的；而所有的信息只有四个位置上的 $f(x)$ 值，如下图所示。



最简单的想法，就是把对于所有满足约束条件的函数进行采样，然后分别找出这些函数的最小值位置，把最常出现位置作为预测的最小值位置。如下图所示。



贝叶斯优化主要包括如下部分

- Prior: 确定函数 $f(x)$ 的先验分布
- Initial space-filling experimental design: 在函数的定义域上找一些尽可能分布均匀的初始点，并且得到它们相应的函数值；
- Posterior：通过一些概率模型（statistical model）根据已有的数据点来确定函数 $f(x)$ 的后验分布；
- Acquisition function：根据求得的后验分布来确定下一个（或者下一批）实验点。

以上过程可以概括为如下算法框架。

Algorithm 1 Basic pseudo-code for Bayesian optimization

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

下面就这几个部分来具体讲解。

3. Statistical model

概率模型最常用的还是高斯过程（Gaussian process），当然也有其他的模型，比如 random forest、t-student process。这里只讲高斯过程。

考虑 $k=n+1$ 个样本点，它们的先验分布为

$$f(x_{1:k}) \sim \text{Normal}\left(\mu_0(x_{1:k}), \Sigma_0(x_{1:k}, x_{1:k})\right), \tag{2}$$

假设已经知道了前 n 个样本点的函数值 $f(x_{1:n})$，可以得到第 $n+1$ 个点 $x=x_{n+1}$ 的后验分布：

$$\begin{aligned} f(x)|f(x_{1:n}) &\sim \text{Normal}(\mu_n(x), \sigma_n^2(x)) \\ \mu_n(x) &= \Sigma_0(x, x_{1:n})\Sigma_0(x_{1:n}, x_{1:n})^{-1}\left(f(x_{1:n}) - \mu_0(x_{1:n})\right) + \mu_0(x) \\ \sigma_n^2(x) &= \Sigma_0(x, x) - \Sigma_0(x, x_{1:n})\Sigma_0(x_{1:n}, x_{1:n})^{-1}\Sigma_0(x_{1:n}, x). \end{aligned} \tag{3}$$

具体计算的过程可以参考 [1]。

一般来说，先验 mean function 取为常数就好，即，$\mu_0(x)=\mu$。先验 kernel function 一般需要满足空间上相近的点关联性更强，比较常用的比如 power exponential / Gaussian kernel：

$$\Sigma_0(x, x') = \alpha_0 \exp\left(-||x - x'||^2\right),$$

或者 Matern kernel：

$$\Sigma_0(x, x') = \alpha_0 \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\sqrt{2\nu}||x - x'||\right)^{\nu} K_{\nu}(\sqrt{2\nu}||x - x'||)$$

这里面的若干系数可以作为算法的超参数。

4. Acquisition functions

通过选定的 statistical model，可以得到任意一个点 x 的后验分布，记该分布为 $\mathcal{N}(\mu(x;\theta,\mathcal{D}),\sigma(x;\theta,\mathcal{D}))$，其中 θ 表示先验分布相关的超参数，$\mathcal{D}$ 表示已有的样本数据。下面需要通过得到的后验分布来确定下一个试验点。下一个试验点可以通过 acquisition function 来得到，它是一个关于 x 的实值函数 $\alpha(x)$，选择一个使得 $\alpha(x)$ 取值最大的 x 作为下一个需要试验的点。

Expected Improvement（EI）

EI 是最为常用的 acquisition function 之一。考虑当我们最后需要从已有的采样点中选取一个点作为问题的解，考虑到每次采样是没有噪声的，因此截止 n 个样本时的最优解为 $y_{best}=\min_{m\le n} f(x_m)$。如果还有一次机会来多做一次采样，如果这一次采样得到的数值 y 比 y_{best} 还好，那么最终的结果就会由于这一次采样而有所改善。改善的数值可以记为 $\max(0, y_{best}-y)$。

因此，选择下一个试验点，使得该改善的期望最大。据此写出相应的 acquisition function

$$\alpha_{EI}(\mathbf{x};\theta,\mathcal{D}) = \int_y \max(0, y_{best} - y)p(y|\mathbf{x};\theta,\mathcal{D})dy$$

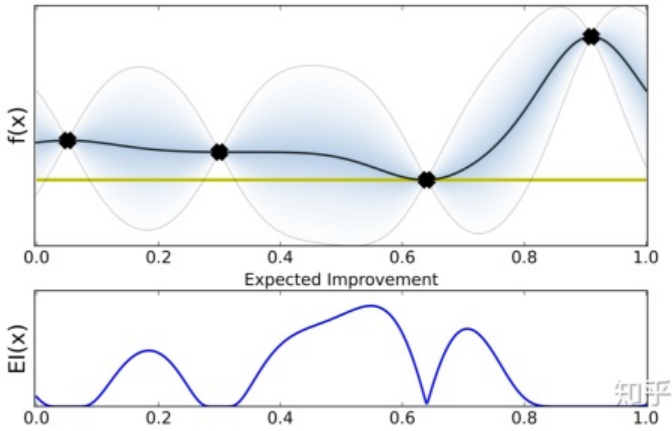
结合高斯模型，可以化简该函数的表达式

$$\alpha_{EI}(\mathbf{x};\theta,\mathcal{D}) = \sigma(\mathbf{x};\theta,\mathcal{D})(\gamma(x)\Phi(\gamma(x))) + \mathcal{N}(\gamma(x);0,1).$$

where

$$\gamma(x) = \frac{f(x_{best}) - \mu(\mathbf{x};\theta,\mathcal{D}) + \psi}{\sigma(\mathbf{x};\theta,\mathcal{D})}.$$

其中 $\Phi(\cdot)$ 为高斯分布的 CDF， ψ 是一个 explorative parameter，鼓励探索性地选择试验点。



Maximum probability of improvement（MPI）

这是最早被提出的一种 acquisition function。和前一种不同的是，它最大化的是下一个采样点能够产生非零 improvement 的概率，即

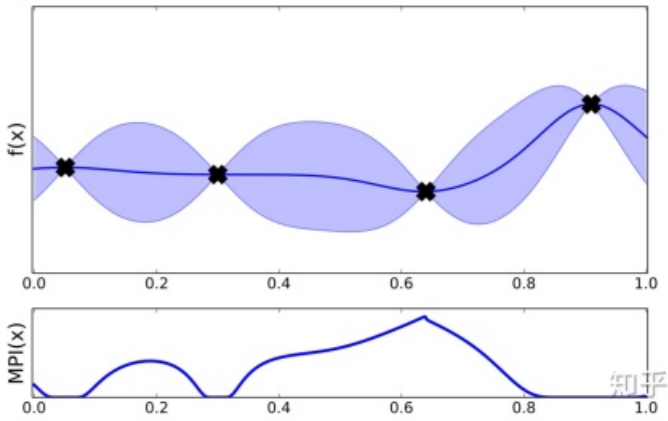
$$\gamma(\mathbf{x}) = \sigma(\mathbf{x};\theta,\mathcal{D})^{-1}(\mu(\mathbf{x};\theta,\mathcal{D}) - y_{best})$$
$$\alpha_{MPI}(\mathbf{x};\theta,\mathcal{D}) = p(f(\mathbf{x}) < y_{best}) = \Phi(\gamma(\mathbf{x}))$$

同样，可以加上一个常数 ψ 鼓励其探索。

$$\alpha_{MPI}(\mathbf{x};\theta,\mathcal{D}) = \Phi(\gamma(x))).$$

where

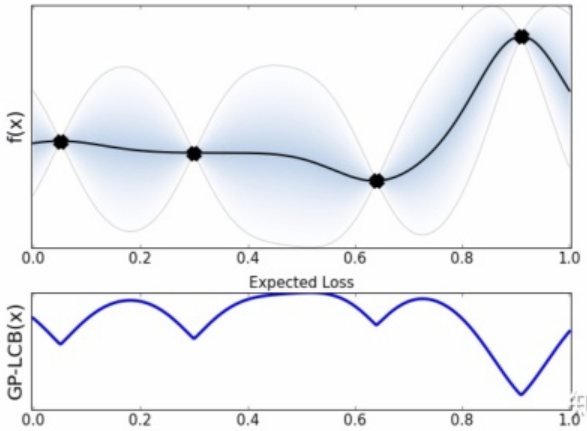
$$\gamma(x) = \frac{f(x_{best}) - \mu(\mathbf{x};\theta,\mathcal{D}) + \psi}{\sigma(\mathbf{x};\theta,\mathcal{D})}.$$



Lower confidence band（LCB）

其 acquisition function 如下

$$\alpha_{LCB}(\mathbf{x};\theta,\mathcal{D}) = -\mu(\mathbf{x};\theta,\mathcal{D}) + \beta_t\sigma(\mathbf{x};\theta,\mathcal{D})$$



Knowledge gradient（KG）

在 EI 中考虑在已有的样本点的值作为最后的解，这里考虑在估计的后验分布均值中找一个最优的值作为最后的解，即

$$\mu_n(\widehat{x^*}) = \max_{x'} \mu_n(x') =: \mu_n^*.$$

这样相应的 acquisition function 可以写为

$$\alpha_{KG}(x) = \mathbb{E}[\mu_n^* - \mu_{n+1}^* | x_{n+1} = x, \mathcal{D}]$$

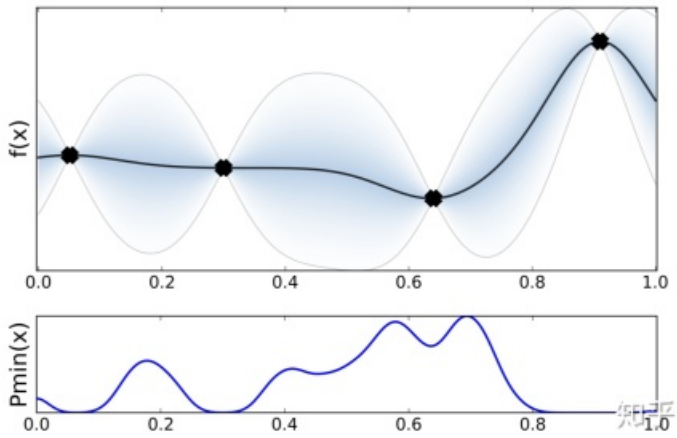
该函数的求解计算比较复杂，具体见第一篇文献。

Entropy search（ES）

其思想是希望找到一个样本点，使得得到改样本点的函数值之后，能够最大程度地更容易确定最优值的位置。即

$$\alpha_{ES}(\mathbf{x}; \theta, \mathcal{D}) = H[p(x_{min}|\mathcal{D})] - \mathbb{E}_{p(y|\mathcal{D},\mathbf{x})}[H[p(x_{min}|\mathcal{D} \cup \{\mathbf{x}, y\})]]$$

其缺点是上述公式不能够直接计算，需要通过采样的方式来近似计算。



Predictive entropy search（PES）

$$\alpha_{PES}(x) = H(p(y|\theta, \mathcal{D})) - \mathbb{E}_{x^*}[H(p(y|\theta, \mathcal{D}, x^*))]$$

其第一项是一个高斯分布的 entropy，能直接算；第二项仍然需要采样来近似。

5. Initial space-filling experimental design

最开始需要选择一些样本点，作为算法的初始样本数据，这部分数据选择的目标是尽可能“均匀”地分布在感兴趣的区域内。

有如下一些方法

- One point in the centre of the domain.
- Uniformly selected random locations.
- Latin design.
- Halton sequences.
- Determinantal point processes.

前两种方法比较 naive，没啥好解释的。

Latin design

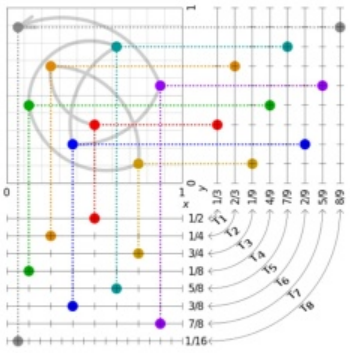
主要是希望每一行（不同的样本）每一列（不同的维度）上的数值都尽可能散开。形象地来说就是下面这样

A	B	F	C	E	D
B	C	A	D	F	E
C	D	B	E	A	F
D	E	C	F	B	A
E	F	D	A	C	B
F	A	E	B	D	C

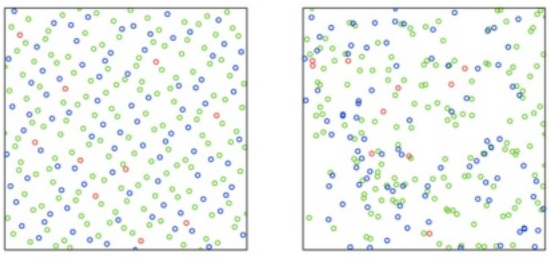
具体的实现可以参考 pyDOE: Randomized Designs。

Halton sequences

通过下面的这种规律来生成样本点，使得生成的点更为均匀地分布在空间中。



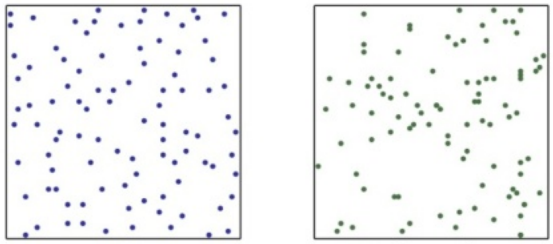
Better coverage than random.



Halton *Random*

Determinantal point processes

没太看明白，只放一张示意图吧。



6. 优化方法

前面定义了各种 acquisition function $\alpha(x)$ ，这些函数都能够通过各种方式计算或者近似（而不需要再对 f 进行采样）。但是我们的目标是找到一个 x 使得 $\alpha(x)$ 最大，因此这中间还需要用到各种优化的方法来辅助我们找到下一个试验点。主要的方法总结如下。

- Gradient descent methods: Conjugate gradient, BFGS, etc.
- Lipschitz based heuristics: DIRECT.
- Evolutionary algorithms: CMA.

需要注意的是有些 acquisition function 能够提供导数，而有些不可以。

7. 贝叶斯优化的推广

以下几方面的扩展也是目前贝叶斯优化研究比较多的方向：

- Noisy evaluation：主要考虑到每次采样得到的函数值可能带有一定的噪声，第一篇文献里面提到说 EI 没有考虑到该问题，但是可以做一个比较自然地拓展，即仍然考虑在已有的样本点上选取一个最优的值作为最后的解，但是考虑选取已有样本点上最优的后验均值函数值，即 $\mu_n^* = \max_{m<n} \mu_n(x_m)$ 。
- Parallel evaluation：前面考虑的是每次给出一个试验点，然后做实验得到该试验点是函数值；现在考虑每次给出一批试验点，然后得到这一批试验点的函数值。这样做的好处是，在实际中，可以并行地对多个试验点进行试验以提高效率。在此情况下，很容易对 EI 做一定的拓展，即

$$\mathrm{EI}_n(x^{(1:q)}) = E_n\left[\left[\max_{i=1,\dots,q} f(x^{(i)}) - f_n^*\right]^+\right], \tag{14}$$

- Multi-fidelity and multi-information source evaluation：考虑到实际中可能对于函数 f 的估计会有一些替代的方案。比如在神经网络超参调节上，可以在整个验证集上进行 evaluate，这样得到的估计比较准确，但是代价会比较大；也可以在部分验证集上进行 evaluate，这样估计的方差可能比较大，但是速度会比较快。该设定在我同学的离子阱实验中也存在，比如可以通过 CCD 拍到的照片大致判断温度的好坏（比如照片上有没有形成类似晶格的结构），也可以通过进一步的实验来直接确定离子阱的温度，但这样实验就会更复杂。这一类拓展就是在研究这个问题，在给出下一个试验点的同时，也希望给出一个方案，告诉我们下一个试验点使用怎样的实验条件来测量。
- Random environmental conditions and multi-task Bayesian optimization：有时候我们希望优化在各种实验条件下的平均性能，即

$$\max_x \int f(x,w)p(w)\,dw,$$
$$\max_x \sum_w f(x,w)p(w),$$

注意到其中 w 为控制不同实验环境的参数。这种情况下，其实可以对于每个试验参数 x 都在不同的环境下测试，然后求和作为 $f(x)$ ，但是这样效率显然不够高。这种拓展条件下，会研究更有效率的方法。

---

[1] Bishop, Christopher M.*Pattern recognition and machine learning*. springer, 2006. Chapter 6.4 Gaussian Process

发布于 2019-07-05

算法　强化学习 (Reinforcement Learning)