贝叶斯优化 - horovod系统整合

研究动机

在使用大量数据进行模型训练时，最好是将训练作业分配给多个GPU（单一实例或者多个实例）来进行分布式训练。深度学习框架提供内置方法以支持多GPU训练或分布式训练。

但除此之外，还有另外一种实现方法，即直接使用分布式深度学习框架（例如Horovod，byteps).。Horovod是Uber公司打造的分布式深度学习开源框架，能够与TensorFlow、Keras、PyTorch以及Apache MXNet等一线热门深度学习工具包协同使用。

bytePS是字节跳动于2019年开源的一款基于参数服务器架构的分布式训练框架。

在分布式训练框架中，有多个参数会对系统性能造成影响。以horovod中的几个参数为例。

参数名称	参数作用	说明
Fusion buffer	Tensor 合并的最大大小	实验表明，Tensor通信能否充分利用带宽和其大小有关，太小的tensor被latency所牵连，所以将不同gradients合并之后一起通信可以提高性能。当然，也不能全合并在一起通信，否则计算与网络通信重叠会受到影响
cycle-time	BackgroundThread 的轮询时间间隔	短的时间会增加Coordinator，并且也会影响fusion buffer的效果；太长的时间则会阻碍通信。
cache-capacity	response cache 的大小	Response cache过小，命中率下降，降低通信效率； Response cache过大，增大内存使用，降低系统性能；

在进行分布式训练时，horovod系统参数的正确选择对性能来说至关重要，然而手动去配置参数是非常繁琐并且复杂的，有时手动去配置参数甚至会降低训练性能。因此在训练过程中进行自动配置参数是非常重要的。

horovod中已经具有autotune的机制，可以在训练过程中进行寻找较优的参数配置，但是horovod autotune机制中可以调节的参数较少。只有以下几种。

1. Fusion-threshold-mb : fusion 大小

2. cycle-time-ms： horovod中一次循环的时间
3. cache-capacity： response cache 的大小
4. hierarchical-allreduce：是否开启层次allreduce
5. hierarchical-allgather： 是否开启层次allgather

并且通过实验发现，通过autotune进行以上几种参数的调整，并不能够选择出较优的参数配置，有时甚至不如默认的效果好, 并且使用autotun会对训练造成一定影响。

下表为不使用autotune、使用autotune、使用autotune获得的最佳参数配置进行训练的结果对照。

训练环境：

机器环境

	A	B	C	D	E	F	G
1	**实例规格**	**vCPU 内存（GiB）**	**内存（GiB）**	**GPU**	**GPU显存（GB）**	**网络带宽能力（出+入）（Gbit/s）**	**网络收发能力（出）（万PP**
2	ecs.gn6e-c12g1.2xlarge	48	368	V100*4	128	16	240

model = resnet50， epoch = 50， batch = 50

1. 系统中设置初始采样点的数量

训练方式	训练总时间/S
不使用autotune	391.1
使用autotune	695.2
使用autotune获得的最佳参数配置	406.9S

在进行分布式训练中，也有着其他参数会影响训练的性能，horovod系统中并没有考虑在内。例如：

1. NCCL相关参数
   a. num-nccl-streams： nccl 中 stream流的个数

   b. NCCL_SOCKET_NTHREADS ： socket连接中的辅助线程数
   c. NCCL_NSOCKS_PERTHREAD： 每个辅助线程中打开套接字的个数
   d. NCCL_BUFFSIZE： 在成对gpu之间传输数据时使用的缓冲区大小
   e. NCCL_NTHREADS： 每个CUDA的线程块中的CUDA线程数
   f. NCCL_ALGO Tree,Ring,Collnet： 选择使用的传输算法
2. MPI相关参数
3. 计算机系统参数
   a. CPU 频率.....
   b. GPU 频率.....

(添加可以调节参数)

如果在自动调整参数的过程中，将以上参数考虑在内进行搜索， 可以进一步提高autotune的效果。

并且horovod中当开启autotune时，对于每一次训练任务，都需要重新开始执行参数优化。对于相同的任务或者相似的任务，在进行参数自动优化时，并没有充分利用之前任务的调节信息，使得调节参数时间过长。

挑战：

1. 目前分布式训练框架在自动参数优化方面，可优化参数少。第一个挑战是包含分布式训练框架中内置参数、nccl参数、mpi参数以及计算机系统参数等参数，进行统一的参数调整。
2. 根据实验结果看，以horovod为例，经过参数自动调节所得到的最优配置，在训练过程中并不一定能够提高训练速度，甚至有时不如默认配置。第二个挑战提高分布式训练框架中autotune的有效性，使autotune能够找到较优的参数配置。
3. 根据实验结果看，以horovod为例，使用autotune，在一定程度上影响了训练速度。第三个挑战是减少autotune过程对训练效率的影响。
4. 目前的分布式框架中，autotune并没有充分使用先前任务调节参数的信息，使得调节参数时间过长。第四个挑战是充分利用先前任务调节参数信息，将当前任务与先前任务进行映射匹配，进一步减少参数搜索时间。

问题定义

在使用horovod进行训练时，有多个参数可以对训练性能进行影响。

设这些参数的个数为d个，则一组参数的值可以表示为 $< x_1, x_2, x_3, ..., x_d >$ 。

对于每个参数 $x_i(1 \leq i \leq d)$ 来说，参数的值的取值范围为 $dom(x_i)$ 。

设 $DOM \subseteq \prod_{i=1}^{d} dom(x_i)$,表示在整个系统中，可调节参数的取值空间。

当d个参数的值确定之后，在horovod系统中采用这些参数进行训练的效能为 y ， 此处 y 的值设置为每轮训练的时间。我们所要做的是在参数的取值空间内，找到最优的一组参数配置，使得每轮训练的时间y值最小。

设在整个系统的训练过程，参数与训练性能y的关系表示为

$y = h(x_1, x_2, x_3, ..., x_d)$,设 $X = (x_1, x_2, x_3, ..., x_d), X \subset DOM$,则

$y^j = h(X^j), X^j \subset DOM$

那么理想的最优参数配置 X^{best} 为y值最小时所对应的参数配置

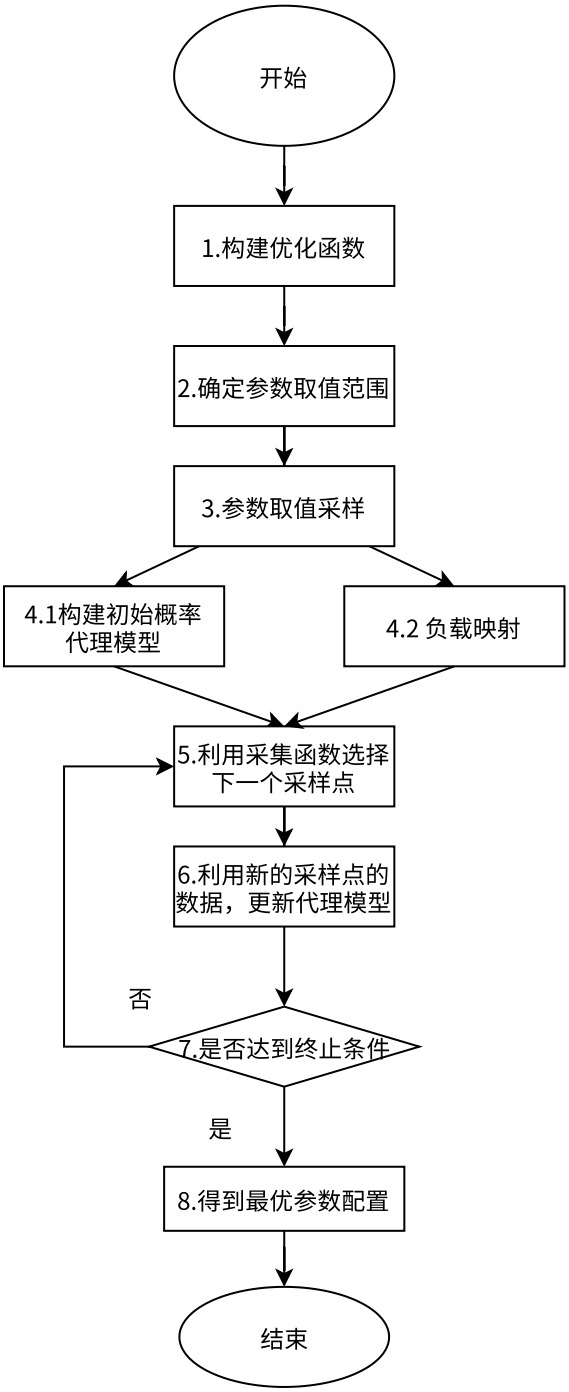
$y_{best} = min\ h(X^j), X^j \subset DOM$ 。

我们需要做的是在训练过程中，在有限的时间内，找到一组配置 X^* ，这组参数所对应的训练性能y，最接近 y^{best}

算法设计

流程图

1. 构建优化函数
2. 确定参数取值范围
3. 基于贝叶斯算法进行参数搜索
    a. 参数取值采样
    b. 基于初始数据构建初始概率代理模型与负载映射
        i. 基于初始数据构建初始概率代理模型
        ii. 基于初始数据进行负载映射
    c. 利用采集函数选择下一个采样点
    d. 获得c步骤中选择采样点的数据，并进行代理模型的更新
    e. 循环执行c,d步骤，直至到达终止条件
4. 得到最优参数配置



1. 构建函数

根据上文中问题定义部分，此系统的目标为寻找到一组参数配置，使得使用horovod进行分布式训练的训练性能，即训练时长最短。

针对此问题定义，构建黑盒函数

Input: $< x_1, x_2, x_3, ..., x_d >$ x_i 代表一个可调节参数的参数值

output: y ,即进行训练时，训练一个epoch所需要花费的时间

2. 确定参数取值范围

在此系统中，目前采取人为设置参数取值范围的方法。

在此系统中，参数根据数据类型可以分为两类，例如

1. 离散型参数
    a. Fusion-threshold-mb： 整数型
    b. cycle-time-ms： horovod中一次循环的时间
    c. num-nccl-streams： nccl 中 stream流的个数
2. 字符型参数
    a. hierarchical-allreduce：是否开启层次allreduce
    b. hierarchical-allgather： 是否开启层次allgather

由于此算法基于贝叶斯算法进行实现，贝叶斯算法只能优化连续性参数，所以需要对离散型参数以及字符型参数进行适配性调整。

1. 在黑盒函数输入时，输入的参数为原类型参数，即为离散或者字符型参数。需要对此进行**离散->连续**的适配性调整。
2. 在流程5-利用采集函数得到下一采样点中，基于贝叶斯的算法输出的数值为连续性数值。需要对此进行**连续->离散的**适配性调整。

由上可以看出，需要进行**离散<->连续**的双向映射。这里先使用一种简单的实现:

1. 在**离散->连续**过程中，直接使用离散的值带入，当做连续型数值。
2. 在**连续->离散**过程中，使用取整或者范围判定方式来进行处理，例如
    a. Fusion-threshold-mb : 6.7 转换为6
    b. Hierarchical-allreduce: 大于的值转换为开启，小于0的值转换为关闭

后续可以再探索更实用的方法。

3. 参数取值采样

在第二步中，我们得到了每个参数的取值空间以及转换办法，下面需要在参数的取值范围进行采样。

基于贝叶斯算法进行参数调节时，需要提供初始几组参数配置，以用来构建初始的代理模型。

采样方法:

1. 随机采样。

随机采样方法实现简单，缺点是对于高维的参数配置，少量的采样对于建立初始代理模型可能是无效的。

2. LHS采样

拉丁超立方采样（Latin hypercube sampling）是一种从多元参数分布中近似随机采样的方法，属于分层采样技术。具体为先分区再在每个分区内均匀采样，可以使采集的样本均匀分布在整个待抽样区域。

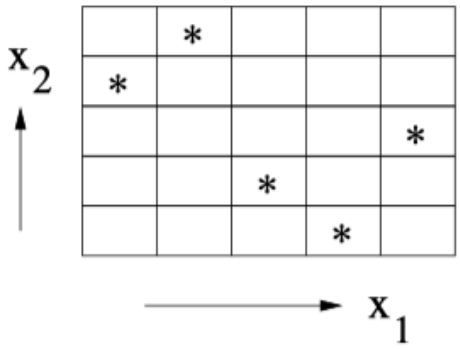


Figure 3: Example set of five LHS samples

优点: 可以通过少量的采样便可以使得样本均匀分布在整个参数空间内。

但是，LHS本身并不排除每次抽样都相同（例如，所有样本均沿对角线传播，这样的效果较差）。通过生成多组LHS样本，然后选择能最大化任何一对样本之间的最小距离的LHS样本来解决此问题。

在系统中，可以事先设置采样点的数量为k。

对于整数型参数，在分区时，可以将取值范围均匀分为k份，然后在每一块区间内随机取值。

对于字符型参数，在分区时，如果取值个数超过k个，则与整数型参数取值类似，先均匀分区，然后进行取值；如果取值个数小于等于k，则直接按取值进行分区。

3. 后续可以再对比选择其他算法。

4. 构建初始概率代理模型与负载映射

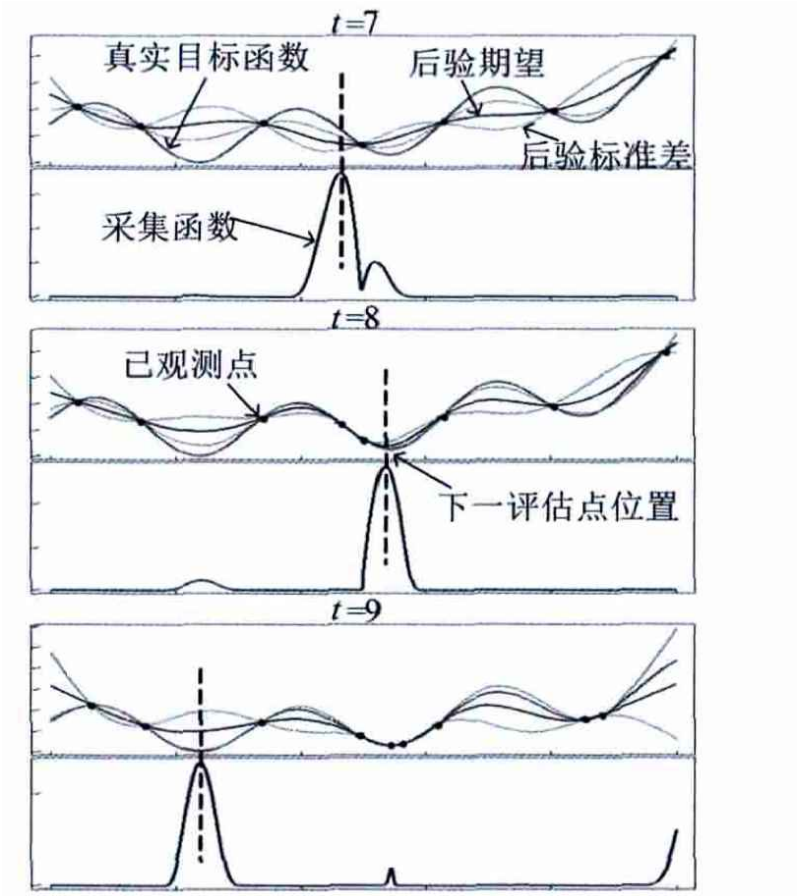
4.1 构建初始代理模型欧

贝叶斯优化框架中主要包含两个核心模块，概率代理模型以及采集函数。这里描述的是构建概率代理模型。

在问题定义中，我们假设 $y = h(x_1, x_2, x_3, ..., x_d)$ ，即h为分布式训练框架中，参数配置与训练性能的映射关系，一个函数。此函数对我们来说是不可知的，我们需要做的就是去尽可能的拟合此函数。

概率代理模型的作用是去代理h函数，即尽可能的去构建出h函数。 从假设先验开始，然后通过迭代的增加信息量，修正先验，从而使得概率代理模型越来越接近h函数。

举个例子，下图为贝叶斯优化框架应用在一维函数 f(x)=(x-0.3)2+0.2*sin(20x)上 3 次迭代的示例.



每一次迭代之后，都会新增加一组数据，得到新的数据之后，便可以对先验进行修正，这样依次迭代，概率代理模型便可以越来越接近真实目标函数。

构建初始概率代理模型与更新概率代理模型的核心步骤相同，区别是在构建初始概率代理模型时，采用的数据是事先采样得到的。

概率代理模型

常见的几种概率代理模型如下：

随机森林

随机森林回归是一种非常适合并行化的回归方法，该方法属于集成学习 ,即通过组合多个弱学习器来提高预测精度 .随机森林回归构造多棵决策树 ,每棵决策树通过从训练数据中有放回的采样进行训练. 当需要预测时,把采样点输入到每棵决策树中,并得到每棵树的预测均值,然后通过投票机制得到最终预测结果.

深度神经网络

深度神经网络通常是指层数超过 2 层的神经网络 ,虽然具有无限多个隐层单元的神经网络等价于高斯过程,但该神经网络具有无穷多个参数,无法训练.为了减少参数个数,一种常用的方法就是增加神经网络的深度。近年来,由于其优越的性能,深度神经网络已成功应用于语音识别 、机器视觉等领域.在贝叶斯优化领域中 ,深度神经网络同样得到重视。

高斯过程

高斯过程是多元高斯概率分布的范化，是一个随机变量的集合 ,存在这样的性质 :任意有限个随机变量都满足一个联合高斯分布。一个高斯过程由一个均值函数和一个协方差函数构成的。

比较

1. 相比较于高斯过程，随机森林算法计算效率高。随机森林回归在训练数据附近能够快速得到高精度预测 , 但在远离训练数据时的预测效果通常很差，即随机森林算法能够很好的利用原来的数据进行exploitation，但是无法对远离训练数据进行有效预测，exploration效果差。

   虽然随机森林算法可以进行并行化，进一步增加效率，但是**并行化不适合此问题场景。**
2. 想要使用深度神经网络来得到理想效果的函数近似,需要合理地设计神经网络架构,如层数、每层的神经元个数等. 较为复杂。

综合考虑，目前使用高斯过程作为概率代理函数。

概率代理模型的更新

假设真实的目标函数f(x)服从高斯分布，即 $f(x) \sim GP\left(E(x), K\left(x, x'\right)\right)$ ，这里为简化描述，x即为一组参数配置，即x为一组向量。 x_i 代表第i次的参数配置。

$E(x)$ 即为高斯分布的均值函数，这里设置均值函数 $E(x)=0$ ， $K\left(x, x'\right)$ 即为协方差函数(也被称为核函数）。

假设经过上述t组训练之后，我们得到了t组数据，并且得到了以下的核函数。

$$K=\begin{bmatrix} k\left(x_1, x_1\right) & \ldots & k\left(x_1, x_t\right) \\ \vdots & \ddots & \vdots \\ k\left(x_t, x_1\right) & \ldots & k\left(x_t, x_t\right) \end{bmatrix}$$

当我们得到一组新的样本 (x_t, y_t) x_t 为参数配置， y_t 为此参数配置下一个epoch训练时长。下面所需要做的就是去更新核函数。

设 $k = [k(x_{t+1}, x_1), k(x_{t+1}, x_2), \ldots, k(x_{t+1}, x_t)]$

则 $K = \begin{bmatrix} K & k^T \\ k & k(x_{t+1}, x_{t+1}) \end{bmatrix}$

这样，利用更新后的协方差矩阵，我们便可以利用前t个样本去估计出f(t+1)的后验概率分布

$$P(f_{t+1} \mid D_{1:t}, x_{t+1}) \sim N(\mu, \sigma^2)$$
$$\mu = k^T K^{-1} f_{1:t}$$
$$\sigma^2 = k(x_{t+1}, x_{t+1}) - k^T K^{-1} k$$

4.2 负载映射

在进行初始化采样点之后，在系统运行期间，便可以进行系统负载信息的采集。负载信息采集完成之后，

收集负载信息

需要收集负载的信息

负载类：

a. 网络负载
   i. 网络速度： Gbps， 每隔1s进行计算
   ii. 流量峰值： 统计1S内的流量峰值
   iii. 一段时间内的流量值（1S，10S，1min..)： 统计一段时间内的流量总值
b. cpu负载
   i. cpu负载： 训练期间使用与等待CPU的任务数 每隔1S进行收集 (top)
   ii. cpu利用率： 训练期间实时占用的CPU百分比 每隔1S进行收集 (top)
   iii. IO： r/s 和 w/s：每秒磁盘读写的次数；avgqu-sze：平均 IO 队列长度 (iostat)
   iv. 训练进程使用的物理 虚拟 共享内存 (top)
c. gpu负载
   i. 显存使用率 (nvidia-smi)
   ii. GPU利用率 (nvidia-smi)
d. 训练相关
   i. tensor数目 (可以在horovod中得到)
   ii. tensor大小 (可以在horovod中得到)

负载映射

对于所收集的负载信息，假设系统中一共有k个负载描述，对于一个需要调节的任务，在已经过初始化采样并执行之后， $S =< w_1, w_2, w_3, ..., w_k >$ ，并且已经存在了很多组负载信息，即存在

$S_1, S_2, S_3, ...S_q$ 。对于新的一个任务，即需要在之前的负载记录中，找到与此任务负载最为接近的信息。则需要计算 $distance(S, S_i)1 <= i <= q$.

目前考虑使用欧几里得距离来计算distance

负载压缩

如上文中提到，需要在根据初始化采样点进行训练期间，进行负载信息收集，在训练过程中，收集负载信息数据量大，对于计算 $distance(S, S_i)$ ，计算成本高，因此可以对收集的负载信息进行压缩，降低计算成本。

PCA降维

因子分析（Factor Analysis）

k-means聚类分析

5.利用采集函数选择下一个采样点

在上一步我们得到了概率代理模型，也就是说，针对对一组参数配置，我们可以得到所对应的训练性能的后验概率分布，即得到所对应训练性能的均值与方差。注意，这里只是估算的值，并不是真正的值，后面我们仍需要将一组参数带入实验，去获取到真正的训练性能，然后再去更新概率代理模型。

上面提到贝叶斯优化算法的核心有两部分，第一部分是概率代理模型，第二部分就是采集函数。

我们可以结合上面得到的均值、方差与采集函数去减少采样的次数。

定义采集函数的目标就是为了有目的的去选取下一次采样点。采集的目标有两个方向：

1. explore，尽可能的探索未知的空间，这样对f(x)的后验概率才会更接近f(x)
2. exploit，强化已有的结果，在现有最大值的附近进行探索，保证找到的f(x)会更大

常见的采集函数有三种：

probability of improvement(POI)

这种方法考虑的是让新的采样能提升最大值的概率最大

假设现在的最大值为 $f(x^+)$ ，那么acquisition函数为：

$$\mathrm{PI}(\mathbf{x}) = P\left(f(\mathbf{x}) \geq f\left(\mathbf{x}^+\right)\right) \quad = \Phi\left(\frac{\mu(\mathbf{x}) - f\left(\mathbf{x}^+\right)}{\sigma(\mathbf{x})}\right)$$

Φ 表示的是正态累计分布函数.

这个函数会更倾向于exploit，而不是explore，因此很有可能会收敛到接近f(x+)附近的位置。为了解决这个问题，可以添加一个trade-off系数。

$$\mathrm{PI}(\mathbf{x}) = P\left(f(\mathbf{x}) \geq f\left(\mathbf{x}^+\right) + \xi\right) \quad = \Phi\left(\frac{\mu(\mathbf{x}) - f\left(\mathbf{x}^+\right) - \xi}{\sigma(\mathbf{x})}\right)$$

这样就会防止 $f(x^+)$ 附近的，非常细微的提升。

这里的系数ξ可以自己定义，可以通过动态调整这个系数的大小来控制偏向explore还是偏向exploit。

Expected improvement(EI)

使用EI作为acquisition function是一个在explore和exploit之间平衡的一个不错选择。explore时，应该选择那些具有比较大方差的点，而在exploit时，则应该优先考虑均值大的点。

POI是一个概率函数，描述的是新的点能比当前最大值大的概率，但是大多少并不关心。

EI则使用的是数学期望，因此"大多少"这个因素也被考虑在内

$$\mathbb{E}\left(\max\left\{0, f_{t+1}(\mathbf{x}) - f\left(\mathbf{x}^+\right)\right\} \mid \mathcal{D}_t\right)$$

$$\mathrm{EI}(\mathbf{x}) = \begin{cases} \left(\mu(\mathbf{x}) - f\left(\mathbf{x}^+\right)\right)\Phi(Z) + \sigma(\mathbf{x})\phi(Z) & \text{if } \sigma(\mathbf{x}) > 0 \\ 0 & \text{if } \sigma(\mathbf{x}) = 0 \end{cases}$$

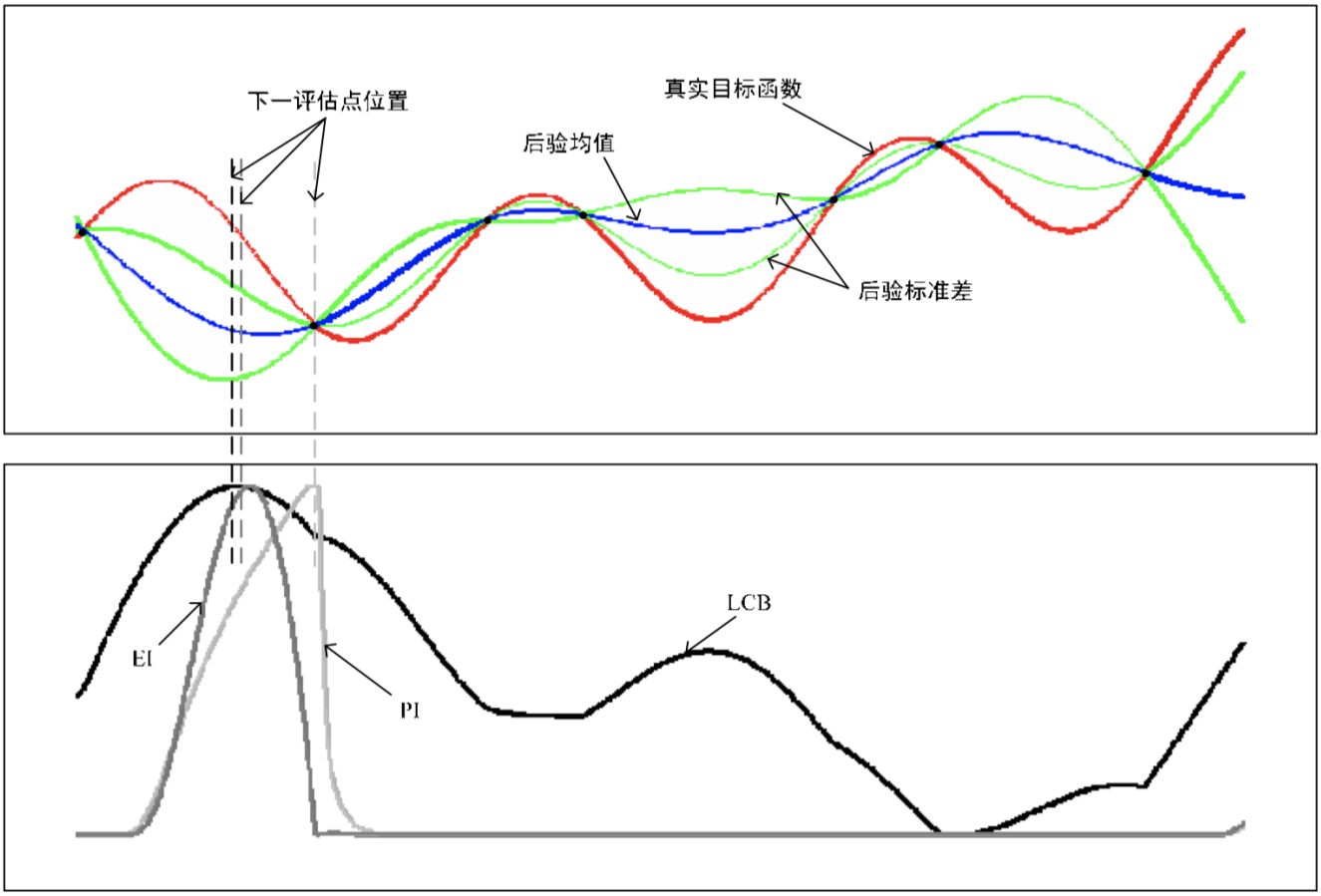
$$Z = \frac{\mu(\mathbf{x}) - f\left(\mathbf{x}^+\right)}{\sigma(\mathbf{x})}$$

Upper confidence bound

除了EI和POI，有一个更简单的算法，直接比较置信空间中的最大值。

$$\mathrm{UCB}(\mathbf{x}) = \mu(\mathbf{x}) + \kappa\sigma(x)$$

在进行选择采样点的时候，确定了采集函数之后，选择可以使得采集函数值最大的点即可，即为下一个采样点。



如上图所示

- 对于PI采集函数，在真实最优解附近时，PI的值很大，并且在远离最优解时，PI的取值小，系数ξ的取值会对PI的曲线造成影响，当ξ较大时，曲线较为平缓，此时更有利于explore；当ξ较小时，曲线较为陡峭，此时更有利于exploit
- 对于EI采集函数，EI即考虑了是否提升，也考虑了提升的效果，从图中可以看到，EI选择了与一个与PI不同的采样点，此点的后验均值较小，并且可能提升的效果好。
- 对于UCB(与LCB近似，分别取得置信空间的最大值与最小值), 从图中可以看出，取了置信空间中的最小值对应的点。

6. 更新概率代理模型

概率代理模型更新方式见第四步-构建初始概率代理模型

7. 是否达到终止条件

在此系统中，可以根据以下几个方面判断是否到达自动优化的终止条件

1. 是否达到调节次数限制
2. 是否已经达到优化目标
3. 是否人为终止

8. 得到最优参数配置

在参数自动优化的过程中，会将各个参数配置以及对应的训练性能记录下来，最终会选取最优训练性能对应的参数配置为最优参数。