超参数调优的方法

笔记本：	机器学习		
创建时间：	2019/4/11 9:33	**更新时间：**	2019/5/13 14:48
标签：	超参优化		
URL：	https://arimo.com/data-science/2016/bayesian-optimization-hyperparameter-tu...		

超参数调优的方法

超参数调整可能是机器学习中最棘手但最有趣的主题之一。对于大多数机器学习从业者而言，掌握调整超参数的艺术不仅需要机器学习算法的扎实背景，还需要使用真实数据集的丰富经验。

在这篇文章中，我将概述超参数和智能调整超参数的各种方法。特别是，我将解释我们如何应用贝叶斯优化来调整真实世界的数据集去预测模型的超参数。

1 什么是超参数？为什么重要？

机器学习是针对特定类别的任务T，利用学习经验（或数据或证明）E，提高性能得分P的过程。实际上，这意味着几乎每个机器学习模型都有一组特定的参数，必须从数据集E中估计这些参数，以便最大化性能评分P。

例如，考虑一个线性回归模型，在给定数据集$E = \{x_i, y_i\}_{i=1}^{N}$的情况下，我们希望准确地预测每个输入向量xi的目标yi。线性回归模型通过假设xi和yi之间的线性关系来实现。通过以下公式：

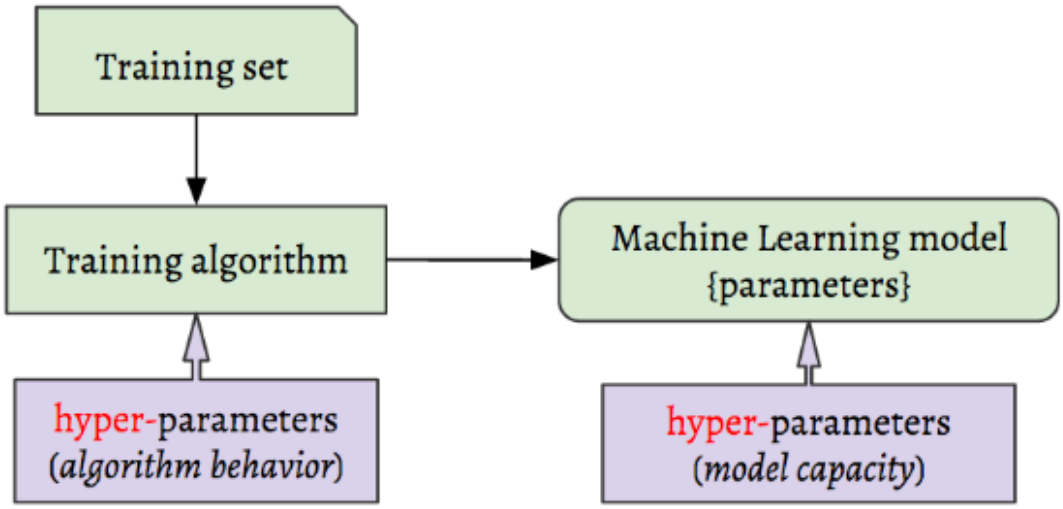
$$y = w^T x$$

其中w是权重向量，其值将由线性回归的训练算法确定。实际上，它被称为线性回归模型的参数，即在训练期间学习的是模型参数。

为了从数据集E中估计w，需要应用训练的算法。训练算法本身也可能有自己的参数，例如，梯度下降具有学习率、提前停止标准等作为参数。为了区分训练算法参数，我们将其称为超参数。超参数很重要，因为它们直接控制训练算法的行为，并对正在训练的模型的性能产生很大的影响。

此外，可能存在控制模型性能的超参数，例如决策树的最大深度，随机森林中树木的数量，岭/lasso回归中的正则化系数，SVM中的惩罚系数，这些性能超参数指定了模型的灵活性，可以看作控制模型偏差与方差的旋钮。

下图描绘了超参数的交互。超参数在训练算法开始之前指定，并且不能在训练算法本身内进行优化。超参数的第一选择不会产生最佳的模型性能，因此通过调整参数并采取另一个训练模型的步骤来调整超参数 - 本质上是一个优化过程。如果训练算法需要很长时间才能运行，则超参数调整的整个过程将花费更长时间，因为它需要多次运行训练算法才能找到接近最优的超参数配置。



最优的超参数集很大程度上取决于模型和数据集。即使使用相同的模型，最佳超参数也可能因数据集而异。因此，必须有一个自动化的步骤，我们为每个新数据集分别调整超参数。

2 超参数调整算法

调整超参数的直接方法是基于人类的专业知识。经验丰富的机器学习从业者大致了解如何选择好的超参数。对于新数据集，他们将遵循具有各种超参数配置的试错过程。这种使用超参数进行实验的过程是启发式的，不同经验的人可能会提出不同的设置，并且该过程不易重现。

还有一个真实的风险是，人类无法实现接近超参数的最佳设置。人类不善于处理高维数据，并且在尝试调整多个超参数时很容易误解或错过趋势和关系。例如，在调整两个参数时，从业者通常会回退到调整一个参数然后调整第二个参数。这可能导致在调整第二个超参数时性能的总体改进趋于平稳，而通过返回更改第一个超参数可以获得更多改进。这种困难需要一种自动且可重复的超参数调整方法。本节概述了最流行的方法。

2.1 网格搜索

在网格搜索中，我们尝试一组超参数配置并相应地训练算法，选择提供最佳性能的超参数配置。在实践中，从业者指定超参数的值之间的界限和步骤，以便它形成配置网格。从业者通常从参数值之间具有相对较大步长的有限网格开始，然后在最佳配置下扩展或使网格更精细并继续搜索新网格。此过程称为手动网格搜索。

网格搜索是一种昂贵的方法。假设我们有n个超参数，每个超参数有两个值，那么配置总数就是2^n。因此，仅在少量配置上进行网格搜索才是可行的。这种方法的真正痛点称为 curse of dimensionality（维数灾难）。这意味着我们添加的维数越多，搜索在时间复杂度上会增加得越多（通常是指数级增长）。

幸运的是，网格搜索是可以并行的，每个工作人员在不同的参数设置上工作，这使得在足够的计算能力下网格搜索更加可行。

2.2 随机搜索

在随机搜索超参数优化中，作者证明了网格搜索的低效率，并且显示了一个令人惊讶的结果：通过随机搜索超参数网格，可以获得与完整网格搜索类似的性能。作者表明，如果超参数的接近最优区域占据整个搜索空间的5%以上的话，通过一定数量的试验（通常为40-60次试验）进行随机搜索将有较高可能找到该最优区域。

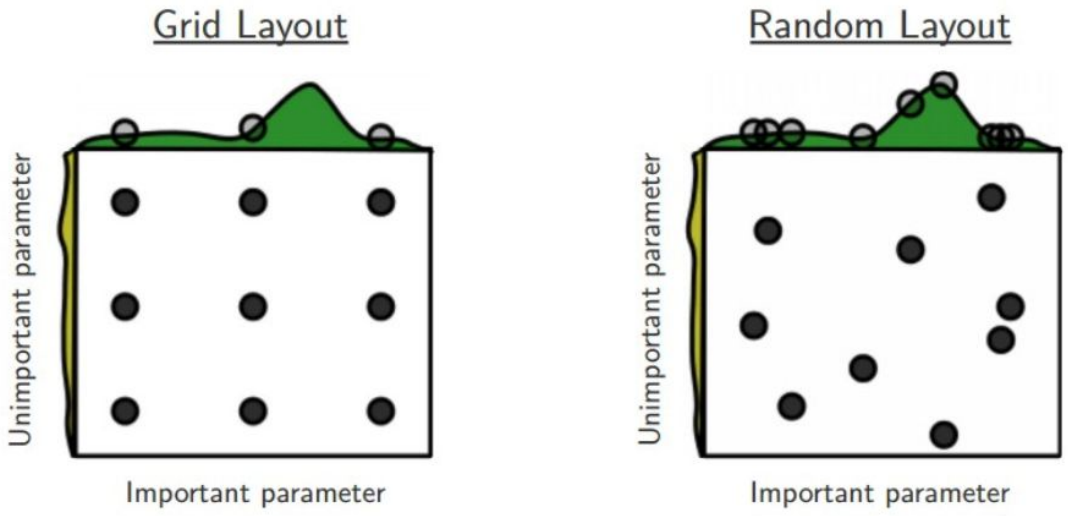


Image from http://jmlr.csail.mit.edu/papers/volume13/bergstra12a/bergstra12a.pdf

在网格搜索中，很容易注意到，即使我们已经训练了 9 个模型，但我们每个变量只使用了 3 个值！然而，使用随机布局，我们不太可能多次选择相同的变量。结果是，通过第二种方法，我们将为每个变量使用 9 个不同的值训练 9 个模型。从每个图像布局顶部的曲线图可以看出，我们使用随机搜索可以更广泛地探索超参数空间（特别是对于更重要的变量）。这将有助于我们在更少的迭代中找到最佳配置。

总结：如果搜索空间包含 3 到 4 个以上的维度，请不要使用网格搜索。相反，使用随机搜索，它为每个搜索任务提供了非常好的基准。 随机搜索非常简单和有效，因此它被许多实践者认为是调优超参数的实际方法。像网格搜索一样，是可以并行的，但试验的次数要少得多，而性能却是相当的。

2.3 贝叶斯优化

在网格搜索和随机搜索中，我们是随机和盲目地尝试配置，下一次试验是独立于之前进行的所有试验。相比之下，贝叶斯优化的下一次试验是参考了之前进行的所有试验而选择的，通常，它将首先在多个配置中收集性能，然后进行一些推断并确定接下来要尝试的配置，目的是在找到良好的最佳状态时尽量减少试验次数，但这个过程本质上是顺序的，不容易并行化。

首先我们把一个具体的机器学习模型的超参数配置记为 λ（一般有多个维度），效果指标记为 $f(\lambda) = g(\lambda, D_{train}, D_{valid})$，则我们的优化问题可以定义为 $\lambda^* = arg \min_{\lambda} f(\lambda)$，其中 $f(\lambda)$ 是我们的优化目标函数，λ^* 是我们的最优机器学习模型超参数配置，如果使用k折交叉验证，优化目标函数就是

$$f(\lambda) = \frac{1}{K}\sum_{k=1}^{K} g(\lambda, D_{train}^k, D_{valid}^k)$$

，而这个 $f(\lambda)$ 是一个黑箱函数，即我们除了执

行模型训练并验证效果指标以外没有其他方法获得$f(\lambda)$的信息。贝叶斯优化通过随机尝试一些配置参数$\lambda_1, \lambda_2, ..., \lambda_n$，并将这些参数应用到优化目标函数，训练验证得到$f_1, f_2, ..., f_n$，然后我们根据这些$f(\lambda)$的采样值，通过贝叶斯公式推断出$f$在任意$\lambda$的后验概率分布$p(f|\lambda)$。根据这个后验分布我们可以选择一个当前信息下最优的λ^*作为下一次训练验证尝试的配置参数。可以看到贝叶斯优化是一个顺序优化的过程，两个关键步骤是：计算$f(\lambda)$的后验分布和在后验分布下寻求最优的λ^*。其中第一个步骤需要对$f(\lambda)$进行统计建模，第二个步骤需要将统计建模产出的后验分布转化为一个可优化的确定性的目标函数，进而优化得到最优解。

通过在目标函数的每次评估之后不断更新替代概率模型来使上面的方法，在高层次上，贝叶斯优化方法是有效的，因为它们以明智的方式选择下一个超参数。通过评估过去结果看起来更有希望的超参数，贝叶斯方法可以在较少的迭代中找到比随机搜索更好的模型设置。我们形成了一个世界的初始视图（称为先验），然后我们根据新的经验更新我们的模型（更新的模型称为后验），贝叶斯超参数优化采用该框架并将其应用于寻找模型设置的最佳值。

3 基于顺序模型的全局优化

而基于顺序模型的全局优化（Sequential model-based optimization，SMBO）是贝叶斯优化的形式化，顺序是指一个接一个地运行试验，每次通过应用贝叶斯推理和更新后验概率分布（代理模型）来尝试更好的超参数，该方法使用代理模型来逼近真正的黑盒函数。

基于模型的超参数优化有五个方面：
1. 要在其上搜索的超参数域
2. 一个目标函数，它接收超参数并输出我们想要最小化（或最大化）的分数
3. 目标函数的代理模型（后验概率分布）
4. 一种称为获得（采集）函数（Acquisition Function）的标准，用于评估下一个从代理模型中选择的超参数
5. 由算法用于更新代理模型的（分数，超参数）对组成的历史记录

在步骤3-4有几种变体 ，即它们在历史参数下如何构建目标函数的代理以及用于选择下一个超参数的标准存在差异，几种常见的方法是高斯过程、随机森林回归和树Parzen估计（TPE），而这三种方法在步骤4的选择的都是期望提升(Expected Improvement)。

- 高斯过程（论文：贝叶斯优化）：使用高斯过程对代理函数进行建模，并且通常会优化预期的改进，即新试验在当前最佳观测的基础上改进的预期概率。高斯过程是函数上的分布，高斯过程的一个样本是一个完整的函数，训练高斯过程包括将这个分布与给定的数据进行拟合，从而生成与观测数据接近的函数。利用高斯过程，可以计算出搜索空间中任意点的期望改进量，给出最高期望的改进将在下一步进行尝试。贝叶斯优化通常为连续超参数(如学习率、正则化系数等)提供非平凡的、网格外的值，并在一些良好的基准数据集上显示出优于人类的性能。著名的实现是Spearmint（或者`bayesian-optimization`）。
- 随机森林回归（论文：基于顺序模型的算法配置 （Sequential Model-based Algorithm Configuration，SMAC））：使用基于回归树的随机树林对目标函数进行建模，从随机森林认为最优的区域（高预期改进）中抽取新点。可以在此处找到实施方案。

- TPE（论文：树形结构Parzen估计（Tree-structured Parzen Estimator，TPE））：是SMAC的改进版本，其中两个分离的模型用于模拟后验。众所周知的TPE实现是hyperopt。

3.1 域

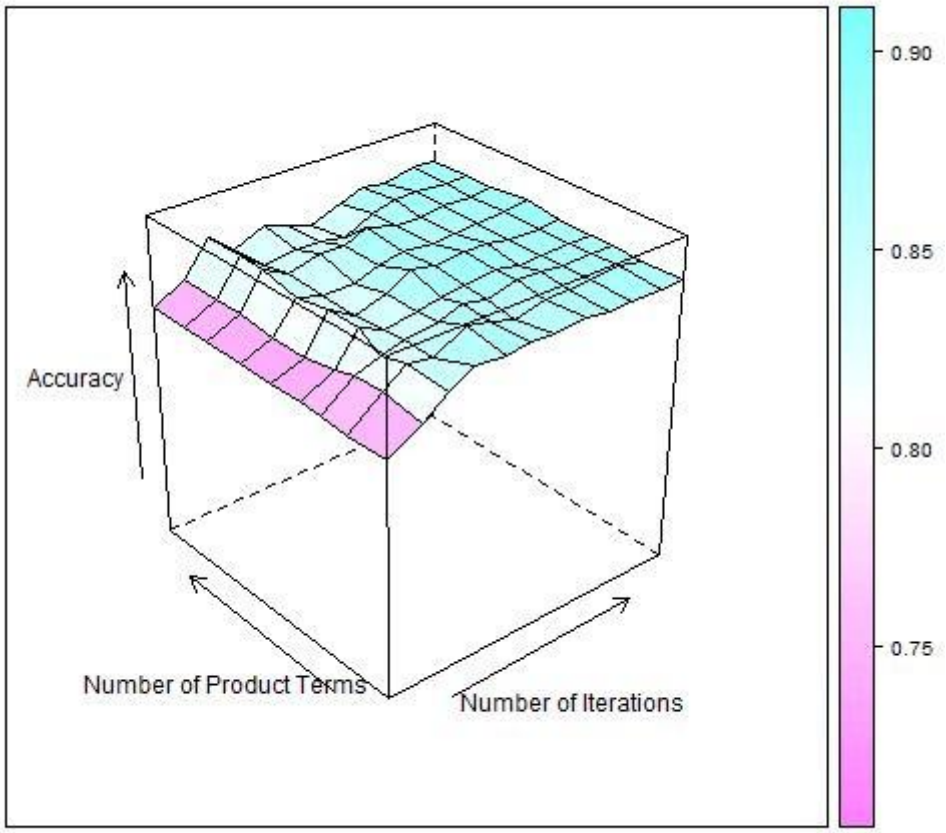
在随机搜索和网格搜索的情况下，我们搜索的超参数域是网格。而对于基于顺序模型的方法，域由概率分布组成。与网格一样，这使我们可以通过在我们认为真正最好的超参数所在的区域中放置更大的概率来将域知识编码到搜索过程中。

3.2 目标函数

目标函数采用超参数并输出我们想要最小化（或最大化）的单个实值得分。例如，让我们考虑为回归问题构建随机森林的情况。我们想要优化的超参数显示在上面的超参数网格中，最小化的分数是均方根误差。虽然目标函数看起来很简单，但计算起来非常昂贵，如果可以快速计算目标函数，那么我们可以尝试每个可能的超参数组合（如在网格搜索中），比如我们使用简单模型、小超参数网格和小数据集，那么这可能是最好的方法。但是，如果目标函数可能需要数小时甚至数天来评估，我们希望限制对它的调用。
基于贝叶斯模型的优化的整个概念是通过仅选择最有希望的超参数集来基于先前对评估函数的调用来减少目标函数需要运行的次数。基于称为代理项的目标函数的模型来选择下一组超参数。

3.3 代理模型（后验概率分布）

代理**模型**（也称为响应表面）是使用先前评估构建的目标函数的概率表示。这有时称为响应面，因为它是超参数与目标函数上得分概率的高维映射。下面是一个只有两个超参数的简单示例：



3.3.1 高斯过程（Spearmint 或 bayesian-optimization实现）

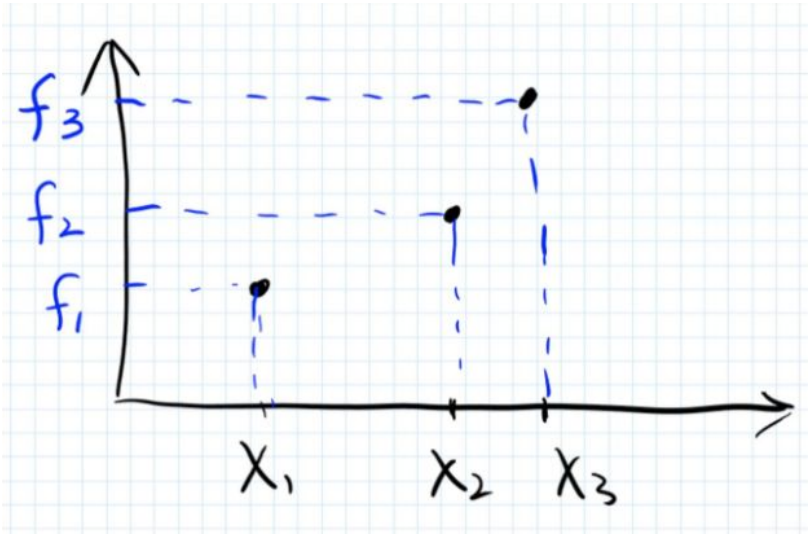
由于f是一个函数, 我们的假设便是对于任意给定的参数组合x′ ， f(x')服从高斯分布, 那么对于整个函数f便有

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

这里的m(x), k(x,x')是关于x的函数.

高斯过程(定义在函数上的正态分布): 是一组随机变量的集合，这个集合里面的任意有限个随机变量都服从联合正态分布。

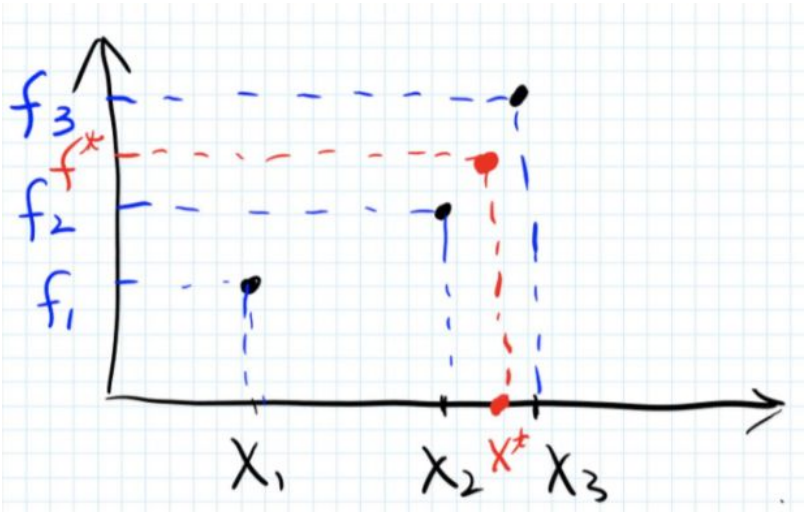
举个例子, 假设我们的机器学习算法只需要对一个超参数调参, 即x为一维矢量, 并且目前我们已将尝试了3个x, (将其记为x1,x2, x3), 得到了3个error rates (记为f1, f2,f3), D1:3={(x1,f1),(x2,f2),(x3,f3)}, 如下图所示:

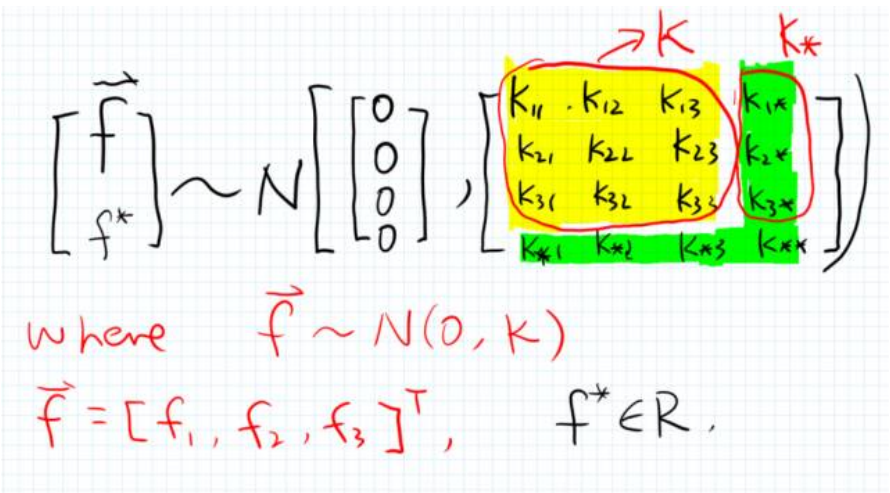


为方便理解, 假设m(x)=0, 那么根据高斯过程的定义, 我们得到:

$$\begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix} \right)$$

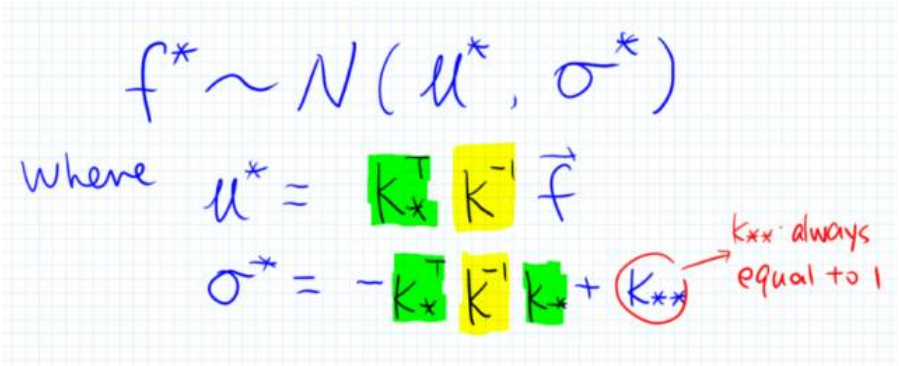
在给定一个x*, 在不进行建模的情况下, 我们可以根据已知的f1, f2,f3和高斯过程的特性得到:





进而根据多元高斯分布可以方便的计算出f*的后验概率P(f*|D1:3). 也就是说我们利用已知信息{(x1, f1),(x2,f2),(x3,f3)}和高斯过程建立起了一个黑箱f的代理函数, 来为

寻找更优的f提供方便.



更严谨的数学表达如下所示:

这里还是假设m(x)=0, 进一步假设模型为无噪音高斯过程(无噪音: f=f(x); 有噪音:f=f(x)+)和k(x, x')(更多的核函数可参考论文A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning 😟)

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

那么多元高斯分布的协方差可以写为

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \dots & k(\mathbf{x}_1, \mathbf{x}_t) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_t, \mathbf{x}_1) & \dots & k(\mathbf{x}_t, \mathbf{x}_t) \end{bmatrix}.$$

依据高斯过程, 可以得到如下联合高斯分布:

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

其中

$$\mathbf{k} = \begin{bmatrix} k(\mathbf{x}_{t+1}, \mathbf{x}_1) & k(\mathbf{x}_{t+1}, \mathbf{x}_2) & \cdots & k(\mathbf{x}_{t+1}, \mathbf{x}_t) \end{bmatrix}$$

利用Sherman-Morrison-Woodbury公式和一些推导过程即可得到后验概率:

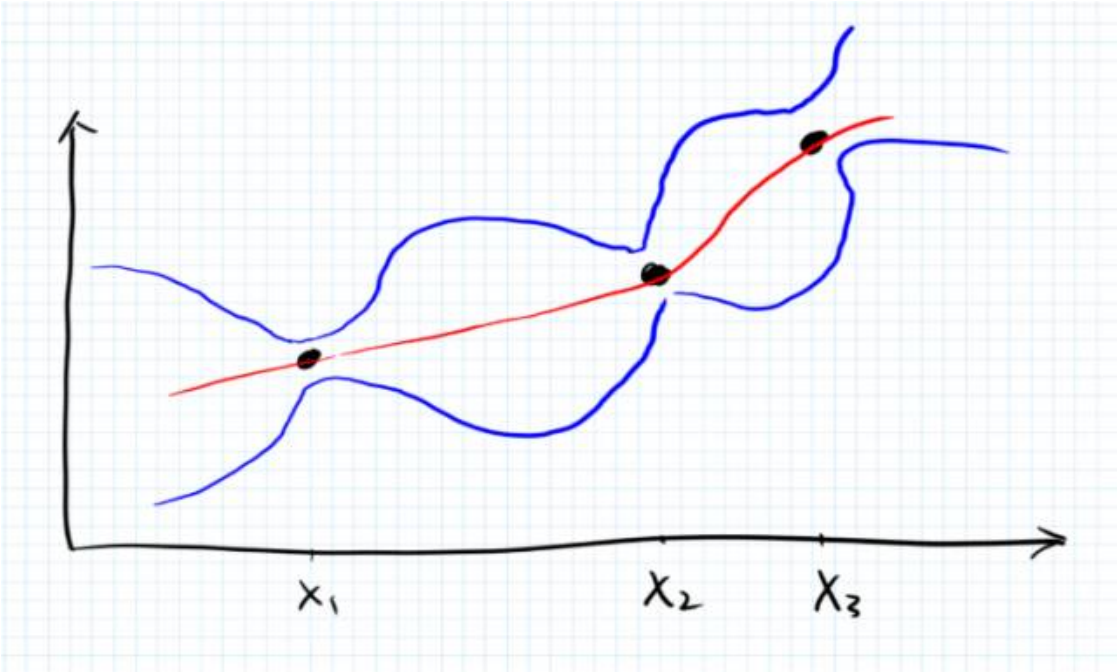
$$P(f_{t+1}|\mathcal{D}_{1:t}, \mathbf{x}_{t+1}) = \mathcal{N}\left(\mu_t(\mathbf{x}_{t+1}), \sigma_t^2(\mathbf{x}_{t+1})\right)$$

其中

$$\begin{aligned} \mu_t(\mathbf{x}_{t+1}) &= \mathbf{k}^T \mathbf{K}^{-1} \mathbf{f}_{1:t} \\ \sigma_t^2(\mathbf{x}_{t+1}) &= k(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}) - \mathbf{k}^T \mathbf{K}^{-1} \mathbf{k}. \end{aligned}$$

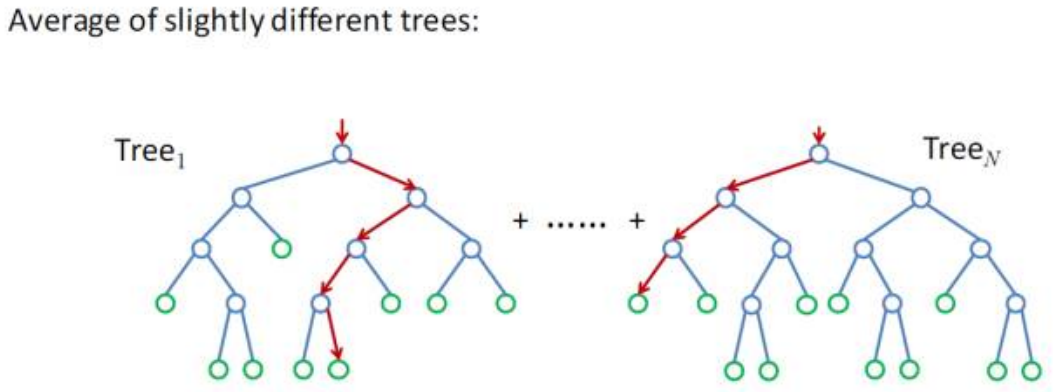
假设我们拍脑袋想出了1000种不同的超参配置，那么利用高斯过程就可以算出1000个x*的均值和标准差，画出来连成线，分别对应红色和蓝色曲线，这样我们对模型误差

的估计有了，对这个估计的确信程度也有了，于是完成了对超参效果认识的更新。



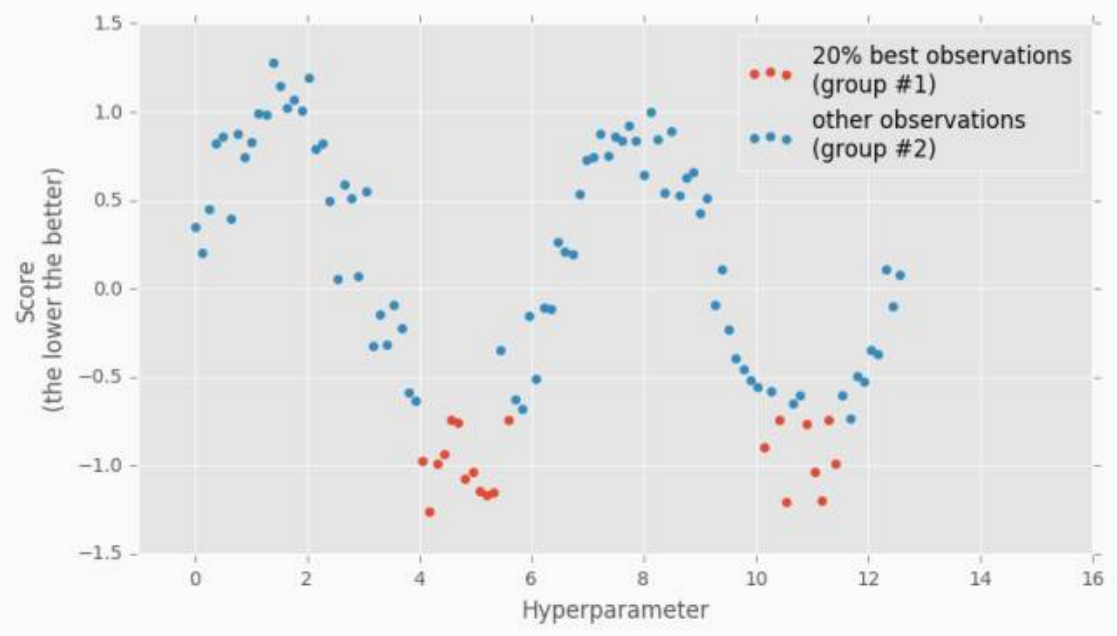
3.3.2 随机森林（SMAC实现）

随机森林的特征是超参配置，目标变量是模型效果，用已经观察到的实验样本做训练，得到1000棵训练好的树。对任意一种新的超参配置，随机森林分别给出1000棵树的预测结果，这1000个点就构成了超参效果的分布，均值和方差都有了。

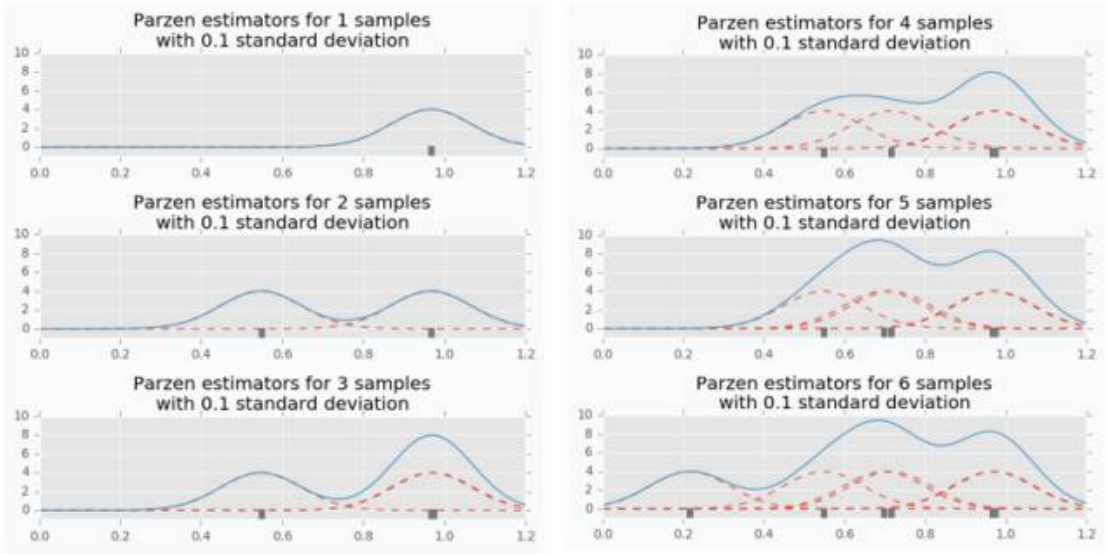


3.3.3 TPE（hyperopt实现）

TPE的想法是，把表现好的超参划到第一梯队，表现不好的超参划到第二梯队，对每个梯队拟合一个分布，那么EI选出来的就是落在第一梯队概率大、落在第二梯队概率小的潜力超参。



对梯队的分布拟合，用的是Parzen Estimtor，简单来讲就是每种超参配置都是一个独立的标准正态分布，超参组合的效果可以通过叠加这些正态分布得到。



现在让我们回到代理函数。SMBO的方法在如何构建替代模型p（y | x）方面不同。树结构Parzen Estimator通过应用贝叶斯规则构建模型。它不是直接表示p（y | x），而是使用：

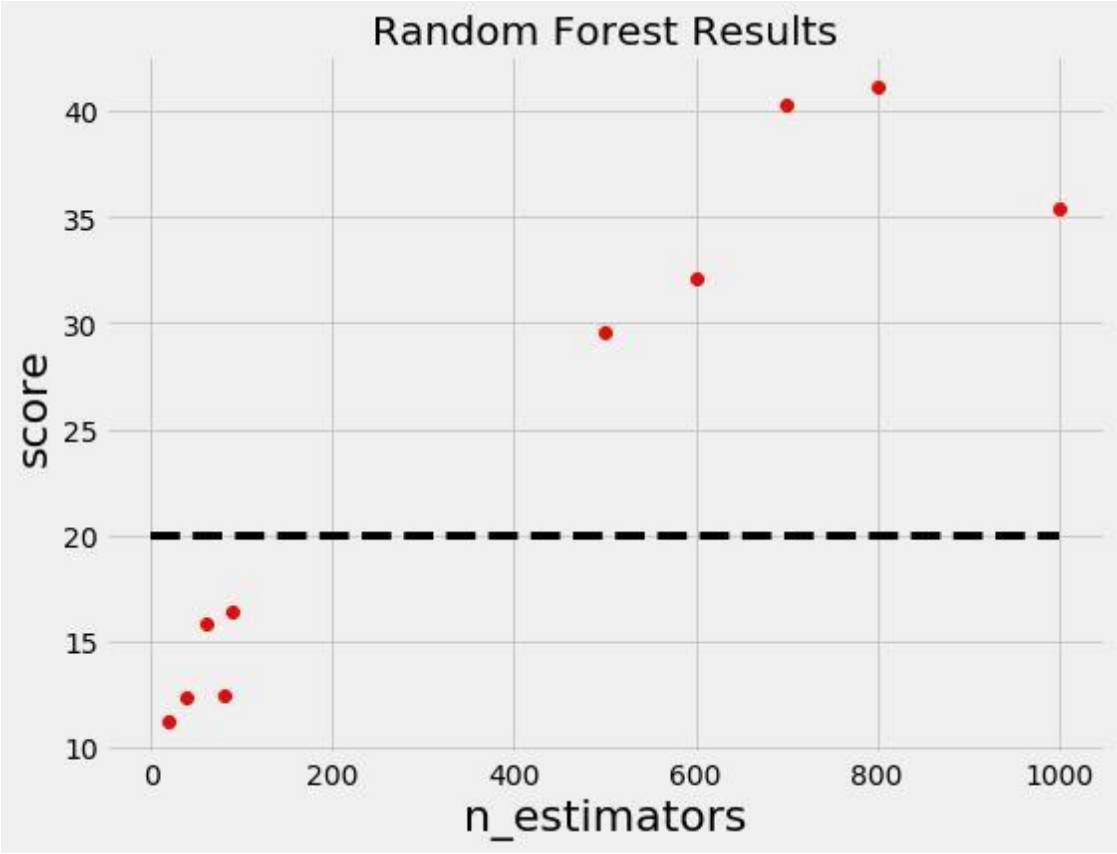
$$p(y|x) = \frac{p(x\ |y) * p(y)}{p(x)}$$

p（x | y），即给定目标函数得分的超参数的概率，反过来表示：

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

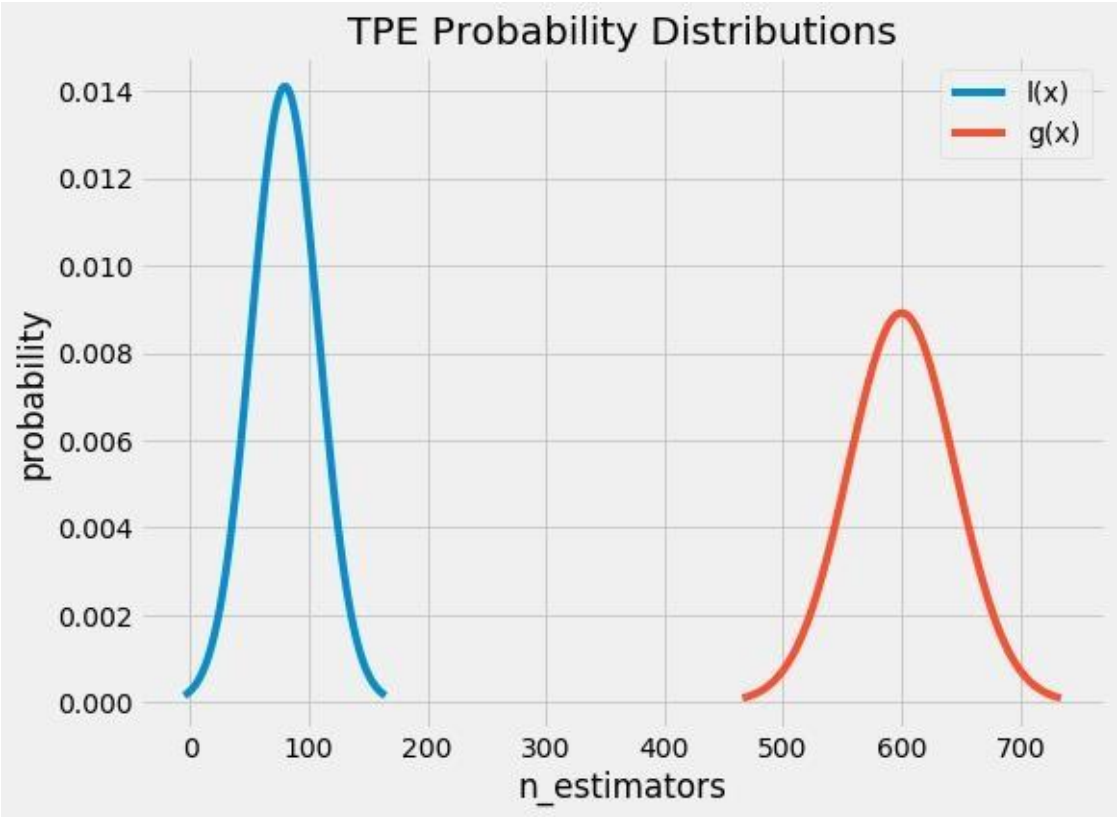
其中y <y *表示目标函数的值低于阈值。这个等式的解释是我们为超参数做出两个不同的分布：一个是目标函数的值小于阈值分布为l（x），另一个是目标函数的值大于阈值分布为g（x）。

让我们更新我们的随机森林图表以包含一个阈值：



现在我们为估计量的数量构建两个概率分布，一个使用在阈值以下产生值的估计量，一个使用产生高于阈值的值的估计量。

直观地，我们似乎想要从l（x）而不是从g（x）绘制x的值，因为这种分布仅基于x的值，其得分低于阈值。事实证明这正是数学所说的使用贝叶斯规则和一些替换，预期的改进方程（我们试图最大化）变为：



最右边的术语是最重要的部分。这说明与比率l（x）/ g（x）成正比，因此，为了最大化预期改进，我们应该最大化这个比率。我们的直觉是正确的：我们应该绘制超参

数的值，这些参数更可能在l（x）下而不是在g（x）下。

Tree-structured Parzen Estimator通过从l（x）绘制样本超参数，以l（x）/ g（x）的形式对它们进行评估，并返回在l（x）/ g（x）下产生最高值的集合。 对应于最大的预期改进。然后在目标函数上评估这些超参数。如果代理函数是正确的，那么这些超参数在评估时应该产生更好的值。

预期的改进标准允许模型平衡勘探与开采。l（x）是分布而不是单个值，这意味着绘制的超参数可能接近但不完全处于预期改进的最大值。此外，因为代理只是对目标函数的估计，所以选择的超参数在评估时实际上可能不会产生改进，并且必须更新代理模型。该更新基于当前替代模型和目标函数评估的历史来完成。

3.4 获得函数

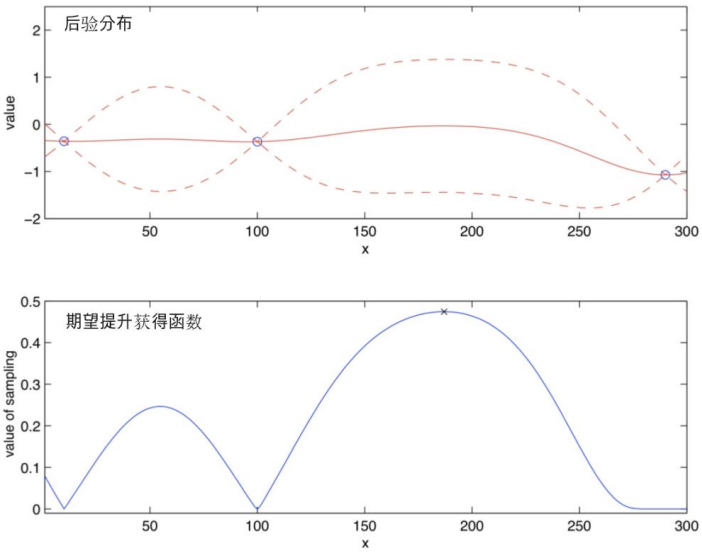
获得函数是从代理函数（后验分布）中选择下一组超参数的标准。最常见的标准选择是预期改进/期望提升（Expected Improvement，简称EI），以老虎机为例，假设当前的判断是，我有七成把握告诉你黑色按一次会吐0~7块钱，有七成把握告诉你红色按一次会吐0~9块钱，你怎么选，当然选红色，风险一样回报更高。但是，假如我还有九成把握告诉你紫色按一次会吐1~4块钱”，现在红色紫色选哪个，是不是得想想了，红色高风险高回报，紫色低风险低回报。EI的目的，就是在风险和回报间做个平衡，也即exploration-exploitation tradeoff，既要靠谱又要回报潜力大。

$$\mathrm{EI}_{y^*}(x) = \int_{-\infty}^{y^*} (y^* - y) p(y|x) dy$$

这里y *是目标函数的阈值，x是建议的超参数集，y是使用超参数x的目标函数的实际值，而p（y | x）是表示y的概率的替代概率模型给出x。如果这一切都有点多，简单来说，目标是最大化x的预期改进。这意味着在代理函数p（y | x）下找到最佳超参数。

如果p（y | x）在y <y *处到处为零，那么超参数x预计不会产生任何改进。如果积分是正的，则意味着超参数x预期产生比阈值更好的结果。

下图为目标函数的后验分布，红实线为后验分布的均值，红虚线为后验分布的置信区间。下图为由后验分布计算出的期望提升获得函数



除了预期改进获得函数意外，还有其他一些获得函数的构造方式，例如上限置信界（Upper confidence bound）和知识梯度（Knowledge gradient），这些方法本质都是探索/利用平衡策略的数学表达。

3.5 历史

每次算法提出一组新的候选超参数时，它就会用实际的目标函数对它们进行评估，并将结果记录在一对（得分，超参数）中。这些记录构成了历史。该算法使用历史建立l（x）和g（x）以得出目标函数的概率模型，该模型随每次迭代而改进。
我们对目标函数的替代进行了初步估计，我们在收集更多证据时会对其进行更新。最终，通过对目标函数的充分评估，我们希望我们的模型准确地反映目标函数，并且产生最大预期改进的超参数对应于最大化目标函数的超参数。

进一步阅读

- MLA Bergstra，James S.，et al。“用于超参数优化的算法。”神经信息处理系统的进展.2011。
- Snoek，Jasper，Hugo Larochelle和Ryan P. Adams。“实用贝叶斯优化机器学习算法。”神经信息处理系统的进展.2012。
- Brochu，Eric，Vlad M. Cora和Nando De Freitas。“关于昂贵成本函数的贝叶斯优化的教程，应用于主动用户建模和分层强化学习。”（2010年）。
- Bergstra，James和Yoshua Bengio。“随机搜索超参数优化。” 机器学习研究期刊 13.1（2012）：281-305。
- Bergstra，James，Daniel Yamins和David Daniel Cox。“制作模型搜索科学：视觉架构的数百个维度的超参数优化。”（2013年）。
- Hutter，Frank，Holger H. Hoos和Kevin Leyton-Brown。“基于序列模型的通用算法配置优化。” 学习和智能优化 .Springer Berlin Heidelberg，2011.507-523。