PRACTICAL BAYESIAN OPTIMIZATION OF MACHINE LEARNING ALGORITHMS

By Jasper Snoek, Hugo Larochelle and Ryan P. Adams

University of Toronto, Université de Sherbrooke and Harvard University

Machine learning algorithms frequently require careful tuning of model hyperparameters, regularization terms, and optimization parameters. Unfortunately, this tuning is often a "black art" that requires expert experience, unwritten rules of thumb, or sometimes brute-force search. Much more appealing is the idea of developing automatic approaches which can optimize the performance of a given learning algorithm to the task at hand. In this work, we consider the automatic tuning problem within the framework of Bayesian optimization, in which a learning algorithm's generalization performance is modeled as a sample from a Gaussian process (GP). The tractable posterior distribution induced by the GP leads to efficient use of the information gathered by previous experiments, enabling optimal choices about what parameters to try next. Here we show how the effects of the Gaussian process prior and the associated inference procedure can have a large impact on the success or failure of Bayesian optimization. We show that thoughtful choices can lead to results that exceed expert-level performance in tuning machine learning algorithms. We also describe new algorithms that take into account the variable cost (duration) of learning experiments and that can leverage the presence of multiple cores for parallel experimentation. We show that these proposed algorithms improve on previous automatic procedures and can reach or *surpass* human expert-level optimization on a diverse set of contemporary algorithms including latent Dirichlet allocation, structured SVMs and convolutional neural networks.

1. Introduction. Machine learning algorithms are rarely parameter-free; whether via the properties of a regularizer, the hyperprior of a generative model, or the step size of a gradient-based optimization, learning procedures almost always require a set of high-level choices that significantly impact generalization performance. As a practitioner, one is usually able to specify the general framework of an inductive bias much more easily than the particular weighting that it should have relative to training data. As a result, these high-level parameters are often considered a nuisance, making it desirable to develop algorithms with as few of these "knobs" as possible.

Another, more flexible take on this issue is to view the optimization of high-level parameters as a procedure to be automated. Specifically, we could view such tuning as the optimization of an unknown black-box function that reflects generalization performance and invoke algorithms developed for such problems. These optimization problems have a somewhat different flavor than the low-level objectives one often encounters as part of a training procedure: here function evaluations are very expensive, as they involve running the primary machine learning algorithm to completion. In this setting where function evaluations are expensive, it is desirable to spend computational time making better choices about where to seek the best parameters. Bayesian optimization (Mockus et al., 1978) provides an elegant approach and has been shown to outperform other state of the art global optimization algorithms on a number of challenging optimization benchmark functions (Jones, 2001). For continuous functions, Bayesian optimization typically works by assuming the unknown function was sampled from a

实用的贝叶斯机器学习优化 ALGORITHMS

由Jasper Snoek，Hugo Larochelle和Ryan P. Adams

多伦多大学，大学德舍布鲁克和哈佛大学

机器学习算法经常需要仔细调整

模型超参数，正则化术语和优化PA−rameters。不幸的是，这种调整往往是一个“黑色艺术”，可以重新询问专家体验，不成文的拇指规则，或有时会强行搜索。更具吸引力是开发自动方法的想法，可以优化给定学习算法对手头的任务的性能。在这项工作中，我们考虑贝叶斯光学框架内的自动调整问题，其中学习算法的泛化性能被建模为来自高斯过程（GP）的样本。
GP引起的贸易后部分布导致有效地利用先前实验收集的信息，从而实现了关于尝试下一步的参数的最佳选择。在这里，我们展示了高斯工艺先前和相关的推理程序的效果如何对贝叶斯优化的成功或失败产生大量影响。我们展示了深思熟虑的选择可以导致超过专家级别的调整机器学习al−摩托车的结果。我们还描述了对学习实验的可变成本（持续时间）的新算法，并且可以利用多个核的存在进行平行实验。我们表明，这些提议的算法改善了先前的自动程序，可以在包括潜在的Dirichlet分配，结构化SVM和卷积神经网络的各种当代算法上达到或超越人类专家级优化。

1.简介。机器学习算法很少是无参数的;是否通过

常规器的属性，生成模型的高度，或基于梯度的优化的步长，学习程序几乎总是需要一组显着影响泛化性能的高级选择。作为从业者，人们通常能够比特定加权更容易地指定归纳偏差的一般框架，它应该相对于训练数据。结果，这些高级参数通常被认为是令人讨厌的，使得希望尽可能少量开发与这些“旋钮”中的少数。

另一个，对此问题的更灵活性是要查看高级参数的优化

作为一种自动化的程序。具体而言，我们可以将这种调谐视为优化未知的黑盒功能，该功能反映了泛化性能并调用为这些问题开发的算法。这些优化问题具有比低级目标的味道有点不同，通常遇到培训程序的一部分：这里的功能评估非常昂贵，因为它们涉及运行主机学习算法完成。在这种情况下，函数评估昂贵，期望花费计　　算时间，使得能够更好地选择寻求最佳参数。贝叶斯优化（Mockus等，1978）提供了优雅的方法，并且已被证明以满足挑战优化基准函数的数字（Jones，2001）的作用，以越优越的本领域的全局优化算法。对于连续功能，贝叶斯优化通常通过假设从A中采样未知功能来工作

Gaussian process (GP) and maintains a posterior distribution for this function as observations are made. In our case, these observations are the measure of generalization performance under different settings of the hyperparameters we wish to optimize. To pick the hyperparameters of the next experiment, one can optimize the expected improvement (EI) (Mockus et al., 1978) over the current best result or the Gaussian process upper confidence bound (UCB)(Srinivas et al., 2010). EI and UCB have been shown to be efficient in the number of function evaluations required to find the global optimum of many multimodal black-box functions (Srinivas et al., 2010; Bull, 2011).

Machine learning algorithms, however, have certain characteristics that distinguish them from other black-box optimization problems. First, each function evaluation can require a variable amount of time: training a small neural network with 10 hidden units will take less time than a bigger network with 1000 hidden units. Even without considering duration, the advent of cloud computing makes it possible to quantify economically the cost of requiring large-memory machines for learning, changing the actual cost in dollars of an experiment with a different number of hidden units. It is desirable to understand how to include a concept of cost into the optimization procedure. Second, machine learning experiments are often run in parallel, on multiple cores or machines. We would like to build Bayesian optimization procedures that can take advantage of this parallelism to reach better solutions more quickly.

In this work, our first contribution is the identification of good practices for Bayesian optimization of machine learning algorithms. In particular, we argue that a fully Bayesian treatment of the GP kernel parameters is of critical importance to robust results, in contrast to the more standard procedure of optimizing hyperparameters (e.g. Bergstra et al. (2011)). We also examine the impact of the kernel itself and examine whether the default choice of the squared-exponential covariance function is appropriate. Our second contribution is the description of a new algorithm that accounts for cost in experiments. Finally, we also propose an algorithm that can take advantage of multiple cores to run machine learning experiments in parallel.

2. Bayesian Optimization with Gaussian Process Priors. As in other kinds of optimization, in Bayesian optimization we are interested in finding the minimum of a function $f(\mathbf{x})$ on some bounded set $\mathcal{X}$, which we will take to be a subset of $\mathbb{R}^D$. What makes Bayesian optimization different from other procedures is that it constructs a probabilistic model for $f(\mathbf{x})$ and then exploits this model to make decisions about where in $\mathcal{X}$ to next evaluate the function, while integrating out uncertainty. The essential philosophy is to use *all* of the information available from previous evaluations of $f(\mathbf{x})$ and not simply rely on local gradient and Hessian approximations. This results in a procedure that can find the minimum of difficult non-convex functions with relatively few evaluations, at the cost of performing more computation to determine the next point to try. When evaluations of $f(\mathbf{x})$ are expensive to perform — as is the case when it requires training a machine learning algorithm — it is easy to justify some extra computation to make better decisions. For an overview of the Bayesian optimization formalism, see, e.g., Brochu et al. (2010). In this section we briefly review the general Bayesian optimization approach, before discussing our novel contributions in Section 3.

There are two major choices that must be made when performing Bayesian optimization. First, one must select a prior over functions that will express assumptions about the function being optimized. For this we choose the Gaussian process prior, due to its flexibility and tractability. Second, we must choose an *acquisition function*, which is used to construct a utility function from the model posterior, allowing us to determine the next point to evaluate.

高斯工艺（GP）并保持对此功能的后部分布，因为进行了观察结果。在我们的情况下，这些观察是我们希望优化的超参数的不同环境下的泛化性能的衡量标准性能。要选择下一个实验的超级参数，人们可以通过目前的最佳结果或高斯过程的上限束缚（UCB）（Srinivas等，2010）来优化预期的改进（ei）（Mockus等，1978）。EI和UCB已被证明是在寻找许多多模块黑匣子功能的全局最佳最佳功能所需的函数评估的数量中有效（Srinivas等，2010; Bull，2011）。

然而，机器学习算法具有区分它们的某些特征

来自其他黑匣子优化问题。首先，每个函数评估可能需要可变的时间：培训具有10个隐藏单元的小型神经网络，比具有1000个隐藏单元的更大的网络时间更少。即使在不考虑持续时间，云计算的出现也使得可以在经济上量化需要大存储器用于学习的成本，以不同数量的隐藏单元改变实验的实际成本。希望了解如何将成本的概念列入优化过程。其次，机器学习实验通常在多个核或机器上并行运行。我们希望建立贝叶斯优化程序，可以利用这种并行性，更快地达到更好的解决方案。

在这项工作中，我们的第一款贡献是识别贝叶斯的良好做法

优化机器学习算法。特别是，我们争辩说，与优化HyperParameters的更高的方法相比，对GP内核参数的完全贝叶斯参数对鲁棒结果至关重要（例如，例如Bergstra等（2011））。我们还检查内核本身的影响，并检查平方指数协方差函数是否适当的默认选择。我们的第二次贡献是对实验中成本计算的新算法的描述。最后，我们还提出了一种算法，可以利用多个核心并行运行机器学习实验。

2.贝叶斯优化与高斯工艺师的优化。与其他类型一样

优化，在贝叶斯优化中，我们有兴趣在某些有限组X上找到一个功能f（x）的兴趣，我们将成为RD的子集。贝叶斯优化与其他程序不同的是，它为f（x）构建了一个概率模型，然后利用该模型来做出关于在X到下一个地点评估功能的决定，同时整合不确定性。基本理念是使用以前评估f（x）的所有信息，而不是简单地依赖于局部梯度和黑森州近似值。这导致可以在执行更多计算的成本中找到具有相对较少的评估的困难非凸函数的过程，以确定尝试的下一个点。当F（x）的评估执行昂贵时 – 就像需要培训机器学习算法的情况一样 –
它很容易证明一些额外的计算来做出更好的决定。有关贝叶斯优化形式主义的概述，请参阅，例如，Brochu等人。（2010）。在本节中，我们简要介绍了贝叶斯优化方法，然后在第3节讨论了我们的新贡献之前。

在进行贝叶斯优化时必须有两种主要选择。

首先，必须在以前的函数中选择一个关于正在优化的功能的假设。为此，由于其灵活性和途径，我们可以选择高斯过程。其次，我们必须选择一个采集函数，它用于构造模型后部的实用程序函数，允许我们确定评估的下一个点。

2.1. *Gaussian Processes.* The Gaussian process (GP) is a convenient and powerful prior distribution on functions, which we will take here to be of the form $f : \mathcal{X} \to \mathbb{R}$. The GP is defined by the property that any finite set of N points $\{\mathbf{x}_n \in \mathcal{X}\}_{n=1}^{N}$ induces a multivariate Gaussian distribution on $\mathbb{R}^N$. The nth of these points is taken to be the function value $f(\mathbf{x}_n)$, and the elegant marginalization properties of the Gaussian distribution allow us to compute marginals and conditionals in closed form. The support and properties of the resulting distribution on functions are determined by a mean function $m : \mathcal{X} \to \mathbb{R}$ and a positive definite covariance function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. We will discuss the impact of covariance functions in Section 3.1. For an overview of Gaussian processes, see Rasmussen and Williams (2006).

2.2. *Acquisition Functions for Bayesian Optimization.* We assume that the function $f(\mathbf{x})$ is drawn from a Gaussian process prior and that our observations are of the form $\{\mathbf{x}_n, y_n\}_{n=1}^{N}$, where $y_n \sim \mathcal{N}(f(\mathbf{x}_n), \nu)$ and ν is the variance of noise introduced into the function observations. This prior and these data induce a posterior over functions; the acquisition function, which we denote by $a : \mathcal{X} \to \mathbb{R}^+$, determines what point in $\mathcal{X}$ should be evaluated next via a proxy optimization $\mathbf{x}_{\mathsf{next}} = \operatorname{argmax}_{\mathbf{x}} a(\mathbf{x})$, where several different functions have been proposed. In general, these acquisition functions depend on the previous observations, as well as the GP hyperparameters; we denote this dependence as $a(\mathbf{x}\,;\,\{\mathbf{x}_n, y_n\}, \theta)$. There are several popular choices of acquisition function. Under the Gaussian process prior, these functions depend on the model solely through its predictive mean function $\mu(\mathbf{x}\,;\,\{\mathbf{x}_n, y_n\}, \theta)$ and predictive variance function $\sigma^2(\mathbf{x}\,;\,\{\mathbf{x}_n, y_n\}, \theta)$. In the proceeding, we will denote the best current value as $\mathbf{x}_{\mathsf{best}} = \operatorname{argmin}_{\mathbf{x}_n} f(\mathbf{x}_n)$, $\Phi(\cdot)$ will denote the cumulative distribution function of the standard normal, and $\phi(\cdot)$ will denote the standard normal density function.

Probability of Improvement. One intuitive strategy is to maximize the probability of improving over the best current value (Kushner, 1964). Under the GP this can be computed analytically as

$$a_{\mathsf{PI}}(\mathbf{x}\,;\,\{\mathbf{x}_n, y_n\}, \theta) = \Phi(\gamma(\mathbf{x})) \qquad \gamma(\mathbf{x}) = \frac{f(\mathbf{x}_{\mathsf{best}}) - \mu(\mathbf{x}\,;\,\{\mathbf{x}_n, y_n\}, \theta)}{\sigma(\mathbf{x}\,;\,\{\mathbf{x}_n, y_n\}, \theta)}. \tag{1}$$

Expected Improvement. Alternatively, one could choose to maximize the expected improvement (EI) over the current best. This also has closed form under the Gaussian process:

$$a_{\mathsf{EI}}(\mathbf{x}\,;\,\{\mathbf{x}_n, y_n\}, \theta) = \sigma(\mathbf{x}\,;\,\{\mathbf{x}_n, y_n\}, \theta)\,(\gamma(\mathbf{x})\,\Phi(\gamma(\mathbf{x})) + \mathcal{N}(\gamma(\mathbf{x})\,;\,0, 1)) \tag{2}$$

GP Upper Confidence Bound. A more recent development is the idea of exploiting lower confidence bounds (upper, when considering maximization) to construct acquisition functions that minimize regret over the course of their optimization (Srinivas et al., 2010). These acquisition functions have the form

$$a_{\mathsf{LCB}}(\mathbf{x}\,;\,\{\mathbf{x}_n, y_n\}, \theta) = \mu(\mathbf{x}\,;\,\{\mathbf{x}_n, y_n\}, \theta) - \kappa\,\sigma(\mathbf{x}\,;\,\{\mathbf{x}_n, y_n\}, \theta), \tag{3}$$

with a tunable κ to balance exploitation against exploration.

In this work we will focus on the expected improvement criterion, as it has been shown to be better-behaved than probability of improvement, but unlike the method of GP upper confidence bounds (GP-UCB), it does not require its own tuning parameter. We have found expected improvement to perform well in minimization problems, but wish to note that the regret formalization is more appropriate for many settings. We perform a direct comparison between our EI-based approach and GP-UCB in Section 4.1.

2.1。高斯过程。高斯过程（GP）是一个方便而强大的先前
我们将在此处将其视为f：x→r。GP由任何有限组n点{xn x x} n定义的GP定义

高斯分布在rn。这些点的第n个被认为是功能值f（xn），高斯分布的优雅边缘化特性允许我们以封闭形式计算边缘和条件。由此产生的函数的支持和属性由平均函数M：X→R和正定的协方差函数K：x×X→R确定。我们将讨论第3.1节中协方差函数的影响。有关高斯流程的概述，请参阅Rasmussen和Williams（2006）。

2.2。贝叶斯优化的采集函数。我们假设功能f（x）
在先前从高斯过程中汲取，我们的观察结果是{xn，yn} n的形式
1，
其中Yn n（f（xn），ν）和ν是引入函数观察中的噪声的方差。此之前和这些数据诱导功能后部;我们表示的采集函数：x→r +，确定x应该通过代理优化xnext = argmaxx
a（x）进行评估x中的哪个点，其中有几种不同的功能已经提出。一般而言，这些采集函数取决于先前的观察，以及GP HyperParameters;我们表示这种依赖性作为（x;
{xn，yn}，θ）。有几个受欢迎的采集功能选择。在高斯进程之前，这些功能仅通过其预测均值μ（x;
{xn，yn}，θ）和预测方差函数σ2（x;
{xn，yn}，θ）来依赖于该模型。在进行中，我们将表示最佳的电流值作为xbest = argminxn
f（xn），ϕ（・）将表示标准正常的累积分布函数，ϕ（・）将表示标准正常密度函数。

改善概率。一个直观的策略是最大化IM-的概率
证明最佳当前值（Kushner，1964）。在GP下，这可以分析地计算为

$$a_{\mathsf{PI}}(\mathbf{x}\,;\,\{\mathbf{x}_n, y_n\}, \theta) = \Phi(\gamma(\mathbf{x})) \qquad \gamma(\mathbf{x}) = \frac{f(\mathbf{x}_{\mathsf{best}}) - \mu(\mathbf{x}\,;\,\{\mathbf{x}_n, y_n\}, \theta)}{\sigma(\mathbf{x}\,;\,\{\mathbf{x}_n, y_n\}, \theta)}. \tag{1}$$

预期改进。或者，可以选择最大化预期的改进 -
最新的ei（ei）。这也在高斯过程下已关闭形式：

$$a_{\mathsf{EI}}(\mathbf{x}\,;\,\{\mathbf{x}_n, y_n\}, \theta) = \sigma(\mathbf{x}\,;\,\{\mathbf{x}_n, y_n\}, \theta)\,(\gamma(\mathbf{x})\,\Phi(\gamma(\mathbf{x})) + \mathcal{N}(\gamma(\mathbf{x})\,;\,0, 1)) \tag{2}$$

GP上部置信度。最近的发展是利用较低的理念
置信范围（上部，在考虑最大化时）构建采集功能，以在优化的过程中最小化遗憾（Srinivas等，2010）。这些常规函数具有表单

$$a_{\mathsf{LCB}}(\mathbf{x}\,;\,\{\mathbf{x}_n, y_n\}, \theta) = \mu(\mathbf{x}\,;\,\{\mathbf{x}_n, y_n\}, \theta) - \kappa\,\sigma(\mathbf{x}\,;\,\{\mathbf{x}_n, y_n\}, \theta), \tag{3}$$

随着调节κ与勘探剥削爆发。

在这项工作中，我们将专注于预期的改进标准，因为它已被展示
要更好地表现，而不是改进的概率，但与GP上置信度（GP-UCB）的方法不同，它不需要其自己的调谐参数。我们发现预期的改进在最小化问题中表现良好，但希望注意到遗憾的正式化更适合许多设置。我们在第4.1节中的基于EI的方法和GP-UCB之间进行直接比较。

3. Practical Considerations for Bayesian Optimization of Hyperparameters. Although an elegant framework for optimizing expensive functions, there are several limitations that have prevented it from becoming a widely-used technique for optimizing hyperparameters in machine learning problems. First, it is unclear for practical problems what an appropriate choice is for the covariance function and its associated hyperparameters. Second, as the function evaluation itself may involve a time-consuming optimization procedure, problems may vary significantly in duration and this should be taken into account. Third, optimization algorithms should take advantage of multi-core parallelism in order to map well onto modern computational environments. In this section, we propose solutions to each of these issues.

3.1. *Covariance Functions and Treatment of Covariance Hyperparameters.* The power of the Gaussian process to express a rich distribution on functions rests solely on the shoulders of the covariance function. While non-degenerate covariance functions correspond to infinite bases, they nevertheless can correspond to strong assumptions regarding likely functions. In particular, the automatic relevance determination (ARD) *squared exponential* kernel

$$K_{\mathsf{SE}}(\mathbf{x},\mathbf{x}') = \theta_0 \exp\left\{-\frac{1}{2}r^2(\mathbf{x},\mathbf{x}')\right\} \qquad r^2(\mathbf{x},\mathbf{x}') = \sum_{d=1}^{D}(x_d - x'_d)^2/\theta_d^2. \tag{4}$$

is often a default choice for Gaussian process regression. However, sample functions with this covariance function are unrealistically smooth for practical optimization problems. We instead propose the use of the ARD Matérn 5/2 kernel:

$$K_{\mathsf{M52}}(\mathbf{x},\mathbf{x}') = \theta_0\left(1+\sqrt{5r^2(\mathbf{x},\mathbf{x}')}+\frac{5}{3}r^2(\mathbf{x},\mathbf{x}')\right)\exp\left\{-\sqrt{5r^2(\mathbf{x},\mathbf{x}')}\right\}. \tag{5}$$

This covariance function results in sample functions which are twice differentiable, an assumption that corresponds to those made by, e.g., quasi-Newton methods, but without requiring the smoothness of the squared exponential.

After choosing the form of the covariance, we must also manage the hyperparameters that govern its behavior (Note that these "hyperparameters" are different than the ones which are being subjected to the overall Bayesian optimization.), as well as that of the mean function. For our problems of interest, typically we would have $D + 3$ Gaussian process hyperparameters: D length scales $\theta_{1:D}$, the covariance amplitude θ_0, the observation noise ν, and a constant mean m. The most commonly advocated approach is to use a point estimate of these parameters by optimizing the marginal likelihood under the Gaussian process

$$p(\mathbf{y}\,|\,\{\mathbf{x}_n\}_{n=1}^N,\theta,\nu,m) = \mathcal{N}(\mathbf{y}\,|\,m\mathbf{1},\mathbf{\Sigma}_\theta+\nu\mathbf{I}),$$

where $\mathbf{y} = [y_1, y_2, \cdots, y_n]^{\mathsf{T}}$, and $\mathbf{\Sigma}_\theta$ is the covariance matrix resulting from the N input points under the hyperparameters θ.

However, for a fully-Bayesian treatment of hyperparameters (summarized here by θ alone), it is desirable to marginalize over hyperparameters and compute the *integrated acquisition function*:

$$\hat{a}(\mathbf{x}\,;\,\{\mathbf{x}_n,y_n\}) = \int a(\mathbf{x}\,;\,\{\mathbf{x}_n,y_n\},\theta)\,p(\theta\,|\,\{\mathbf{x}_n,y_n\}_{n=1}^N)\,\mathrm{d}\theta, \tag{6}$$

3.贝叶斯优化的普带的普带优化的实践考虑因素。

虽然优雅的框架优化昂贵的功能，但有几种限制可以防止它成为优化机器学习问题中超参数的广泛使用的技术。首先，对于实际问题，不清楚适当的选择是协方差函数及其相关的普通公共表。即，由于函数评估本身可能涉及耗时的优化程序，因此持续时间可能会显着变化，因此应考虑到这一点。第三，优化算法应利用多核并行性，以便映射到现代计算环境中。在本节中，我们向每个问题提出解决方案。

3.1。协方差函数和协方差普遍参数的治疗。的力量

高斯进程表达丰富的职能分配完全基于协方差函数的肩部。虽然非退化的协方差函数对应于无限基础，但它们仍然可以对应于有可能功能的强烈假设。特别地，自动相关性确定（ARD）平方指数核

$$K_{\mathsf{SE}}(\mathbf{x},\mathbf{x}') = \theta_0 \exp\left\{-\frac{1}{2}r^2(\mathbf{x},\mathbf{x}')\right\} \qquad r^2(\mathbf{x},\mathbf{x}') = \sum_{d=1}^{D}(x_d - x'_d)^2/\theta_d^2. \tag{4}$$

通常是高斯进程回归的默认选择。但是，具有此协方差功能的示例功能对于实际优化问题而言是不切实际的平滑。我们建议使用ARD Mat'ern 5/2内核：

$$K_{\mathsf{M52}}(\mathbf{x},\mathbf{x}') = \theta_0\left(1+\sqrt{5r^2(\mathbf{x},\mathbf{x}')}+\frac{5}{3}r^2(\mathbf{x},\mathbf{x}')\right)\exp\left\{-\sqrt{5r^2(\mathbf{x},\mathbf{x}')}\right\}. \tag{5}$$

这种协方差函数导致样本功能，这些功能是两次可分辨率，其对应于由例如准牛顿方法制造的体验，但不需要平方指数的平滑度。

在选择协方差形式之后，我们还必须管理Quand参数

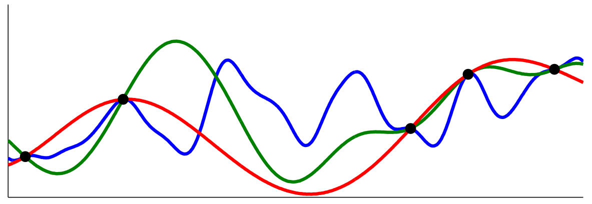
管理其行为（请注意，这些"HyperParameters"与正在受到整体贝叶斯优化的行为不同。）以及平均功能的行为。对于我们感兴趣的问题，通常我们将拥有D + 3高斯过程超级参数：D长度级θ1：D，协方差幅度θ0，观察噪声ν和恒定的平均值。最常见的倡导方法是通过优化高斯过程下的边际可能性来利用这些参数的点估计

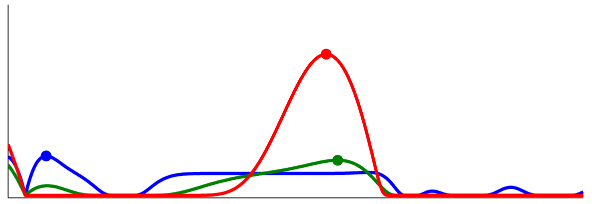
$$p(\mathbf{y}\,|\,\{\mathbf{x}_n\}_{n=1}^N,\theta,\nu,m) = \mathcal{N}(\mathbf{y}\,|\,m\mathbf{1},\mathbf{\Sigma}_\theta+\nu\mathbf{I}),$$

其中Y = [Y1，Y2，·······YN] T和σ θ是由HyperParameters θ下的n个输入点产生的协方差矩阵。

然而，对于普遍的贝叶斯治疗的普通人（θ单独总结），
希望通过普遍参数边缘化并计算集成采集功能：

$$\hat{a}(\mathbf{x}\,;\,\{\mathbf{x}_n,y_n\}) = \int a(\mathbf{x}\,;\,\{\mathbf{x}_n,y_n\},\theta)\,p(\theta\,|\,\{\mathbf{x}_n,y_n\}_{n=1}^N)\,\mathrm{d}\theta, \tag{6}$$


(a) Posterior samples under varying hyperparameters


(b) Expected improvement under varying hyperparameters

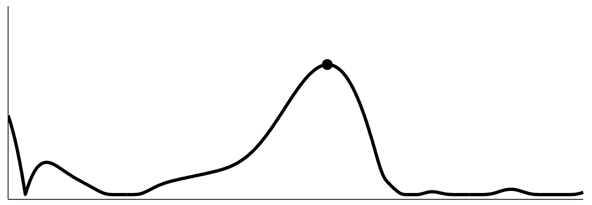
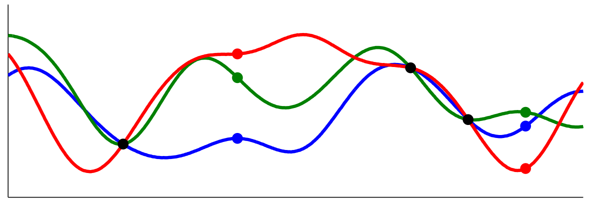
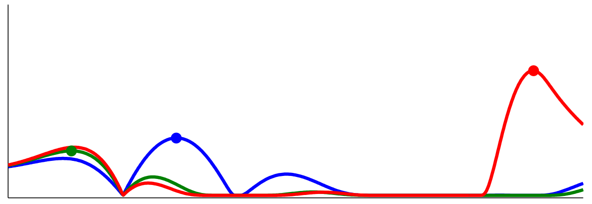

(c) Integrated expected improvement

Fig 1: Illustration of integrated expected improvement. (a) Three posterior samples are shown, each with different length scales, after the same five observations. (b) Three expected improvement acquisition functions, with the same data and hyperparameters. The maximum of each is shown. (c) The integrated expected improvement, with its maximum shown.


(a) Posterior samples after three data


(b) Expected improvement under three fantasies

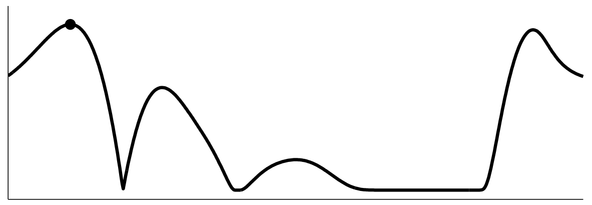
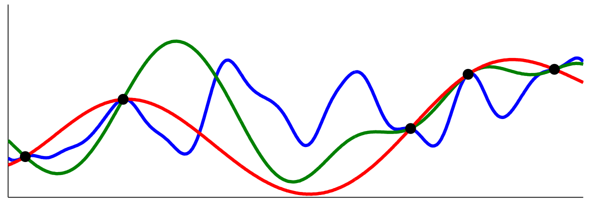
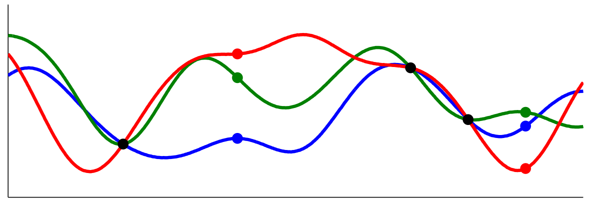

(c) Expected improvement across fantasies

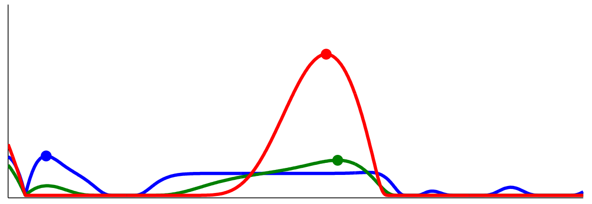
Fig 2: Illustration of the acquisition with pending evaluations. (a) Three data have been observed and three posterior functions are shown, with “fantasies” for three pending evaluations. (b) Expected improvement, conditioned on the each joint fantasy of the pending outcome. (c) Expected improvement after integrating over the fantasy outcomes.

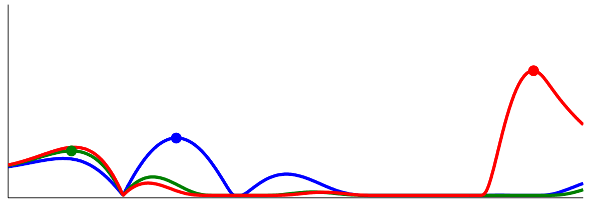
where $a(\mathbf{x})$ depends on θ and all of the observations. For probability of improvement and expected improvement, this expectation is the correct generalization to account for uncertainty in hyperparameters. We can therefore blend acquisition functions arising from samples from the posterior over GP hyperparameters and have a Monte Carlo estimate of the integrated expected improvement. These samples can be acquired efficiently using slice sampling, as described in Murray and Adams (2010). As both optimization and Markov chain Monte Carlo are computationally dominated by the cubic cost of solving an N-dimensional linear system (and our function evaluations are assumed to be much more expensive anyway), the fully-Bayesian treatment is sensible and our empirical evaluations bear this out. Figure 1 shows how the integrated expected improvement changes the acquistion function.

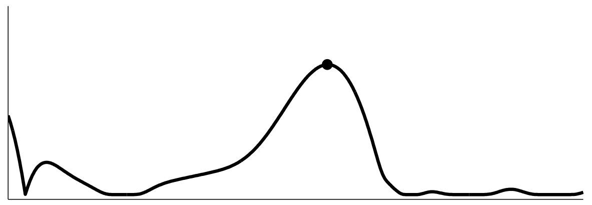
3.2. *Modeling Costs.* Ultimately, the objective of Bayesian optimization is to find a good setting of our hyperparameters as quickly as possible. Greedy acquisition procedures such as expected improvement try to make the best progress possible in the next function evaluation. From a practial point of view, however, we are not so concerned with function evaluations as with wallclock time. Different regions of the parameter space may result in vastly different execution times, due to varying regularization, learning rates, etc. To improve our performance


（a）在不同的近似数率下的后样品


（a）三个数据后的后样品


（b）在不同的近似数目下的预期改善


（b）在三个幻想下预期改善


(c) Integrated expected improvement

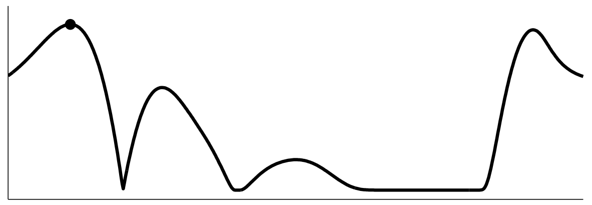

（c）预期跨幻想的改善

图1：综合预期改进的插图。（a）示出了三个后部样品，每个后部具有不同的长度尺度，在相同的五个OB−套件之后。（b）三个预期的改进函数，具有相同的数据和自行列度。每个显示每个的最大值。（c）综合预期改进，其最大值显示。

图2：采集与案件评估的插图。（a）已经观察到三个数据，并显示了三个后函数，具有三个待定评估的“泛态”。（b）预期改进，条件在待处理结果的每个联合诉讼中。（c）在整合幻想之后，预期的预期提出了。

其中（x）取决于θ和所有观察结果。对于改进和预期改进的可能性，这种预期是正确的概括，以解释超公路中的不确定性。因此，我们可以将采集函数从GP超公数的后后部产生的采集功能融合，并具有综合预期改进的蒙特卡罗估计。如Murray和Adams（2010）所述，可以使用切片采样有效地获取这些样本。由于优化和马尔可夫链蒙特卡罗通过解决N维线性系统的立方成本来计算地支配（并且我们的功能评估无论如何都假设更昂贵），完全贝叶斯治疗是明智的，我们的经验评估持有这出了。图1显示了集成的预期改进如何改变收获功能。

3.2。建模成本。最终，贝叶斯优化的目标是找到一个好的

尽快设置我们的超级参数。贪婪的收购程序，例如预期的改进尝试在下一个函数评估中尽可能取得最佳进步。然而，从实地的角度来看，我们并不是如此关注杂货时间的函数评估。参数空间的不同区域可能导致不同的执行时间，由于变化正则化，学习率等。提高我们的性能

in terms of wallclock time, we propose optimizing with the *expected improvement per second*, which prefers to acquire points that are not only likely to be good, but that are also likely to be evaluated quickly. This notion of cost can be naturally generalized to other budgeted resources, such as reagents or money.

Just as we do not know the true objective function $f(\mathbf{x})$, we also do not know the *duration function* $c(\mathbf{x}) : \mathcal{X} \to \mathbb{R}^+$. We can nevertheless employ our Gaussian process machinery to model $\ln c(\mathbf{x})$ alongside $f(\mathbf{x})$. In this work, we assume that these functions are independent of each other, although their coupling may be usefully captured using GP variants of multi-task learning (e.g., Teh et al. (2005); Bonilla et al. (2008)). Under the independence assumption, we can easily compute the predicted expected inverse duration and use it to compute the expected improvement per second as a function of $\mathbf{x}$.

3.3. *Monte Carlo Acquisition for Parallelizing Bayesian Optimization.* With the advent of multi-core computing, it is natural to ask how we can parallelize our Bayesian optimization procedures. More generally than simply batch parallelism, however, we would like to be able to decide what $\mathbf{x}$ should be evaluated next, even while a set of points are being evaluated. Clearly, we cannot use the same acquisition function again, or we will repeat one of the pending experiments. We would ideally perform a roll-out of our acquisition policy, to choose a point that appropriately balanced information gain and exploitation. However, such roll-outs are generally intractable. Instead we propose a sequential strategy that takes advantage of the tractable inference properties of the Gaussian process to compute Monte Carlo estimates of the acquisiton function under different possible results from pending function evaluations.

Consider the situation in which N evaluations have completed, yielding data $\{\mathbf{x}_n, y_n\}_{n=1}^N$, and in which J evaluations are pending at locations $\{\mathbf{x}_j\}_{j=1}^J$. Ideally, we would choose a new point based on the expected acquisition function under all possible outcomes of these pending evaluations:

$$(7)\quad \hat{a}(\mathbf{x}\,;\{\mathbf{x}_n, y_n\}, \theta, \{\mathbf{x}_j\}) = \int_{\mathbb{R}^J} a(\mathbf{x}\,;\{\mathbf{x}_n, y_n\}, \theta, \{\mathbf{x}_j, y_j\})\, p(\{y_j\}_{j=1}^J \,|\, \{\mathbf{x}_j\}_{j=1}^J, \{\mathbf{x}_n, y_n\}_{n=1}^N)\, \mathrm{d}y_1 \cdots \mathrm{d}y_J.$$

This is simply the expectation of $a(\mathbf{x})$ under a J-dimensional Gaussian distribution, whose mean and covariance can easily be computed. As in the covariance hyperparameter case, it is straightforward to use samples from this distribution to compute the expected acquisition and use this to select the next point. Figure 2 shows how this procedure would operate with queued evaluations. We note that a similar approach is touched upon briefly by Ginsbourger and Riche (2010), but they view it as too intractable to warrant attention. We have found our Monte Carlo estimation procedure to be highly effective in practice, however, as will be discussed in Section 4.

4. Empirical Analyses.

In this section, we empirically analyse[1] the algorithms introduced in this paper and compare to existing strategies and human performance on a number of challenging machine learning problems. We refer to our method of expected improvement while marginalizing GP hyperparameters as "GP EI MCMC", optimizing hyperparameters as "GP EI Opt", EI per second as "GP EI per Second", and N times parallelized GP EI MCMC as "Nx GP EI MCMC".

[1] All experiments were conducted on identical machines using the Amazon EC2 service.

在壁画时间方面，我们提出了每秒预期的改进优化，这更喜欢获取不仅是良好的点，而且还可能很快评估。这种成本概念可以自然地推广到其他预算资源，例如试剂或金钱。

就像我们不知道真正的目标函数f（x）一样，我们也不知道持续时间
功能c（x）：x→r +。然而，我们可以使用我们的高斯工艺机械来模拟LN
C（x）旁边的f（x）。在这项工作中，假设这些功能彼此独立，尽管它们的耦合可以使用多任务学习的GP变体进行有效捕获（例如，Teh等人（2005）；
Bonilla等人（2008））。在独立假设下，我们可以轻松地计算预测的预期逆持续时间，并使用它来计算每秒的预期改进作为x的函数。

3.3。Monte Carlo采集并行化贝叶斯优化。随着出现
多核计算，询问我们如何并将我们的贝叶斯优化程序平行化是自然的。然而，更一般的批次并行性，但是，我们希望能够决定接下来应该评估x的x，即使正在评估一组点。显然，我们不能再次使用相同的获取功能，或者我们将重复其中一个待处理的实验。我们理想地介绍我们的收购策略，选择适当平衡信息增益和剥削的点。然而，这种滚筒通常是棘手的。相反，我们提出了一种顺序策略，该策略利用高斯过程的易诊推理属性来计算来自不同可能的函数评估的不同可能结果的默认函数的Monte Carlo估计。

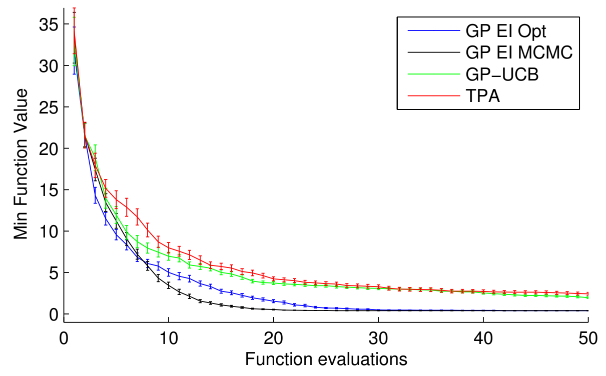
考虑n评估已完成的情况，产生数据{xn，yn} n 1，
并且其中J评估在位置{XJ} J j = 1。理想情况下，我们会选择一个新的
根据这些待定评估的所有可能结果，基于预期采集功能的点：

$$(7)\quad \hat{a}(\mathbf{x}\,;\{\mathbf{x}_n, y_n\}, \theta, \{\mathbf{x}_j\}) = \int_{\mathbb{R}^J} a(\mathbf{x}\,;\{\mathbf{x}_n, y_n\}, \theta, \{\mathbf{x}_j, y_j\})\, p(\{y_j\}_{j=1}^J \,|\, \{\mathbf{x}_j\}_{j=1}^J, \{\mathbf{x}_n, y_n\}_{n=1}^N)\, \mathrm{d}y_1 \cdots \mathrm{d}y_J.$$

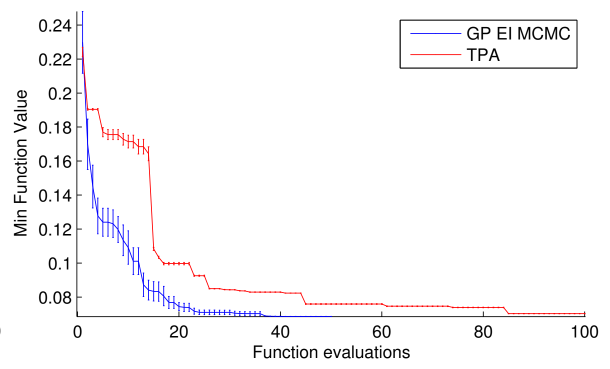
这只是在J维高斯分布下对（X）的期望，其平均值和协方差可以很容易地计算。与在协方差HyperParameter的情况一样，使用此分发中的样本是简单的，以计算预期的采集并使用此选中下一个点。图2显示了如何使用排队的评估运行该过程。我们注意到，Ginsbourger和Riche（2010）简单地触及了类似的方法，但他们认为这太棘手了，不保证关注。我们发现我们的蒙特卡罗估计程序在实践中非常有效，但在第4节中将讨论。

4.经验分析。在本节中，我们经验验证分析1算法介绍
本文提出，并与现有战略和人类表现进行比较，以众多具有挑战性的机器学习问题。我们指的是我们的预期改进方法，同时将GP高级参数边缘化为“GP EI MCMC”，优化作为“GP EI选择”，EI每秒作为“GP EI每秒”的“GP EIPT”，以及N次并行化GP EI MCMC为“NX GP”ei mcmc“。

使用亚马逊EC2服务在相同机器上进行1ALL实验。



(a)



(b)

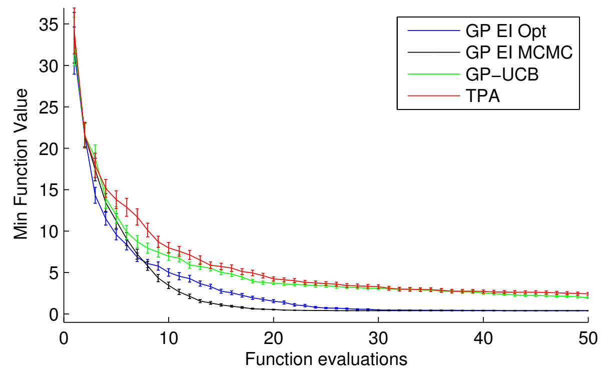
Fig 3: A comparison of standard approaches compared to our GP EI MCMC approach on the Branin-Hoo function (3a) and training logistic regression on the MNIST data (3b).

4.1. *Branin-Hoo and Logistic Regression.* We first compare to standard approaches and the recent Tree Parzen Algorithm[2] (TPA) of Bergstra et al. (2011) on two standard problems. The Branin-Hoo function is a common benchmark for Bayesian optimization techniques (Jones, 2001) that is defined over $x \in \mathbb{R}^2$ where $0 \leq x_1 \leq 15$ and $-5 \leq x_2 \leq 15$. We also compare to TPA on a logistic regression classification task on the popular MNIST data. The algorithm requires choosing four hyperparameters, the learning rate for stochastic gradient descent, on a log scale from 0 to 1, the ℓ_2 regularization parameter, between 0 and 1, the mini batch size, from 20 to 2000 and the number of learning epochs, from 5 to 2000. Each algorithm was run on the Branin-Hoo and logistic regression problems 100 and 10 times respectively and mean and standard error are reported. The results of these analyses are presented in Figures 3a and 3b in terms of the number of times the function is evaluated. On Branin-Hoo, integrating over hyperparameters is superior to using a point estimate and the GP EI significantly outperforms TPA, finding the minimum in fewer than half as many evaluations, in both cases.

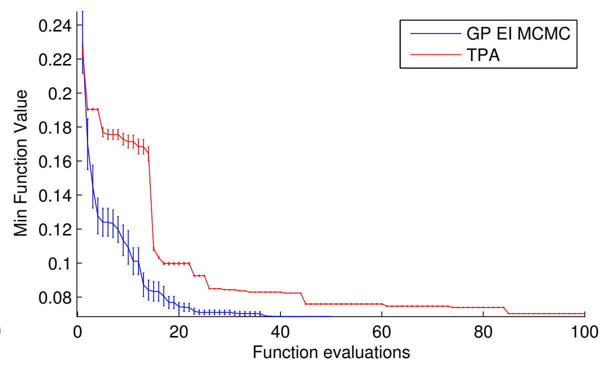
4.2. *Online LDA.* Latent Dirichlet allocation (LDA) is a directed graphical model for documents in which words are generated from a mixture of multinomial "topic" distributions. Variational Bayes is a popular paradigm for learning and, recently, Hoffman et al. (2010) proposed an online learning approach in that context. Online LDA requires two learning parameters, τ_0 and κ, that control the learning rate $\rho_t = (\tau_0 + t)^{-\kappa}$ used to update the variational parameters of LDA based on the t^{th} minibatch of document word count vectors. The size of the minibatch is also a third parameter that must be chosen. Hoffman et al. (2010) relied on an exhaustive grid search of size $6 \times 6 \times 8$, for a total of 288 hyperparameter configurations.

We used the code made publically available by Hoffman et al. (2010) to run experiments with online LDA on a collection of Wikipedia articles. We downloaded a random set of 249,560 articles, split into training, validation and test sets of size 200,000, 24,560 and 25,000 respectively. The documents are represented as vectors of word counts from a vocabulary of 7,702 words. As reported in Hoffman et al. (2010), we used a lower bound on the per word perplexity of the validation set documents as the performance measure. One must also specify the number of topics and the hyperparameters η for the symmetric Dirichlet prior over the topic

[2]Using the publicly available code from `https://github.com/jaberg/hyperopt/wiki`



(a)



(b)

图3：与我们在Branin−Hoo函数（3A）上的GP EI MCMC方法相比，标准方法的比较和MNIST数据（3B）上的训练逻辑回归。

4.1。Branin−Hoo和Logistic回归。我们首先与标准方法进行比较 Bergstra等人的最近树Parzen算法2（TPA）。（2011）在两个标准问题上。Branin−Hoo函数是贝叶斯优化技术（JONES，2001）的共同基准，其定义为0≤x1≤15和−5≤x2≤15.我们还与TPA相比逻辑回归流行Mnist数据的分类任务。该算法需要选择四个超级参数，为时码梯度下降的学习率，在0到1的日志比例上，χ2正则化参数，在0到1之间，迷你批量大小，从20到2000和学习的数量之间和学习人数从5到2000年的时期。每种算法在Branin−hoo和Logistic回归问题上运行100和10次，并且报告了均值和标准错误。这些分析的结果在图3A和3B中呈现在评估功能的次数方面。在Branin−Hoo上，整合到HyperParameters优于使用点估计，GP EI显着优于TPA，在这两种情况下，在少于一半的评估中发现最小值。

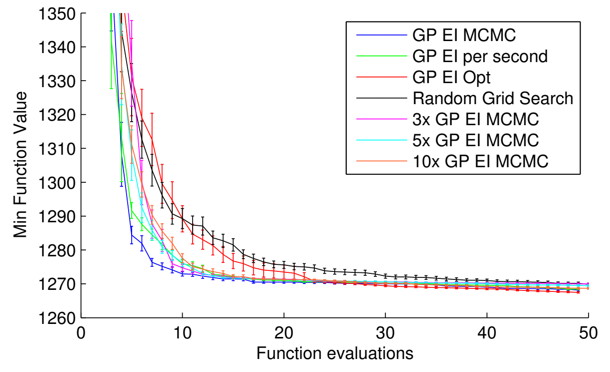
4.2。在线LDA。潜在的Dirichlet分配（LDA）是Doc−的指导图形模型

从多项式"主题"分布的混合中生成单词的解剖。变差贝叶斯是一个受欢迎的学习范式，最近霍夫曼等人。（2010）在该背景下提出了在线学习方法。在线LDA需要两种学习PA − rameters，τ0和κ，控制速度ρt=（τ0+t）−κ基于文档字数向量的Tth小匹匹匹匹匹匹匹配项来更新LDA的变分参数。小靶的大小也是必须选择的第三个参数。Hoffman等人。（2010）依赖于尺寸为6×6×8的详尽网格搜索，总共288个超级参数配置。

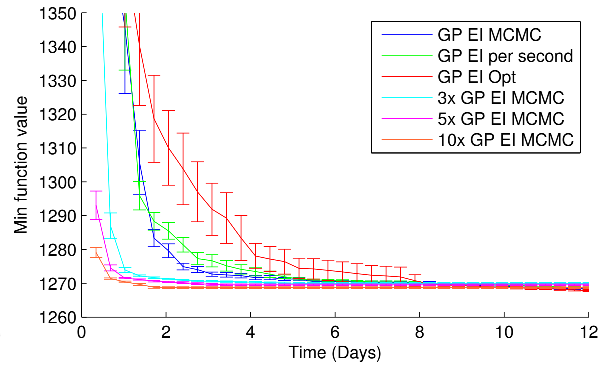
我们使用了Hoffman等人发布的代码。（2010）运行实验

在网上LDA与一系列维基百科文章。我们下载了一个随机组的249,560篇文章，分为训练，验证和测试尺寸为200,000,24,560和25,000套。这些文件表示为来自7,702字词汇的词汇量的载体。据霍夫曼等人报道。（2010），我们在验证集文档的每个单词下面的单词上使用了下限作为性能测量。一个必须指定在主题之前的对称Dirichlet的主题和超级参数η的数量

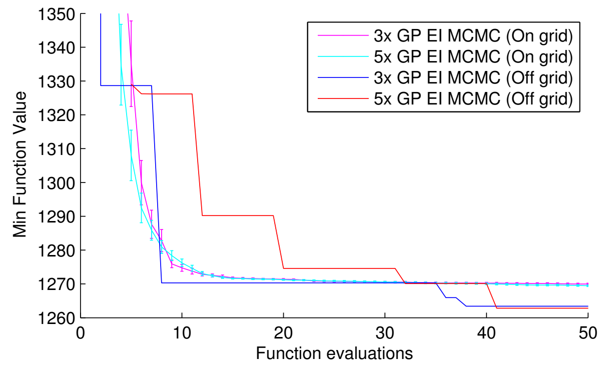
2使用https://github.com/jaberg/hyperopt/wiki的公开可用的代码



(a)



(b)



(c)

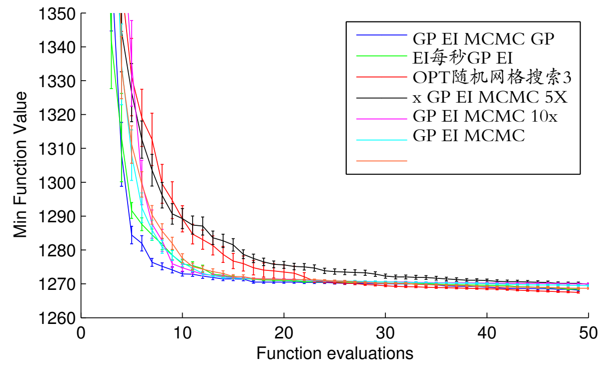
Fig 4: Different strategies of optimization on the Online LDA problem compared in terms of function evaluations (4a), walltime (4b) and constrained to a grid or not (4c).

distributions and α for the symmetric Dirichlet prior over the per document topic mixing weights. We followed Hoffman et al. (2010) and used 100 topics and $\eta = \alpha = 0.01$ in our experiments in order to emulate their analysis and repeated exactly the grid search reported in the paper[3]. Each online LDA evaluation generally took between five to ten hours to converge, thus the grid search requires approximately 60 to 120 processor days to complete.

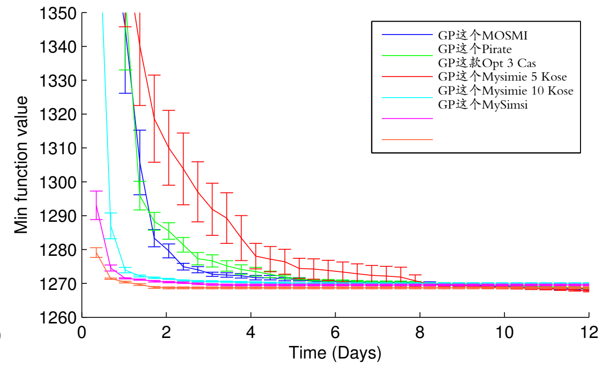
In Figures 4a and 4b we compare our various strategies of optimization over the same grid on this expensive problem. That is, the algorithms were restricted to only the exact parameter settings as evaluated by the grid search. Each optimization was then repeated one hundred times (each time picking two different random experiments to initialize the optimization with) and the mean and standard error are reported. Figures 4a and 4b respectively show the average minimum loss (perplexity) achieved by each strategy compared to the number of times online LDA is evaluated with new parameter settings and the duration of the optimization in days. Figure 4c shows the average loss of 3 and 5 times parallelized GP EI MCMC which are restricted to the same grid as compared to a single run of the same algorithms where the algorithm can flexibly choose new parameter settings within the same range by optimizing the expected improvement.

In this case, integrating over hyperparameters is superior to using a point estimate. While

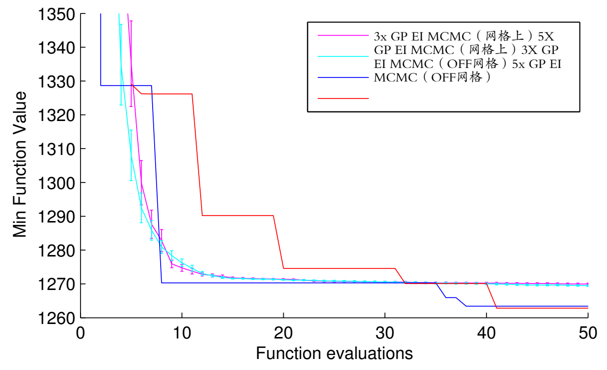
[3] i.e. the only difference was the randomly sampled collection of articles in the data set and the choice of the vocabulary. We ran each evaluation for 10 hours or until convergence.



(a)



(b)



(c)

图4：在功能评估（4A），WallTime（4B）方面比较了在线LDA问题的不同优化策略，并且被约束到网格（4C）。

分布和 α 对对称Dirichlet以每份文档主题混合重量。我们跟随Hoffman等。（2010）并使用了100个主题和 $\eta = \alpha = 0.01$，以便模拟其分析并重复在Paper3中报告的网格搜索。每个在线LDA评估通常需要五到十个小时才能收敛，因此网格搜索需要大约60到120个处理器天数来完成。

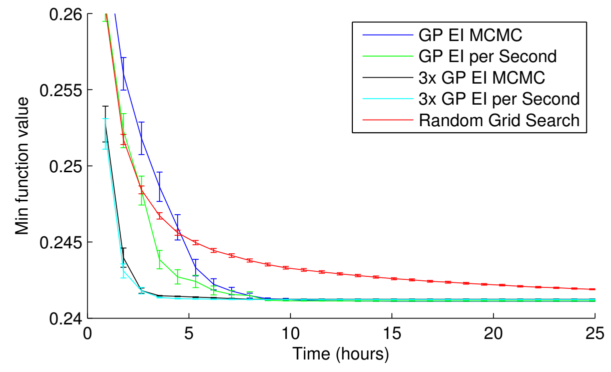
在图4A和4B中，我们将我们的各种优化策略与同一网格进行比较

在这个昂贵的问题上。也就是说，算法仅限于仅由网格搜索评估的确切参数设置。然后，每次优化重复一百次（每次挑选两个不同的随机实验以初始化优化）和平均值和标准误差。图4A和4B分别示出了与在线LDA的次数与新参数设置和在几天内的优化持续时间进行评估的次数相比，每个策略所实现的平均最小损耗（困惑）。图4C显示了3和5次并行化GP EI MCMC的平均损耗，该GP EI MCMC被限制在相同的网格上，与单个运行相同的算法相比，通过优化预期的改进，算法可以灵活地选择相同范围内的新参数设置。

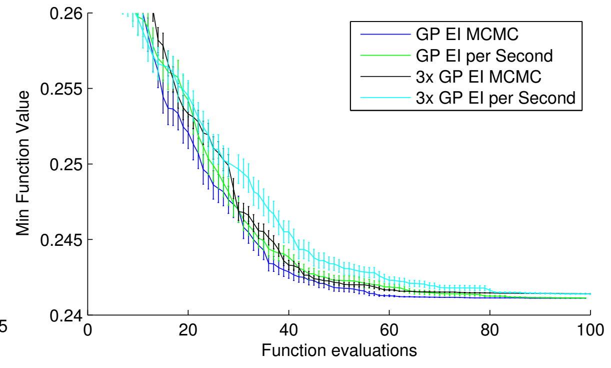
在这种情况下，整合到HyperParameters优于使用点估计。尽管

3i.e.唯一的区别是数据集中随机采样的文章集合和选择词汇。我们每次评估10小时或直到收敛。

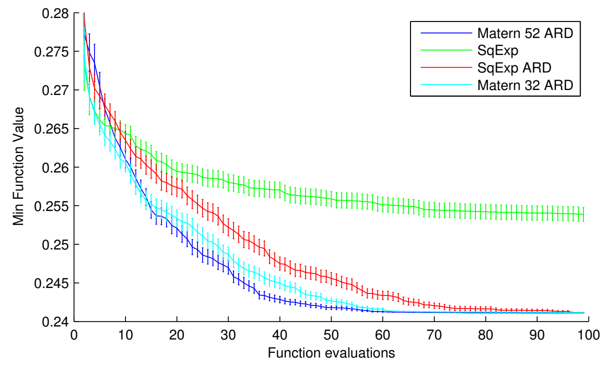
GP EI MCMC is the most efficient in terms of function evaluations, we see that parallelized GP EI MCMC finds the best parameters in significantly less time. Finally, in Figure 4c we see that the parallelized GP EI MCMC algorithms find a significantly better minimum value than was found in the grid search used by Hoffman et al. (2010) while running a fraction of the number of experiments.



(a)



(b)



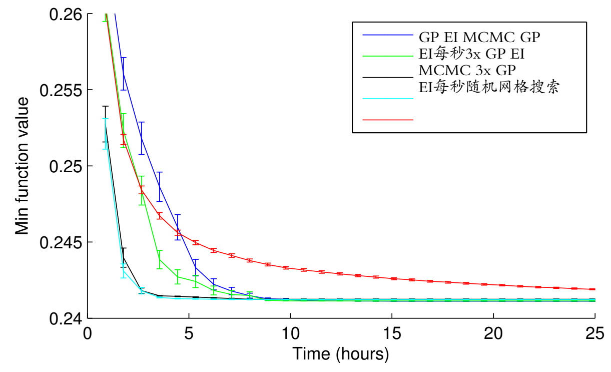
(c)

Fig 5: A comparison of various strategies for optimizing the hyperparameters of M3E models on the protein motif finding task in terms of wallclock time (5a), function evaluations (5b) and different covariance functions(5c).

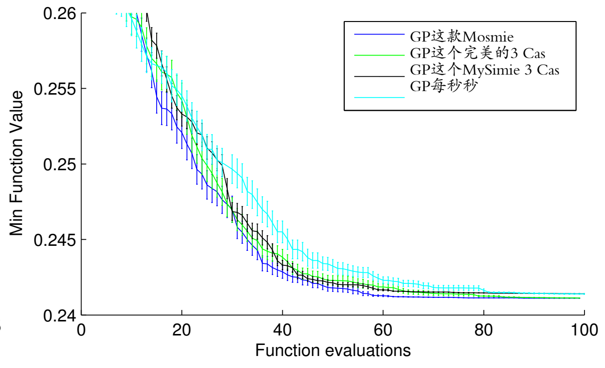
4.3. *Motif Finding with Structured Support Vector Machines.* In this example, we consider optimizing the learning parameters of Max-Margin Min-Entropy (M3E) Models (Miller et al., 2012), which include Latent Structured Support Vector Machines (Yu and Joachims, 2009) as a special case. Latent structured SVMs outperform SVMs on problems where they can explicitly model problem-dependent hidden variables. A popular example task is the binary classification of protein DNA sequences (Miller et al., 2012; Yu and Joachims, 2009; Kumar et al., 2010). The hidden variable to be modeled is the unknown location of particular subsequences, or *motifs*, that are indicators of positive sequences.

Setting the hyperparameters, such as the regularisation term, C, of structured SVMs remains a challenge and these are typically set through a time consuming grid search procedure as is done in Miller et al. (2012) and Yu and Joachims (2009). Indeed, Kumar et al. (2010) report that hyperparameter selection was avoided for the motif finding task due to being too computationally expensive. However, Miller et al. (2012) demonstrate that classification

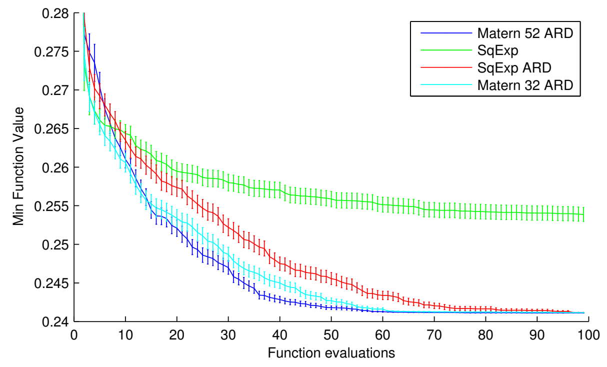
GP EI MCMC在功能评估方面是最有效的，我们看到并行化GP EI
MCMC在显着更少的时间内找到最佳参数。最后，在图4C中，我们看到并行化GP EI
MCMC算法发现比Hoffman等人使用的网格搜索中的最小值明显更好的最小值。（2010），同时运行一小部分实验。



(a)



(b)



(c)

图5：在壁克服时间（5A），功能评估（5B）和不同协方差功能（5C）方面，在蛋白质图案发现任务上优化M3E模型的超额参数的各种策略的比较。

4.3。主题查找结构化支持向量机。在这个例子中，我们考虑

优化MAX−ramgin
MIN−entopy（M3E）型号的学习参数（Miller等，2012），包括潜在结构化支持向量机（YU和Joachims，2009）作为一个特殊情况。潜在结构化的SVMS概率表达了SVMS的问题，它们可以明确地模拟问题依赖性隐藏变量。一个流行的例子任务是蛋白质DNA序列的二进制分类（Miller等，2012年; Yu和Joachims，2009; Kumar等，2010）。要建模的隐藏变量是特定子序列或图案的未知位置，或者是正序列的指标。

设置具有结构化SVM的正则化术语C等reledParameters（如正则化术语）

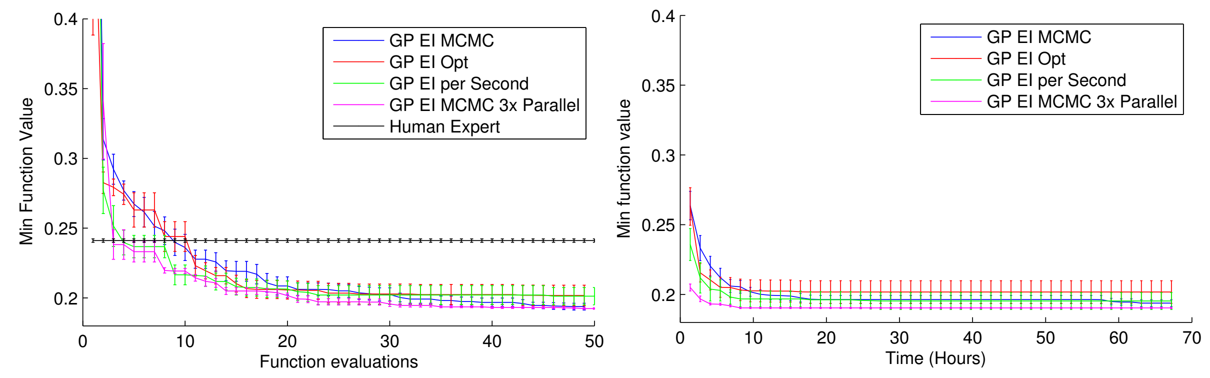
主电源是挑战，而这些挑战通常通过米勒等人完成的耗时的网格搜索程序来设置。（2012）和Yu和Joachims（2009）。的确，Kumar等人。（2010）报告称，由于过于计算昂贵，因此避免了为主题查找任务避免了超级参数选择。但是，米勒等人。（2012）证明分类

results depend highly on the setting of the parameters, which differ for each protein.

M3E models introduce an entropy term, parameterized by α, which enables the model to significantly outperform latent structured SVMs. This additional performance, however, comes at the expense of an additional problem-dependent hyperparameter. We emulate the experiments of Miller et al. (2012) for one protein with approximately 40,000 sequences. We explore 25 settings of the parameter C, on a log scale from 10^{-1} to 10^{6}, 14 settings of α, on a log scale from 0.1 to 5 and the model convergence tolerance, $\epsilon \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$. We ran a grid search over the 1,400 possible combinations of these parameters, evaluating each over 5 random 50-50 training and test splits.

In Figures 5a and 5b, we compare the randomized grid search to GP EI MCMC, GP EI per Second and their 3x parallelized versions, all constrained to the same points on the grid, in terms of minimum validation error vs wallclock time and function evaluations. Each algorithm was repeated 100 times and the mean and standard error are shown. We observe that the Bayesian optimization strategies are considerably more efficient than grid search which is the status quo. In this case, GP EI MCMC is superior to GP EI per Second in terms of function evaluations but GP EI per Second finds better parameters faster than GP EI MCMC as it learns to use a less strict convergence tolerance early on while exploring the other parameters. Indeed, 3x GP EI per second is the least efficient in terms of function evaluations but finds better parameters faster than all the other algorithms.

Figure 5c compares the use of various covariance functions in GP EI MCMC optimization on this problem. The optimization was repeated for each covariance 100 times and the mean and standard error are shown. It is clear that the selection of an appropriate covariance significantly affects performance and the estimation of length scale parameters is critical. The assumption of the infinite differentiability of the underlying function as imposed by the commonly used squared exponential is too restrictive for this problem.



(a) (b)

Fig 6: Validation error on the CIFAR-10 data for different optimization strategies.

4.4. *Convolutional Networks on CIFAR-10.* Neural networks and deep learning methods notoriously require careful tuning of numerous hyperparameters. Multi-layer convolutional neural networks are an example of such a model for which a thorough exploration of architechtures and hyperparameters is beneficial, as demonstrated in Saxe et al. (2011), but often computationally prohibitive. While Saxe et al. (2011) demonstrate a methodology for efficiently exploring model architechtures, numerous hyperparameters, such as regularisation

结果高度依赖于对每个蛋白质不同的参数的设置。

M3E模型引入熵项，由α参数化，这使得该模型能够

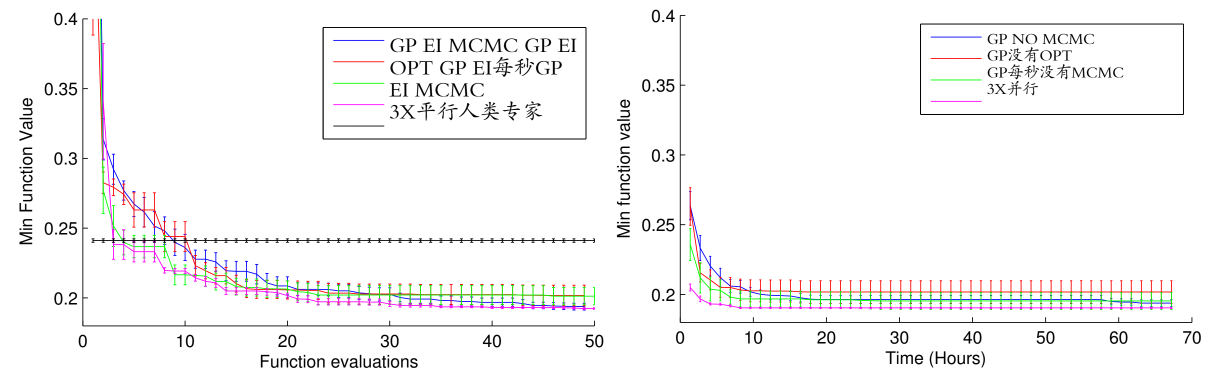
显着优于延迟潜在结构化的SVM。然而，这种额外的性能是以额外的问题依赖的覆盖率为代价。我们模拟了Miller等人的实验。（2012）对于具有约40,000个序列的蛋白质。我们探索参数c的25个设置，在10−1到106,14的日志比例上，14个α设置，在0.1到5的日志比例和模型会聚公差，ε ∈{10−4,10−3，10−2,10−1}。我们在这些参数的1,400种可能的组合中运行了网格搜索，每次评估超过5个随机50−50训练和测试分裂。

在图5A和5B中，我们将随机网格搜索与GP EI MCMC进行比较，GP EI

第二个及其3x并行化版本，所有在网格上的相同点都受到最小验证误差与壁点时间和函数评估。每种算法重复100次，显示平均值和标准误差。我们观察到贝叶斯优化策略比网格搜索更有效，这是现状。在这种情况下，GP EI MCMC在功能评估方面优于GP EI，但每秒GP EI可以比GP EI MCMC从GP EI MCMC获取更好的参数，因为它在探索其他参数时使用不太严格的收敛耐受性。实际上，每秒3x GP EI在功能评估方面是最不高的效率，而是比所有其他算法更快地找到更好的参数。

图5C比较了在GP EI MCMC优化中使用各种协方差函数的使用

关于这个问题。对每个协方差重复优化100次，并显示平均值和标准误差。很明显，选择适当的协方差显着影响性能，长度比例参数的估计至关重要。普遍使用的平方指数施加的潜在功能的无限差异的假设对于这个问题来说太限制了。



(a) (b)

图6：用于不同优化策略的CIFAR−10数据上的验证错误。

4.4。CiFar−10上的卷积网络。神经网络和深层学习方法

臭名昭着地需要仔细调整众多的覆盖物。多层卷积神经网络是这种模型的一个例子，其彻底探索AR−Chiture和HyperParameters是有益的，如Saxe等人所证明的那样。（2011年），但经常计算令人望而却步。虽然Saxe等人。（2011）展示了一种有效地探索模型architechtures的方法，众多超参数，如正则化

parameters, remain. In this empirical analysis, we tune nine hyperparameters of a three-layer convolutional network, described in Krizhevsky (2009) on the CIFAR-10 benchmark dataset using the code provided[4]. This model has been carefully tuned by a human expert (Krizhevsky, 2009) to achieve a highly competitive result of 18% test error, which matches the published state of the art[5] result (Coates and Ng, 2011) on CIFAR-10. The parameters we explore include the number of epochs to run the model, the learning rate, four weight costs (one for each layer and the softmax output weights), and the width, scale and power of the response normalization on the pooling layers of the network.

We optimize over the nine parameters for each strategy on a withheld validation set and report the mean validation error and standard error over five separate randomly initialized runs. Results are presented in Figure 6 and contrasted with the average results achieved using the best parameters found by the expert. The best hyperparameters[6] found by the GP EI MCMC approach achieve an error on the *test set* of 14.98%, which is over 3% better than the expert and the state of the art on CIFAR-10.

5. Conclusion. In this paper we presented methods for performing Bayesian optimization of hyperparameters associated with general machine learning algorithms. We introduced a fully Bayesian treatment for expected improvement, and algorithms for dealing with variable time regimes and parallelized experiments. Our empirical analysis demonstrates the effectiveness of our approaches on three challenging recently published problems spanning different areas of machine learning. The code used will be made publicly available. The resulting Bayesian optimization finds better hyperparameters significantly faster than the approaches used by the authors. Indeed our algorithms *surpassed* a human expert at selecting hyperparameters on the competitive CIFAR-10 dataset and as a result beat the state of the art by over 3%.

Acknowledgements. This work was supported by a grant from Amazon Web Services and by a DARPA Young Faculty Award.

References.

J Mockus, V Tiesis, and A Zilinskas. The application of Bayesian methods for seeking the extremum. *Towards Global Optimization*, 2:117–129, 1978.

D.R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21(4):345–383, 2001.

Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *ICML*, 2010.

Adam D. Bull. Convergence rates of efficient global optimization algorithms. *JMLR*, (3-4):2879–2904, 2011.

James S. Bergstra, Rémi Bardenet, Yoshua Bengio, and Bálázs Kégl. Algorithms for hyper-parameter optimization. In *NIPS*. 2011.

Eric Brochu, Vlad M. Cora, and Nando de Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *pre-print*, 2010. arXiv:1012.2599.

Carl E. Rasmussen and Christopher Williams. *Gaussian Processes for Machine Learning.* MIT Press, 2006.

H. J. Kushner. A new method for locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86, 1964.

[4]Available at: `http://code.google.com/p/cuda-convnet/` using the architechture defined in `http://code.google.com/p/cuda-convnet/source/browse/trunk/example-layers/layers-18pct.cfg`

[5]Without augmenting the training data.

[6]The optimized parameters deviate interestingly from the expert-determined settings; e.g., the optimal weight costs are asymmetric (the weight cost of the second layer is approximately an order of magnitude smaller than the first layer), a learning rate two orders of magnitude smaller, a slightly wider response normalization, larger scale and much smaller power.

参数仍然存在。在本实证分析中，我们使用所提供的代码在Cifar−10基准数据集中描述了三层卷积网络的九个冗余网络的九个超级参数。该模型由人类专家（Krizhevsky，2009）仔细调整，以实现18%的测试错误的高竞争结果，与CiFar−10上的ART5结果（Coate和Ng，2011）的公布状态匹配。我们探索的参数包括纪念模型的数量，学习率，四重重量（每个层和软MAX输出权重的一个），以及汇集层上响应标准化的宽度，比例和功率网络。

我们在持续验证集上为每种策略进行了优化的九个参数

报告平均验证误差和标准错误五个单独的随机初始化运行。结果如图6所示，与使用专家发现的最佳参数的平均结果形成鲜明对比。GP EI
MCMC方法发现的最佳超参数6在测试集中实现了14.98%的错误，比CiFar−10的专家和最先进的优于专家和最先进的测试集。

结论。在本文中，我们提出了执行贝叶斯optimiza的方法 −

与一般机器学习算法相关的近似参数。我们介绍了一个完全贝叶斯治疗的预期改进，以及处理可变时间制度和并行化实验的算法。我们的实证分析表明，我们在三个挑战最近发表了机器学习的不同AR的问题的三个挑战的方法的有效性。使用的代码将公开可用。由此产生的贝叶斯优化得到了更好的近似数目，明显比作者使用的方法更快。实际上，我们的算法超过了在竞争激烈的CiFar−10数据集上选择了Quand参数的人类专家，结果以超过3%击败了最新技术。

致谢。亚马逊Web服务的授权支持这项工作

由DARPA年轻的教师奖。

References.

j mockus，v tiesis和zilinskas。贝叶斯方法寻求极值的应用。向
Global Optimization, 2:117–129, 1978.

D.R.琼斯。基于响应曲面的全局优化方法分类。全球杂志
Optimization, 21(4):345–383, 2001.

Niranjan Srinivas，Andreas Krause，Sham Kakade和Matthias Seeger。高斯流程优化
强盗设置：无后悔和实验设计。在ICML，2010年。

亚当D.公牛。高效全局优化算法的收敛速率。JMLR，（3−4）：2879−2904,2011。詹姆斯·贝加斯特拉，鲁梅巴·贝加特，Yoshua Bengio和B'al'azs K'egl。超参数OPTI的算法 −
mization. In *NIPS*. 2011.

Eric Brochu，Vlad M. Cora和Nando de Freitas。贝叶斯优化昂贵成本的教程
函数，应用于主动用户建模和分层强化学习。预先打印，2010. Arxiv：1012.2599。

Carl E. Rasmussen和Christopher Williams。高斯机器学习工艺。MIT Press，2006. H. J. Kushner。一种在存在下定位任意多跳曲曲线的最大点的新方法
噪音。中国基础工程学报，86,1964。

4avaIsable
at：http：//code.google.com/p/cuda−convnet/使用http：//代码中定义的architechture。google.com/p/cuda−convnet/source/browse/trunk/example−layers/layers−18pct.cfg.

5推出培训数据。6优化的参数有趣地偏离专家确定的设置;例如，最佳

重量成本不对称（第二层的重量成本大约比第一层小的数量级），学习率较小两个数量级，略较宽的响应归一化，较大的规模和更小的功率。

Iain Murray and Ryan Prescott Adams. Slice sampling covariance hyperparameters of latent Gaussian models. In *NIPS*, pages 1723–1731. 2010.

Yee Whye Teh, Matthias Seeger, and Michael I. Jordan. Semiparametric latent factor models. In *AISTATS*, 2005.

Edwin V. Bonilla, Kian Ming A. Chai, and Christopher K. I. Williams. Multi-task Gaussian process prediction. In *NIPS*, 2008.

David Ginsbourger and Rodolphe Le Riche. Dealing with asynchronicity in parallel Gaussian process based global optimization. 2010.

Matthew Hoffman, David M. Blei, and Francis Bach. Online learning for latent Dirichlet allocation. In *NIPS*, 2010.

Kevin Miller, M. Pawan Kumar, Benjamin Packer, Danny Goodman, and Daphne Koller. Max-margin min-entropy models. In *AISTATS*, 2012.

Chun-Nam John Yu and Thorsten Joachims. Learning structural SVMs with latent variables. In *ICML*, 2009.

M. Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *NIPS*. 2010.

Andrew Saxe, Pang Wei Koh, Zhenghao Chen, Maneesh Bhand, Bipin Suresh, and Andrew Ng. On random weights and unsupervised feature learning. In *ICML*, 2011.

Alex Krizhevsky. Learning multiple layers of features from tiny images. *Technical report, Department of Computer Science, University of Toronto*, 2009.

Adam Coates and Andrew Y. Ng. Selecting receptive fields in deep networks. In *NIPS*. 2011.

Jasper Snoek
Department of Computer Science
University of Toronto
E-mail: jasper@cs.toronto.edu

Hugo Larochelle
Département d'informatique
Université de Sherbrooke
E-mail: hugo.larochelle@usherbrooke.ca

Ryan P. Adams
School of Engineering and Applied Sciences
Harvard University
E-mail: rpa@seas.harvard.edu

Iain Murray和Ryan Prescott Adams。潜伏高斯模型的切片采样协方差超公数。
In *NIPS*, pages 1723–1731. 2010.

Yee Whye Teh，Matthias Seeger和Michael I. Jordan。Semiparametric潜在因子模型。在奥斯特，
2005.

Edwin V. Bonilla，Kian Ming A. Chai和Christopher K. I. Williams。多任务高斯过程预测。
In *NIPS*, 2008.

David Ginsbourger和Rodolphe Le Riche。处理并行高斯过程中的异步性
global optimization. 2010.

Matthew Hoffman，David M. Blei和Francis Bach。在线学习潜在的Dirichlet分配。在尼斯，
2010.

Kevin Miller，M. Pawan Kumar，Benjamin Packer，Danny Goodman和Daphne Koller。max−margin min−
entropy models. In *AISTATS*, 2012.

Chun−Nam John Yu和Thorsten Joachims。学习具有潜在变量的结构SVM。在ICML，2009年。M. Pawan
Kumar，Benjamin Packer和Daphne Koller。潜在变量模型的自定节奏学习。在
NIPS. 2010.

巨大的性，庞伟酸，Zahenh Chen，Manish Bhark，Bipin Suresh和Agekng。随机
重量和无监督的特征学习。在ICML，2011年。

Alex Krizhevsky。从小图像学习多层特征。技术报告，部门
计算机科学，多伦多大学，2009。

亚当凯斯和安德鲁Y.NG。选择深网络中的接受字段。在尼斯。2011年。

Jasper
Snoek电脑科学大学多伦多大学电
子邮件：jasper@cs.toronto.edu

Hugo Larochelle
Département d'informatique
Université de Sherbrooke
E-mail: hugo.larochelle@usherbrooke.ca

Ryan P.
Adams工程学院和应用科学哈佛大学电子邮件
：RPA@seas.harvard.edu