分类号 _____ 编号_____

U D C_____ 密级_____

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

本科生毕业设计（论文）

题　　目： 基于 GP-hedge 的贝叶斯优化采集函数集成算法研究

姓　　名： 李辰曦

学　　号： 11911813

系　　别： 统计与数据科学系

专　　业： 统计学

指导教师： 杨丽丽 教授

2023 年 5 月 30 日

CLC _____ Number _____

UDC _____ Available for reference ☐Yes ☐No

SUSTech Southern University of Science and Technology

Undergraduate Thesis

Thesis Title: **A study of potofolio method for bayesian optimization**

Student Name: **Chenxi Li**

Student ID: **11911813**

Department: **Department of Statistics and Data Science**

Program: **Statistics**

Thesis Advisor: **Professor Lili Yang**

Date: May 30, 2023

诚信承诺书

1. 本人郑重承诺所呈交的毕业设计（论文），是在导师的指导下，独立进行研究工作所取得的成果，所有数据、图片资料均真实可靠。

2. 除文中已经注明引用的内容外，本论文不包含任何其他人或集体已经发表或撰写过的作品或成果。对本论文的研究作出重要贡献的个人和集体，均已在文中以明确的方式标明。

3. 本人承诺在毕业论文（设计）选题和研究内容过程中没有抄袭他人研究成果和伪造相关数据等行为。

4. 在毕业论文（设计）中对侵犯任何方面知识产权的行为，由本人承担相应的法律责任。

作者签名:

_____ 年____ 月____ 日

COMMITMENT OF HONESTY

1. I solemnly promise that the paper presented comes from my independent research work under my supervisor's supervision. All statistics and images are real and reliable.

2. Except for the annotated reference, the paper contents no other published work or achievement by person or group. All people making important contributions to the study of the paper have been indicated clearly in the paper.

3. I promise that I did not plagiarize other people's research achievement or forge related data in the process of designing topic and research content.

4. If there is violation of any intellectual property right, I will take legal responsibility myself.

Signature:

Date:

基于 GP-hedge 的贝叶斯优化采集函数集成算法研究

李辰曦

（统计与数据科学系　指导教师：杨丽丽）

[摘要]：贝叶斯优化是一种高效的黑盒函数优化框架，常被用于处理评估代价昂贵的复杂黑盒优化问题。该优化框架由代理模型和采集函数组成，其中采集函数决定了算法的探索策略，因而采集函数的选取对于算法的优化效果有着重要影响。然而，在优化前往往很难确定哪种采集函数适合当前的优化问题。针对这一问题，本文首先在 GP-Hedge 算法的基础上加以改进提出了 Improved GP-Hedge 算法。前者基于多臂摇赌博机问题中的 Hedge 算法构建了一个采集函数集成框架，本文对其进行了分析与改进。此外，本文还提出了一种基于投票选择机制的采集函数集成框架 vote 算法。在本文的数值实验中，这两种算法在大多数测试函数上都展现出了较好的优化性能。

[关键词]：贝叶斯优化，黑盒优化，采集函数集成算法，高斯过程，GP-Hedge，多臂摇赌博机

[**ABSTRACT**]: Bayesian optimization is an efficient black box function optimization framework that is often used to handle complex black box optimization problems with high evaluation costs. The optimization framework consists of surrogate model and acquisition function, in which the acquisition function determines the exploration strategy of the algorithm, thus the selection of acquisition function has an important impact on the optimization effect of the algorithm. However, it is often difficult to determine which acquisition function is suitable for the current optimization problem before optimization. To solve this problem, we first proposes an improved GP-Hedge algorithm based on the GP-Hedge algorithm. GP-Hedge builds a acquisition function integration framework based on the Hedge algorithm in the multi arm gambling machine problem, and we analyzes and improves it. In addition, we also proposes a acquisition function integration framework called Vote algorithm based on voting selection mechanism. In the numerical experiments of this article, these two algorithms demonstrated good optimization performance on most test functions.

[**Key words**]: Bayesian optimization, black-box optimization, acquisition function integration algorithm, Gaussian process, GP-Hedge,multi-armed bandits

目录

1. 引言

贝叶斯优化是黑盒优化领域中较为先进的优化框架，其特点在于能够以较少的目标函数评估次数来尽可能地获得接近全局最优值的解，因而被广泛使用在药物研发，神经网络参数调节等目标函数评估代价昂贵的优化问题上。该算法是一种基于模型的序贯优化[1]，在每一步迭代时，贝叶斯优化都会根据当前已掌握的信息来制定平衡探索和开发的策略用以选择下一个评估点。传统的贝叶斯优化框架通过设计一个采集函数来实行这种策略。常用的采集函数有基于概率提升的 PI[2]，基于期望提升的 EI[3]以及基于贝叶斯置信上界的 GP-UCB[4]，本文将在 2.3 节详细介绍这三种采集函数。

对于不同的优化问题，不同的采集函数往往会表现出不同的优化性能。我们很难去找到一个万能的采集函数去适应所有种类的优化问题。基于这一问题，Hoffman 等人于 2011 年提出了 GP-hedge[5]算法。其算法的核心思想是让贝叶斯优化的每一步迭代不仅仅只依赖于某一个采集函数，而是通过某种策略在多个采集函数中选择最合适的使用。该算法实质上是将多臂摇老虎机问题 (multi-armed bandits)[6]中的 Hedge 算法[7]扩展到了贝叶斯优化领域。具体来说，GP-Hedge 算法将待选的每一个采集函数视为一个臂 (bandit)，根据历史信息设计奖励函数 (reward function) 来指导每一步迭代的 bandit 选择。其文章给出的数值实验中 GP-Hedge 算法在多种测试函数中的表现均要显著优于单一的采集函数。

然而，由于 GP-Hedge 的奖励函数是基于历史信息中后验分布的均值累加来进行决策，这种决策方式会使得 GP-Hedge 算法在优化的初期倾向于选择注重开发的采集函数（因其早期得到的 reward 更大），因而可能会过早地陷入局部最优值，经过一段时间的迭代后才能跳出。为解决这一问题，Wang 等人在 hedge 的基础上提出了 ESP 算法[8]，通过计算各个采集函数推荐评估点对于最优值位置的熵的影响来选择采集函数。这种策略并不基于历史的奖励信息，因而有效地避免了优化初期的错误的函数选择。该算法在多数优化问题上的表现均优于 GP-Hedge 算法，但引入熵的计算会极大地增加计算所需要的时间成本。

本文的贡献主要有以下两点：首先，我们深入分析了 GP-Hedge 算法所存在的缺

陷，并在此基础上对其进行改进，提出了改进后的 Improved GP-Hedge 算法。此外，我们跳出 GP-Hedge 的基于历史累积数据的算法策略，提出了一种基于投票机制的采集函数集成框架。在数值实验部分我们复现了 GP-Hedge 原文的测试实验，并增加了更多的复杂测试函数来研究各个采集函数算法策略的优化表现。我们还测试了 Improved GP-Hedge 对于添加较差采集函数的鲁棒性，并发现相较于原 GP-Hedge 算法其鲁棒性有了较大的提升。本文的相关代码均已上传至 github[1], 其中贝叶斯优化算法框架是基于 Bayesian Optimization 包[2]源代码的基础上对其加以改进优化。具体的改进措施见本文的实验部分。

本文的主要结构如下：在第 2 节我们介绍了贝叶斯优化算法框架，包括其代理模型高斯过程和常用的采集函数。第 3 节是对 GP-Hedge 算法的介绍。第 4 节我们深入分析了 GP-Hedge 算法的优劣之处并在其基础上对其存在的缺陷进行针对性的改进，提出了 Improved GP-Hedge 算法。第 5 节我们提出了基于投票机制的采集函数集成算法框架 vote 算法。第 6 节为数值实验部分，我们共测试了 9 组测试函数对比各个算法的优化表现，并对比测试了 GP-Hedge 算法和 Improved GP-Hedge 算法对较差采集函数的鲁棒性。第 7 节是对全文的概括总结。

2. 贝叶斯优化

贝叶斯优化框架主要由两部分组成：代理模型和采集函数，用来解决复杂黑盒函数的全局优化问题。具体来说，记 $f(x)$ 为未知的目标黑盒函数，考虑优化问题：

$$x^* = \arg\max_{x \in \mathcal{X}} f(x) \tag{1}$$

其中 $\mathcal{X}$ 为搜索空间。目标函数 $f(x)$ 无需拥有显示表达式，只需满足对于任意的 $x_0 \in X$, 均可以计算出对应的目标函数值 $f(x_0)$。主流的贝叶斯优化框架均采用高斯过程 (见 2.1 节) 作为代理模型，用于构建目标函数的后验分布。高斯过程模型具有较强的灵活性，已被证明其等价于存在无限隐藏层单元的神经网络[9]。利用高斯过程建模之后对于搜索空间的任何一点，高斯过程模型均能返回该点对应的预测均值以及其不

[1]https://github.com/Silver439/PortfolioBO

[2]https://github.com/fmfn/BayesianOptimization

确定性。利用其返回的预测均值和不确定性，我们就能设计采集函数（见 2.2 节）来权衡开发 (exploration) 与探索 (exploitation)，以此来选择下一个观测点的位置。贝叶斯优化的具体算法流程见 **Algorithm1**。

Algorithm 1: Bayesian optimization

Input: number of initial point n_0, number of max iteration n

Randomly get initial data $D_{1:n_0}$ and update GP model;

for $t = 1, 2, ...n$ **do**

  Find x_t by optimizing the acquisition function over the GP
  $x_t = \arg\max_x u(x|D_{1:n_0+n})$;

  Sample the objective function: $y_t = f(x_t) + \epsilon_t$;

  Augment the data $D_{1:n_0+n} = D_{1:n_0+n-1}, (x_t, y_t)$ and update GP model;

end

Result: $(x_{t^*}, y_{t^*}) \in D_{1:n_0+n}, t^* = \arg\max y_{1:n_0+n}$

2.1 高斯过程

高斯过程[10]是一种非参数模型，由均值函数 $m(x) : \mathcal{X} \to \mathbb{R}$ 以及正定核函数（也可视为协方差函数）$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ 组成。对于搜索空间内任何有限点集 $x_{1:n}$，定义 $f_i := f(x_i)$ 为 x_i 处对应的目标函数值，$y_i := f(x_i) + \epsilon_i$ 为 x_i 处对应的带噪声的观测值。在高斯过程模型中，我们假设 $\mathbf{f} := f_{1:n}$ 服从联合高斯分布，且在给定 $\mathbf{f}$ 的情况下 $\mathbf{y} := y_{1:n}$ 服从正态分布，具体数学表达形式如下：

$$\mathbf{f} \mid \mathcal{X} \sim \mathcal{N}(\mathbf{m}, \mathbf{K}) \tag{2}$$

$$\mathbf{y} \mid \mathbf{f}, \sigma^2 \sim \mathcal{N}(\mathbf{f}, \sigma^2\mathbf{I}]) \tag{3}$$

该模型的特点在于我们将目标函数 $f(x)$ 视为随机变量，equation(2) 代表了变量 $f(x)$ 的先验分布。其中 $\mathbf{m}$ 代表先验的均值函数,$K_{i,j} := k(x_i, x_j)$ 为协方差矩阵。得到观测数据 $\mathcal{D}_n = (x_i, y_i)_{i=1}^n$ 后，$f(x)$ 在给定观测数据 $\mathcal{D}_n$ 的条件下，对于新的待预测值 $f(x^*)$, 由高斯过程的性质，我们有如下联合分布：

$$\begin{bmatrix} \mathbf{y} \\ f(x^*) \end{bmatrix} \sim \mathcal{N}\left(\mathbf{m}, \begin{bmatrix} \mathbf{K} + \sigma^2\mathbf{I} & \mathbf{k}(x^*) \\ \mathbf{k}(x^*)^T & k(x^*, x^*) \end{bmatrix}\right)$$

由正态分布的条件概率分布公式容易推出 $f(x^*)$ 的均值与方差函数如下：

$$m_n(x^*) = m(x^*) + \mathbf{k}(x^*)^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}(\mathbf{y} - \mathbf{m}(\mathbf{x})) \tag{4}$$

$$\sigma_n^2(x^*) = k(\mathbf{x}, \mathbf{x}) - \mathbf{k}(\mathbf{x})^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{k}(\mathbf{x}) \tag{5}$$

公式中的 $\mathbf{k}(x^*)$ 是 x^* 与 $\mathbf{x}_{1:n}$ 之间的协方差向量。值得一提的是，若我们对目标函数并没有十分具体的先验信息，则先验均值函数 **m** 一般直接取 **0**[11]。有了高斯过程模型后，我们对于搜索空间上的每一个点均能获得其预测均值和预测方差，这给采集函数的设计提供了重要基础。

2.2 协方差函数与超参数优化

在 2.1 节中我们提到高斯过程由均值函数和协方差函数组成。其中协方差函数的选取对于模型的拟合效果至关重要，因其描述了搜索空间中不同位置的点之间的相关关系。传统的贝叶斯优化框架一般假设目标函数利普希兹连续且具有平稳性。在我们对目标函数没有特别的先验信息时，主流的贝叶斯优化框架大多从 Matérn 协方差函数簇中选取[10][1]。常用的 Matérn 协方差函数如下：

$$k_{Matrn\frac{1}{2}}(\mathbf{x}, \mathbf{x}') = \theta_0^2 exp(-r) \tag{6}$$

$$k_{Matrn\frac{3}{2}}(\mathbf{x}, \mathbf{x}') = \theta_0^2 exp(-\sqrt{3}r)(1 + \sqrt{3}r) \tag{7}$$

$$k_{Matrn\frac{5}{2}}(\mathbf{x}, \mathbf{x}') = \theta_0^2 exp(-\sqrt{5}r)(1 + \sqrt{5}r + \frac{5}{3}r^2) \tag{8}$$

$$k_{squre-exp}(\mathbf{x}, \mathbf{x}') = \theta_0^2 exp(-\frac{1}{2}r^2) \tag{9}$$

其中 $\frac{1}{2}$，$\frac{3}{2}$，$\frac{5}{2}$ 为控制 Matérn 协方差函数平滑性的参数 v 的取值，当 $v \to +\infty$ 时 Matérn 等价于平方指数协方差函数 (即 equation(9))。$r^2 = (x - x')^T\Lambda(x - x')$,$\Lambda$ 为对角矩阵，其每一行的对角项值 θ_i^2 为对应的每个维度上的平方长度尺度。θ_0 为振幅参数。我们用 θ 来记录集合 $\{\ \theta_0, \theta_1, ..., \theta_d\ \}$。则对于维度为 d 的搜索空间，我们建模所需要调整的超参数数量为 $d + 1$ 个。关于超参数优化，我们通常采用最大化边际似然的方法，

具体来说，由 equation(2) 和 equation(3) 我们可以推导出边际似然分布：

$$p(\mathbf{y} \mid \mathbf{X}, \theta) = \int p(\mathbf{y} \mid \mathbf{f}) p(\mathbf{f} \mid \mathbf{X}, \theta) df = \mathcal{N}(\mathbf{m}, \mathbf{K} + \sigma^2 \mathbf{I}) \tag{10}$$

其中 **X** 为当前已观测评估点 $\{ x_1, x_2, ..., x_n \}$,**f** 为其对应的函数值集合 $\{ f(x_1), f(x_2), ..., f(x_n) \}$,**y** 为对应的观测值集合 $\{ y_1, y_2, ..., y_n \}$。将 (10) 式取对数后展开得到表达式：

$$log\mathbf{p}(\mathbf{y} \mid \mathbf{X}, \theta) = -\frac{1}{2}(\mathbf{y} - \mathbf{m})^T (\mathbf{K}^\theta + \sigma^2 \mathbf{I})^{-1} (\mathbf{y} - \mathbf{m}) - \frac{1}{2} log \mid \mathbf{K}^\theta + \sigma^2 \mathbf{I} \mid - \frac{n}{2} log(2\pi) \tag{11}$$

取 $\theta^* = \arg\max_\theta log\mathbf{p}(\mathbf{y} \mid \mathbf{X}, \theta)$ 用以更新模型的超参数集。

2.3 采集函数

在贝叶斯优化中，采集函数被设计用来有策略地从搜索空间中选择下一个评估点。该函数将会利用高斯过程模型对当前已观测数据集的后验分布所提供的搜索空间各个点的预测均值和预测方差，在开发和探索之间自动权衡以选出最优的评估点。具体的数学形式如下：

$$\mathbf{x}_{t+1} = \underset{\mathbf{x} \in \mathcal{X}}{max}\, u(\mathbf{x}; D_{1:t}) \tag{12}$$

其中 $u(\mathbf{x}; D_{1:t})$ 代表采集函数，它将输入值 **x** 映射到实数空间上，通过最大化采集函数 u 来寻找下一轮的待评估点 $\mathbf{x}_{t+1}$。常用的采集函数共有三种类型，接下来将分别介绍这三类采集函数。

2.3.1 Probability of improvement(PI)[2]

Probability of improvement 是最早使用的采集函数，其思想是选择一个待评估点，使得其相对于当前最优值 (即在 n 个观测值中的最大观测值) 的提升概率最大。具体的数学表达形式为：

$$PI(\mathbf{x}; D_{1:n}) = P(f(\mathbf{x}) \geq \mu_n^+ + \xi) = \Phi\Big(\frac{\mu_n(\mathbf{x}) - \mu_n^+ - \xi}{\sigma_n(\mathbf{x})}\Big) \tag{13}$$

其中 μ^+ 为当前最优值，$\mu_n(\mathbf{x}), \sigma_n(\mathbf{x})$ 分别为高斯过程模型对输入值 **x** 所返回的预测均值和预测方差。Φ 为标准正态分布的累积分布函数 (CDF)。ξ 是一个人为设置的参数，用于调整采集函数对于开发和探索的倾向性大小,ξ 越大，该采集函数越倾向于探索。若 $\xi = 0$，则 PI 将会完全注重于开发，这通常不利于算法的优化效果。

2.3.2 Expected improvement(EI)[3]

Expected improvement 是基于 PI 改进而来的采集函数。在上一节对 PI 的介绍中我们不难发现：PI 的搜索策略只考虑了待评估点相对于当前最优值提升的概率大小，却并没有考虑其相对于当前最优值提升量的大小。EI 将提升量的大小也考虑进来，具体来说，我们先定义改进函数 I(improvement function):

$$I(\mathbf{x}, v, \theta) := (v - \mu_n^+)\mathbb{I}(v > \mu_n^+) \tag{14}$$

其中 $v \sim \mathcal{N}(\mu(\mathbf{x}), \sigma^2(\mathbf{x}))$,$\mathbb{I}$ 为示性函数，θ 为超参数集。对 v 求期望即得到采集函数 EI：

$$u(\mathbf{x}; D_{1:n}) := \mathbb{E}[I(\mathbf{x}, v, \theta)] = (\mu_n(\mathbf{x}) - \mu_n^+)\Phi\Big(\frac{\mu_n(\mathbf{x}) - \mu_n^+}{\sigma(\mathbf{x})}\Big) + \sigma_n(\mathbf{x})\phi\Big(\frac{\mu_n(\mathbf{x}) - \mu_n^+}{\sigma_n(\mathbf{x})}\Big) \tag{15}$$

其中 Φ 和 ϕ 分别为标准正态分布的累积分布 (CDF) 和概率密度分布 (PDF) 与 EI 类似，我们同样可以在 EI 中添加人为设置的平衡参数 ξ 以调整采集函数的性能，添加后 EI 表达式如下：

$$u(\mathbf{x}; D_{1:n}) = (\mu_n(\mathbf{x}) - \mu_n^+ - \xi)\Phi\Big(\frac{\mu_n(\mathbf{x}) - \mu_n^+ - \xi}{\sigma_n(\mathbf{x})}\Big) + \sigma_n(\mathbf{x})\phi\Big(\frac{\mu_n(\mathbf{x}) - \mu_n^+ - \xi}{\sigma_n(\mathbf{x})}\Big) \tag{16}$$

2.3.3 GP-UCB[4]

置信边界策略是一种权衡开发与探索的方法，被广泛使用在 K 臂摇赌博机和蒙特卡罗搜索树等决策问题中。而 GP-UCB 是将该策略应用于高斯过程。具体来说，应用 UCB 策略的采集函数表达式为：

$$u(\mathbf{x}; D_{1:n}) = \mu_n(\mathbf{x}) + \beta_n\sigma_n(\mathbf{x}) \tag{17}$$

其中 β_n 是一个人为设置的超参数，用于权衡开发与探索。

3. GP-Hedge 算法介绍

在 2.3 节中我们介绍了三种不同类型的采集函数，不同的采集函数在面对不同的优化问题时会有着不同的优化效果。事实上，在面对实际的优化问题时，我们很难事先判断哪一种采集函数能够起到最好的优化效果。同样地，我们也不可能找到一种采集函数能够完美地解决所有的优化问题[8]。需要注意的是，贝叶斯优化算法往往被应用到评估代价较为昂贵的优化问题当中，这就导致我们不可能将计算资源花费在评估采集函数的性能上。然而如果我们仅仅随意地使用一种采集函数去完成优化，我们将很难保证它能够有较好的优化效果。GP-Hhedge 算法[5]正是为解决这一问题而设计。其核心思想在于事先准备多种采集函数，在每一轮迭代中依照某种策略选择使用其中一种采集函数，以此来解决使用单一采集函数在一些优化问题中效果不佳的问题。接下来将具体介绍该算法的细节部分。

假设在优化前已经设置好了 N 种采集函数 $\{ u_1, u_2, ...u_N \}$,GP-Hedge 算法借鉴了多臂摇赌博机问题中 Hedge 算法[7]的优化思想，将每一种采集函数都视为多臂摇赌博机问题中的一个臂 (bandit)。在传统的多臂摇赌博机问题中[6]，算法每一轮都会依照一个概率分布去选择一个臂，而每选择一个臂都会得到相应的回报 (reward)，算法会根据所得到的回报数据来调整各个臂被选择的概率分布。GP-Hedge 算法将这种方式扩展到了贝叶斯优化中。不同之处在于，在传统的多臂摇赌博机问题中，每一轮迭代都只有一个臂会得到回报数据 (因为我们每一轮只能选择一个臂)，而在贝叶斯优化中，GP-Hedge 将回报定义为更新后验分布后各个采集函数所选评估点的预测均值。在每一轮迭代中，GP-Hedge 算法会按照既定的概率分布随机挑选一种采集函数，使用该采集函数计算出下一个评估点，计算出该评估点所对应的观测值并更新高斯过程模型的后验分布。GP-Hedge 算法使用计算出的观测值作为所选采集函数在这一轮迭代中所获得的回报。而对于其他未选择到的采集函数，则会使用其推荐评估点的后验均值作为该轮迭代的回报值。这样一来，每一轮迭代中每种采集函数都会获得相应的回报值，GP-Hedge 算法可以利用这些回报值数据来更新决定各个采集函数被选概

率的概率分布函数。具体的算法框架见 **Algorithm2**[3]

Algorithm 2: GP-Hedge

Select parameter $\eta \in \mathbb{R}^+$;
Set $g_0^i = 0$ for $i = 1, ..., N$;
for $t = 1, 2, ...$ **do**
    Nominate points from each acquisition function: $\mathbf{x}_t^i = argmax_{\mathbf{x}} u_i(\mathbf{x} \mid D_{1:t-1})$;
    Select nominee $\mathbf{x}_t = \mathbf{x}_t^j$with probability:
    $p_t(j) = exp(\eta g_{t-1}^j) / \sum_{l=1}^k exp(\eta g_{t-1}^l)$;
    Sample the objective function $y_t = f(\mathbf{x}_t) + \epsilon_t$;
    Augment the data $D_{1:t} = \{D_{1:t-1}, (\mathbf{x}_t, y_t)\}$;
    Receive rewards $r_t^i = \mu_t(\mathbf{x}_t^i)$from the updated GP;
    Update gains $g_t^i = g_{t-1}^i + r_t^i$
end

4. Improved GP-Hedge 算法

4.1 GP-Hedge 算法分析

在第 3 节我们详细介绍了 GP-Hedge 算法，该算法最明显的优点在于提供了一个集成采集函数的算法框架，使得我们能够尽可能地避免因选择单一采集函数而造成的优化效果不佳的问题。同时，值得注意的是，GP-Hedge 算法相比于选择单一采集函数而言并未增加太多的计算量，这也是其算法的优势所在。

然而，我们需要指出的是，GP-Hedge 算法在设计上也同样存在着不少缺陷。具体列举如下：

(1) 该算法在优化初期对倾向于探索的采集函数并不友好，因为相对于倾向开发的采集函数，它们在优化的初期获得的回报往往较少，使得 GP-Hedge 算法框架在优化的初期更倾向于选择开发性的采集函数。这显然在优化问题的初期是不合理的，因为优化初期我们对目标函数的信息知之甚少，更应该鼓励探索而并非开发。优化初期鼓励开发可能会导致过早地陷入局部最优。

(2) 该算法没有区分每一轮回报信息的重要性。由于优化初期我们对目标函数的建模还十分不精确，所以该阶段获得的回报 (即后验均值) 并不准确，对优化中后期的决策参考价值不大。而 GP-Hedge 算法对这些信息全部一视同仁，这可能导致优化

[3]该算法框架伪代码直接引用自 GP-Hedge 的原文

初期的不准确信息干扰中后期做出正确决策。

(3) 该算法同样没有考虑计算资源这一重要信息。理想的优化过程应当是在计算资源充裕时尽量探索，而在计算资源不足时 (即剩余迭代次数不多时) 倾向于开发以充分利用以掌握的信息获得较好的结果。

(4) 该算法中由人为设置的超参数 η 在我们对目标函数没有先验的信息时难以设置。该参数主要用于调整指数幂的大小使得指数不至于过大。但若不了解函数的取值范围，我们将很难为该参数选取一个合适的值。

4.2 GP-Hedge 算法改进

接下来我们将针对上一小节提到的问题对 GP-Hedge 算法进行优化改进。我们将改进后的算法命名为 Improved GP-Hedge 算法。具体的改进措施列举如下：

(1) 针对问题 (1)，我们考虑将各个采集函数所选评估点的不确定性也加入到对应采集函数的回报中去，即采集函数每一轮迭代的回报不仅仅只由后验均值决定，还会与所选点的预测方差有关。值得注意的是对于第 n 轮迭代，我们将使用第 n-1 轮的 GP 模型去计算各个采集函数所选点的不确定性，而非使用第 n 轮更新后的 GP 模型计算。否则该轮被选择的采集函数其所选点的不确定性将始终为零。由于优化初期后验均值一般相对较小，而预测方差则相对较大，这样的改进方式将鼓励算法在优化初期倾向于进行探索而非开发。

(2) 针对问题 (2)，我们借鉴了指数移动平均的思想，在每一轮迭代中，我们都会在 **Algorithm2** 中的累积回报函数 g_t^i 之前乘上一个常数 c ($0 < c < 1$, 在我们的实验中取 $c = 0.95$)。这项改动会使得随着优化过程的进行，较早轮次的回报值在总回报值中所占的比重会越来越小，从而避免早期的不精确信息影响优化中后期的决策。

(3) 针对问题 (3)，我们考虑利用最大迭代次数这一信息辅助决策。从直观的逻辑上来说，我们应当在优化初期鼓励探索，优化末期鼓励开发，因此在改进 (1) 中不确定性在回报函数中所占的比重大小应当随着优化轮次的增加而逐渐下降。这里我们考虑在预测方差前乘上 $log_m(m - n + 1)$ 来实现这一目标。其中 m 为最大迭代次数，n 为当前迭代轮次。在优化开始时该式值为 1，结束时其值下降为 0。

最后，我们还需要说明的是，在我们的改进算法中，我们弃用了原算法中依靠概

率分布选择采集函数的方法。事实上，依靠概率分布的选择策略是一把双刃剑，它的确可以让算法在陷入局部最优时能够有概率选择其他正确的采集函数跳出局部最优，但也可能在已经掌握足够信息的情况下因概率随机地选择了错误的采集函数。在我们的改进算法中，由于我们已经减轻了优化初期信息对当前优化的影响，且将方差信息也考虑其中，不易出现重复选择某一错误函数的现象。因此，在我们的改进算法中，我们每一轮迭代直接选择累积回报值最高的采集函数，以此来提高算法表现的稳定性。由于概率分布的弃用，问题 (4) 也随之迎刃而解 (不再需要 η 这一参数)。算法的具体流程见 **Algorithm3**。

Algorithm 3: Improved GP-Hedge

Input: number of max iteration m, attenuation parameter c

Set $g_0^i = 0$ for $i = 1, ..., N$;

for $t = 1, 2, ..., m$ **do**

  Nominate points from each acquisition function: $\mathbf{x}_t^i = argmax_{\mathbf{x}} u_i(\mathbf{x} \mid D_{1:t-1})$;

  Obtain the predicted variance corresponding to each Nominate points $v_t^i = \sigma_{t-1}(\mathbf{x}_t^i)$ using the current GP model;

  Select nominee $\mathbf{x}_t = \mathbf{x}_t^j, \quad j = \arg\max_i g_t^i$;

  Sample the objective function $y_t = f(\mathbf{x}_t) + \epsilon_t$;

  Augment the data $D_{1:t} = \{D_{1:t-1}, (\mathbf{x}_t, y_t)\}$;

  Receive posterior mean $\mu_t(\mathbf{x}_t^i)$from the updated GP;

  Receive rewards $r_t^i = \mu_t(\mathbf{x}_t^i) + log_m(m - t + 1) * v_t^i$;

  Update gains $g_t^i = c * g_{t-1}^i + r_t^i$

end

5. 基于投票机制的采集函数集成框架

5.1 概述

在前文我们已经完整地介绍了 GP-Hedge 优化框架及其改进算法。现在让我们把注意力放在 GP-Hedge 算法的最初假设上：将各个备选的采集函数视为多臂摇老虎机的一个臂。这一假设可以使得我们将多臂摇老虎机领域的优化算法直接迁移到 GP-Hedge 的优化框架中去。事实上 GP-Hedge 的原文中还测试了诸如 Exp3 等多臂摇老虎机领域的其他算法。然而，在传统的多臂摇老虎机问题中，每个臂给出回报的概率是固定的，并不会随着优化的进行而发生改变。因此我们可以根据历史的累积数据来推断各个臂的回报概率大小。而在贝叶斯优化中，采集函数的表现随着优化的

进行是会发生改变的。例如，有些采集函数可能在优化初期效果不佳，但在中后期有着很好的表现。事实上，即便一个采集函数在之前的优化轮次中表现优异，也并不能说明在当前轮次中它所给出的点就是最好的选择。基于上述事实我们可以做出推断：基于历史累积数据的采集函数优化框架与贝叶斯优化并不完全适配。我们需要寻找到一种新的方法，使得它能够基于当下已掌握的所有信息直接对采集函数做出选择，而非依赖各个采集函数的历史表现作为判断依据。基于这一思路，Wang 等人创造性地提出了基于信息增益的采集函数集成框架 ESP 算法[8]，即在每轮迭代中选择的采集函数其推荐评估点能够最大程度地减少目标函数最优值位置的熵。该算法框架不依赖于历史累积数据，在其原文中给出的诸多基准测试中表现均优于 GP-Hedge。然而该框架引入了熵的计算使得算法的计算复杂度大大增加，因而无法向解决高维问题进一步扩展。接下来我们将介绍一种基于投票机制的采集函数集成算法框架，该框架对采集函数的选择并不依赖于其历史累积数据，且对于计算的复杂度影响不大。

5.2 vote 算法

接下来我们将详细介绍我们所提出的基于投票机制的采集函数集成框架，我们将其称之为”vote 算法”。其算法思路灵感来源于机器学习中的经典集成模型随机森林算法。在随机森林算法中，每组决策树都会给出其相应的预测分类结果，而算法最终的输出结果遵循着” 少数服从多数” 的原则，即输出被预测数最多的类别。这种机制就好似组成随机森林算法的各组决策树在对预测类别进行投票。在 GP-Hedge 算法中，每个采集函数都会给出一个推荐点，该点能够使其对应的采集函数值取到最大值。我们注意到事实上我们也可以使用其他的采集函数去计算某一个采集函数所给出的推荐点对应的函数值。这个函数值自然会比它们自身给出的推荐点对应的函数值小，但我们可以去判断它们之间的相似程度。这一思路基于一个事实：在所有的采集函数都能够优化当前问题的条件下，若高斯过程模型对于目标函数的拟合已经较为精确，那么目标函数的最优值所对应的采集函数值也会相对较大。简而言之，若所有的采集函数都认为该点是一个相对不错的点 (即采集函数值较大)，那么我们就可以选择这个点做该轮迭代的待评估点。该算法具体流程如下：

假设共有 n 个待选采集函数 $\{u_1, u_2, ..., u_k, ..., u_n\}$, 在第 t 轮迭代中各个采集函数

给出的推荐点记为 $\{x_t^1, x_t^2, ..., x_t^n\}$。并假设前 k 个采集函数为 PI 或 EI 类的采集函数，其余为 UCB 类的采集函数。对于 PI 或 EI 类的采集函数，我们定义一个 loss 函数如下：

$$loss_1(x_t^i) = \sum_{j=1, j\neq i}^{k} \frac{u_j(x_t^j) - u_j(x_t^i)}{u_j(x_t^j)} \tag{18}$$

该函数代表了在第 t 轮迭代中第 i 个采集函数所给出的点 x_t^i 在其他采集函数上计算出的采集函数值相对于各个采集函数的最大值的损失程度。$loss(x_t^i)$ 的值越小，代表 x_t^i 越被其他采集函数所认同。值得注意的是，基于采集函数 PI 和 EI 本身的非负性质，无论 PI 或者 EI 的参数如何选取，loss 函数的分母都一定非负。然而，当优化轮次较多优化接近收敛时，有可能出现分母接近甚至等于 0 的情况 (若平衡参数 ξ 较大这种情况会更早出现)。出现该情况说明在当前轮次中采集函数认为优化已接近完成，找不到更好的推荐点。因此我们考虑在具体的算法实现中设置一个阈值 (本文实验中取的是 1e-16)，当分母小于该值时我们直接令该项为 0，代表在当前轮次该采集函数对于我们的决策已经起不到指导作用。

对于 UCB 类型的采集函数，我们需要调整 loss 函数的形式。原因在于 ucb 并无非负特性，其取值大小范围直接取决于目标函数的取值范围。这使得我们需要找到新的方式去衡量 x_t^i 在其他采集函数中的认可程度，且要注意控制函数的长度尺度。事实上，我们很难将函数的长度尺度控制到与 PI 或 EI 相似的程度，只能够选取一种折中的方式将 UCB 类型的函数添加到我们的算法框架当中。具体来说，我们先定义一个 score 函数，它代表了点 x_t^i 在其他采集函数中所对应的函数值与各采集函数最大值的相似程度：

$$score(x_t^i) = \sum_{j=k+1, j\neq i}^{n} \frac{u_j(x_t^j) - m_t^j}{u_j(x_t^j) - m_t^j} \tag{19}$$

式中的 m_t^j 是在第 t 轮迭代中第 j 个采集函数对于搜索空间内某一随机点的采集函数值，用于保证分母为正。Equation(19) 实际上与 GP-Hedge 原文中所定义的 Gap matric 完全类似，后者是用来表示当前优化最优值与真实全局最优值的接近程度。这里如果我们将 m_t^j 的值取第 j 个采集函数的全局最小值，该式将代表一般的归一化计算，此时 equation(19) 的值将会严格限制在 (0,1) 之间。但考虑到我们并不知道目标黑盒

函数具体的取值范围，若其长度尺度过大 (m_t^j 过小)，则对于搜索空间内任意一点 equation(19) 的值将会始终接近于 1，没有区分度。这种情况下在我们的投票机制中基于 UCB 的采集函数将始终" 保持中立"，对所有点都一视同仁。这就丧失了优化的决策意义。因此我们为 m_t^j 引入随机性，这样在部分优化轮次中 UCB 类型的采集函数可能会保持中立，而在其他轮次中能够起到一定的指导作用。

这里的 score 函数代表的是相似程度，要计算损失程度，我们直接用 1 减去 score 函数的各个项，得到 loss 函数如下：

$$loss_2(x_t^i) = \sum_{j=k+1, j\neq i}^{n} 1 - \frac{u_j(x_t^j) - m_t^j}{u_j(x_t^j) - m_t^j} = \sum_{j=1, j\neq i}^{n} \frac{u_j(x_t^j) - u_j(x_t^i)}{u_j(x_t^j) - m_t^j} \tag{20}$$

最终 (20) 式相比于 (18) 式只是分母上多减去了一个 m_t^j。这一处理能够保证 (20) 式始终非负，注意若 $u_j(x_t^i) < m_t^j$，(20) 式可能大于 1。我们并没有强行将 UCB 采集函数的 loss 值放缩到 $(0, 1)$ 之间, 但考虑到 x_t^i 毕竟是其他采集函数所推荐的点，而 m_t^j 则只是一个随机选取的点，在大多数情况下 (20) 式的值都是在 $(0, 1)$ 之间。

将 (18) 式与 (20) 式相加即可得到点 x_t^i 的总损失值:

$$loss(x_t^i) = loss_1(x_t^i) + loss_2(x_t^i) \tag{21}$$

定义 loss 函数之后，我们的算法框架就也已经搭建完毕：只需在每轮迭代中选择推荐点中 loss 函数值最小的点作为下一个评估点即可。具体的算法流程见 **Algorithm4**

Algorithm 4: Vote algorithm

Input: number of max iteration m

for $t = 1, 2, ..., m$ **do**

  Nominate points from each acquisition function: $\mathbf{x}_t^i = argmax_{\mathbf{x}} u_i(\mathbf{x} \mid D_{1:t-1})$;

  Calculate loss function for every nominate points Select nominee $\mathbf{x}_t = \mathbf{x}_t^j, \quad j = \arg\min_i loss(\mathbf{x}_t^i)$;

  Sample the objective function $y_t = f(\mathbf{x}_t) + \epsilon_t$;

  Augment the data $D_{1:t} = \{D_{1:t-1}, (\mathbf{x}_t, y_t)\}$ and updated GP;

end

6. 数值实验

本文实验中的贝叶斯优化代码框架是基于 bayesian-optimization 包改进而成。具体来说，我们对其算法框架中的采集函数内部优化进行了改进。原算法框架在寻找采集函数最优值时会先在搜索空间中随机地生成 10000 个点，计算各个点的采集函数值，选取最大的值暂时作为当前的采集函数最大值。然后再随机选取十个点作为起始点进行梯度下降算法 (一般为 L-BFGS-B 算法) 寻找最优值。若梯度下降算法寻找到的最优值大于暂定的最大值则替换之，否则就输出暂定值作为采集函数最大值。在我们的改进中，我们首先采用索波尔序列 (Sobol sequence)[12]在搜索空间中生成 10000 个点，该序列能够使得采样点较为均匀地分布在搜索空间中以便更好的探索搜索空间获取更多信息。同样地，我们会算出各个点的采集函数值，然后选取最大的十个值为起点进行梯度下降算法，而非随机选取十个点。我们相信这样的改进能够更好地优化采集函数, 改进前后的效果对比图见**图 1**。

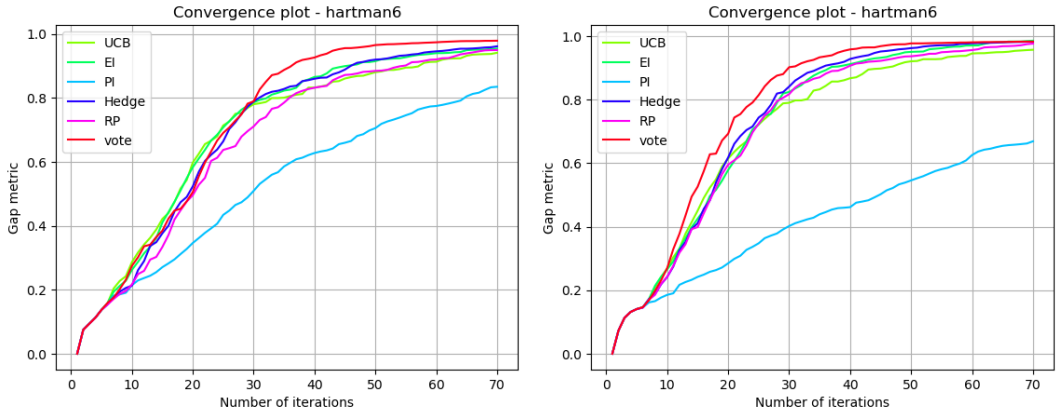
在所有的集成算法中，采集函数的选取以及参数设置方面，为了与 GP-Hedge 算法形成对比，我们沿用了该算法的采集函数架构，PI,EI,UCB 三种采集函数各自取三个不同的参数值形成九个不同的采集函数 (对于 PI 和 EI，$\xi = 0.01, 0.1, 1$; 对于 UCB，$\beta = 1.96, 2.58, 3.10$)[13]。优化的每一轮迭代都将从这 9 个采集函数中选取一个使用。特别地，针对 Hedge 算法中超参数 η 的设置，我们将根据目标函数的取值范围进行调整使得其计算过程中指数大小不至于过大而导致计算中断。

在测试函数的选取方面，我们使用了 GP-Hedge 原文中的三个测试函数，并在此基础上增加了其他 6 个常见的智能优化领域复杂测试函数作为黑箱函数进行优化测试。这些函数的表达式及搜索空间取值范围详见**附录**。

我们每个测试函数都会使用七种采集函数进行优化，分别为基础采集函数中的 PI($\xi = 0$)，EI($\xi = 0$)，UCB($\beta = 2.58$)，集成算法中的 GP-Hedge,vote,improved GP-Hedge，以及用于直接进行集成策略效果对比的纯随机的集成算法 Random Pick(每轮迭代从九种采集函数中随机选取一种使用)。

GP-Hedge 原文中使用了 Gap-metric[14]来计算表示优化算法所给出的结果与目标函数的真实全局最优值之间的接近程度。以 Gap-metric 作为纵坐标在各个算法优化

性能较为接近时其图线容易重合导致难以区分，因此在我们的实验结果中将采取常用的对数遗憾作为图像的纵坐标。具体来说就是计算真实全局最优值与优化算法所给出的值的差的对数值。我们取 10 作为对数的底。



a) 采集函数优化改进前　　**b) 采集函数优化改进后**

图 1 该图简单展示了采集函数内部优化方法改进前后各采集函数的表现对比情况。其中纵坐标代表算法给出的最优值与全局最优值的接近程度。我们可以看出，除 PI 外，改进后的各采集函数在相同轮次下相较于改进前优化效果有了明显提升。(例如改进前在第 30 轮迭代时除 PI 外各算法收敛度都在 0.8 以下，而改进后均超过了 0.8)

表 1,2,3 和**图 2** 中所有测试函数的实验数据都是在取不同随机种子重复 30 次实验取均值而成。其中随机种子是任意截取的一段整数区间 (161-190)。对于各个采集函数算法，每次实验的初始采样点完全相同 (初始采样点个数统一设置为 5 个)，每轮迭代不同采集函数进行内部优化时随机生成的索波尔序列完全相同 (不同轮次生成的序列不同)，这意味着每组实验各个采集函数算法都处在一个绝对公平的竞争环境中。我们的源代码已全部上传至 GitHub[4]，能够保证各组实验数据均能完全复现。

表 1 GP-Hedge 原文测试函数

function/method	branin	hartman3	hartman6
true maximum	-0.397887	3.86278	3.32190
PI	-1.045 ± 1.300	3.714 ± 0.1282	2.304 ± 0.7158
EI	-0.3981 ± 0.0004778	3.853 ± 0.01623	3.127 ± 0.2129
UCB	-0.4083 ± 0.01307	3.850 ± 0.01730	3.099 ± 0.2007
$RandomPick$	-0.3986 ± 0.001222	3.844 ± 0.02750	3.052 ± 0.4059
$GPHedge$	-0.3988 ± 0.001386	3.848 ± 0.02756	3.154 ± 0.2196
$vote$	-0.4008 ± 0.002943	3.853 ± 0.007716	3.155 ± 0.2334
$ImprovedGPHedge$	-0.3989 ± 0.001496	3.826 ± 0.1380	3.162 ± 0.2645

[4]https://github.com/Silver439/PortfolioBO

表 2 复杂测试函数

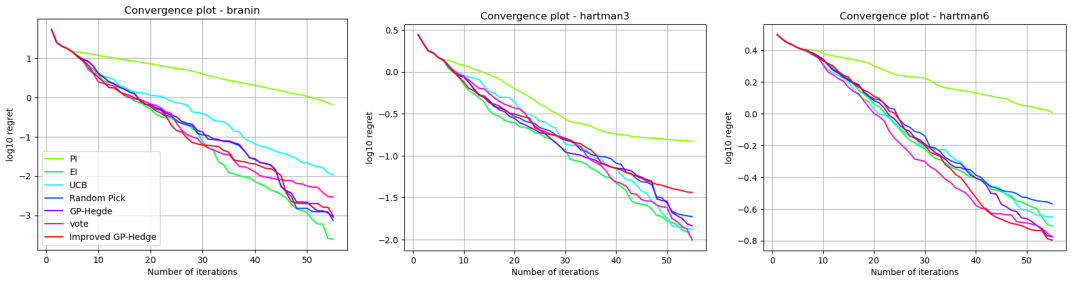
function/method	beale	rosenbrock	griewank
true maximum	0	0	-0.397887
PI	-3.812 ± 4.617	-13.87 ± 16.07	-32.80 ± 19.54
EI	-0.5367 ± 0.4812	-2.601 ± 1.875	-0.9647 ± 0.1693
UCB	-0.5854 ± 0.4200	-12.701 ± 8.080	-0.5630 ± 0.2114
$RandomPick$	-0.8884 ± 0.6351	-3.843 ± 2.615	-0.7265 ± 0.2089
$GPHedge$	-1.123 ± 1.313	-2.118 ± 1.981	-1.699 ± 3.884
$vote$	-0.4956 ± 0.6129	-2.536 ± 1.350	-0.6504 ± 0.1893
$ImprovedGPHedge$	-0.5790 ± 0.6529	-1.778 ± 1.519	-0.6429 ± 0.2814

表 3 复杂测试函数

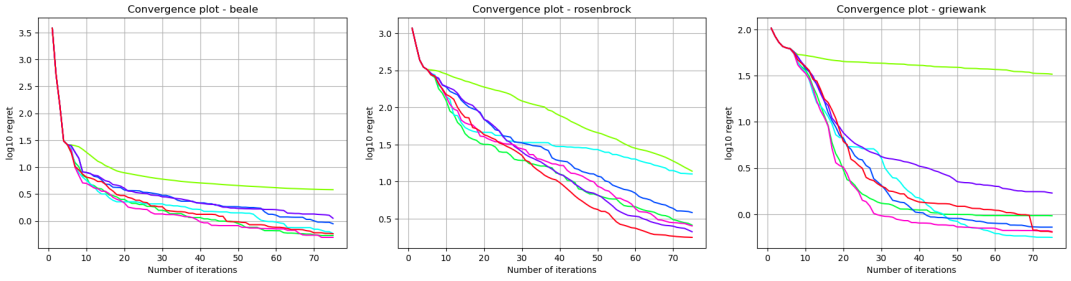
function/method	levy5	Ackley	levy10
true maximum	0	0	0
PI	-5.293 ± 2.537	-18.25 ± 3.802	-22.06 ± 12.41
EI	-1.981 ± 2.052	-12.15 ± 6.761	-14.35 ± 7.122
UCB	-3.274 ± 2.281	-9.760 ± 5.720	-12.08 ± 6.350
$RandomPick$	-2.137 ± 1.763	-11.55 ± 6.720	-13.74 ± 8.161
$GPHedge$	-2.812 ± 2.337	-12.05 ± 5.900	-13.83 ± 7.083
$vote$	-1.483 ± 1.682	-9.288 ± 6.638	-10.69 ± 7.250
$ImprovedGPHedge$	-1.731 ± 1.762	-8.372 ± 6.877	-11.28 ± 8.143

由于**表 1** 中的三个测试函数优化难度较低，因而我们的测试数据都是在设置最大迭代次数为 50 的条件下得到的。对于其他 6 个较为复杂的测试函数，我们统一将最大迭代次数设置为 70 次。从**表 1** 中可以看到，对比 GP-Hedge 原文的三个测试函数，前两个函数由于优化难度较低，除 PI 外其余采集函数算法的优化结果并无太大区别。对于稍复杂的 hartman6 函数，GP-Hedge 算法确实与原文所展现的一样优于三种基础采集函数以及随机算法，我们文章中所提出的两种算法在优化终值上与 GP-Hedge 区别不大，但通过观察 hartman6 的结果图像可以看出，在优化的中期 vote 算法以及改进后的 GP-Hedge 算法要明显优于原 GP-Hedge 算法。因此在进一步减小计算资源的情况下我们应该使用本文所提出的两种算法。

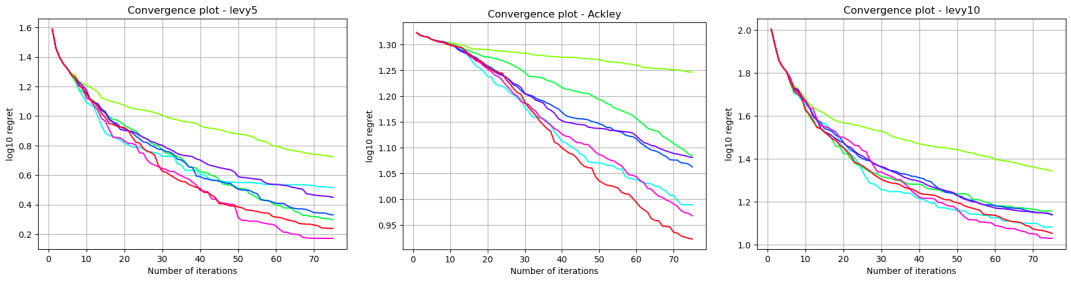
对于其他的较为复杂的测试函数，我们发现在一些测试函数上 (如 beale,griewank,levy5) 原 GP-Hedge 算法要明显弱于随机选择的集成算法。这种现象在 ESP 算法的原文中对 GP-Hedge 的复现实验中也有出现。这说明了我们在 4.1 节中分析的 GP-Hedge 的问题的确影响了其算法稳定性，导致在一些函数的测试实验中 GP-Hedge 的选择策略甚至不如随机挑选。本文提出的两种算法在这 6 种复杂测试函数的优化实验中互有



a) brain(d=2) **b) hartman3(d=3)** **c) hartman6(d=6)**



d) beale(d=2) **e) rosenbrock(d=4)** **f) griewank(d=4)**



g) levy5(d=5) **h) Ackley(d=8)** **i) levy10(d=10)**

图 2 九组测试函数实验结果图像

优劣，vote 算法在 beale,levy5 和 levy10 的优化中占有优势，而 Improved GP-Hedge 则在 rosenbrock 和 Ackley 的优化中明显处于上风。可以看到 6 组实验中这两个集成算法的优化效果都要明显优于完全随机的集成算法，这证明了我们的优化策略的确起到了明显的正向作用。

对于三种基础采集函数，由于最原始的 PI($\xi = 0$) 完全注重于开发而忽略探索，因此在各组实验中 PI 的表现都表现不佳。而 EI 和 UCB 两个采集函数算法则显示出了明显的不稳定性：UCB 在 levy10,Ackley 以及 griewank 函数的优化中表现优异，能够接近甚至超越本文所提出的两种集成算法，但却在 levy5 和 rosenbrock 的优化中表现极为糟糕。EI 则是在 levy5 和 rosenbrock 的优化中表现较好，但在其余的除 beale 外的函数优化中表现不尽人意。这恰好印证了我们在引言中所提出的问题，即每种采集函数在面对不同的优化问题时会有不同的优化效果。与之形成对比的是，我们可以

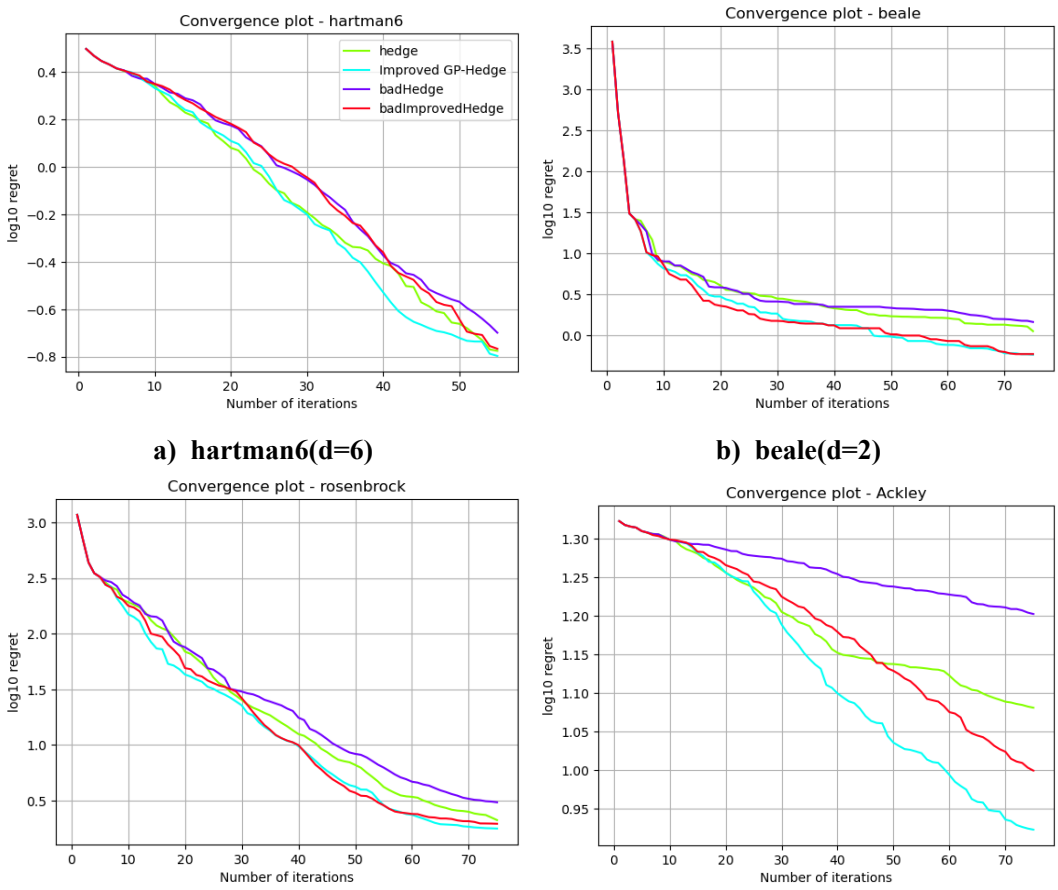
看到本文所提出的两种采集函数集成算法在所有的测试函数上都给出了相对较好的优化结果，展现出了较强的稳定性。这也正是集成算法的意义所在：即在我们事先并不知道哪种采集函数能够有着较好优化效果的情况下，可以选择直接使用集成算法来解决问题。

此外，我们还测试了 Improved GP-Hedge 与原 GP-Hedge 算法对于较差采集函数的鲁棒性。在已有的 9 个采集函数基础上，我们添加了六个作为干扰项的采集函数，这些采集函数会从搜索空间中随机地挑选点作为推荐的评估点，显然它们对于寻找目标函数的最优值起不到任何优化作用。我们将含有干扰项的 GP-Hedge 算法命名为 badHedge，将含有干扰项的 Improved GP-Hedge 算法命名为 badImprovedHedge。我们选择 hartman6，beale，rosenbrock 以及 Ackley 函数作为测试函数，得到的实验对比图表见**表 4** 和**图 3**。

表 4 鲁棒性测试

function/method	hartman6	beale	rosenbrock	Ackley
true maximum	3.32190	0	0	0
$GPHedge$	3.154 ± 0.2196	-1.123 ± 1.313	-2.118 ± 1.981	-12.05 ± 5.900
$badHedge$	3.122 ± 0.1946	-1.452 ± 1.505	-3.061 ± 8.161	-15.93 ± 4.300
$ImprovedGPHedge$	3.162 ± 0.2645	-0.5790 ± 0.6529	-1.778 ± 1.519	-8.372 ± 6.877
$badImprovedHedge$	3.151 ± 0.2645	-0.5899 ± 0.5636	-1.964 ± 1.518	-9.983 ± 6.953

从实验数据中可以看出改进后的 Improved GP-Hedge 算法相比于原 GP-Hedge 算法具有更强的鲁棒性。在较低维度的测试函数中 (2 维和 4 维)Improved GP-Hedge 算法在添加干扰采集函数后几乎没有受到任何影响。对于 6 维的 hartman6 函数，从图像上看 Improved GP-Hedge 算法在添加干扰项后在优化前中期明显受到了干扰项的影响，但在最终给出的优化结果与原算法几乎没有太大差距。而对于更高维度的 Ackley 函数，Improved GP-Hedge 算法在添加干扰项后优化结果均值比不加干扰项要小 1.611，与之相比 GP-Hedge 算法则在添加干扰项后小了 3.88，远高于 Improved GP-Hedge 算法。



a) hartman6(d=6) **b) beale(d=2)**

c) rosenbrock(d=4) **d) Ackley(d=8)**

图 3 鲁棒性测试

7. 总结与展望

本文通过总结分析 GP-Hedge 算法，在其基础上加以改进提出了 Improved GP-Hedge 算法，并进一步跳出了 GP-Hedge 的框架，提出了全新的基于投票机制的采集函数集成框架 vote 算法。这两种算法在各个测试函数上都取得了较为优异的优化表现。值得一提的是，vote 算法也有着较为明显的缺陷：若集成算法中只有很少的采集函数对优化起作用，那么 vote 算法可能无法有较好的表现，因其少数服从多数的特性会使得算法选择错误的采集函数所提供的推荐点。这种情况下使用 Improved GP-Hedge 会更加合理，因其对于添加较差的采集函数有着更强的鲁棒性。当然，上述问题是针对极端情况下的判断，对于一般的优化问题，每种采集函数都应当能够起到一定的优化作用。

同时，对于 vote 算法框架，考虑到不同种类的采集函数其取值的长度尺度不同，

该框架更适用于种类相同的采集函数集成框架，例如所有的采集函数均选用有着不同超参数值的 EI. 由于本文需要与 GP-Hedge 算法进行对比，所以将三种采集函数都添加到了算法的集成框架中。Vote 框架的优势在于在确定集成的采集函数后，该算法框架本身不含有任何需要调整的超参数，可以直接使用。从本文的实验中可以看出，虽然存在长度尺度的问题，vote 算法依旧表现出了不错的优化性能。

最后，我们想要说明的是，本文所提出的两个算法最大的优势是在于其可扩展性。例如，我们的算法可以很简单地扩展到高维贝叶斯优化中。贝叶斯优化有很多高维扩展算法，只要该算法使用传统的采集函数 (例如 BOCK 算法[15])，我们的集成算法就能够直接被其使用。而对于 ESP 算法来说，由于其引用了熵的计算，高维条件下该计算将会极为复杂，因而很难扩展到高维的问题中。事实上本文在采集函数集成算法这一问题上还做出了一些其他的尝试，例如将 knowledge gradient(KG)[16]算法扩展到采集函数集成算法上，以及基于 vote 算法框架，利用信赖域算法[17]在整个搜索空间中寻找 loss 值最小的点。然而我们的实验结果显示前者存在着与 ESP 同样的计算量过大的问题，后者虽然计算量没有增加太多，但其优化性能与 vote 算法没有明显差距。因此本文并未介绍这两个想法。当然，我们认为采集函数的集成算法方向还存在不少工作可以进行。例如，当前的集成算法在优化前就已经确定好了各个采集函数内部的超参数值，这些超参数值对于采集函数的优化表现有着极为重要的影响。我们可以跳出集成框架的思维，改为寻找一种策略，利用当前以掌握的信息使得在每轮迭代中算法能够自适应地调整采集函数超参数值。这或许能进一步地增加采集函数的优化性能。

参考文献

[1] SHAHRIARI B, SWERSKY K, WANG Z, et al. Taking the Human Out of the Loop: A Review of Bayesian Optimization[J]. Proceedings of the IEEE, 2016, 104(1): 148-175. DOI: 10.1109/JPROC.2015.2494218.

[2] KUSHNER H J. A New Method of Locating the Maximum Point of an Arbitrary Multipeak Curve in the Presence of Noise[J]. Journal of Basic Engineering, 1964, 86: 97-106.

[3] MOCKUS J, TIESIS V, ZILINSKAS A. The application of Bayesian methods for seeking the extremum[M]. Towards Global Optimisation 2, 1978.

[4] SRINIVAS N, KRAUSE A, KAKADE S M, et al. Information-Theoretic Regret Bounds for Gaussian Process Optimization in the Bandit Setting[J]. IEEE Transactions on Information Theory, 2012, 58(5): 3250-3265. DOI: 10.1109/TIT.2011.2182033.

[5] BROCHU E, HOFFMAN M W, de FREITAS N. Portfolio allocation for Bayesian optimization[J]. arXiv preprint arXiv:1009.5419, 2010.

[6] CESA-BIANCHI N, LUGOSI G. Prediction, Learning, and Games[M]. Cambridge University Press, 2006. DOI: 10.1017/CBO9780511546921.

[7] AUER P, CESA-BIANCHI N, FREUND Y, et al. Gambling in a rigged casino: The adversarial multi-armed bandit problem[C]//Proceedings of IEEE 36th Annual Foundations of Computer Science. 1995: 322-331. DOI: 10.1109/SFCS.1995.492488.

[8] SHAHRIARI B, WANG Z, HOFFMAN M W, et al. An entropy search portfolio for Bayesian optimization[J]. arXiv preprint arXiv:1406.4625, 2014.

[9] NEAL R M. Bayesian Learning for Neural Networks[D]. AAINN02676. CAN: University of Toronto, 1995.

[10] SEEGER M. Gaussian processes for machine learning[J]. International journal of neural systems, 2004, 14(02): 69-106.

[11] 崔佳旭, 杨博. 贝叶斯优化方法和应用综述[J]. 软件学报, 2018, 29(10): 23.

[12] BRATLEY P, FOX B. Implementing sobols quasirandom sequence generator (algorithm 659)[J]. ACM Transactions on Mathematical Software, 2003, 29(1): 49-57.

[13] NANDY A, KUMAR C, MEWADA D, et al. Bayesian Optimization–Multi-Armed Bandit Problem[J]. arXiv preprint arXiv:2012.07885, 2020.

[14] HUANG D, ALLEN T T, NOTZ W I, et al. Global optimization of stochastic black-box systems via sequential kriging meta-models[J]. Journal of global optimization, 2006, 34(3): 441-466.

[15] OH C, GAVVES E, WELLING M. BOCK : Bayesian Optimization with Cylindrical Kernels[C]//International Conference on Machine Learning. 2018.

[16] FRAZIER P I, POWELL W B, DAYANIK S. A Knowledge-Gradient Policy for Sequential Information Collection[J]. SIAM Journal on Control and Optimization, 2008, 47(5): 2410-2439. DOI: 10.1137/070693424.

[17] ERIKSSON D, PEARCE M, GARDNER J, et al. Scalable global optimization via local bayesian optimization[J]. Advances in neural information processing systems, 2019, 32.

附录

优化测试函数

branin:

Input domain: $x_1 \in [-5, 10]$,$x_2 \in [0, 15]$

Global Maximum: $f(x) = -0.397887$ at $x^* = (-\pi, 12.275)$,$(\pi, 2.275)$ and $(9042478, 2.475)$

$$f(x) = -\frac{1}{51.95}[(\hat{x_2} - \frac{5.1\hat{x_1}^2}{4\pi^2} + \frac{5\hat{x_1}}{\pi} - 6) + (10 - \frac{10}{8\pi})cos(\hat{x_1}) - 44.81] \tag{22}$$

where $\hat{x_1} = 15x_1 - 5$,$\hat{x_2} = 15x_2$

hartman3:

Input domain: $x_i \in (0, 1)$ for all $i = 1, 2, 3$

Global Maximum: $f(x) = 3.86278$ at $x^* = (0.114614, 0.555649, 0.852574)$。

$$f(x) = \sum_{i=1}^{4} \alpha_i exp\Big(- \sum_{j=1}^{3} A_{ij}(x_j - P_{ij})^2 \Big) \tag{23}$$

where $\alpha = (1.0, 1.2, 3.0, 3.2)^T$,$A = \begin{pmatrix} 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \end{pmatrix}$,$P = 10^{-4} \begin{pmatrix} 3689 & 1170 & 2673 \\ 4699 & 4387 & 7470 \\ 1091 & 8732 & 5547 \\ 381 & 5743 & 8828 \end{pmatrix}$

hartman6:

Input domain: $x_i \in (0, 1)$ for all $i = 1, ..., 6$

Global Maximum: $f(x) = 3.32237$ at $x^* = (0.2017, 0.1500, 0.4769, 0.2753, 0.31165, 0.6573)$。

$$f(x) = \sum_{i=1}^{4} \alpha_i exp\Big(- \sum_{j=1}^{6} A_{ij}(x_j - P_{ij})^2 \Big) \tag{24}$$

where $\alpha = (1.0, 1.2, 3.0, 3.2)^T$,$A = \begin{pmatrix} 10 & 3 & 17 & 3.50 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 17 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{pmatrix}$,

$P = 10^{-4} \begin{pmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5586 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{pmatrix}$

beale:

Input domain: $x_i \in [-4.5, 4.5]$ for all $i = 1, 2$

Global Maximum: $f(x) = 0$ at $x^* = (3, 0.5)$

$$f(x) = -[(1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2] \quad (25)$$

rosenbrock:

Input domain: $x_i \in [-2.048, 2.048]$ for all $i = 1, 2, 3, 4$

Global Maximum: $f(x) = 0$ at $x^* = (1, 1, 1, 1)$

$$f(x) = -\sum_{i=1}^{3}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2] \quad (26)$$

griewank:

Input domain: $x_i \in [-600, 600]$ for all $i = 1, 2, 3, 4$

Global Maximum: $f(x) = 0$ at $x^* = (0, 0, 0, 0)$

$$f(x) = -\sum_{i=1}^{4}\frac{x_i^2}{4000} + \prod_{i=1}^{4} cos(\frac{x_i}{\sqrt{i}}) - 1 \quad (27)$$

levy5 and levy10:

Input domain: $x_i \in [-10, 10]$ for all $i = 1, ..., d$,in our article $d = 5$ or $d = 10$

Global Maximum: $f(x) = 0$ at $x^* = (1, ..., 1)$

$$f(x) = -sin^2(\pi\omega_1) - \sum_{i=1}^{d-1}(\omega_i - 1)^2[1 + 10sin^2(\pi\omega_i + 1)] - (\omega_d - 1)^2[1 + sin^2(2\pi\omega_d)] \quad (28)$$

where $\omega_i = 1 + \frac{x_i - 1}{4}$ for all $i = 1, ..., 5$

Ackley:

Input domain: $x_i \in [-32.768, 32.768]$ for all $i = 1, ..., 8$

Global Maximum: $f(x) = 0$ at $x^* = (0, ..., 0)$

$$f(x) = 20exp\Big(-0.2\sqrt{\frac{1}{8}\sum_{i=1}^{8} x_i^2}\Big) + exp\Big(\frac{1}{8}\sum_{i=1}^{8} cos(2\pi x_i)\Big) - 20 - exp(1) \quad (29)$$

致谢

感谢南方科技大学统计与数据科学系的老师本科阶段对我的教导与培养，感谢家人朋友一直以来的支持与鼓励。感谢我的学术导师杨丽丽教授对我毕业设计的悉心指导。同时感谢信管系的王松昊老师，在我完成毕业论文的过程中他也给予了我很多帮助。