关于 Super-Level Set Estimation 的一些算法研究

林海涛 201828014628036

摘要：

高水平集估计一直以来是一个很受关注且具有挑战性的问题，其主要目的是通过对样本空间内一些点进行观测，从而估计样本空间内哪些点对应的取值高于某一个阈值，我们也把这些点的集合叫做高水平集（Super Level Set）。一般的算法采用高斯过程对样本背后的值函数进行模拟，通过增加观测点的个数来更新高斯过程的参数，使其对于样本值函数进行更好的拟合。在本次实验中，我们对于三种经典的算法（LSE, TRUVAR, RMILE）进行了代码的复现，并使用一个经典的三角函数进行模拟实验，深入分析三种算法的优劣。另外，我们还探究了在两种限制条件下的高水平集估计问题，包括给定隐性阈值以及考虑距离损失。在这两种条件下我们对于 LSE 算法做了一些优化与改进，并与其他算法进行比较，取得了不错的结果。

1. 背景介绍：

在科学实验中，数据收集，或者说信息收集一直以来都是一个很关键的问题。人们往往关心在一堆样本点中，那些对应的取值高于某一阈值的样本点。这样的样本集合我们一般把它叫做高水平集（Super-Level Set），而相应的，样本值低于某一阈值的点的集合被称作低水平集（Sub-Level Set）。估计高水平集最简单直接的方法就是对所有样本点进行观测，获得观测结果，根据观测结果判断该样本点是否属于高水平集。然而，该方法在实际应用中存在两个很大的问题：一是每一次观测很可能带来很大的花销，对于所有样本点进行观测往往是不切实际的；二是每次观测存在着额外的误差（噪声），这些误差会影响算法对于一个点是否属于高水平集的判断。因此在实际研究中，人们会利用高斯过程（Gaussian Process）去对样本值进行模拟估计，从而选择最合适的点进行观测，利用少部分点的观测值来估计整个样本空间中哪些样本点属于高水平集。

在实际应用中，除了想利用尽可能少的观测点来给出高正确率的判别结果外，在某些情况下，问题可能会有如下这样的限制条件：（1）界定是否属于高水平集的阈值不是显式给定的，而是和样本可能取的最大值的比例相关，称作隐性阈值（Implicit Threshold）。比如我们想知道某一个区域内环境污染比较严重的地方，但我们并不知道 PM2.5 值达到多少

对于该区域算严重，因此我们只能给出一个相对的界限，例如我们把超过该区域内最高PM2.5 值的 80%的值定做污染严重的界限。在给定隐性阈值的条件下，如何正确地估计高水平集是我们非常关心的一个问题。（2）在观测的过程中需要考虑相邻两次观测所带来的距离损失。我们考虑一个测量湖水水质的问题，如果我们对湖划分成不同区域，每个区域选取一个代表点，这些点就构成了样本点集。在选取集合中某一个点进行测量时，仪器需要从上一个测量点移动到下一个测量点，这中间的运输成本随着距离的增加而逐渐增大。如何合理地将这一因素添加到算法中是另一个很关键的问题。

本文主要针对现有的几种估计高水平集的方法进行代码复现，通过实验模拟进行数值分析，对比，分析不同算法的优劣。另外，针对上述的两种条件限制对已有算法进行改进，比较改进前后算法性能上的差异。章节的结构划分如下：第一章介绍问题的背景和大体思路；第二章介绍近些年的相关工作；第三章介绍实验所使用的三种算法以及对于其中某些算法的改进；第四章介绍实验及结果；第五章是对全文的总结和未来工作的展望。

2. 相关工作：

最早的研究如何估计高水平集的工作是由[1]提出的。[1]给出了估计高水平集问题的具体定义形式，并首次利用高斯过程对该问题进行建模，认为选择一个恰当的点进行观测对于模型的效果有着很重要的影响。作者分析了包括错误率，信息熵，方差，信息增益等在内的多种选点算法，最后提出一种启发式的方法（Straddle），有效地对高水平集进行准确的估计。[2]对于上述的研究进行了进一步的探索，提出了 LSE (Level Set Estimation)算法，是 Straddle 方法的一般表示形式，并给出了其收敛性的证明和测量次数上界的估计。另外[2]还考虑了隐性阈值和批测量的设定，并说明 LSE 算法在这两种设定下都能取得较好的结果。[3]针对另外一个问题进行研究，考虑在一个区域内平均取值超过某一阈值的情况，被称作“主动区域搜索”。该算法使用了一步向前的贪心算法很直接地最大化期望的主动区域个数，取得了不错的结果。[4]同样借助了一步向前的思想，提出了 TRUVAR(Truncated Variance Reduction)算法，通过最小化后验方差来选择下一个目标点。该工作的主要贡献在于将贝叶斯最优化问题（Bayes Optimization）和高水平集估计（Level Set Estimation）两个问题统一起来，在同一个算法框架下进行实现。[5]继续对高水平集估计这一问题进行优化，参考了[3]的选点方法，提出了 RMILE(Robust Maximum Improvement for Level-set Estimation)算法，通过最大化高水平集的元素个数期望值来选择下一个目标点。实验证明

该算法在多种设定下均优于传统的 LSE 和 Straddle 的方法。

3. 算法设计：

3.1 问题定义：

对于一个给定的样本点集D，集合中每一个点 x 都对应一个取值$f(x)$。函数 f 是隐含的，无法直接获得的。定义高水平集$H = \{x|x \in D, P(f(x) > t) > \delta\}$，其中 t 是给定的阈值。由于在实际应用中对$f(x)$的每次观测都存在误差，因此这里给出的是一个概率的形式，即考虑当 f(x)的取值以大概率高于 t 的时候，认定 x 属于高水平集，δ也被称作置信度。我们的目的就是用尽可能少的观测次数对 H 进行精确的估计。

3.2 函数估计：

在建模这样一个问题的时候，一种常用的办法是对函数 f 进行一个先验的估计，然后根据观测值调整对 f 的估计。在第 2 章提到的所有的研究都使用高斯过程[6]进行建模。高斯过程由于自身天然存在着熵值最大的分布特性，很好地符合一个先验分布所需的条件。对于给定高斯过程的先验均值 $\mu_0(x)$ 以及核函数（方差） $k_0(x, x')$，通过已观测样本 $\{(x_i, y_i)\}_{i=1}^{n}$ 可以计算出后验高斯过程的均值和方差如公式（3.1），（3.2）所示：

$$\mu_n(x) = \mu_0(x) + k_n(x)^T (K_n + \sigma^2 I)^{-1} \left(y_{1:n} - \mu_0(x_{1:n})\right) \tag{3.1}$$

$$k_n(x, x') = k_0(x, x') - k_n(x)^T (K_n + \sigma^2 I)^{-1} k_n(x') \tag{3.2}$$

其中 $k_n(x) = \left(k_0(x, x_1), \ldots, k_0(x, x_n)\right)^T$，$\sigma^2$代表观测噪声对应的方差，这里我们假设观测噪声$\epsilon_i \sim N(0, \sigma^2)$。通过不断进行观测，我们就能对某一个样本点的均值和方差进行精确地估计，从而判断其是否属于高水平集。

3.3 算法基本框架：

LSE, TRUVAR, RMILE 这三种算法都是基于同一个核心框架，其算法伪代码如算法 1 所示。

算法核心由四个函数组成，其中 update 函数用来更新后验高斯过程的分布，具体更新方法如公式（3.1）（3.2）所示，这里不再赘述。observe 函数模拟了一次测量的过程，给 f(x)添加了一个噪声误差，在所有算法中均相同。关键的差别在于 estimate 函数和 select point 函数，分别用来估计高水平集和选择下一个观测点。在接下来的两个小节中我会分析

不同算法在这两个函数上的异同。

Algorithm 1 Estimate Super-level Set

Input: data points D, GP prior(μ_0, k_0), threshold t, start point (x, y), max step m_step.
Output: Output super-level set H, sub-level set L.

$U \leftarrow D, H \leftarrow \Phi, L \leftarrow \Phi$
$query\ set \leftarrow \{x, y\}$
$step \leftarrow 0$
while U is not ϕ and $step < m\_step$ **do**
    $GP \leftarrow$ UPDATE$(GP, query\ set)$
    $U, H, L \leftarrow$ ESTIMATE(GP, U, t)
    $x \leftarrow$ SELECTPOINT(GP, U)
    $y \leftarrow$ OBSERVE(x)
    Add (x, y) to $query\ set$
    $step \leftarrow step + 1$
end while

算法 1：估计高水平集的核心框架

3.4 高水平集估计：

当我们有了高斯过程的后验分布参数之后，我们就可以对于每一个样本点 x 计算其对应高斯分布的期望 $\mu(x)$和方差 $\sigma^2(x)$。不同算法的不同估计方法实际上就是设定了不同的判别条件。下面给出算法判别条件的具体定义：

- LSE：算法对每个未分类的样本 x 给出一个置信区间 C，代表算法目前认为 x 可能取值的范围，每一次更新高斯过程的参数后，给出 x 的新置信区间$\mathrm{Q} = [\mu(\mathrm{x}) - \beta^{1/2}\sigma(\mathrm{x}), \mu(\mathrm{x}) + \beta^{1/2}\sigma(\mathrm{x})]$。这里 β 代表置信度，在实验中一般取 1.96。最后我们取两个置信区间的交集作为最后的判断条件，赋值给 C。如果$\min(\mathrm{C}(\mathrm{x})) > t$，则判定$\mathrm{x} \in \mathrm{H}$；如果$\max(\mathrm{C}(\mathrm{x})) < t$，则判定 $\mathrm{x} \in \mathrm{L}$；否则不做判定，继续下一次观测。
- TRUVAR：大体思路与 LSE 相似，不同的是这里直接使用每次预测的结果 Q 作为最终的置信区间。
- RMILE：表面上来看该算法使用的判定方法与前两者稍有不同，其公式如下：$\mathrm{H} = \{\mathrm{x} \in \mathrm{D} | \mathrm{P}_{\mathrm{GP}}(\mathrm{f}(\mathrm{x}) > \mathrm{t}) > \delta\}$，即以某一概率大于阈值 t 的点被看作属于高水平集。但是利用高斯分布的性质可得：

$$\mathrm{P}_{\mathrm{GP}}(\mathrm{f}(\mathrm{x}) > \mathrm{t}) > \delta \Leftrightarrow \mu(\mathrm{x}) - \beta \cdot \sigma(x) > t$$

其中 δ 和 β 是一一对应的，特别的，$\delta = 0.5$对应 $\beta = 0$。因此如果我们将δ 和 β 设

定成对应的值的时候，这两种判定方法的结果是相同的。

3.5 观测点选取：

从上一小节可以看出，不同算法对应的高水平集估计方法都大同小异，那么影响算法效率的最主要的一个因素就是如何选取下一个待观测的样本点了。下一个样本点需要尽可能地给模型带来更多的信息，使得对应的高斯过程能够更准确地拟合数据背后隐藏的函数。因此，算法一般采用贪心的策略来达到这一目的，具体的实现方法如下：

- LSE：算法的目标是最大化公式（3.3）

$$\mathrm{a}(x) = \min\{\max(C(x)) - t, t - \min(C(x))\} \tag{3.3}$$

a(x) 的定义实际上代表着 x 的置信区间的边界和阈值 t 的最小距离。[2]中把 a(x) 叫做模糊度，代表目前模型判断该样本点是否属于高水平集的困难程度。这样一种最大化的优化目标实际上是想探索样本点集中那些不确定性较高的地方，同时又兼顾到了样本点取值是否在阈值附近这一特征，相比于单纯最大化方差来讲对问题有更强的针对性，且速度很快，算法复杂度较低，一直以来作为其他算法实验对比时的 baseline 之一。

- TRUVAR：算法的优化目标如（3.4）所示：

$$\mathrm{x_t} = argmax_{x\in D} \frac{\sum_{x'\in U} \max\{\beta_i \sigma^2_{t-1}(x'), \eta^2\} - \sum_{x'\in U} \max\left\{\beta_i \sigma^2_{(t-1|x)}(x'), \eta^2\right\}}{c(x)} \tag{3.4}$$

式子中分子的第一项代表先验的方差，后一项代表观测完 x 后的方差，两者相减取最大值表示模型希望能够通过这一次的观测减小未分类样本点的方差之和。这里 η^2 代表一个方差的下界，即如果一个点的方差已经足够小了，那么再减小它也没有太大的实际意义了。分母 c(x)表示观测 x 所需要的花费，使得模型还能够考虑到不同观测点花销的影响。与此同时，如果$\max\left(\beta^{1/2}\sigma(\mathrm{x})\right) < (1+\delta)\eta$，说明当前的方差都已经很小了，需要降低方差下界，即令 $\eta = \mathrm{r} \times \eta$。该算法相比于 LSE 考虑了选择某一样本点之后的方差变化，使得估计结果更加准确，但由于算法需要对 D 中的每个样本点进行遍历，并计算样本点的方差之和，一步运算的算法复杂度大概在$\mathrm{O(n^3)}$左右，时间开销较大。

- RMILE：算法的优化目标如（3.5）所示：

$$\mathrm{x^+} = argmax_{x\in D} \max\{E|I_{GP^+}| - |I_{GP}|, \gamma \cdot \sigma_{GP}(x)\} \tag{3.5}$$

其中$|I_{GP}|$代表当前被划分到高水平集的样本点个数，$E|I_{GP^+}|$表示观测了点 x 后新的高水平集样本点个数的期望值。直观的来讲，模型的目的是最大化高水平集的元素个数。当检测任何一个点都不会大幅度提高高水平集中样本点数目的时候，算法考虑那些方差较大的点，通过检测它们来对于高斯过程的参数进行修正，也就是$\gamma \cdot \sigma_{GP}(x)$所起的作用。在实际实现的过程中，期望值是由公式（3.6）计算而得，其中函数$\Phi(\cdot)$是标准正态分布的累计分布函数（cumulative distribution function）。

$$\mathrm{E}|\mathrm{I}_{\mathrm{GP}^+}| = \sum_{x \in D} \Phi\left(\frac{\sqrt{\sigma_{GP}^2(x^+) + \sigma_\epsilon^2}}{|k_{GP}(x, x^+)|} \times (\mu_{GP}(x) - \beta \sigma_{GP^+}(x) - t) \right) \tag{3.6}$$

3.6 算法改进：

上述三种算法都对于高水平集的估计问题给出了不错的解决办法，但这里我们更关心的是在两个特殊设定下的估计问题，即隐性阈值和距离损失。

3.6.1 隐性阈值：

隐性阈值这一问题首先在[2]中被提出。在这种情况下，给定的阈值 t 是不确定的，取而代之的是百分比 w，而一个样本点如果属于高水平集，其观测值需高于所有样本点最大观测值的 w 倍。

[2]给出了一个关于隐性阈值的改进算法，如算法 2 所示。算法通过估计最大值的上界和下界来确定最大值的范围，进而对高水平集进行估计。然而算法在进行观测点选取的时候，仅考虑选取置信区间最大的点进行观测，等价于选择方差最大的点进行观测。原论文中表示这样的设定是为了尽可能地对样本空间探索，准确地估计模型的最大值及对应的阈值。在实际实验中我们发现，该目标函数会显著增加算法收敛所需的步数，最后的判别准确率也会有显著下降。

因此，基于这一点，我们尝试对于需选择目标点的函数进行修改，让其兼顾对于样本空间的探索和对于高水平集边界的估计。基于原有 LSE 算法的影响，我们给出以下的算法，如算法 3 所示。算法的目标函数和传统的 LSE 算法类似，但在阈值上选用了我们所估计的两个阈值 $\mathrm{h}^{\mathrm{pes}}$ 和 $\mathrm{h}^{\mathrm{opt}}$ 。通过最大化该目标函数，我们依旧想寻找那些在边界附近却有较大置信区间的点，作为算法下一个目标检测的候选点，兼顾了对样本空间的探索和对目标问题的优化这两个方面。第 4 章的实验说明改进过后的 LSE 算法相比于改进前有一定的提升。与此同时，我们也实现了 TRUVAR 在隐性阈值的方法，用于和前两种方法进行对

比。

Algorithm 1 LSE implicit threshold Estimate

Input: Unclassified points U, possible limit points Z, GP prior(μ_0, k_0), threshold w.

Output: Output Unclassified points U, possible limit points Z, super-level set H, sub-level set L.

for all $x \in U$ **do**
  Get condition range $C(x)$
  $f^{opt} \leftarrow max_{x \in Z} max(C(x)), f^{pes} \leftarrow max_{x \in Z} min(C(x))$
  $h^{opt} \leftarrow w f^{opt}, h^{pes} \leftarrow w f^{pes}$
  if $min(C(x)) \geq h^{opt}$) **then**
    $U \leftarrow U - \{x\}$
    $H \leftarrow H \cup \{x\}$
    if $max(C(x)) > f^{pes}$ **then**
      $Z \leftarrow Z \cup \{x\}$
    end if
  else if $max(C(x)) \leq h^{pes}$) **then**
    $U \leftarrow U - \{x\}$
    $L \leftarrow L \cup \{x\}$
    if $max(C(x)) > f^{pes}$ **then**
      $Z \leftarrow Z \cup \{x\}$
    end if
  else
    $Z \leftarrow Z \cup \{x\}$
  end if
end for
return U, Z, H, L

Algorithm 2 LSE implicit threshold Select Point

Input: Unclassified points U, GP prior(μ_0, k_0).

Output: Output point x

$x \leftarrow argmax_{x \in U}\ max(C(x)) - min(C(x))$
return x

算法 2：LSE 在隐性阈值下的算法伪代码

Algorithm 1 LSE_new implicit threshold Select Point

Input: Unclassified points U, GP prior(μ_0, k_0).

Output: Output point x

Calculate confindece range C, estimated maximum boundary h^{opt}, h^{pes}
$x \leftarrow argmax_{x \in U}\ min(max(C(x)) - h^{pes}, h^{opt} - min(C(x)))$
return x

算法 3：LSE 在隐性阈值下的改进算法伪代码

3.6.2 距离损失：

通常情况下，我们考虑使用最少的步数让算法收敛，得到结果。在这种设定下，每一步的损失开销都是相同的。但在实际应用中，相邻两步之间的开销很可能是与测量点的位置相关的。这里我们给出基于位置的一种开销定义方式：

$$\mathrm{cost}(\mathrm{x},\mathrm{x}') = \sqrt{\sum_{i=1}^{n}(x_i - x_i')^2} \tag{3.7}$$

其中点 x, x'均是 n 维的向量，之间的开销就是 n 维空间上两点之间的欧几里得距离。对于这样的设定，问题的优化变成如何在累计损失最小的前提下让算法给出更好的分类结果。

论文[4]中对于考虑相邻两个观测点之间开销的高水平集估计问题给出了解决方案，如 3.5 小节中提到的那样，将距离损失 c(x) 考虑进优化目标函数当中。文中指出这样的解决方法使得算法比 LSE 在同等分类精度所产生的开销要小很多。然而，作者并没有对 LSE 算法进行距离损失上的额外修正，这样的对比是有失公允的。因此我们对于 LSE 和 RMILE 两种算法同样进行了修正，将距离损失融入算法选点的公式中，并且形式上和 TRUVAR 保持一致，即在目标函数上除以 c(x)。实验的对比结果同样地在第 4 章给出。

4. 实验及结论：

4.1 实验具体设置：

参考论文[5]的实验设定，我们采用三角函数 $\mathrm{f}(\mathrm{x}_1,\mathrm{x}_2) = \sin(10x_1) + \cos(4x_2) - \cos(3x_1x_2)$作为样本的待拟合函数，定义域区间为$\mathrm{x}_1 \in [0,1], x_2 \in [0,2]$。该函数大致的取值分布如图 1 所示。实验设定的阈值为 1，也就是图中绿色线条包裹的区域。我们使用网格法采取了 20*40 共 800 个样本点构成样本点集。在这 800 个样本点中，对应的取值最大值接近于 3，因而我们设定隐性阈值为 1/3，转换成显性阈值后是同样的界限。对于先验高斯过程，我们给定初始均值$\mu_{\mathrm{x}} = 0$，核函数$\mathrm{k}(\mathrm{x},\mathrm{x}') = \sigma_{kernel}^2 \cdot \exp(\frac{\left\|x-x'\right\|_2^2}{2l^2})$，其中$\ln(\sigma_{kernel}) = 1.0, \ln(\mathrm{l}) = -1.5$，设定与[5]中的实验设定类似，目的是给高斯过程一个较为合理的先验参数，便于模型进行后续的学习。另外测量误差这里设定为$\epsilon_{\mathrm{i}} \sim N(0, \exp(-2))$。所有算法的最大步数均设为 100 步，每种算法均运行 10 次，以均值和标准差作为参考的指标，消除初始点等随机因素对于算法效率的影响。

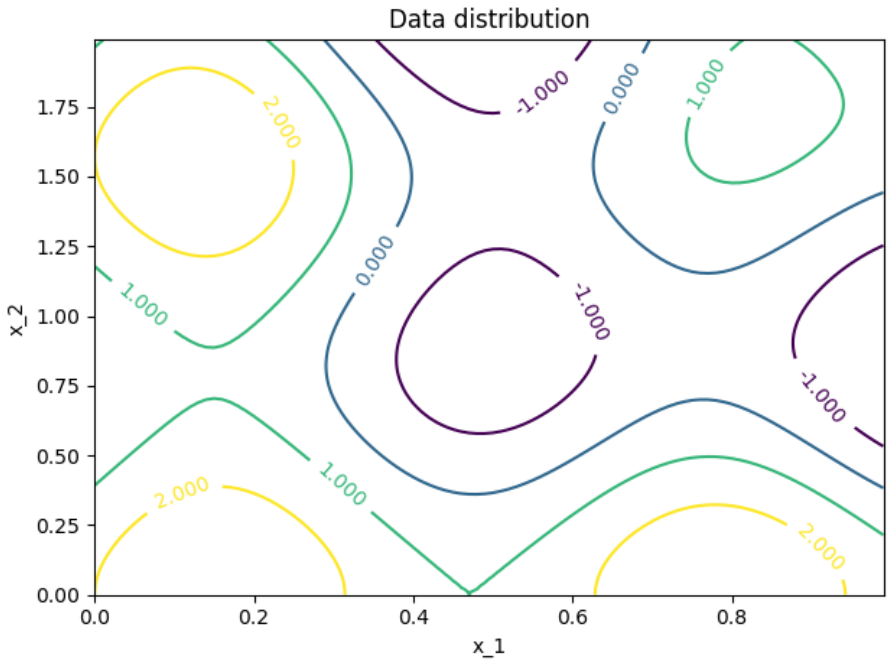


图 1：函数$\sin(10x_1) + \cos(4x_2) - \cos(3x_1x_2)$的数值分布

对于不同算法，其所需参数也略有不同，详细的试验参数设定如表 1 所示。

LSE	$\beta = 1.96$
TRUVAR	$\beta = 1.96, \delta = 0, \eta = 1, r = 0.1$
RMILE	$\eta = 0.01, \delta = 0.975 (\beta = 1.96)$

表 1：不同算法的一些特殊参数设定

所有的代码均采用 Python 编写，源码可参见 github 上的链接（https://github.com/xiaolinAndy/LSE），里面包含三种算法的复现以及对应的改进版本，包括考虑隐性阈值和距离损失两种情况下的选项，希望能为之后的研究提供一定的帮助。

4.2 评价标准：

实验采用的常见的 F1 值作为最终的评价结果，其计算方法如下所示：

$$\text{prec} = \frac{TP}{TP + FP} \tag{4.1}$$

$$\text{recall} = \frac{TP}{TP + FN} \tag{4.2}$$

$$\text{f1} = \frac{2 * prec * recall}{prec + recall} \tag{4.3}$$

其中 TP 代表被正确判断成高水平集的样本点个数，FP 代表被误判成高水平集的样本点的个数，FN 代表属于高水平集但未成功被判断出来的样本点个数。F1 值通过计算准确

率（precision）和召回率（recall）的调和平均值而得。值得注意的是，算法在未收敛的时候对于某些点不能判别其属于的类别，如果这些点也属于高水平集，我们把这些情况也算作 FN。

4.3 实验结果：

4.3.1 标准设定：

原始的 LSE, TRUVAR, RMILE 三种算法在我们实验下的结果如图 2 所示。横坐标代表迭代步数，纵坐标代表当前的 F1 值，值越高表示算法效果越好。蓝色，橘黄色，绿色分别代表 LSE, TRUVAR, RMILE 三种算法。图中每一个点代表对应步数时计算得到 F1 值的均值，竖线代表一个标准差。从图中我们可以看出，三条线在最后基本重合，说明不同算法最终都能较好地对高水平集进行区分和判断。其主要的差别在于前 30 步的效率，很明显 RMILE 在前面有着更高的 F1 值，然后是 TRUVAR，最后是传统的 LSE 算法。实验结果表明采用一步向前的贪心方法确实能在开始的时候提供更好的选点方法，但同样的，其计算复杂度也随之增高。表 2 给出了三种算法运行一次所需的平均时间。我们发现 TRUVAR 大约是 LSE 的 400 多倍，RMILE 更是慢了有 1000 倍以上。从理论分析也可以看出，这两种算法比 LSE 的时间复杂度要高出一个多项式级别（$O(n^2), O(n^3)$，这里 n=800）。因此实际应用中 LSE 算法显然更为实用。

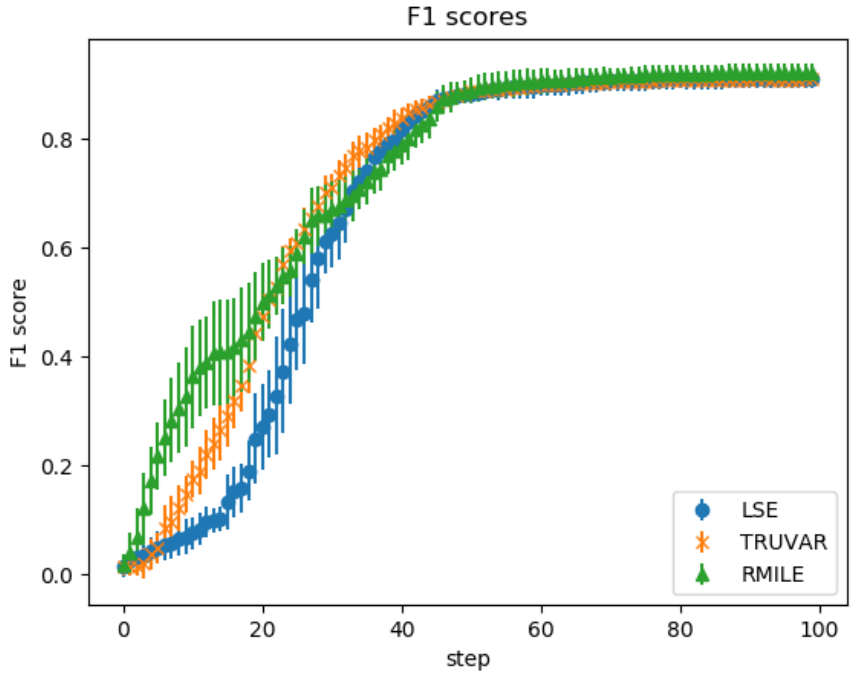


图 2：标准设定下三种算法性能的比较

算法	平均时间 (s)
LSE	1.58
TRUVAR	455.0
RMILE	1307.9

表 2：三种算法运行时间对比

4.3.2 隐性阈值：

首先我们给出 LSE 算法在显性阈值和隐性阈值下的实验结果，如图 3 所示。LSE 代表显性阈值条件，LSE_imp 代表隐性阈值条件。实验结果比较符合我们的直观预期，在显性阈值下无论是模型收敛速度还是最后的分类精度都要优于隐性阈值的情况，多次实验的方差也较小，这也是因为隐性阈值多了上界估计这样一个问题，其难度相对来讲要大很多。

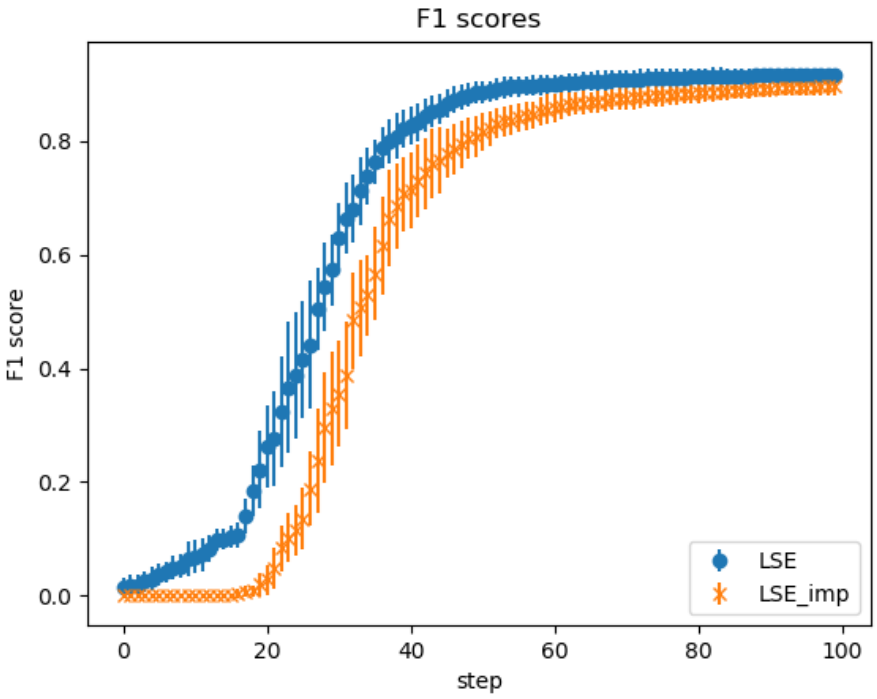


图 3：LSE 在显性阈值和隐性阈值下的实验结果比较

从图中我们还发现，在算法运行前期，隐性阈值算法在前 20 步依然没有给出有用的信息，这说明算法在前期更多地选择探索整个样本空间，效率较低。因此我们对算法做了改进，具体的改进方法如 3.6.1 所述。我们将改进后的算法用 LSE_imp_mod 表示，和 LSE 方法还有 TRUVAR 的隐性阈值算法进行了比较，结果如图 3 所示。橘黄色的是改进后的 LSE 算法，算法虽然在前 20 步也没有获得有用的信息，但在 20 到 40

步给出的结果要略优于原来的 LSE 算法（曲线在蓝色曲线上方），甚至要优于 TRUVAR 算法，但优势并不明显。另外 LSE 及其改进算法在最后收敛时给出的精度要略低于 TRUVAR 算法的，说明其还有一定的改进空间。

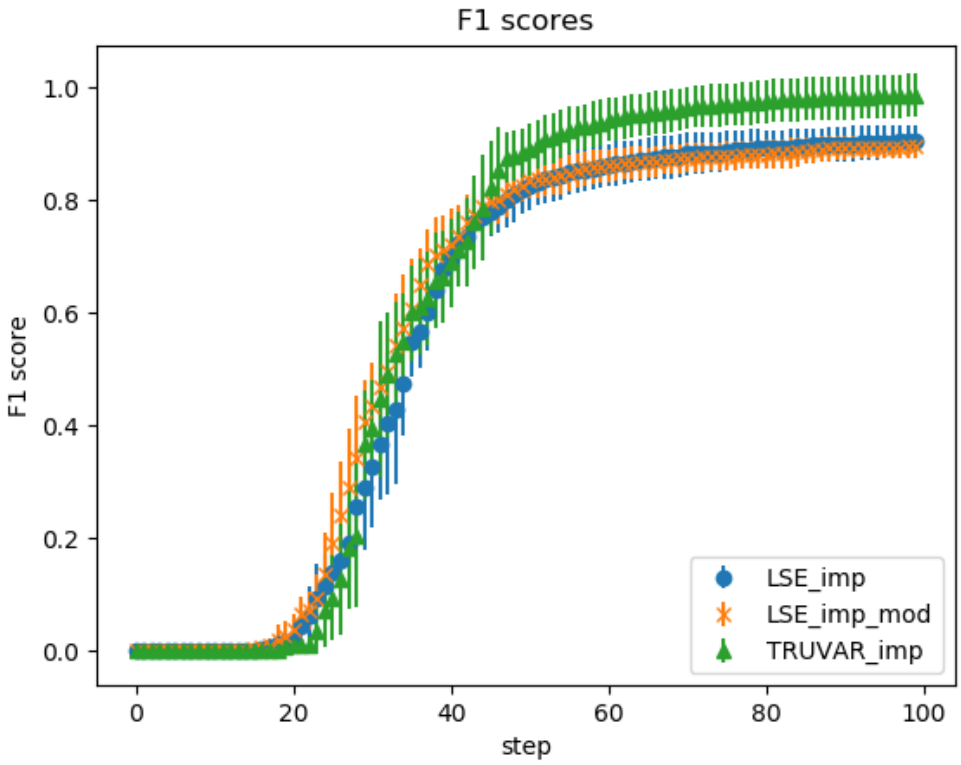


图 3：LSE 及改进版本和 TRUVAR 在隐性阈值条件下的结果

4.3.3 距离损失

在考虑距离的影响下，我们对比距离损失和 F1 值之间的关系，考虑当距离开销相同时，算法给出的 F1 值之间的差距。首先我们给出 LSE 考虑距离损失和不考虑距离损失条件下的结果比较，如图 4 所示，分别用蓝线和橘黄色线来表示。图中没有采用和上面几张图类似的均值和方差来表示，是因为距离损失在算法运行不同次数的时候取值不固定，因此我们只给出了一次运行的结果。图 4 很直观的告诉我们考虑距离损失后算法可以用很小的开销来得到很高的分类精度（cost=15, F1=0.8 与 cost=50, F1 =0.8）。具体的分析在 4.4 中还会有所提及。

另外，由于在论文[4]中，作者没有将 LSE 改进成考虑距离损失的版本，并用原来的算法和考虑距离损失的 TRUVAR 进行比较，我们认为这样是不合理的。因此我们修改了 LSE 算法使之能够考虑距离损失，并重新和 TRUVAR, RMILE 进行比较，结果如图 5 所示。实验结果显示 LSE 能达到和 TRUVAR 与 RMILE 相似的性能，差别不大，这也就此否定了论

文[4]中提到的，TRUVAR 在考虑距离损失后所特有的优势。只要在算法的选点目标函数上除以一个损失函数 c(x)，就可以将距离损失合理地融于算法当中了。

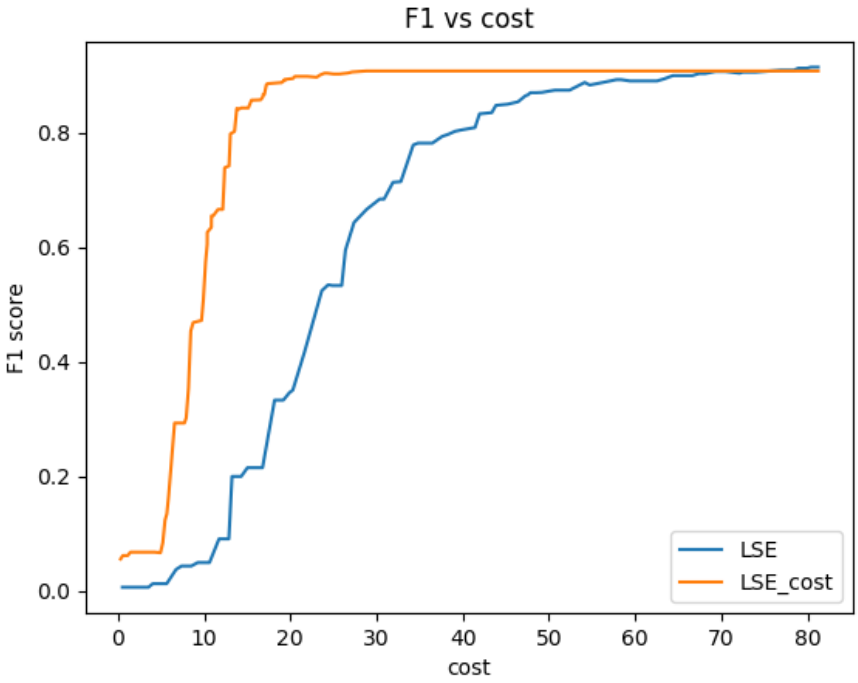


图 4：LSE 在是否考虑距离损失的结果差异

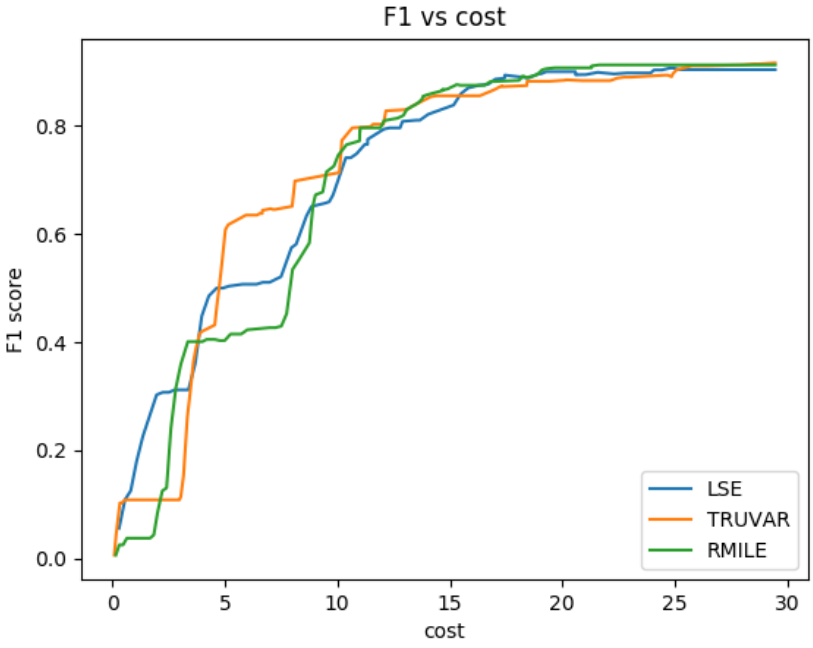
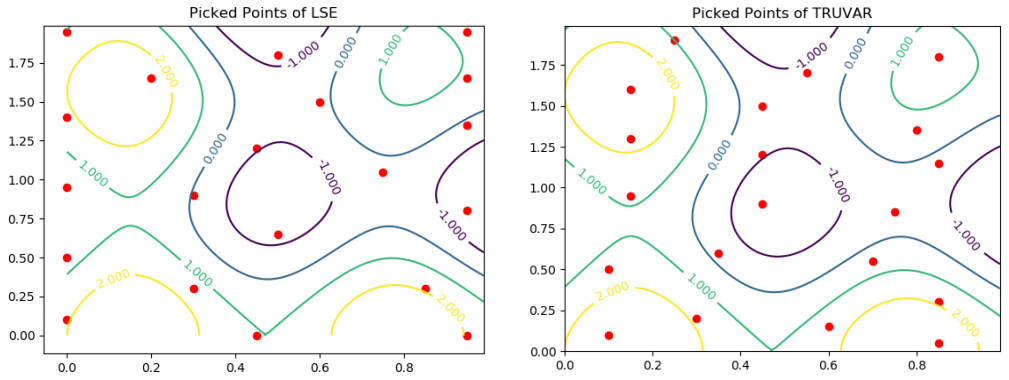


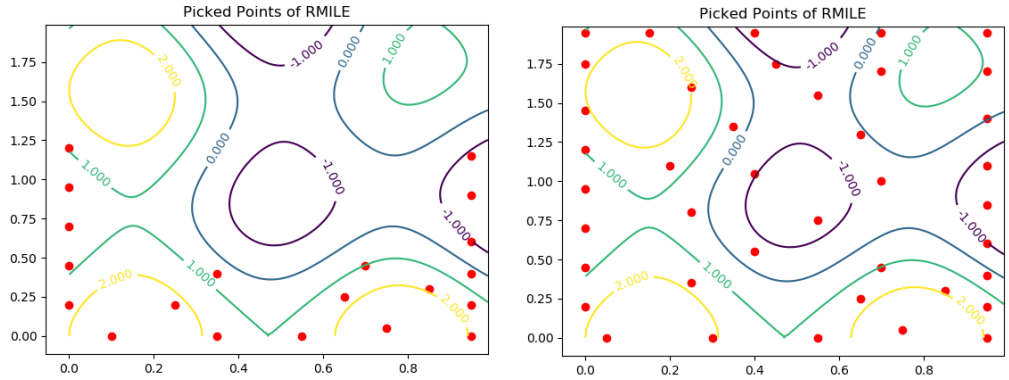
图 5：三种算法在考虑距离损失后的结果比较

4.4 实验分析

这里我们主要对于不同模型在开始的前 20 个点的选择做一些分析，意在通过分析不同算法的选点方法来直观地感受算法上的差异。首先给出三种基本算法的前 20 个点的选择结果，如图 6 所示。直观地来看，LSE 和 TRUVAR 的选点较为类似，LSE 的点更集中在阈值边缘区域（绿色等高线周围），而 TRUVAR 的选点稍微分散一点。RMILE 的选点则比较不同，前 20 个点都集中在下方，且大多在绿色线以内。不过从图(d)我们可以看到，算法已经能够探索到上方区域中高水平集对应的区间了。这些点的分布其实都与算法本身选点的优化目标相关。LSE 考虑那些在阈值边缘附近且具有较大方差的点，因此选点大多在阈值边缘，且较为分散；TRUVAR 的优化目标是最小化后验方差，因此选择的点要尽可能带来更多的分布信息，选点更为分散；RMILE 的优化目标是最大化高水平集元素个数的期望，因此选点大多集中在高水平集范围内。三种算法的实验结果和理论分析基本吻合。



(a) (b)



(c) (d)

图 6：三种算法的前 20 个点选取的分布情况，

(a) LSE, (b) TRUVAR, (c) RMILE, (d)RMILE 40 个点

接下来我们分析考虑距离损失对于选点的影响。我们这里给出 LSE 算法及考虑距离损

失的 LSE 算法的选点路线上的差异，如图 7 所示。我们将算法每次选取的点按先后顺序画出对应的路线图，左边是不考虑距离损失，右边是考虑距离损失的结果。很明显我们可以看出，考虑距离损失的算法的选点策略是不走回头路，路线之间没有交叉，且都围绕着阈值范围附近进行探索，即绿色的等高线（h=1），大大提高了搜索效率。而不考虑距离损失的算法在选择路线上就较为随意，存在着大量的路线相交的情况，搜索效率较低。

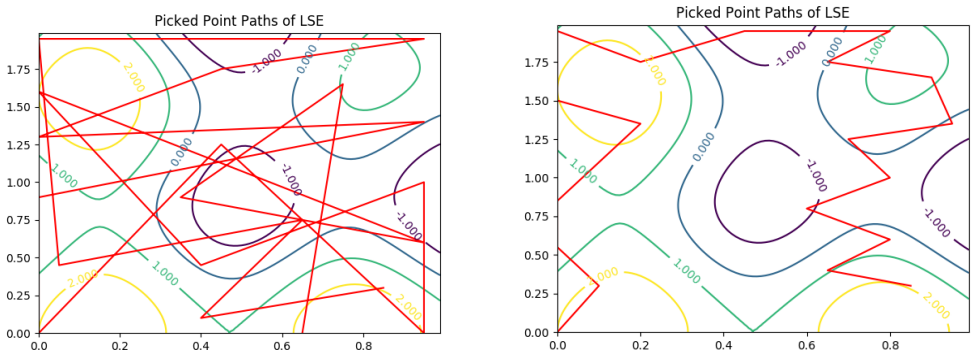


图 7：LSE 在是否考虑距离损失的选点路线差异

（左：不考虑距离损失， 右：考虑距离损失）

5 总结与展望：

本次实验中，我们对于高水平集的估计问题进行了详细的探索和研究。我们独立复现了 LSE, TRUVAR, RMILE 这三种算法，进行了横向对比分析，并对于考虑隐性阈值和距离损失两种特殊情况做了针对性的改进。实验结果表明，三种算法均能对于高水平集给出较好的估计，LSE 的优点在于速度快，TRUVAR, RMILE 的优势在于前期给出的估计较好，但速度较慢。针对 LSE 在隐性阈值情况下的改进能够在一定程度上提高算法在前期的预测精度。与[4]的结论不同，LSE 算法同样可以考虑距离损失并且保证一定的精度，其效果与 TRUVAR 和 RMILE 算法相近。通过本次实验，我们认为在未来的工作研究中，如何有效地提高 TRUVAR, RMILE 这两种基于一步向前贪心机制算法的速度是一个很值得研究的方向。

6 参考文献：

[1] Bryan B, Nichol R C, Genovese C R, et al. Active learning for identifying function threshold boundaries[C]//Advances in neural information processing systems. 2006: 163-170.

[2] Gotovos A, Casati N, Hitz G, et al. Active learning for level set estimation[C]//Twenty-Third International Joint Conference on Artificial Intelligence. 2013.

[3] Ma Y, Garnett R, Schneider J. Active area search via Bayesian quadrature[C]//Artificial

intelligence and statistics. 2014: 595-603.

[4] Bogunovic I, Scarlett J, Krause A, et al. Truncated variance reduction: A unified approach to bayesian optimization and level-set estimation[C]//Advances in neural information processing systems. 2016: 1507-1515.

[5] Zanette A, Zhang J, Kochenderfer M J. Robust Super-Level Set Estimation using Gaussian Processes[C]//Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, Cham, 2018: 276-291.

[6] Rasmussen C E. Gaussian processes in machine learning[C]//Summer School on Machine Learning. Springer, Berlin, Heidelberg, 2003: 63-71.