# 基于 XGBoost 的特征选择算法

李占山[1,2,3]，刘兆赓[2,3]

（1. 吉林大学计算机科学与技术学院，吉林 长春 130012；2. 吉林大学软件学院，吉林 长春 130012；
3. 吉林大学符号计算与知识工程教育部重点实验室，吉林 长春 130012）

**摘 要**：分类问题中的特征选择一直是一个重要而又困难的问题。这类问题中要求特征选择算法不仅能够帮助分类器提高分类准确率，同时还要尽可能地减少冗余特征。因此，为了在分类问题中更好地进行特征选择，提出了一种新型的包裹式特征选择算法 XGBSFS。该算法借鉴极端梯度提升（XGBoost）算法中构建树的思想过程，通过从 3 个重要性度量的角度来衡量特征的重要性，避免单一重要性度量的局限性；然后通过改进的序列浮动前向搜索策略（ISFFS）搜索特征子集，使最终得到的特征子集有较高的质量。在 8 个 UCI 数据集的对比实验中表明，所提算法具有很好的性能。

**关键词**：特征选择；极端梯度提升；序列浮动搜索策略

**中图分类号**：TP18

**文献标识码**：A

**doi:** 10.11959/j.issn.1000−436x.2019154

# Feature selection algorithm based on XGBoost

LI Zhanshan[1,2,3], LIU Zhaogeng[2,3]

1. College of Computer Science and Technology, Jilin University, Changchun 130012, China
2. College of Software, Jilin University, Changchun 130012, China
3. Key Laboratory of Symbolic Computation and Knowledge Engineering, Ministry of Education, Jilin University, Changchun 130012, China

**Abstract:** Feature selection in classification has always been an important but difficult problem. This kind of problem requires that feature selection algorithms can not only help classifiers to improve the classification accuracy, but also reduce the redundant features as much as possible. Therefore, in order to solve feature selection in the classification problems better, a new wrapped feature selection algorithm XGBSFS was proposed. The thought process of building trees in XGBoost was used for reference, and the importance of features from three importance metrics was measured to avoid the limitation of single importance metric. Then the improved sequential floating forward selection (ISFFS) was applied to search the feature subset so that it had high quality. Compared with the experimental results of eight datasets in UCI, the proposed algorithm has good performance.

**Key words:** feature selection, XGBoost, sequential floating forward selection

## 1 引言

特征选择（feature selection）也称属性选择（attribute selection），是从原始特征中选择一些有效特征来降低数据集维度的过程[1]，是机器学习和数据挖掘中关键的预处理步骤。对于特征选择的设计，国内外学者已进行了大量的研究。常见的特征选择方法大致有 3 类：过滤式（filter）、嵌入式

**收稿日期**：2019–01–04；**修回日期**：2019–04–04

**通信作者**：刘兆赓，lzgalex777@foxmail.com

**基金项目**：国家自然科学基金资助项目（No.61672261）；吉林省自然科学基金资助项目（No.2018010143JC）；吉林省发改委产业技术研究与开发专项基金资助项目（No.2019C053-9）

**Foundation Items:** The National Natural Science Foundation of China (No.61672261), The Natural Science Foundation of Jilin Province (No.2018010143JC), Industrial Technology Research and Development Special Project of Jilin Province Development and Reform Commission (No.2019C053-9)

（embedding）和包裹式（wrapper）[2]。过滤式方法在训练学习器之前会对数据集进行筛选，特征选择过程与后续学习器无关；嵌入式方法将特征选择过程与学习器训练过程融为一体，在学习器训练过程中自动地进行特征选择[3]；包裹式方法直接将最终要使用的学习器的性能作为特征子集的评价标准，通常被证明搜寻特征子集的分类准确性优于前二者[4]。

搜索最优特征子集是特征选择过程中最关键、最具有挑战性的一环。对于不同的搜索策略，特征选择方法又可以分为穷举法、启发式法、基于信息理论的方法、基于演化计算的方法等。Almuallim 等[5]基于穷举法提出的 FOCUS 算法是在整个搜索空间中进行搜寻，直到找出一个最小的特征子集将训练数据分成单纯的类。但是 FOCUS 的时间复杂度是 $O(2^n)$，当特征数量非常大时，评价所有特征子集的时间开销几乎是不可接受的[6]，因此，在实际任务中很少使用穷举法。特征选择的启发式方法主要包括爬山法、分支限界法、定向搜索法和最佳优先搜索法[7-8]。相较于穷举法而言，启发式方法更加高效，在解决实际问题的过程中，人们往往将启发式方法集成到包裹式方法中，以便权衡运算效率和特征子集的质量，获得一个好的平衡点，继而得到近似最优解。基于信息理论的方法主要是应用不同的信息策略来过滤特征，其中具有代表性的有基于联合互信息（JMI, joint mutual information）的特征选择方法和基于互信息最大化（MIM, mutual information maximization）的特征选择方法。基于演化计算的方法是近年来应用较广的一类方法，相比于其他方法，这类方法具有更强的全局搜索能力[9]，最近的研究表明基于森林优化的特征选择方法[10]和基于收益成本的萤火虫方法[11]均具有良好的性能。基于演化计算的方法虽然性能较好，但也有不足的地方：首先只有迭代次数足够大时才可能找到比较好的结果，其次如何设置参数也是一个问题。本文基于 XGBoost 提出了一种新型的包裹式特征选择算法 XGBSFS（XGBoost sequential floating selection）。XGBoost 是 Chen 等[12]在前人关于梯度提升算法的大量研究工作基础上提出的一个基于提升树的机器学习系统，它包含一个迭代残差树的集合，每一棵树都在学习前 $N-1$ 棵树的残差，将每棵树预测的新样本输出值相加就是样本最终的预测值[13]。但不同于常用的梯度提升决策树（GBDT, gradient boosting decision tree）在优化时仅用一阶导数信息，XGBoost 对代价函数进行了二阶泰勒展开，同时用到了一阶导数和二阶导数，使 XGBoost 具有良好的结果。所提算法的主要特点如下。

1) XGBoost 具有良好的防过拟合特性。

2) XGBoost 拥有较高的计算效率。

3) XGBoost 的计算过程中有一定的启发性。

目前在数据挖掘和机器学习竞赛中，获胜的队伍大多使用 XGBoost 系统，解决了诸如网络内容分类、广告竞价排名、顾客行为预测等实际问题，这表明该系统适用范围广泛。基于以上分析，本文以 XGBoost 为基本工具，完成如下工作。

1) 基于 XGBoost 构建了用于求解特征选择问题的启发式策略。

2) 提出了基于 2 种重要性度量的序列浮动前向选择策略 ISFFS。

3) 提出了一个新型的包裹式特征选择算法 XGBSFS。

4) 结合理论与实验对 XGBSFS 的有效性做了全面分析与验证。

## 2 XGBoost 概述

XGBoost 是梯度提升（gradient boosting）思想的一种高效的系统实现，其基学习器既可以是线性分类器，也可以是树。本文利用其树模型的特点作为量化每个特征重要性的依据进行特征选择。

### 2.1 树模型

对于给定的数据集，在树模型构建的过程中，每一层贪心地选取一个特征分割点作为叶子节点，使在分割之后整棵树增益值最大，这意味着特征被分割次数越多，该特征给整棵树带来的效益越大，该特征也越重要；相似地，特征每次被分割时的平均增益越大，该特征越重要。在分割过程中，每个叶子节点的权值可以表示为 $w(g_i,h_i)$，$g_i$ 和 $h_i$ 分别为

$$g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}) \tag{1}$$

$$h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}) \tag{2}$$

其中，训练误差 $l(y_i,\hat{y}_i)$ 表示目标值 $y_i$ 和预测值 $\hat{y}_i$ 之间的差距。为了使分割后的树代价最小，根据所有叶子节点的权值，考虑每个特征作为分割点的增益 Gain，有

$$\text{Gain}=\sum_{\text{left}} w+\sum_{\text{right}} w-\sum_{\text{nosplit}} w \tag{3}$$

式(3)说明了对于每一个分割点而言，其增益可表示为分割后的总权值（叶子节点左子树的总权值与右子树的总权值之和）减去分割前的叶子节点的总权值。

### 2.2　树模型的组合

决策树模型作为一种非参数监督式学习模型，常用于分类与回归，该模型不需要对数据有任何的先验假设，就可以快速地根据数据的特征找到决策规则[14]。而 XGBoost 在决策树的基础上采用了集成策略，利用梯度提升算法不断减小前面生成的决策树的损失，并产生新树构成模型，确保了最终决策的可靠性。XGBoost 在每一次迭代的时候都会增加一棵树，则构建 $K$ 棵树的线性组合为

$$\hat{y}_i^{(t)}=\sum_{k=1}^{K} f_k(x_i)=\hat{y}_i^{(t-1)}+f_t(x_i),\ f_k\in F \tag{4}$$

其中，$F$ 表示包含所有树的函数空间，$f_k(x_i)$表示第 $i$ 个样本在第 $k$ 棵树中被分类到所在叶子节点的权重。

### 2.3　重要性度量指标

重要性度量指标是评估每个特征在所属特征集中重要程度的一种衡量方式。XGBoost 根据特征分裂的次数 FScore、特征平均增益值 AverageGain 和特征平均覆盖率 AverageCover 来作为其构建决策树的依据，以便准确地完成分类任务。

对于上述 3 种重要性度量指标，有

$$\text{FScore}=|X| \tag{5}$$

$$\text{AverageGain}=\frac{\sum \text{Gain}_X}{\text{FScore}} \tag{6}$$

$$\text{AverageCover}=\frac{\sum \text{Cover}_X}{\text{FScore}} \tag{7}$$

其中，$X$ 是所求特征分类到叶子节点的集合；Gain 是 $X$ 中每个叶子节点由式(3)得到的在分割时节点增益值；Cover 是 $X$ 中落在每个节点的样本个数。

## 3　XGBSFS 算法

### 3.1　算法描述

通过以上分析，本文提出了一种基于 XGBoost 的包裹式特征选择方法 XGBSFS，利用 XGBoost 算法构建树的过程，分别根据 FScore、AverageGain 和 AverageCover 计算特征重要性度量，然后提出一种改进的序列浮动前向选择策略（ISFFS, improved sequential floating forward selection）进行搜索，最后把分类准确率最高的特征子集作为特征选择结果。XGBSFS 用于特征选择特点如下。

1) 对于缺失值的处理

利用 XGBoost 独特的树模型，使 XGBSFS 能够对数据集中的一些缺失值进行处理。XGBSFS 先将缺失值转换为稀疏矩阵，将缺失值数据分到左子树和右子树分别计算损失，并选择损失较小的子树；如果训练中没有数据缺失，但预测时出现了数据缺失，那么缺失数据被默认分类到右子树[12]。如此便有效降低了树模型对缺失值的敏感度。

2) 对于零重要度特征的处理

在树模型构建中，零重要度的特征，即特征节点分裂次数 FScore=0 的特征不会被用于分割任何节点，移除它们不会影响最终的模型表现，因此 XGBSFS 在得到特征的重要性度量后优先过滤零重要度特征。

3) 对于特征的重要性处理顺序

利用给定结构的树模型集合，对非零重要度特征进行排序；考虑到单向搜索的局限性，因此 XGBSFS 采用不同的重要性度量双向进行搜索，以获得更好的效果。

### 3.2　构建树模型计算重要性度量

本文提出的 XGBSFS 在构建树的过程中，同时对重要性度量进行计算，具体构建步骤如下。

1) 从根节点开始，对每一节点都遍历所有的特征。

2) 对于特征 $a_i\in A$，先按照样本值进行排序 $\{a_i^{(1)},\cdots,a_i^{(n)}\}$，线性扫描 $a_i$ 确定增益最好的分割点 $(a_i^{(k)},\text{gain}_i)$。

3) 从所有选择好的特征分割点中选取增益最高的特征进行分割 $\max\text{Gain}(a_1^{(k_1)},\cdots,a_m^{(k_m)})$，并更新样本到树节点的映射关系。

4) 一直分割到最大深度，并计算构建下一棵树的残差。

5) 将生成的所有树集成，完成最终树模型的构建。

6) 根据式(5)~式(7)计算重要性度量。

在 XGBoost 的相关研究中，许多学者已经验证了该树模型的优势及其高效的原因[12,15-16]。本文在此基础之上，结合所提出的 ISFFS 搜索策略，将 XGBoost 在特征选择问题的求解中进行了更好的扩展应用。

### 3.3 ISFFS 搜索策略

对全部特征根据重要性度量进行排序，可以得到原始的候选子集。传统的搜索策略有 3 种：前向搜索（forward search）、后向搜索（backward search）和双向搜素（bidirectional search）。逐渐增加相关特征的策略称为前向搜索[17]：首先取当前最优的特征作为第一轮的选定集，接下来每一轮轮番添加一个特征，对每次生成的集合判断是否优于上一轮，若是，保留本轮的结果作为新的选定集；若不是，则停止搜索，并将上一轮选定的集合作为特征选择的结果[18]。相反，若从完整的候选集合开始，每轮尝试舍弃一个无关特征，像这样逐渐减少特征的策略称为后向搜索[19]。而双向搜索结合了前向与后向搜索策略，每轮根据评价算法并行地选出最优特征和无关特征：在增加相关最优特征的同时去掉一个无关特征，且新加入的特征在后续搜索过程中不会被淘汰[20]。然而，上述策略容易陷入局部最优[18,21]。本文通过对传统搜索策略的研究，提出了一种改进的序列浮动前向搜索方法 ISFFS。一般的序列浮动前向搜索方法只使用一种重要性度量指标搜索候选集合，而 ISFFS 同时使用了 2 种不同的重要性度量 $i_1$ 和 $i_2$ 进行搜索，从而避免被单一重要性度量所约束，在一定程度上减少了问题的局限性。该策略主要包含 2 个步骤，具体如下。

**步骤 1** 序列前向添加。建立一个起始为空的目标特征集合，每次从原始的候选子集中，依据重要性度量 $i_1$ 从大到小搜索一个最重要的特征添加到目标集合中，使在这个特征加入之后，目标集合的分类准确性提高。

**步骤 2** 浮动后向筛除。从目标集合中根据重要性度量 $i_2$ 从小到大搜索并剔除一个特征，使在去除该特征之后，目标集合的分类准确性提高，一直筛到不存在能使准确性提高的特征，返回步骤 1。

经过若干次迭代后，最终将得到特征数目最少、分类准确性最高的目标特征集合作为特征选择结果。

### 3.4 XGBSFS 伪代码

结合前文描述，本文给出了 XGBSFS 算法，其伪代码如算法 1 所示。

**算法 1** XGBSFS

**输入** 原始特征集合 $A$

**输出** 目标特征集合 $O$

$O \leftarrow \varnothing$; depth ← 0

初始化重要性度量

建立根节点

while depth < 树的最大深度 do

for each $a \in A$ do

计算最佳分割点 $a^k$ 并为最佳分割点建立孩子节点

end for

depth ← depth + 1

end while

根据式(4)将所建树加入树模型中

for each tree do

for each 分割节点 do

计算分割节点产生的增益值 gain 和落在其中的样本个数 cover

更新每一个特征的重要性度量

FScore ← FScore + 1, Gain ← Gain + gain, Cover ← Cover + cover

end for

end for

for each $a \in A$ do

$$\text{AverageGain} \leftarrow \frac{\text{Gain}}{\text{FScore}}$$

$$\text{AverageCover} \leftarrow \frac{\text{Cover}}{\text{FScore}}$$

end for

根据重要性度量分别对特征进行排序

选取 2 个候选特征子集 $I_1$ 和 $I_2$，其中 $I_1$ 是根据重要性度量 $i_1$ 从大到小排列的集合，$I_2$ 是根据重要性度量 $i_2$ 从小到大排列的集合

从集合 $I_1$ 中获取特征 $x_b$，使目标集合 $O$ 加入 $x_b$ 后，准确性评价函数 $J$ 提高

while $x_b$ 存在 do

$O \leftarrow O \cup x_b$

从 $I_2$ 中获取特征 $x_w$，使 $O$ 剔除 $x_w$ 后，$J$ 提高

while $x_w$ 存在 do

$O \leftarrow O - x_w$

重复从 $I_2$ 中获取 $x_w$

end while

重复从 $I_1$ 中获取特征 $x_b$

end while

return $O$

### 3.5 复杂度分析

考虑 $D$ 为树的最大深度，$K$ 为树的个数，给定无缺失值的数据集，其样本量为 $N$，特征数量为 $M$，

非零重要性特征的比例为 $\beta$，令 $m=M\beta$，则对于精确贪婪算法（exact greedy）[12]下的全局特征预排序，有排序复杂度 $O(Mn\log N)$；由于后期节点分裂时都可以复用全局预排序的结果，不需要消耗额外的时间来进行排序。对于树模型构建的过程，每一层遍历分割点的时间复杂度为 $O(MN)$，故建立 $K$ 棵树的时间复杂度为 $O(MNKD)$。对所得非零重要性特征子集进行排序，采用快速排序算法的平均时间复杂度为 $O(m\log m)$。在搜索策略 SFFS 中，序列前向添加过程从头遍历到尾，由于$|S|=m$，故添加过程时间复杂度为 $O(m)$；浮动后向筛除过程从尾遍历到头，由于$|O|\leqslant m$，故筛除过程时间复杂度为 $O(m)$；ISFFS 总的时间复杂度为 $O(m^2)\times$基分类器的时间复杂度。当基分类器为 *K*NN（*k*-nearest neighbor）时，主流的 *K*NN 分类器均是经过 KD tree 或是 ball tree 优化后的，其复杂度可近似表示为 $O(mN\log N)$。所以 ISFFS 过程的时间复杂度可近似表示为 $O(m^3N\log N)$。故 XGBSFS 算法总的时间复杂度可近似表示为

$$O(MN\log N+m\log m+MNKD+m^3N\log N) \tag{8}$$

## 4　实验分析

### 4.1　实验数据与方法

本文从 UCI 机器学习数据库（UCI machine learning repository）[22]中选择了 8 个数据集进行测试，这些数据集的特征数、样本数和分类数如表 1 所示。

**表 1　　UCI 的数据集**

| 数据集 | 特征数/个 | 样本数/个 | 分类数/类 |
|---|---|---|---|
| Wine | 13 | 178 | 3 |
| Vehicle | 18 | 846 | 4 |
| Segmentation | 19 | 2 310 | 7 |
| Ionosphere | 34 | 351 | 2 |
| Sonar | 60 | 208 | 2 |
| LSVT | 310 | 126 | 2 |
| CNAE-9 | 856 | 1 080 | 9 |
| Arcene | 10 000 | 200 | 2 |

为了验证 ISFFS 策略的有效性，分别独立使用 FScore、AverageGain 和 AverageCover 做了序列浮动前向搜索，对比 SFFS 策略的详细说明如表 2 所示。为了便于比较本文所提算法的性能，选择其他具有代表性的特征选择算法进行对比，对比算法的详细信息如表 3 所示。

**表 2　　对比 SFFS 策略的详细说明**

| SFFS 策略 | 特征重要性判别描述 |
|---|---|
| SFFS1 | 仅使用 FScore 进行序列浮动前向搜索 |
| SFFS2 | 仅使用 AverageGain 进行序列浮动前向搜索 |
| SFFS3 | 仅使用 AverageCover 进行序列浮动前向搜索 |
| ISFFS | 应用本文提出的改进的序列浮动前向搜索 |

**表 3　　对比算法的详细信息**

| 算法名称 | 描述/发表年份 |
|---|---|
| Rc-BBFA[1] | 基于收益成本的萤火虫算法的特征选择[11]/2017 |
| FSFOA[2] | 基于森林优化算法的特征选择[10]/2016 |
| JMI[3] | 基于联合互信息的特征选择[23]/2012 |
| MIM[4] | 基于互信息最大化的特征选择[23]/2012 |

注：1. Rc-BBFA (return-cost-based binary feature selection method based on firefly algorithm)。

2. FSFOA (feature selection using forest optimization algorithm)。

3. JMI (joint mutual information)。

4. MIM (mutual information maximization)。

在本文的实验中，使用了 python 3.6 实现算法，同时使用了公开的工具包 XGBoost、scikit-feature 和 scikit-learn。所有实验均在一台配置为 AMD R7 1700 CPU、16 GB 内存、500 GB 硬盘的电脑上完成。

### 4.2　实验结果分析

在实验结果的统计上，本文使用了分类准确率和维度缩减率这 2 个常用于检验特征选择算法性能的指标[10,24]，其中分类准确率的定义如式(9)所示，维度缩减率的定义如式(10)所示。

$$\mathrm{CA}=\frac{\mathrm{NCC}}{\mathrm{NAS}} \tag{9}$$

$$\mathrm{DR}=1-\frac{\mathrm{NSF}}{\mathrm{NAF}} \tag{10}$$

其中，NCC 代表正确的分类数，NAS 代表数据集的总实例，NSF 代表选择的特征数，NAF 代表总的特征数。

在分类器的选择上，本文使用了 KNN 分类器作为基分类器。之所以使用 KNN 是因为其是一个通用简便的分类方法，并且由于 KNN 分类器相较于其他分类器来说，并不需要调整过多的参数，因此较容易进行对比实验来验证特征选择算法的性能。目前提出的很多包裹式特征选择算法，仅使用了 *K*NN 作为唯一的基分类器[11,25-26]，它们之间唯一区别在于参数 $K$ 的取值。本文中，为了与 Rc-BBFA 对比方便，在参数 $K$ 的选取上和 Rc-BBFA

保持了一致，将 $K$ 设置为 1。表 4 和表 5 是验证 ISFFS 策略有效性的实验结果，表 6~表 13 给出了 XGBSFS 与其他特征选择算法的对比实验结果，其中，加粗字体表示对应的最优值。

**表 4　改进前后的 SFFS 策略对分类准确率的影响结果**

| 数据集 | SFFS1 | SFFS2 | SFFS3 | ISFFS |
|---|---|---|---|---|
| Wine | 91.67 | 88.34 | 84.26 | **97.97** |
| Vehicle | 71.48 | 69.17 | 70.12 | **75.95** |
| Segmentation | 95.09 | 96.64 | 95.17 | **96.67** |
| Ionosphere | 94.15 | 94.15 | 93.59 | **96.42** |
| Sonar | 84.76 | 89.21 | 90.32 | **95.71** |
| LSVT | 86.05 | 87.89 | 88.42 | **91.32** |
| CNAE-9 | 89.54 | 91.57 | 87.78 | **91.88** |
| Arcene | 95.67 | 95.33 | 94.00 | **97.83** |

**表 5　改进前后的 SFFS 策略对维度缩减率的影响结果**

| 数据集 | SFFS1 | SFFS2 | SFFS3 | ISFFS |
|---|---|---|---|---|
| Wine | 63.85 | **71.54** | 59.23 | 60.77 |
| Vehicle | 59.45 | 61.11 | 61.11 | **63.89** |
| Segmentation | 52.63 | 66.31 | 62.63 | **67.37** |
| Ionosphere | **85.00** | 83.82 | 81.47 | 81.77 |
| Sonar | **87.67** | 87.00 | 85.83 | 82.17 |
| LSVT | 98.58 | 98.23 | 98.10 | **98.61** |
| CNAE-9 | 94.19 | **94.42** | 92.22 | 92.73 |
| Arcene | 99.87 | **99.89** | **99.89** | 99.86 |

**表 6　XGBSFS 及其对比算法在 Wine 上的结果**

| Wine | CA | DR | 验证方法 |
|---|---|---|---|
| XGBSFS | 97.97% | 60.77% | 70%为训练集，30%为验证集 |
| Rc-BBFA | **99.66%** | 38.46% | 70%为训练集，30%为验证集 |
| FSFOA | 98.07% | 50.00% | 70%为训练集，30%为验证集 |
| JMI | 92.41% | 58.46% | 70%为训练集，30%为验证集 |
| MIM | 94.81% | **74.61%** | 70%为训练集，30%为验证集 |

**表 7　XGBSFS 及其对比算法在 Vehicle 上的结果**

| Vehicle | CA | DR | 验证方法 |
|---|---|---|---|
| XGBSFS | **75.95%** | **63.89%** | 70%为训练集，30%为验证集 |
| Rc-BBFA | 75.79% | 61.11% | 70%为训练集，30%为验证集 |
| FSFOA | 73.81% | 61.11% | 70%为训练集，30%为验证集 |
| JMI | 65.28% | 38.89% | 70%为训练集，30%为验证集 |
| MIM | 67.83% | 28.33% | 70%为训练集，30%为验证集 |

**表 8　XGBSFS 及其对比算法在 Segmentation 上的结果**

| Segmentation | CA | DR | 验证方法 |
|---|---|---|---|
| XGBSFS | 96.67 | **67.37** | 70%为训练集，30%为验证集 |
| Rc-BBFA | **98.27** | 36.84% | 70%为训练集，30%为验证集 |
| FSFOA | 96.51 | 36.84% | 70%为训练集，30%为验证集 |
| JMI | 95.49 | 26.84% | 70%为训练集，30%为验证集 |
| MIM | 96.10 | 19.47% | 70%为训练集，30%为验证集 |

**表 9　XGBSFS 及其对比算法在 Ionosphere 上的结果**

| Ionosphere | CA | DR | 验证方法 |
|---|---|---|---|
| XGBSFS | **96.42%** | **81.77%** | 70%为训练集，30%为验证集 |
| Rc-BBFA | 96.18% | 58.82% | 70%为训练集，30%为验证集 |
| FSFOA | 89.52% | 54.28% | 70%为训练集，30%为验证集 |
| JMI | 91.98% | 64.71% | 70%为训练集，30%为验证集 |
| MIM | 90.28% | 47.65% | 70%为训练集，30%为验证集 |

**表 10　XGBSFS 及其对比算法在 Sonar 上的结果**

| Sonar | CA | DR | 验证方法 |
|---|---|---|---|
| XGBSFS | **95.71%** | **82.17%** | 70%为训练集，30%为验证集 |
| Rc-BBFA | 95.57% | 53.33% | 70%为训练集，30%为验证集 |
| FSFOA | 85.43% | 57.37% | 70%为训练集，30%为验证集 |
| JMI | 88.89% | 51.33% | 70%为训练集，30%为验证集 |
| MIM | 90.00% | 63.17% | 70%为训练集，30%为验证集 |

**表 11　XGBSFS 及其对比算法在 LSVT 上的结果**

| LSVT | CA | DR | 验证方法 |
|---|---|---|---|
| XGBSFS | 91.32% | 98.61% | 70%为训练集，30%为验证集 |
| Rc-BBFA | **94.60%** | 50.35% | 70%为训练集，30%为验证集 |
| FSFOA | 89.47% | 98.71% | 70%为训练集，30%为验证集 |
| JMI | 83.42% | **99.29%** | 70%为训练集，30%为验证集 |
| MIM | 82.37% | 99.16% | 70%为训练集，30%为验证集 |

**表 12　XGBSFS 及其对比算法在 CNAE-9 上的结果**

| CNAE-9 | CA | DR | 验证方法 |
|---|---|---|---|
| XGBSFS | 91.88% | **92.73%** | 70%为训练集，30%为验证集 |
| Rc-BBFA | **95.06%** | 50.35% | 70%为训练集，30%为验证集 |
| FSFOA | 91.05% | 24.88% | 70%为训练集，30%为验证集 |
| JMI | 87.04% | 70.19% | 70%为训练集，30%为验证集 |
| MIM | 87.22% | 77.08% | 70%为训练集，30%为验证集 |

**表 13　XGBSFS 及其对比算法在 Arcene 上的结果**

| Arcene | CA | DR | 验证方法 |
|---|---|---|---|
| XGBSFS | **97.83%** | **99.86%** | 70%为训练集，30%为验证集 |
| Rc-BBFA | 92.5% | 48.66% | 70%为训练集，30%为验证集 |
| FSFOA | 88.33% | 61.9% | 70%为训练集，30%为验证集 |
| JMI | 84.52% | 99.59% | 70%为训练集，30%为验证集 |
| MIM | 83.33% | 99.82% | 70%为训练集，30%为验证集 |

从表 4 可以发现，本文提出的同时使用 2 种重要性度量的 ISFFS 在全部数据集的分类准确率上均能带来不同程度的提升。在效果较为明显的 Wine 数据集上，ISFFS 甚至比仅使用单一重要性度量指标的序列浮动前向搜索策略中表现最好的 SFFS1 高出了 6%。从表 5 可以看出，ISFFS 在 Vehicle、Segmentation 和 LSVT 这 3 个数据集上的表现均位于最优，因此在这 3 个数据集上本文所提改进策略无疑是最佳的。至于在其他数据集上，ISFFS 的表现不如表 4 明显，是因为搜索过程中的主要目标是分类准确率，所以并不能保证维度缩减率也达到最优。总而言之，ISFFS 不仅可以提升分类准确率，而且在一定程度上能保证算法在 CA 取最优时仍使 DR 达到最优。唯一比较遗憾的是，在进行搜索之前，没有很好的方法能够得知由 XGBSFS 所计算出的 3 个重要性指标构成的 6 个组合中哪个是最优的。不过，总共仅有 6 种情况尚属于可枚举的范围，所以在实际实验操作中，本文恰好可以利用多核 CPU 的计算特性，采用 6 个内核并行运算的方式来降低计算时间开销，最后从中选择出分类准确率最佳的结果即为最终结果。

通过对比表 6~表 13 可以看出，XGBSFS 在 Vehicle、Ionosphere、Sonar 和 Arcene 这 4 个数据集上的平均分类准确率均达到了最高，在 Wine、Segmentation、LSVT 和 CNAE-9 这 4 个数据集上的平均分类准确率稍微落后于目前性能较好的基于萤火虫算法的特征选择算法，总体来看，在分类准确率上 XGBSFS 和 Rc-BBFA 处于伯仲之间。对比 JMI 和 MIM 这 2 个常用的基于信息理论的启发式特征选择策略，很容易发现本文提出的基于 XGBoost 的启发式方法 XGBSFS 在这些数据集上的准确率性能是占优势的。观察 DR 值，除了 Wine 和 LSVT 数据集之外，XGBSFS 的维度缩减率也均是最优的。另外结合式(8)中提出的复杂度进一步分析发现，对于绝大部分的实验数据集，均有 $M\leqslant[(1-\mathrm{DR})\times M]^3$，由此可以近似得出 $M\leqslant m^3$。又因为在通常情况下 $m\log m<m^3N\log N$，根据时间复杂度的性质，可以将式(8)简化成 $O(MNKD+m^3N\log N)$。这样基于简化后的形式，可以合理做一假设，当 XGBSFS 处理 $M$ 或 $N$ 较大的数据集时，$KD$ 会趋近于常数，所以 $m^3N\log N$ 的大小是最终决定算法性能的关键，而这一项的出现，是经由基于 $K$NN 分类器的时间复杂度提出的，再者绝大多数分类器的时间复杂度都是大于 $K$NN 分类器的，因此可以说本文所提算法仍主要受限于包裹式特征选择框架。但由于 XGBSFS 在 DR 值上有优良表现，因此这从侧面验证了本文基于 XGBoost 提出启发式策略的合理性，也很好地印证了本文所采用的启发式策略的高效性。

综合 8 个数据集的实验结果分析发现，大多数数据集的准确率都在 90%以上，只有 Vehicle 数据集上的结果是 75.95%。之所以有这种情况，是因为 Vehicle 数据集用于 1NN 分类器进行分类时效果并不是很好，但这并不能证明本文所提特征选择算法 XGBSFS 不适用于这一数据集，相反，通过 XGBSFS 对 Vehicle 进行特征选择之后，会达到全部对比算法中最高的分类准确率和维度缩减率，这说明了 XGBSFS 在该数据集上的优越性。而像在 LSVT 数据集上，虽然 CA 和 DR 值都在 90%以上，但 XGBSFS 的总体对比并没有十分突出，反而不能说明 XGBSFS 在该数据集上性能优秀。

总体而言，XGBSFS 之所以在有些数据集上表现良好，有些数据集上的结果不太理想，主要是因为通过 XGBoost 计算的 3 种重要性指标符合数据内在分布规律时，结果就令人满意，不符合数据的内在规律时，即使通过 ISFFS 策略混合了不同的重要性指标，也只能在有限的程度上改善结果；另一个导致实验结果不同的因素是数据集中存在的离群数据的数量，离群数据越多，对算法特征重要性计算的准确性干扰越大，进而影响实验结果。

## 5　结束语

本文提出了一种基于 XGBoost 的包裹式特征选择算法 XGBSFS，该算法利用 XGBoost 的重要性量度作为特征子集搜索启发依据，同时通过采用提出的 ISFFS 策略较好地完成了特征选择任务。实验结果表明，本文提出的特征选择算法与其他特征选择算法相比具有一定的竞争优势。在今后的工作

中，将进一步提高本文所提算法在超高维数据集以及庞大实例数据集上的性能，同时尝试采用并行计算或其他策略，努力寻求能让算法突破包裹式特征选择框架限制的方法。

## 参考文献：


[1] ZHOU T, LU H L, WANG W W, et al. GA-SVM based feature selection and parameter optimization in hospitalization expense modeling[J]. Applied Soft Computing, 2019(75): 323-332.

[2] LI J D, CHENG K W, WANG S H, et al. Feature selection: a data perspective[J]. ACM Computing Surveys, 2017, 50(6): 1-45.

[3] 周志华. 机器学习[M]. 北京: 清华大学出版社, 2016.
ZHOU Z H. Machine learning[M]. Beijing: Tsinghua University Press, 2016.

[4] LIU H, YU L. Toward integrating feature selection algorithms for classification and clustering[J]. IEEE Transactions on Knowledge and Data Engineering, 2005, 17(4): 491-502.

[5] ALMUALLIM H, DIETTERICH T G. Learning boolean concepts in the presence of many irrelevant features[J]. Artificial Intelligence, 1994, 69(1-2): 279-305.

[6] KAMATH U, DE J K, SHEHU A. Effective automated feature construction and selection for classification of biological sequences[J]. Plos One, 2014, 9(7): e99982.

[7] GUYON I, ELISSEEFF A. An introduction to variable and feature selection[J]. Journal of Machine Learning Research, 2003, 3(6): 1157-1182.

[8] ZAKERI A, HOKMABADI A. Efficient feature selection method using real-valued grasshopper optimization algorithm[J]. Expert Systems with Applications, 2019(119): 61-72.

[9] XUE B, ZHANG M, BROWNE W N, et al. A survey on evolutionary computation approaches to feature selection[J]. IEEE Transactions on Evolutionary Computation, 2016, 20(4): 606-626.

[10] GHAEMI M, FEIZI-DERAKHSHI M R. Feature selection using forest optimization algorithm[J]. Pattern Recognition, 2016(60): 121-129.

[11] ZHANG Y, SONG X, GONG D. A return-cost-based binary firefly algorithm for feature selection[J]. Information Sciences, 2017, 418-419: 561-574.

[12] CHEN T, GUESTRIN C. XGBoost: a scalable tree boosting system[C]//ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2016: 785-794.

[13] FRIEDMAN J, HASTIE T, TIBSHIRANI R. Special invited paper. additive logistic regression: a statistical view of boosting[J]. The Annals of Statistics, 2000, 28(2): 337-374.

[14] LU Y, LIU L, LUAN S, et al. The diagnostic value of texture analysis in predicting WHO grades of meningiomas based on ADC maps: an attempt using decision tree and decision forest[J]. European Radiology, 2019, 29(3): 1318-1328.

[15] PANG L, WANG J, ZHAO L, et al. A novel protein subcellular localization method with CNN-XGBoost model for alzheimer's disease[J]. Frontiers in Genetics, 2019(9): 1-7.

[16] PAN B. Application of XGBoost algorithm in hourly PM2.5 concentration prediction[J]. IOP Conference Series: Earth and Environmental Science, 2018(113): 012127.

[17] MACEDO F, ROSÁRIO OLIVEIRA M, PACHECO A, et al. Theoretical foundations of forward feature selection methods based on mutual information[J]. Neurocomputing, 2019(325): 67-89.

[18] VERGARA J R, ESTÉVEZ P A. A review of feature selection methods based on mutual information[J]. Neural Computing and Applications, 2014, 24(1): 175-186.

[19] SHI F, YAO Y, BIN Y, et al. Computational identification of deleterious synonymous variants in human genomes using a feature-based approach[J]. BMC Medical Genomics, 2019, 12(1): 81-88.

[20] XUE B, ZHANG M, BROWNE W N. Novel initialisation and updating mechanisms in PSO for feature selection in classification[J]. Applications of Evolutionary Computation, 2013(7835): 428-438.

[21] GHOSH A, DATTA A, GHOSH S. Self-adaptive differential evolution for feature selection in hyperspectral image data[J]. Applied Soft Computing, 2013, 13(4): 1969-1977.

[22] DUA D, EFI K T. UCI machine learning repository[Z]. The UCI Machine Learning Repository, 2019.

[23] BROWN G, POCOCK A C, ZHAO M J, et al. Conditional likelihood maximisation: a unifying framework for information theoretic feature selection[J]. Journal of Machine Learning Research, 2012(13): 27-66.

[24] CADENAS J M, GARRIDO M C, MARTÍNEZ R. Feature subset selection filter–wrapper based on low quality data[J]. Expert Systems with Applications, 2013, 40(16): 6241-6252.

[25] MAFARJA M M, MIRJALILI S. Hybrid whale optimization algorithm with simulated annealing for feature selection[J]. Neurocomputing, 2017(260): 302-312.

[26] EMARY E, ZAWBAA H M, HASSANIEN A E. Binary grey wolf optimization approaches for feature selection[J]. Neurocomputing, 2016(172): 371-381.


[作者简介]


**李占山**（1966− ），男，吉林长春人，博士，吉林大学教授、博士生导师，主要研究方向为约束优化与约束求解、机器学习、基于模型的诊断、智能规划与调度等。

**刘兆赓**（1993− ），男，吉林吉林人，吉林大学硕士生，主要研究方向为机器学习。