Hans Journal of Data Mining 数据挖掘, 2017, 7(2), 51-63
Published Online April 2017 in Hans. http://www.hanspub.org/journal/hjdm
https://doi.org/10.12677/hjdm.2017.72006

Hans汉斯

A Kind of FAST Feature Selection Algorithm Considering Feature Interaction

Biyun Lu, Lei Zhang

School of Mathematics, Sun Yat-sen University, Guangzhou Guangdong
Email: mcszl@mail.sysu.edu.cn

Received: Apr. 7th, 2017; accepted: Apr. 27th, 2017; published: Apr. 30th, 2017

Abstract

Interacting features are those appear to be irrelevant or weakly relevant with the class individually, but it may highly correlate to the class when it combined. Feature interaction is almost everywhere, but discovering feature interaction is a challenging task in feature selection. The purpose of this paper is to improve the FAST feature selection algorithm based on cluster by considering feature interaction. Firstly, deleted the irrelevant feature removal section, then brought in an interaction weight factor, so that we can retain interacted features when removed the irrelevant and redundant ones. In order to do the comparison between this two algorithms, we selected 16 public data sets which cover 5 different domains on the empirical analysis, and used 4 types of classifier to evaluate the results, namely, C5.0, Bayes Net, Neural Net and Logistic. Finally, we compared these two algorithms according to the number of selected features, running time of algorithm and the accuracy of classifier. The experimental result showed that it has little difference on the number of selected features, and sometimes IWFAST can produce smaller subsets of features. Meanwhile, IWFAST can improve the accuracy of the classifier, especially for the high-dimensional data set, or especially for the Game and Life area. The defect is that the running time of IWFAST is long, but is acceptable computational complexity.

Keywords

Feature Selection, Feature Interaction, Interaction Weight Factor, Filter, Feature Clustering, Graph Theory

考虑了特征协同作用的FAST特征选择算法的改进

陆碧云，张　磊

中山大学数学学院，广东 广州
Email: mcszl@mail.sysu.edu.cn

文章引用: 陆碧云, 张磊. 考虑了特征协同作用的 FAST 特征选择算法的改进[J]. 数据挖掘, 2017, 7(2): 51-63.
https://doi.org/10.12677/hjdm.2017.72006

收稿日期：2017年4月7日；录用日期：2017年4月27日；发布日期：2017年4月30日

摘　要

交互的特征是指那些分开考虑对目标集不相关或弱相关，但合在一起考虑却对目标集高度相关的特征。特征交互现象广泛存在，但找出有交互作用的特征却是一项具有挑战性的任务。本文旨在对基于聚类的FAST特征选择算法进行改进，在其基础上考虑特征的交互作用，首先去掉FAST的移除不相关特征的部分，接着加入交互权值变量，使得在移除不相关和冗余特征的同时，保留有交互作用的特征。为了对两个算法进行对比分析，我们选取了5个不同领域的16个公开数据集进行实证分析，并使用4种分类器对实验结果进行评估，包括C5.0、Bayes Net、Neural Net和Logistic，接着从选择的特征个数、算法运行时间和分类器的准确率3个方面对两个算法进行比较。实验结果表明，两者选择的特征个数相差不大，有时IWFAST甚至可以减少特征个数，同时IWFAST能提高分类器的准确率，尤其对于特征数量较多的情形，以及Game和Life领域。美中不足的是，IWFAST的运行时间较长，但仍在可接受的范围内。

关键词

特征选择，特征交互，交互权值变量，Filter，特征聚类，图论

1. 引言

特征选择是模式识别和机器学习中的数据预处理的一个步骤，在近些年来受到各个领域中诸多学者的关注。特征选择的目标，是从特征集合中选择一个尽可能小的特征子集，并且在表达原始特征时保留一个适当高的准确率。特征选择有很多优点，比如避免出现过拟合现象、提高数据的可视化、减少储存需求、减少训练时间、提高学习的准确率、提高结果的可理解性等[1]。

特征选择的方法可以分为四大类：Filter、Wrapper、Embedded 和 Hybrid。Filter 方法是选定一个指标来评估特征，根据特征指标值来对特征排序，同时设定阈值，将达不到该阈值的特征去掉。这类方法只考虑特征 X 和目标 Y 之间的关联，相对另两类特征选择方法 Wrapper 和 Embedded 计算开销最少。Wrapper 方法和 Filter 不同，它不考虑特征 X 和目标 Y 直接的关联性，而是在添加一个特征后观察模型最终的表现来评估特征的好坏。它使用一个明确的分类器去评估特征子集，以不同的搜索策略下分类器的准确率来评价特征。这类方法的准确率一般较高，但是所选择的特征子集依赖于分类器的选择，计算复杂度较高，而且容易出现过拟合。Embedded 方法将特征选择与算法本身紧密结合，在模型的训练过程中完成特征的选择，具体来说，先使用某些机器学习算法对模型进行训练，得到各个特征的权值系数，根据系数的大小选择特征。Hybrid 方法是 Filter 和 Wrapper 的结合，它通过使用 Filter 来产出一个特征子集，再运用 Wrapper 来对该子集进行特征选择。在这四类方法中，Filter 方法因其具有普遍性而受到广泛使用，特别是在处理大数据的情形下。

在 Filter 的特征选择算法中，基于聚类的特征选择算法被证实了比传统的算法更为有效，Pereira *et al.* [2]、Baker 和 McCallum [3]以及 Dhillon *et al.* [4]利用了词的分布聚类来减少文本数据的维度。在聚类分

析中，图论被广泛地运用于实践中。图论聚类的思路很简单：计算样本的邻近图，然后根据某一标准删除明显比邻边更大或更小的边，由此可以得到一个森林，森林中的每一颗树代表了一个簇。Qinbao Song, Jingjie Ni 和 Guangtao Wang [5]将图论聚类运用于特征选择中，采用了最小生成树(MST)理论，提出了一种基于快速聚类的特征选择算法(FAST)。FAST 算法有两个步骤：首先，通过图论聚类方法将特征划分成几个类，然后，从每一个类中选出一个最具有代表性，与目标集最相关的特征，组成最终的特征子集。在这一算法中，不同类中的特征各自独立，因此该算法极可能产生有用而相互独立的特征子集。

然而，与大多数特征选择方法一样，FAST 算法关注的是尽可能地去除不相关和冗余的特征。但除了鉴别不相关和冗余的特征之外，一个重要而又常常被忽视的概念就是特征间的信息交互[6]。有交互作用的特征是指各自对目标集不相关，但一旦结合起来却对目标集高度相关的特征。XOR 问题就是一个典型的例子。尽管最近有研究指出交互作用的存在，但是很少有文献给出交互作用的具体解决方案。Zilin Zeng，Hongjun Zhang，Rui Zhang 和 Chengxiang Yin [7]给出了信息交互作用的定义，提出了基于交互权值的特征选择算法，引入交互权值变量来衡量特征的冗余和交互作用。

本文拟对 FAST 算法进行改进，在其基础上考虑特征的交互作用，在算法中加入交互权值变量，用这一变量去修改邻近图的边的权重，并调整相应的特征筛选标准，构造一个新的特征选择算法，称为 IWFAST。

为了证实 IWFAST 算法的有效性，我们用了 5 个领域的 16 个公开的数据集进行检验并与 FAST 算法进行对比分析。实证结果表明，与 FAST 算法相比，IWFAST 算法在某些情况能够减少特征的个数，还能提高在不同分类器下算法的准确率。

本文接下来的内容按以下结构展开：第 2 章，介绍相关的工作。第 3 章，给出相关、冗余和交互的定义。第 4 章，分别介绍 FAST 算法和 IWFAST 算法的框架和具体流程。第 5 章，将两个算法分别运用于实际数据中，以做对比分析。最后，第 6 章，作出一个简要的结论并且给出进一步的工作计划。

2. 相关工作

近些年来，层次聚类被用于文本分类的选词中。分布聚类可将词划分成不同的群组，在这类研究中，Pereira *et al*. [2]是基于词之间特殊的语法关系，而 Baker 和 McCallum [3]则基于目标集的分布。由于词的分布聚类会自然地凝聚，从而导致次最优的词聚类以及较高的计算开销，Dhillon *et al*. [4]提出了一种新的信息理论划分算法用于词聚类，并且将其应用于文本分类中。Butterworth *et al*. [8]在特征的每一个类中使用一种特殊的测量标准，叫 Barthelemy-Montjardet 距离，然后在得到的聚类层级中使用系统树图来选择最相关的特征。不幸的是，这种基于 Barthelemy-Montjardet 距离的聚类评估标准不能识别出能够允许分类器提高原始准确率的特征子集，进一步说，与其他特征选择算法相比，这一方法得到的准确率较低。

层次聚类还能用于光谱数据的特征选择中。Van Dijck 和 Van Hulle [9]提出了一种 Hybrid 特征选择回归算法。Krier *et al*. [10]展示了一个结合了光谱变量的分层约束聚类和基于互信息的簇的选取的方法。这一特征聚类方法和 Van Dijck 和 Van Hulle [9]的类似，唯一不同的是前者要求每个类中必须只包含连续的特征。这两种方法都使用了凝聚层次聚类来剔除冗余的特征。

与上面基于层次聚类的特征选择算法不同，Qinbao Song，Jingjie Ni 和 Guangtao Wang 提出的 FAST 算法[5]在聚类时运用了最小生成树的理论。这也是本文重点关注的内容。

在很多分类问题中，一个特征单独地看对目标集并不提供有用的信息，但当和另一个特征结合在一起时，会对分类的准确性产生明显的提升效果。如果我们只考虑相关性和冗余而忽视交互作用，我们就会损失一些有价值的特征。

于是，特征的交互作用逐渐引起了学者们的关注。Zhao and Liu [11]提出了一种基于数据一致性的特

征评分标准，发展了一种 Filter 算法，叫 INTERACT，以便在选择相关特征的同时探索特征的交互作用。最近，Wang *et al.* [12]提出了一个基于 FOIL 规则的算法 FRFS，它不仅保留了相关的特征，剔除不相关和冗余的特征，还考虑了特征的交互作用。FRFS 首先将 FOIL 规则中所有在前的特征合并，得到一个排除了冗余特征并保留了交互特征的候选特征子集。接着，使用一个新的 CoverRatio 标准去识别和剔除候选特征子集中不相关的特征，得到最终的特征子集。

Zilin Zeng，Hongjun Zhang, Rui Zhang 和 Chengxiang Yin [7]在信息理论框架中重新定义了相关性、冗余和交互作用，接着引入一个交互权值变量，该变量可以反映出特征是冗余的还是信息交互的，从而提出了基于交互权值的特征选择算法(IWFAST)。本文将利用 IWFAST 中的相关定义及交互权值变量，对 FAST 算法进行改进，使得在保留相关特征和剔除冗余特征的同时，仍能保留有交互作用的特征。

3. 相关、冗余和交互的定义

在介绍相关、冗余和交互的定义之前，首先要引入一个概念，叫交互增益(即交互信息)。交互增益的引进利用了信息增益(即互信息)的思想。

信息增益是指期望信息或信息熵的有效减少量。设 $X=\{x_1,x_2,\cdots,x_n\}$ 和 $Y=\{y_1,y_2,\cdots,y_m\}$ 是两个离散的随机变量，他们的联合概率分布是 $p(x_i,y_j)$，其中 $i=1,2,\cdots,n;\ j=1,2,\cdots,m$。则信息增益 $I(X;Y)$ 的定义为

$$I(X;Y)=\sum_{i=1}^{n}\sum_{j=1}^{m}p(x_i,y_j)\log_2\frac{p(x_i\mid y_j)}{p(x_i)} \tag{1}$$

信息增益 $I(X;Y)$ 可被视为当两个随机变量有交互作用时对信息交互的衡量。Jakulin [13] [14]概括了当有三个随机变量有交互作用时对信息交互的衡量，介绍了交互增益的定义：

$$I(X;Y;Z)=I(X,Y;Z)-I(X;Z)-I(Y;Z) \tag{2}$$

该定义又可以变为

$$I(X;Y;Z)=\sum_{k}\sum_{j}\sum_{l}p(a_k,b_j,c_l)\log\frac{p(a_k,b_j)\,p(b_j,c_l)\,p(a_k,c_l)}{p(a_k)\,p(b_j)\,p(c_l)\,p(a_k,b_j,c_l)} \tag{3}$$

与互信息不同，交互信息可以为正、为 0 或为负。当两个随机变量合在一起可以提供一些在他们分开时不能提供的信息，则交互信息取正数；当两个随机变量可以提供一些相同的信息时，交互信息取负数；当 X (或 Y)的引入不能改变 Y (或 X)与 Z 的关系时，交互信息为 0。

Zilin Zeng，Hongjun Zhang，Rui Zhang 和 Chengxiang Yin [7]利用上述交互信息的概念给出了相关、冗余和交互的定义：

定义 1 (相关)设 F 是一个特征全集，$F_i\in F$ 且 $F'=F-\{F_i\}$。特征 F_i 对于目标集 C 是相关的当且仅当 $\exists S\subseteq F'$，使得 $I(F_i;C\mid S)>0$。否则，特征 F_i 是不相关的。

依据定义 1，相关性有条件地依赖于 S。一个特征 F_i 对 C 是相关的，只要存在 $F-\{F_i\}$ 的一个子集 S，使得对于 S，F_i 和 C 的条件互信息是非负的。这和传统的相关特征的定义[15]是一致的，并且包括了强相关和弱相关两种情况。

定义 2 (冗余)设 $F=\{F_1,F_2,\cdots,F_k\}$ 是一个拥有 k 个特征的特征集，$F'\subset F$ 且 $F''=F-F'$。F 中的特征是相互冗余的当且仅当 $\forall F'\subset F$，$I(F;C)\le I(F';C)+I(F'';C)$。

依据定义 2，$I(F;C)\le I(F';C)+I(F'';C)$ 表明 F' 和 F'' 是相互冗余的，换句话说，他们都提供了部分关于 C 的重叠的信息。因此，这一不等式暗示了 k 个特征中存在着冗余。

定义 3 (交互)设 $F=\{F_1,F_2,\cdots,F_k\}$ 是一个拥有 k 个特征的特征集， $F'\subset F$ 且 $F''=F-F'$ 。 F 中的特征是相互交互的当且仅当 $\forall F'\subset F$ ， $I(F;C)\geq I(F';C)+I(F'';C)$ 。

依据定义 3， $I(F;C)\geq I(F';C)+I(F'';C)$ 意味着特征子集 F' 和 F'' 间存在协同作用，即他们合在一起会比分开时提供更多的信息。换句话说， F 中任意一个特征的缺失都会降低预测 C 的能力。因此，交互的定义是合理的。特别地，当 $k=2$ 时，有， $I(F;C)\geq I(F_1;C)+I(F_2;C)$ 。

4. 算法介绍

4.1. FAST 特征选择算法

Qinbao Song，Jingjie Ni 和 Guangtao Wang [5]提出了一种新奇的 FAST 特征选择算法，它能有效地处理不相关和冗余的特征，得到一个好的特征子集。图 1 的左图展示了 FAST 算法的框架，其中包括了移除不相关特征和剔除冗余特征两个部分。前者通过移除不相关特征以得到对目标集相关的特征子集，后者通过从不同的类中选择代表性的特征来剔除冗余特征，从而得到最终的特征子集。

相关性准则一经确定，就可以直接移除不相关特征。而剔除冗余特征稍微复杂一些，它包括：在加权完成图中构造最小生成树；将最小生成树划分成森林，每一棵树代表一个类；从每个类中选择一个代表特征。

为了更详细地描述这一算法，需要介绍一些定义。

首先，从 3 节我们可以知道，相关的特征与目标集有很强的相关性，而冗余的特征相互之间是相关的。因此，特征的冗余和相关性分别对应了特征间的相关性和特征与目标集的相关性。

互信息衡量了特征与目标集的分布的统计独立性的不同程度。这是一个对特征之间或者特征与目标集的相关性的非线性估计。通过特征或目标集的熵将互信息标准化，得到对称不确定性(SU)。FAST 算法使用了对称不确定性(SU)作为两个特征之间或者特征与目标集之间的相关性的度量， SU 的定义如下：

$$SU(X,Y)=\frac{2I(X;Y)}{H(X)+H(Y)} \tag{4}$$

其中 $H(X)$ 为 X 的熵，定义为

$$H(X)=-\sum_{i=1}^{n}p(x_i)\log_2 p(x_i) \tag{5}$$

对称不确定性将一对变量视为是对称的，它对变量的信息增益的偏倚进行了补偿，并将其标准化使得它的值落在 $[0,1]$ 区间内。虽然该度量是对离散变量的定义，但他也可以用于连续变量中，只要将连续变量进行离散化即可。

在引入了对称不确定性的概念后，我们便可以用它来衡量相关性。特征间的相关性表示为 $SU(F_i,F_j)$ ，其中 $i\neq j$ ；特征与目标集的相关性表示为 $SU(F_i,C)$ ，在 FAST 算法中，如果 $SU(F_i,C)$ 的值大于某个预定的阈值 θ ，则可以认为 F_i 对目标值是强相关的，小于该阈值的 F_i 就可以移除了。

对称不确定性还可以衡量变量间的冗余[5]。 $S=\{F_1,F_2,\cdots,F_i,\cdots,F_{k<|F|}\}$ 是特征集的一个聚类，如果 $\exists F_j\in S$ ，使得对每一个 $F_i\in S(i\neq j)$ ，都有：

$$SU(F_j,C)\geq SU(F_i,C)\wedge SU(F_i,F_j)>SU(F_i,C),$$

则 F_i 对于 F_j 是冗余的。

当然，也可以通过对称不确定性筛选出每个类的代表特征，也就是说，对于一个类 S ， $F_j\in S$ ，选择使得 $SU(F_j,C)$ 最大的那一个 F_j 作为该类的代表特征。

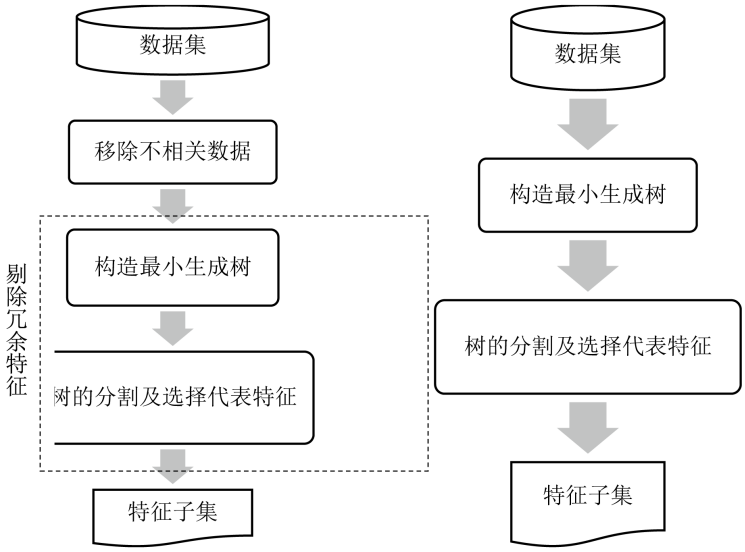

Figure 1. Frame of FAST Algorithm and IWFAST Algorithm
图 1. FAST 算法和 IWFAST 算法的框架

FAST 算法一共有三个步骤：1) 移除不相关的特征；2) 构造最小生成树；3) 分割最小生成树并选择每个类的代表特征。

移除不相关特征和选择代表特征上面已经解决，只剩下构造最小生成树和分割最小生成树。首先，为什么要构造最小生成树呢？对于一个含有 m 个特征的数据集，每两个特征求 $SU\left(F_i,F_j\right)$ 作为他们的边的权值，则可以得到一个无向的加权完成图 G， G 反映了各个特征间的相关性。不幸的是， G 含有 k 个顶点 $k\left(k-1\right)/2$ 条边，对于高维数据集，则过于密集且各条边强烈地交织在一起。此外，分解这样的完成图被证实是一个 NP-hard 问题。因此，对 G，首先要生成最小生成树，使得在包含所有顶点的情况下让边的权值的总和达到最小。在 FAST 算法中，运用 Prim 算法[16]生成最小生成树。

接下来要解决如何分割最小生成树。利用上面的基于对称不确定性的冗余的定义，只要我们将最小生成树中所有权值小于 $SU\left(F_i,C\right)$ 和 $SU\left(F_j,C\right)$ 的边去掉，我们就能保证对于剩下的每一条边 $\left(F_i,F_j\right)$，都有 $SU\left(F_i,F_j\right)\geq SU\left(F_i,C\right)$ 或 $SU\left(F_i,F_j\right)\geq SU\left(F_j,C\right)$，从而可以知道同一类中的所有特征都是冗余的。经过上面的叙述，FAST 算法的伪代码如表 1：

在伪代码第 9 行中，让 $SU\left(F_i',F_j'\right)$ 加 1 是为了避免当 $SU\left(F_i',F_j'\right)$ 的值为零时相应的边的权值无效。

4.2. IWFAST 特征选择算法

首先，需要给出交互权值变量的定义[7]。

从等式(2)我们可以看到，特征 F_j 的引入会影响特征 F_i 和目标集 C 之间的独立性。正的交互增益意味着我们只有将两个特征合在一起考虑，才能描述他们的关系，同时会增加独立性的值。也就是说，特征 F_j 的引入对预测目标集有正的影响。相应地，我们需要增加特征 F_j 的权重。负的交互增益意味着新的特征的加入会抑制独立性的大小，也就是说，特征 F_j 的引入对预测目标集有负的影响。相应地，我们需要减少特征 F_j 的权重。因此，可以基于交互增益定义一个交互权值变量，它使得分析特征间的关系成为可能。

定义 7 (交互权值变量)两个特征 F_i 和 F_j 之间的交互权值变量定义为：

$$IW\left(F_i,F_j\right)=1+\frac{2I\left(F_i;F_j;C\right)}{H\left(F_i\right)+H\left(F_j\right)} \tag{6}$$

Table 1. Pseudocode of FAST Algorithm
表 1. FAST 算法的伪代码

算法： *FAST*
输入： $D(F_1,F_2,\cdots,F_m,C)$-已知数据集； θ-预定的阈值
输出： S-最终的特征子集
//Part1：剔除不相关特征
1 *For* $i=1$ *to* m *do*
2 $T\text{-}Relevance = SU(F_i,C)$;
3 *If* $T\text{-}Relevance > \theta$ *do*
4 $S = S \cup \{F_i\}$;
5 *End*
6 *End*
//Part2：最小生成树的构造
7 $G = NULL$; // G 是一个完成图
8 *For* 每一对特征 $\{F_i', F_j'\} \subset S$ *do*
9 $F-Correlation = 1 + SU(F_i', F_j')$;
10 将 F_i' 和 F_j' 加入 G 中并将 $F-Correlation$ 作为他们的边的权值;
11 *End*
12 $minSpanTree = Prim(G)$; //使用 *Prim* 算法生成最小生成树
//Part3：树的分割
13 $Forest = minSpanTree$;
14 *For* 每一条边 $E_{i,j} \in Forest$ *do*
15 *If* $SU(F_i,F_j) > SU(F_i,C) \wedge SU(F_i,F_j) > SU(F_j,C)$ *do*
16 $Forest = Forest - E_{i,j}$; //剪枝
17 *End*
18 *End*
//Part4：代表特征的选择
19 $S = \varnothing$;
20 *For* 每一棵树 $T_i \in Forest$ *do*
21 $F_R = argmax_{F_k' \in T_i} SU(F_k', C)$;
22 $S = S \cup \{F_R\}$;
23 *End*
24 *return* S

对于交互权值变量，有以下三个定理[7]：

定理 1 $0 \le IW(F_i,F_j) \le 2$。

定理 2 如果特征 F_i 相对于特征 F_j 是冗余的，则有 $0 \le IW(F_i,F_j) \le 1$。

定理 3 如果特征 F_i 相对于特征 F_j 是交互的，则有 $1 \le IW(F_i,F_j) \le 2$。

FAST 算法在考虑特征的冗余时并没有考虑特征的交互作用，这会导致在特征选择过程中损失某些有用的特征变量。为了解决这一问题，我们使用交互权值变量(*IW*)去修正特征之间或者特征与目标集之间的对称不确定性(*SU*)。

有了交互权值变量的定义和定理之后，我们给出 IWFAST 算法的伪代码如表 2 所示：

在 IWFAST 算法中，我们去掉了剔除不相关特征这一步骤，将所有特征形成完全图，这是因为由于交互作用的影响，不相关特征跟其他特征一起也可能为预测目标集提供有用的信息，因此，在一开始就剔除不相关特征有可能会损失有价值的特征。所以 IWFAST 算法的框架如图 1 的右图所示：

Table 2. Pseudocode of IWFAST Algorithm
表 2. IWFAST 算法的伪代码

算法： *IWFAST*
输入： $D(F_1,F_2,\cdots,F_m,C)$-已知数据集； θ-预定的阈值
输出： S-最终的特征子集
//Part1：最小生成树的构造
1 $G=NULL$; // G 是一个完成图
2 *For* 每一对特征 $\{F_i,F_j\}\subset F$ *do*
3 $adj-correlation(F_i,F_j)=IW(F_i,F_j)\times(1+SU(F_i,F_j))$;
4 将 F_i 和 F_j 加入 G 中并将 $adj-correlation(F_i,F_j)$ 作为他们的边的权值;
5 *End*
6 $minSpanTree=Prim(G)$; //使用 *Prim* 算法生成最小生成树
//Part2：树的分割
7 $Forest=minSpanTree$;
8 *For* 每一条边 $E_{i,j}\in Forest$ *do*
9 *If* $SU(F_i,F_j)>SU(F_i,C)\wedge SU(F_i,F_j)>SU(F_j,C)$ *do*
10 $Forest=Forest-E_{i,j}$; //剪枝
11 *End*
12 *End*
//Part3：代表特征的选择
13 $S=\varnothing$;
14 *For* $i=1$ *to* m *do*
15 $F'=F-\{F_i\}$;
16 $F_R=argmax_{F_j\in F}IW(F_i,F_j)$;
17 $adjSU(F_i,C)=IW(F_i,F_R)\times(1+SU(F_i,C))$;
18 *If* $IW(F_i,F_R)>1.05$ *do*
19 $flag(F_i)=1$;
20 $interact(F_i)=\{F_R\}$;
21 *End*
22 *If* $adjSU(F_i,C)>\theta$ *do*
23 *If* $flag(F_i)=1$ *do*
24 $S=S\cup\{F_i\}\cup interact(F_i)$;
25 *Else*
26 $S=S\cup\{F_i\}$;
27 *End*
28 *End*
29 *End*
30 给 S 去重得到 FS；
31 *return* FS

在构造最小生成树这一部分，在确定边的权值时(即伪代码第 3 行)，我们在 FAST 算法的基础上乘以了 $IW(F_i,F_j)$，因为如前所述，如果 F_i 和 F_j 相互冗余，则 $0\le IW(F_i,F_j)\le 1$，于是调整后的权值比 FAST 算法的权值小；若 F_i 和 F_j 有交互作用，则 $1\le IW(F_i,F_j)\le 2$，那么调整后的权值比 FAST 算法的权值大。于是，在生成最小生成树时，交互特征的边就更可能被去掉，而更可能保留冗余特征的边，从而使得冗余的特征更可能被分在同一类中，交互的特征被分在不同的类中。

对于树的分割部分，跟 FAST 算法相同，以使得在同一类中的特征都是冗余的。而在代表特征的选

择这一部分中，FAST 算法是直接将每一类中使得 $SU(F_i,C)$ 最大的特征作为代表特征，同样的，我们要对 $SU(F_i,C)$ 进行调整，调整方法是：对于每一个特征 F_i，找出使得 $IW(F_i,F_j)$ 最大的特征 F_j，从而用这个最大值乘以 $1+SU(F_i,C)$ 得到调整后的标准，如伪代码 14-17 行所示。如果 $IW(F_i,F_j)$ 的最大值足够大，表明 F_i 和 F_j 的交互作用值得被重视，就要将 F_i 和 F_j 一起放入特征子集中。

在本文中，经过多次实验，我们决定将 $IW(F_i,F_j)\geq 1.05$ 视为是否重视该交互作用的一个标准。如果未达到 1.05，则忽略这两个特征的交互作用；如果满足条件，就要把两个特征捆绑在一起考虑。伪代码第 18-21 行就实现了这一想法。接着，我们已经对每个类中的特征都求得了一个调整后的对称不确定 $adjSU(F_i,C)$，在每个类中取使 $adjSU(F_i,C)$ 达到最大值的 F_i，但与 FAST 算法不同，在 IWFAST 算法中，只有最大的这个 $adjSU(F_i,C)$ 超过了一个特定的阈值，我们才将相应的特征加入特征子集，而该特征是否有交互的另一特征，由 $flag(F_i)$ 的值决定，这也就是上面 22-29 行的内容。最后，由于有交互作用，得到的特征子集可能会有重复的特征，因此再对该特征子集去重，得到最终的特征子集 FS。

5. 实证分析

5.1. 数据来源

为了很好地评估我们的 IWFAST 算法的性能和效力，以证实该算法是否能有效地用于实践中，并且能够使其他学者信服我们的结果，我们从 UCI 数据库中选择了 16 个数据集来进行实证对比分析。

这 16 个数据集的特征个数的范围从 4 到 259，样本个数的范围从 32 到 41188，目标集的范围从 2 到 17。此外，这 16 个数据集囊括了 5 个领域，分别为商业领域(Business)、社会领域(Social)、游戏领域(Game)、生物领域(Life)和物理领域(Physical)。表 3 展示了这些数据集的相关信息。

其中 F、I 和 T 分别表示特征的个数，样本的个数和目标分类的个数。

5.2. 实验准备

为了评估 IWFAST 算法的性能和效力以及将其与 FAST 算法进行比较，我们首先要进行以下实验准备与分析：数据离散化，缺失值的处理，阈值的选择。

FAST 和 IWFAST 都涉及阈值的设定。对于 FAST 算法，将阈值设定为在所有 SU 的值中从大到小排名第 $\left[\sqrt{m}*\lg m\right]_{th}$ 的元素[5]所对应的 SU 值，若 $\left[\sqrt{m}*\lg m\right]_{th}$ 为小数，对其四舍五入取整。对于 IWFAST 算法，我们将阈值设定为在所有 $adjSU$ 值中从大到小排名第 $\left[\sqrt{m/\ln m}\right]_{th}$ 的元素所对应的 $adjSU$ 值，若 $\left[\sqrt{m/\ln m}\right]_{th}$ 为小数，同样对其取四舍五入取整。

5.3. 结果与分析

为了更好地比较两个算法选择的特征子集的优劣，我们选择了 4 个分类器对其进行评估，最后再求出 4 个分类器所得结果的平均值。这 4 个分类器分别为决策树 C5.0、贝叶斯网络(Bayes Net)、神经网络(Neural Net)和 Logistic 回归。评估过程依然使用软件 Clementine 12.0。同时，所有数据集以 70%作为训练集，30%作为测试集。此外，在决策树 C5.0 中，我们使用 10 折交叉验证法建立模型。我们将从三个方面分析两个算法的优劣：1) 选择的子集的特征个数；2) 获得特征子集所需要的运行时间；3) 分类器的准确率。特征个数与运行时间的比较统一记录于表 4 中。

从表 4 可以看出，有时 FAST 比 IWFAST 筛选的特征多，有时又少。而对于运行时间，IWFAST 需要的时间总是比 FAST 多，这也与本文第 4 章对算法时间复杂度的理论分析结果一致。

4 种分类器所得到的准确率及平均值则在表 5 中记录。

Table 3. Summarize of sixteen datasets
表 3. 16 个数据集的概况

No.	Data Name	F	I	T	Area
1	Iris	4	150	3	Life
2	Balance Scale	4	625	3	Social
3	Istanbul Stock Exchange	7	536	3	Business
4	Tic-Tac-Toe Endgame	9	958	2	Game
5	Wine	13	178	3	Physical
6	Statlog (Australian Credit Approval)	14	690	2	Business
7	Congressional Voting Records	16	435	2	Social
8	Bank Marketing	20	41188	2	Business
9	Ionosphere	33	351	2	Physical
10	Chess (King-Rook vs. King-Pawn)	36	3196	2	Game
11	Spectf Heart	44	267	2	Life
12	Lung-cancer	56	32	3	Life
13	Audiology	68	226	17	Life
14	Ozone Level Detection	72	2158	2	Physical
15	Musk1	166	476	2	Physical
16	Arrhythmia	259	452	16	Life

Table 4. Number of features and running time
表 4. 特征个数和运行时间

No.	Instances	Features	Number of features selected		Running time (in s)	
			IWFAST	FAST	IWFAST	FAST
1	150	4	3	1	0.989	0.768
2	625	4	4	4	1.184	0.908
3	536	7	3	1	1.498	0.977
4	958	9	5	3	1.574	1.376
5	178	13	3	4	2.524	0.909
6	690	14	3	3	3.212	1.225
7	435	16	4	3	1.872	0.995
8	41188	20	4	1	2916.119	427.275
9	351	33	5	9	17.135	3.492
10	3196	36	5	1	11.277	3.061
11	267	44	5	3	21.384	3.441
12	32	56	10	6	36.294	4.071
13	226	68	7	12	171.739	5.309
14	2158	72	6	1	54.017	9.285
15	476	166	9	12	236.195	32.328
16	452	259	11	31	5119.651	123.520

Table 5. Accuracy of four classifiers and the average accuracy
表 5. 4 种分类器的准确率以其平均值

No.	C5.0		Bayes Net		Neural Net		Logistic		Average	
	IWFAST	FAST	IWFAST	FAST	IWFAST	FAST	IWFAST	FAST	IWFAST	FAST
1	95.74	100	95.74	100	100	100	95.74	100	96.81	100
2	82.92	82.92	85.94	85.94	91.15	91.15	88.54	88.54	87.14	87.14
3	65.62	66.62	68.12	66.62	70	66.62	71.25	66.62	68.75	66.62
4	81.4	73.33	75.44	70.88	68.07	66.67	60.35	61.4	71.32	68.07
5	92.98	98.25	94.74	94.74	96.49	100	94.74	100	94.74	98.25
6	83.01	83.01	83.01	83.01	83.01	83.01	83.01	83.01	83.01	83.01
7	99.28	99.28	97.12	98.56	99.28	99.28	97.12	98.56	98.2	98.92
8	90.03	89.94	90.12	89.94	90.25	89.94	90.23	89.94	90.16	89.94
9	91.89	91.89	90.99	85.59	95.5	95.5	83.78	90.99	90.54	90.99
10	94.66	67.19	94.66	67.19	94.66	67.19	94.66	67.19	94.66	67.19
11	76.47	76.47	80	76.47	80	76.47	80	76.47	79.12	76.47
12	75	59.38	81.25	87.5	93.75	84.38	87.5	78.12	84.38	77.35
13	72.22	69.44	73.61	65.28	77.78	76.39	69.44	55.56	73.26	66.67
14	96.6	96.6	96.6	96.6	97.19	96.6	96.45	96.6	96.71	96.6
15	75	75	79.73	76.35	74.32	78.38	77.7	76.35	76.69	76.52
16	64.14	68.97	53.1	11.93	69.66	77.24	62.07	42.76	62.24	50.23

从平均值来分析，对于数据集 1、5、7、9，FAST 比 IWFAST 的准确率高；对于 2、6，两者的准确率相同；对于数据集 3、4、8、10、11、12、13、14、15、16，IWFAST 比 FAST 的准确率高。可见对于特征个数较多的数据集，IWFAST 更有优势，这也是因为在特征较多的情形下，FAST 更有可能将一些有用的特征剔除，而 IWFAST 由于考虑了交互作用就更可能保留一些有用的特征。此外，就准确率而言，在 FAST 优于 IWFAST 的情形下，FAST 与 IWAFST 的准确率之差并不大于 4%，而 IWFAST 优于 FAST 的情形，IWFAST 与 FAST 的准确率之差最大可达到 27.47%。因此可以大致认为 IWFAST 分类能力优于 FAST。

为了更好地比较两个算法的优劣，我们按领域分类取平均并求出了所有数据集总的平均值，三种评判标准的各领域平均值及总平均值记录如表 6 所示。

下面对表 6 进行分析。

首先是运行时间，IWFAST 的运行时间都比 FAST 要长，但仍在可接受的范围内。

接着对特征个数和准确率进行讨论。对于 Life 和 Physical 领域，IWFAT 能够减少 FAST 选择的特征的个数，并且提高预测的准确率，IWFAST 分别使准确率提高了 5.02%和 0.11%。然而在 Social 领域中，IWFAST 增加了特征的个数，但是却导致准确率下降 0.36%，不过这一变化幅度较小，也就是说，两个算法相差不大。对于 Business 和 Game 领域，IWFAST 通过增加特征个数使准确率分别提高了 0.78%和 13.3%。在各个领域中两个算法的特征个数相差不大，而准确率相差较大的是 Life 和 Game 领域，从这一角度，在 Life 和 Game 领域中，IWFAST 的优势更为明显，尤其是 Game 领域。

对于总的平均值，IWFAST 的特征个数小于 FAST，运行时间长于 FAST，而准确率比 FAST 算法高 3.36%，因此，本文对 FAST 算法的改进是合理与可行的。

Table 6. Average values for five kinds of items
表 6. 5 个领域的各项平均值的比较

Average	Number of features selected		Running time (in s)		Accuracy (%)	
	IWFAST	FAST	IWFAST	FAST	IWFAST	FAST
Life	7.2	10.4	1117.633	27.422	79.16	74.14
Social	4	3.5	1.508	0.952	92.67	93.03
Business	3.33	1.67	980.22	143.159	80.64	79.86
Game	5	2	6.427	2.219	80.93	67.63
Physical	5.75	6.5	81.092	11.504	90.7	90.59
Total average	5.44	5.88	537.29	38.684	84.23	80.87

6. 结论

特征选择的主要目的是找出一个尽可能小的特征子集，使得该特征子集能够准确地预测目标集。特征交互存在于很多实际情形中，但找出有交互作用的特征是一项具有挑战性的任务。本文旨在对 FAST 算法进行改进，在 FAST 算法的基础上考虑特征的交互作用，使得在移除不相关和冗余特征的同时，保留有交互作用的特征。改进的 IWFAST 算法去掉了 FAST 中移除不相关特征这一步骤，对所有特征构造完成图，并在 FAST 的权值与相关度量的基础上加入交互权值变量，将调整后的度量作为完成图的权值，使得算法能够反映出特征的冗余或交互。接着，IWFAST 算法仍需要生成最小生成树，分割最小生成树，选择代表特征，最终得到特征子集。

我们将 FAST 算法和 IWFAST 算法运用 5 个领域的 16 个公开数据集中，包括商业领域(Business)、社会领域(Social)、游戏领域(Game)、生物领域(Life)和物理领域(Physical)，并运用决策树 C5.0、贝叶斯网络(Bayes Net)、神经网络(Neural Net)和 Logistic 回归这 4 个分类器对两个算法的结果进行评估，从特征个数、运行时间和准确率三个方面进行对比分析。实验结果表明，IWFAST 算法的性能和效力要优于 FAST 算法，特别是在 Game 和 Life 领域，或者是对于特征数量较多的数据集。但是美中不足的是，IWFAST 的运行时间较长，这是考虑交互作用所要付出的代价，不过也是在可接受的范围内。因此，IWFAST 为基于聚类的特征选择算法提供了参考，也可以作为处理高维数据的一种有效方法。

当然，本文的算法还存在着一些不足，比如，IWFAST 算法的运行时间偏长；算法只能运用于离散数据等，对连续数据离散化可能会损失部分信息。因此，我们将来的工作首先要重点关注 IWFAST 算法的优化，考虑是否有其他的编译环境或者编译语言能够缩减算法的运行时间。接着，要考虑是否可以将 IWFAST 推广到连续数据或混合型数据中。

参考文献 (References)

[1] Guyon, I. and Elisseeff, A. (2003) An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research*, **3**, 1157-1182.

[2] Pereira, F. Tishby, N. and Lee, L. (2007) Distributional Clustering of English Words. *Proceedings of Sixth International Conference on Advanced Language Processing and Web Information*, Luoyang, 22-24 August 2007, 123-128.

[3] Baker, L.D. and McCallum, A.K. (1998) Distributional Clustering of Words for Text Classification. *Proceedings of the* 21*st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Melbourne, 24-28 August 1998, 96-103. https://doi.org/10.1145/290941.290970

[4] Dhillon, I.S., Mallela, S. and Kumar, R. (2003) A Divisive Information Theoretic Feature Clustering Algorithm for Text Classification. *Machine Learning Research*, **3**, 1265-1287.

[5] Song, Q.B., Ni, J.J. and Wang, G.T. (2013) A Fast Clustering-Based Feature Subset Selection Algorithm for

High-Dimensional Data. *IEEE Transactions on Knowledge and Data Engineering*, **25**, 1. https://doi.org/10.1109/TKDE.2011.181

[6] Jakulin, A. and Bratko, I. (2003) Analying Attribute Dependencies. *Proceedings of Seventh European Conference on Principles and Practice of Knowledge Discovery in Databases*, Cavtat-Dubrovnik, 22-26 September 2003, 229-240.

[7] Zeng, Z.L., Zhang, H.J., Zhang, R. and Yin, C.X. (2015) A Novel Feature Selection Method Considering Feature Interaction. *Pattern Recognition*, **48**, 2656-2666. https://doi.org/10.1016/j.patcog.2015.02.025

[8] Butterworth, R., Piatetcky-Shapiro, G. and Simovici, D.A. (2005) On Feature Selection through Clustering. *Proceedings of the Fifth IEEE International Conference on Data Mining*, Washington DC, 27-30 November 2005, 581-84. https://doi.org/10.1109/ICDM.2005.106

[9] Van Dijck, G. and Van Hulle, M.M. (2006) Speeding Up the Wrapper Feature Subset Selection in Regression by Mutual Information Relevance and Redundancy Analysis. 16*th International Conference*: *Artificial Neural Networks*. Athens, 10-14 September 2006.

[10] Krier, C., Francois, D., Rossi, F. and Verleysen, M. (2007) Feature Clustering and Mutual Information for the Selection of Variables in Spectral Data. *Proceedings of European Symposium on Artificial Neural Networks*, Bruges, 25-27 April 2007, 157-162.

[11] Zhao, Z. and Liu, H. (2013) Searching for Interacting Features in Subset Selection. *Intelligent Data Analysis*, **13**, 207-228.

[12] Wang, G., Song, Q., Xu, B. and Zhou, Y. (2013) Selecting Feature Subset for High Dimension Data via the Propositional FOIL Rules. *Pattern Recognition*, **46**, 199-214. https://doi.org/10.1016/j.patcog.2012.07.028

[13] Jakulin, A. and Bratko, I. (2004) Testing the Significance of Attribute Interactions. *Proceedings of the Twenty-First International Conference on Machine Learning*, Banff, 4-8 July 2004, 409-416. https://doi.org/10.1145/1015330.1015377

[14] Jakulin, A. (2003) Attribute Interactions in Machine Learning. Master Thesis, University of Liubljana, Kongresni.

[15] John, G.H., Kohavi, R. and Fleger, K.P. (2012) Irrelevant Features and the Subset Selection Problem. *Proceedings of the Eleventh International Conference on Machine Learning*, Boca Raton, 12-15 December 2012, 121-129.

[16] Prim, R.C. (1957) Shortest Connection Networks and Some Generalizations. *Bell System Technical Journal*, **36**, 1389-1401. https://doi.org/10.1002/j.1538-7305.1957.tb01515.x