一种基于树型贝叶斯网络的集成多标记分类算法

张志东 王志海 刘海洋 孙艳歌
(北京交通大学计算机与信息技术学院 北京 100044)

摘 要 在多标记分类问题中,有效地利用标记间的依赖关系是进一步提升分类器性能的主要途径之一。基于分类器链算法,利用互信息度量理论构造分类对象的类属性之间明确的多标记关系依赖模型,并依据建立的标记依赖模型将分类器链中的线性依赖拓展成树型依赖,以适应更为复杂的标记依赖关系;同时,在此基础上利用Stacking集成学习方法建立最终训练模型,提出了一种新的针对树型依赖表示模型的Stacking算法。在多个实验数据集上的实验结果表明,与原有的Stacking集成学习相比,该算法提升了分类器的相应评价指标。

关键词 多标记分类,标记依赖,Stacking,树型贝叶斯网络

中图法分类号 TP181 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.03.030

Ensemble Multi-label Classification Algorithm Based on Tree-Bayesian Network

ZHANG Zhi-dong WANG Zhi-hai LIU Hai-yang SUN Yan-ge
(School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China)

Abstract The performance of learning algorithm can be improved by utilizing existing label dependencies in multi-label classification. Based on the strategy of classifier chain and stacking ensemble learning, this paper built a model to explain the dependency of different labels, and extended the linear dependency into tree dependency to deal with much more complicated label relations. Compared with the original Stacking algorithm, the performance of the proposed algorithm is improved in the experiments.

Keywords Multilabel classification, Label dependency, Stacking, Tree-Bayesian network

1 引言

在传统的有监督分类问题中处理的一般都是单标记的数据集,$X^d \subseteq R$ 是数据集特征向量空间。在单标记分类中,每个实例由一个 d 维向量表示,$x=[x_1, x_2, \cdots, x_d]$,其属性只有一个,因此单标记分类问题就是确定待训练实例标记属性的取值。随着大数据技术的兴起以及海量数据存储方案的逐渐成熟,在目前比较流行的领域中出现了很多标记分类任务。

在多标记分类[1-2]这个机器学习问题中,分类器对于每个实例赋予一组标记,而不像传统的单标记分类那样只对单个属性进行预测。对常见的多标记分类任务的描述如下:在自然语言处理中,给文章分类或者指派关键词,以及在搜索引擎中对抓取到的文本进行分类并建立索引,其核心都是对文本等非结构化数据进行多标记分类[3];社交网络中的人物、活动等实体也需要进行多标记分类以确定其分类[4];图像或视频领域中的动作或目标的识别[5]也需要进行多标记分类。这些多标记分类任务的标记之间都是有关系的,我们并不能直接把它们当作多个单标记分类任务,因此传统的分类方法并不能直接用于多标记分类。

目前,经典的多标记分类方法主要分为两大类:算法适应和问题转化。算法适应主要包含懒惰式学习(如ML-kNN[6])、决策树(如ML-DT[7])、核学习方法(如Rank-SVM[8])以及基于信息论的学习方法(如CML[9])。常用的问题转化方法包括将多标记问题转化为多个单标记分类问题或者标记评价[10]等。问题转化包括BR转化和LP转化。Alavares等人提出了一种基于BR转换的方法BR$^+$[11]。Tsoumakas等在LP转化的基础上找出了随机K标记集RAKEL算法[12]。Read等人提出了一种基于标记相关的分类器链算法[13]。Tsoumakas等人将基于 ϕ 相关系数的剪枝技术应用于Stacking分类中[14]。还有算法分析了标记之间的依赖关系,标记间的依赖关系主要包括条件依赖[15]和非条件依赖[16]。另外有算法探索了不同标记子集之间的关系[17]。有些方法基于标记对(Pairwise)对标记属性之间的相关性及非相关性进行评分[9]。付彬等人提出了一种树型依赖结构的多标记分类算法TCC[18]。

到稿日期:2016-12-03 返修日期:2017-04-20 本文受国家自然科学基金(61672086),北京市自然科学基金(4182052)资助。

张志东(1992-),男,硕士生,主要研究领域为数据挖掘和机器学习,E-mail:15120471@bjtu.edu.cn;**王志海**(1963-),男,博士,教授,博士生导师,CCF会员,主要研究领域为数据挖掘和机器学习,E-mail:zhhwang@bjtu.edu.cn(通信作者);**刘海洋**(1987-),男,博士生,主要研究领域为数据挖掘和机器学习,E-mail:11112096@bjtu.edu.cn;**孙艳歌**(1982-),女,博士生,讲师,主要研究领域为数据挖掘和机器学习,E-mail:13112074@bjtu.edu.cn。

Stacking 也属于问题转化中的一种，与 Stacking 方法相比，BR$^+$算法对于类标记的预测顺序并没有明确规定，仅仅是将其他标记当作普通属性对待，而在 TCC 中建立了多个分类器链以应对不同的情况，但是增加了算法的时间复杂度。本文设计并实现了一种基于树型贝叶斯网络的 Stacking 多标记分类算法，主要贡献如下：

1)在建立基层分类器的过程中，对多个类标间的依赖关系建模，明确类标预测顺序，使得算法更具可解释性。

2) 在确定依赖关系的过程中，减少了无关属性对于当前标记分类效果及性能的影响。

3) 相对于 TCC 中建立多棵树的方法，本算法只建立一棵树，有效地降低了标记数较大的情况下的算法复杂度。

本文第 2 节介绍了多标记分类的研究背景及现状；第 3 节对本文提出的基于树型贝叶斯网络的分类器算法(TreeSta-cking)进行了详细描述；第 4 节分析了实验结果；最后总结全文并对未来工作进行展望。

2 研究背景

在详细叙述本文工作之前，本节首先对多标记分类中的主流方法以及一些常用的术语进行说明。在明确多标记分类任务之后，本文主要讨论多标记分类方法中的问题转化算法，主要包括 BR 转化和 LP 转化、分类器链和 Stacking。

2.1 多标记分类方法

在多标记分类中，标记集合是数据集所有标记属性的集合，用 $\boldsymbol{L}$ 表示，且 $\boldsymbol{L}=\{l_1,l_2,\cdots,l_m\}$；$\boldsymbol{Y}$ 表示数据实例的实际标记集合；训练集用 $\boldsymbol{D}$ 表示，且 $\boldsymbol{D}=\{(\boldsymbol{x}_1,\boldsymbol{y}_1),(\boldsymbol{x}_2,\boldsymbol{y}_2),\cdots,(\boldsymbol{x}_n,\boldsymbol{y}_n)\}$，$\boldsymbol{x}$ 是实例的特征空间；$\boldsymbol{y}$ 是对应实例的标记向量，并且 $\boldsymbol{y}\subseteq\boldsymbol{Y},\boldsymbol{Y}\subseteq\boldsymbol{L},\boldsymbol{D}$ 中的每个元素都是独立且随机的。对于一个分类器输入 $\boldsymbol{D}$，建立相应的模型 h；然后对于输入的待分类实例 $\boldsymbol{x}$，分类器 h 输出 $\boldsymbol{x}$ 的标记集合 $\boldsymbol{y}$。方法转化就是在此分类任务上针对某个环节进行转化。

BR 转化(Binary Relevance，BR)是最直接和最简单的处理多标记的分类问题的方法。在 BR 中，将一个有 m 个标记的实例转换成 m 个二值分类问题，即建立 m 个分类器 $h_1,h_2,h_3,\cdots,h_m$，并且使用分类器 h_j 对第 j 个标记进行分类，预测过程中实例的每个类标由训练期间建立的分类器分别进行处理，最后得到最终结果。

标记幂集转化(Label PowerSet，LP)是转化策略中的另外一种常用的思想。BR 转化忽略了标记之间可能存在的依赖，而 LP 考虑了标记之间的依赖。它的思想是将全部的标记集合转换为单标记分类问题。因此，新标记的取值集合代表原来多标记分类结果的取值空间。代价就是在形成的单标记实例中标记取值的个数，标记数量呈指数增长，在标记数量超过一定数值后，这种简单方案变得不可行。

分类器链(Classifier Chain，CC)是多标记分类中基于 BR 转化的另外一种处理方式。在分类器链中，按照一定的搜索方式确定标记空间中的标记处理顺序。在训练时，初始的特征空间即为实例特征空间，当某个标记被处理之后，CC 使用它来拓展原始特征空间，重复以上步骤，最后得到所有标记的处理结果。

Stacking 是一种集成的训练方法，可以在 BR 或者 LP 转化的基础上进行集成，这种方法广泛用于传统分类以及多标记分类，具体做法是训练一个模型用于组合其他各个模型，即首先训练多个不同的模型，然后再以训练的各个模型的输出为输入来训练一个模型，从而得到最终输出。Tsoumakas 等人提出了一种多标记的 Stacking 分类方法[14]。在训练过程中 Stacking 算法会建立两组分类器：1)基分类器；2)做元分类器。基层的分类器由 BR 方法或其变种方法转换成二值分类器。

$$h_{\text{base}}(x)=(h_1(x),h_2(x),\cdots,h_m(x)) \tag{1}$$

其中，h_{base} 代表基层分类器，h_i 是对于第 i 个标记建立的基分类器。

元层的分类器同样也是二值分类器，不同的是它们的输入实例是在特征空间拓展之后得到的。

$$h_{\text{meta}}(x,\hat{y})=(h_1'(x,\hat{y}),h_2'(x,\hat{y}),\cdots,h_m'(x,\hat{y})) \tag{2}$$

其中，h'代表使用基层分类结果来拓展原始特征空间之后建立的分类器。

从本质上说，Stacking 通过拓展原始特征空间方法来考虑标记之间的依赖。在预测时，最终的结果是元层输出的结果。

2.2 互信息

信息论中的信息熵用来度量信息的多少，由于数据集中变量的出现服从一定分布，因此可以根据信息熵的多少来衡量不确定性，从而来解决多标记中的依赖问题。

假设 $\boldsymbol{X}$ 和 $\boldsymbol{Y}$ 是两个随机变量，$\boldsymbol{P}(\cdot)$ 表示随机变量出现的概率，对于 $\boldsymbol{X}$ 与 $\boldsymbol{Y}$，有如下定义。

对于任意一个随机变量 $\boldsymbol{X}$，它的熵的定义如下：

$$H(\boldsymbol{X})=\sum_{x\in\boldsymbol{X}}P(x)\cdot\log P(x) \tag{3}$$

其中，H 表示随机变量 $\boldsymbol{X}$ 所含有的信息熵，用来表示随机变量 $\boldsymbol{X}$ 的不确定性。

如果知道 $\boldsymbol{Y}$ 随机变量与 $\boldsymbol{X}$ 随机变量出现的联合概率分布，以及 $\boldsymbol{Y}$ 在不同条件下的 $\boldsymbol{X}$ 的分布概率 $P(\boldsymbol{X}|\boldsymbol{Y})$，则定义 $\boldsymbol{X}$ 在 $\boldsymbol{Y}$ 的条件下的条件熵为：

$$H(\boldsymbol{X}|\boldsymbol{Y})=-\sum_{x\in\boldsymbol{X},y\in\boldsymbol{Y}}P(x,y)\cdot\log P(x|y) \tag{4}$$

使用互信息来量化两个随机事件的相关性：

$$I(\boldsymbol{X};\boldsymbol{Y})=\sum_{x\in X,y\in Y}P(x,y)\cdot\log\frac{P(x,y)}{P(x)P(y)} \tag{5}$$

其中，$I(\boldsymbol{X};\boldsymbol{Y})$表示 $\boldsymbol{X}$ 和 $\boldsymbol{Y}$ 这两个随机变量之间的互信息的值，表示两个随机变量中已知其中一个信息熵之后另外一个减少的不确定度。

互信息 $I(\boldsymbol{X};\boldsymbol{Y})$其实是随机事件 $\boldsymbol{X}$ 的不确定性与随机事件在 $\boldsymbol{Y}$ 条件下的不确定性的差异：

$$I(\boldsymbol{X};\boldsymbol{Y})=H(\boldsymbol{X})-H(\boldsymbol{X}|\boldsymbol{Y}) \tag{6}$$

因此两个随机变量的互信息，就是在 $\boldsymbol{Y}$ 的条件下对消除另外一个随机变量 $\boldsymbol{X}$ 的不确定度的度量，并且 $I(\boldsymbol{X};\boldsymbol{Y})$满足以下性质：

$$0\leqslant I(\boldsymbol{X};\boldsymbol{Y})\leqslant\min(H(\boldsymbol{X}),H(\boldsymbol{Y})) \tag{7}$$

3 基于树型贝叶斯网络的 Stacking 分类算法

大多数学习和利用标记间的依赖关系的方法主要包括两个步骤：1)为每个标记确定其可能依赖的若干其他标记；2)基于

BR 转换，为每个标记构建相应的训练集。其中步骤 1）是研究的关键。

基本的 BR 转化忽略了标记间的依赖关系，并且标记数目较大时可能会导致数据倾斜的问题。利用标记之间的依赖关系有助于提高分类的准确度，例如：在对新闻数据进行分类时，若一个新闻属于娱乐类别，则它属于明星类别的概率会远远大于属于政治类别的概率。虽然 LP 转化方法考虑到了标记间的相互关系，在某些情况下效果较好，但其计算时间复杂度较高。分类器链简单且有效，但是从本质上看分类器链的处理顺序是随机选取的[19]，某些依赖较弱的标记的加入反而可能会降低效果；另外，标记间的依赖并不一定是线性的，也可能存在更为复杂的依赖关系。而拓展的贝叶斯网络（TAN）可以用来表示标记属性之间的依赖关系。BR^- 算法在训练期间对一个标记进行预测时将其他标记添加到特征空间中，并且随着标记预测的进行，已添加到特征空间的标记会被动态更新。该算法本身并没有发现依赖关系的机制，只是通过不同的类属性来拓展原始特征属性空间，让分类器自适应地发现标记之间的依赖，这种做法存在一定的盲目性。这是因为单纯地将预测结果加入到基分类器中，没有考虑标记之间的依赖关系，也没有考虑拓展之后的属性空间对于建立分类器是否合适。基于以上分析，本文提出了 TreeStacking 算法。TreeStacking 算法主要解决了两个问题：1）确定了每个类标可能依赖的其他的若干标记；2）确定了每个类标的预测顺序，并根据这些信息为每个标记构建相应的训练集。

3.1　TreeStacking 中的标记依赖模型

分类器链算法的本质是为每个标记随机标定其依赖标记集合，并非通过度量依赖程度的大小来标定，标记之间可能并不存在较强的依赖关系。因此，为了表示多个特征和标记之间的依赖关系，Bielza 等人提出了多维贝叶斯网络模型（Multi-dimensional Bayesian Network）[19]，但是这种没有限制的贝叶斯网络模型是一个 NP-Hard 问题，复杂性极高。付彬等人也提出了一种基于随机游走策略的类标依赖方法[20]，但是其迭代过程是基于图的，成本相对较高。另外，多个标记间也可能存在着相互依赖关系，因此这种单向线性的依赖并不能正确地表示真实的依赖关系[22]。TAN 算法放宽了这种要求，其并不要求每个属性都相互独立，每个属性可以依赖多个其他属性。TAN 基于信息熵公式对各个属性进行度量，类似地，De Campos 等人基于贝叶斯网络，结合信息熵及卡方统计量提出了一个新的评价函数[21]，但是它的计算方式成本较高。因此，本文将 TAN 算法形成的树状贝叶斯网络应用于 Stacking。本文使用式（5）来衡量两个类标记之间的相关性，对于 $\boldsymbol{X}$ 和 $\boldsymbol{Y}$ 两个类标记，x 和 y 分别表示 $\boldsymbol{X}$ 和 $\boldsymbol{Y}$ 标记的所有的取值，$P(\cdot)$ 表示某个取值出现的概率，$I(\boldsymbol{X};\boldsymbol{Y})$ 表示 $\boldsymbol{X}$ 和 $\boldsymbol{Y}$ 这两个标记之间的互信息，表示在两个随机变量中已知其中一个属性条件之后对另外一个减少的不确定度。互信息越大，表明在已知一个标记的情况下，对另外一个标记不确定度性的减少量。

假设一个数据集有 m 个标记，则互信息的结果可以保存到一个 $m*m$ 的矩阵中，并且这个矩阵是对称的。随机挑选一个标记为根节点，那么根据这个节点就可以利用 Prim 算法求得最大生成树，其中树的节点为训练实例的每个属性，假如一个节点 n 的父结点用 p 表示，n 与 p 的相关性相对于 n 与其他属性间的相关性更大。

如果选择不同的节点，则可以生成不同的树。若有 m 个类属性，则有 m 棵树。图 1 中共有 5 个节点，则共有 5 种生成树，每个节点都可作为根节点，选中根节点之后，使用方向来表示依赖关系，图 2(a)中 y_1 与 y_2 的关系表示 y_2 属性依赖于 y_1 属性。因此以每个节点为根都可以形成一棵树，图 2 示出了可能的 3 种树。

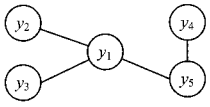

图 1　多个类标间可能的依赖关系对应的最大生成树

Fig. 1　Maximum spinning tree of possible dependenly relation between class attributes

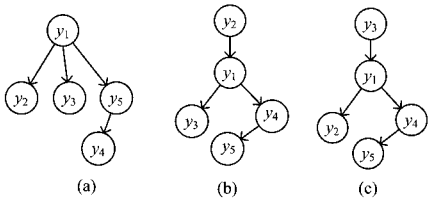

图 2　属性之间可能的依赖关系树

Fig. 2　Possible dependency tree among different attributes

由图 2 可知，选择不同的决策树时其中标记依赖的情况也是不同的。在得到最大生成树之后，由信息熵的定义可知，互信息量最小的标记相对于其他的标记较为独立，因此选择互信息量最小的两个节点中的任意一个作为根节点。图 2(a)所示的生成树应当先对标记 y_1 进行处理，而图 2(b)所示的生成树应先对 y_2 进行处理，同理，图 2(c)所示的生成树应该先对 y_3 进行处理。当确定标记之间的依赖关系之后，下一步的工作就是根据树确定的顺序进行处理，以图 2(c)为例来说明基层的预测算法。根据树的结构可以得到树的拓扑排序：$y_3 \rightarrow y_1 \rightarrow y_2 \rightarrow y_4 \rightarrow y_5$（其中 y_2 与 y_4 的顺序没有要求，取决于算法实现），根据这个顺序，在对标记进行预测时，应当先预测 y_3，在预测完 y_3 后获取 y_3 的预测结果，将 y_3 的预测结果添加到原来的特征空间。接下来预测 y_1，得到 y_1 的预测结果，并将其同样添加到特征空间。当预测到 y_2 时，这一个结点有两个父节点，一个是 y_1（称为 y_2 的直接父结点），另外一个是 y_3（称为 y_5 的祖先节点），这时有两个策略可以选择：

1）对某非根节点进行预测时，将该节点所有的父节点的预测值加入到特征空间中，称为 All Pa 策略（见图 3）；

2）对某非根节点进行预测时，只将该节点的直接父结点的预测值加入到特征空间中，称为 One Pa 策略（见图 4）。

由 All Pa 与 One Pa 两种策略可以看出，实例的特征空间是不断扩大的，最大扩展空间取决于形成的树的结构，最为极端的两种情况有两种：一种就像 Naïve Bayes 那种，每个节点只依赖于一个标记；另外一种情况是像分类器链那样，树型贝叶斯网络退化成线性依赖关系。

通过这样的方式就避免了将所有的标记加入到特征集而

增加计算复杂度的问题，又通过利用信息熵理论而考虑了标记间的依赖。如果只加入直接节点，那么算法所花费的时间复杂度较小，而如果加入所有结点，那么算法的时间复杂度较高，这取决于最终树的深度，在实际情况中，需要根据实际情况来衡量使用的策略。

在训练数据集时，当形成元层的训练实例(如图 3 或图 4 所示)之后，即在预测或者训练阶段形成元层训练实例之后，为防止树型网络退化为线性网络而造成的性能下降，可以根据实际情况的需要对属性或者标记进行剪枝操作，剪枝操作有不同的策略，总体来说有以下 4 种：

1)只对特征属性剪枝，类标记使用真实值；

2)只对特征属性剪枝，类标记使用预测值；

3)对全部属性剪枝，使用真实的标记取值；

4)对全部属性剪枝，使用预测的标记取值。

第 1 种情况即对形成的实例的原始特征空间进行剪枝，(扩展之后的)标记属性保持不变；第 2 种情况与第 1 种情况类似，只是它的标记值是真实的原始输入数据，而不是用分类器预测出来的值。第 3 种情况和第 4 种情况就是将扩展之后的标记视为原始属性，并与原始属性一起进行剪枝操作。当然也可以对标记和属性分别进行剪枝，然后将结果合并为新的实例，这样可以分别使标记与原来的特征空间分别保持一定的比例，更为灵活。

本文的 TreeStacking 根据不同的输入参数可以选择 2)和 4)两种策略。其本质是通过提高基层的分类器输出的可信度，来优化元层实例的构造过程。在使用元层分类器预测时，在理论上基层和元层分类器的建立过程都不同程度地考虑了标记之间的依赖关系，使得评价指标有所改善。

综上所述，根据树型贝叶斯网络可以确定类标之间依赖的拓扑顺序，其过程如算法 1 所示。

算法 1 树型贝叶斯网络类标依赖拓扑排序算法

输入：数据集 Insts

输出：类标依赖关系数组 dependency

1. 初始化 LabelSet = Insts 类标集合；
2. for i in LabelSet do
3. begin
4. for j in LabelSet do
5. 根据式(5)计算类标 i 和 j 的互信息 Info(i,j)
6. End
7. 利用 Prim 算法找出 Info 表示的最大生成树 G；
8. 任意选择一个结点为根结点，由 G 得到 Info 对应的有向树 Tree；
9. 对于 Tree 拓扑排序，确定类标依赖数组 dependency；
10. 返回 dependency.

3.2 TreeStacking 训练及预测

在 Stacking 的基层中，训练分类器时先对实例进行转换。原始的转换方法只是简单地使用 BR 转化，修改之后，在转换时需根据上述算法返回的 *dependency* 数组来确定待预测标记的顺序。如图 3 所示，假如最大生成树是[-1,2,5,4,2,0,1]，则说明第 0 个标记是根节点、第 5 个标记依赖第 0 个节点、第 2 个节点依赖第 5 个节点等，因此生成的 *dependency* 数组应该是[0,5,2,1,4,6,3]。首先预测索引为 0 的节点，当预测节点是树中的根节点时，使用原始特征空间直接进行预测，预测完成之后，将预测的结果添加到原始的特征空间中以拓展原始的特征空间。然后预测索引为 5 的标记，并根据生成树找到第 5 个节点的父节点(此时，父节点一定被预测过)，将父节点的预测结果添加到原始属性空间中，然后得到索引为 5 的标记的预测结果。重复以上步骤，直到所有的标记都预测完成。

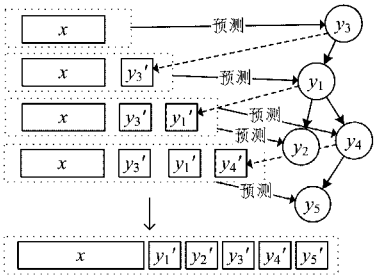

图 3 Tree tacking(All Pa)元层实例的形成

Fig. 3 Formation of meta instances in meta level(All Pa)

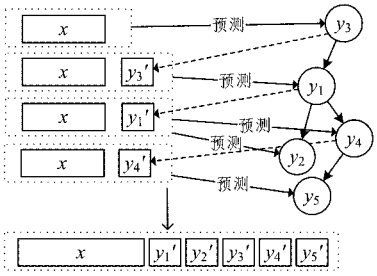

图 4 树型 Stacking(One Pa)元层实例的形成

Fig. 4 Formation of meta instances in meta level(One Pa)

TreeStacking 训练时的算法伪代码如算法 2 所示。

算法 2 TreeStacking 分类器训练过程

输入：待训练实例集 Insts

输出：TreeStacking 分类器 H

1. 由算法 1 得到类标依赖关系的拓扑排序 dependency；
2. 初始化 LabelSet=Insts 类标集合；
3. for l in LabelSet do
4. Begin
5. 将 l 视为类标，把 Insts 转换为单类标数据集 D_l，进行交叉验证得到 D_l 的分类结果 $Prob_l$ 及基层分类器 B_l；
6. 以上 $Prob_l$ 根据 dependency 形成元层数据集 $meta_l$；
7. 利用 $meta_l$ 数据集建立元层的分类器 M_l；
8. End
9. 输出建立的 TreeStacking 模型 H.

使用分类器进行预测时，首先也是将待预测实例进行转换。按照 *dependency* 数组指定的顺序进行预测，由前文的叙述并参照训练过程，不难总结出分类过程算法。在分类过程中，同样需要对实例进行相应的转化，TreeStacking 分类器对一个实例进行分类的伪代码如算法 3 所示。

算法 3 TreeStacking 分类器的分类过程

输入：待分类实例 Inst

输出：该实例的概率分布 Prob

1. 初始化 LabelSet = Inst 类标集合；
2. for l in LabelSet do
3. Begin

4.　将 l 视为类标，找到 l 依赖类标 Pa(l)，使用基层分类器预测并将预测结果添加到特征空间(元层实例 meta)；

5. End

6. 使用元层分类器对 meta 进行分类，最后得到最终结果 Prob；

7. 返回最终的概率分布数组 Prob.

4　实验

本节主要描述实验平台和实验方法，并对实验结果进行分析与总结；同时，验证 TreeStacking 算法，将其与原来基于 BR 转换方法的 Stacking 算法在不同的指标上进行了对比分析。

4.1　实验环境

本实验是基于 Weka 平台[1] 以及 Mulan 多标记学习框架[2] 的。实验程序是在 CPU 为 i7-6700 3.40GHz、内存为 8 GB、操作系统为 Windows 7 的 PC 机上进行的。

4.2　实验指标

为了描述分类器指标，首先给出将要用到的数学符号，设 $\boldsymbol{T}=\{(x_1,C_1),(x_2,C_2),(x_3,C_3),\cdots,(x_N,C_n)\}$ 是一个包含 n 个测试实例的测试集，Λ 代表数据集的标记集合，m 代表标记空间大小。x_k 代表第 k 个实例，$C_k\subseteq\Lambda$ 表示实例 x_k 的真实标记集合。对于一个给定的分类器 h 和测试实例 x_k，分类器 h 的输出为 Y_k，$Y_k\subseteq\Lambda$ 表示分类器对其预测的结果。由以上定义，测试中的指标可以表示如下。

1)汉明损失。该指标用于统计分类器在所有实例上被错分的标记个数的均值，其定义如式(8)所示：

$$HanmingLoss(h,T)=\frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i\oplus C_i}{m}\right) \tag{8}$$

2)子集正确率。子集正确率用于比较实例的真实标记空间和预测标记空间，只有两个集合完全相等才认为分类正确，否则认为分类错误，这个指标表示被完全分类正确的实例的比例，其定义如式(9)所示：

$$SubsetAccuracy(h,T)=\frac{1}{n}\sum_{i=1}^{n}I(\boldsymbol{c}_i=\hat{\boldsymbol{c}}_i) \tag{9}$$

3)召回率。召回率表示每个测试实例真实标记空间和相应预测标记空间交集的大小与真实标记空间大小的比值，并在所有预测实例上求均值，其定义如式(10)所示：

$$Recall(h,T)=\frac{1}{n}\sum_{i=1}^{n}\frac{|\boldsymbol{y}_i\cap\hat{\boldsymbol{y}}_i|}{|\boldsymbol{y}_i|} \tag{10}$$

4)精确率。精确率表示每个测试实例的真实标记空间和相应的预测标记空间交集的大小与预测标记空间大小的比值，其定义如式(11)所示：

$$Precison(h,T)=\frac{1}{n}\sum_{i=1}^{n}\frac{|\boldsymbol{y}_i\cap\hat{\boldsymbol{y}}_i|}{|\hat{\boldsymbol{y}}_i|} \tag{11}$$

5)F1 测量。精确率和召回率相互制约，为了平衡这两个指标，以便能更准确地评价分类器性能，从而提出 $F1$ 测试值，其定义如式(12)所示：

$$F1(h,T)=\frac{2\cdot Recall(h,T)\cdot Precision(h,T)}{Recall(h,T)+Precision(h,T)} \tag{12}$$

6)准确率。准确率表示每个真实标记空间和预测标记空间的交集的大小与真实标记空间和预测标记集的并集大小的比值，并取求所有的实例的均值，其定义如式(13)所示：

$$Accuracy(f,T)=\frac{1}{n}\sum_{i=1}^{n}\frac{|\boldsymbol{y}_i\cap\hat{\boldsymbol{y}}_i|}{\boldsymbol{y}_i\cup\hat{\boldsymbol{y}}_i} \tag{13}$$

4.3　实验数据集

本实验采用的 9 个数据集及其详细信息如表 1 所列。

表 1　实验数据集描述

Table 1　Description of data sets used in experiments

数据集	领域	实例数	特征数	标记数
birds	视频	645	260	19
CAL500	音乐	502	68	174
emotions	音乐	593	72	6
flags	图像	194	19	7
genbase	生物	1876	1186	27
medical	医疗	978	1449	45
yeast	生物	2417	103	14
enron	文本	1702	1001	53
bibtex	文本	7359	1836	159

4.4　实验设计

在实验中，对 Stacking 方法中的元层属性数据进行剪枝，为了保留不同的比率属性对分类精度的影响，选择不同 P 值在不同数据集上进行实验。根据表 2 的结果，考虑到有些数据集的标记数量过少，遵循在损失相当的情况下选择较小 P 值的原则，选择的 P 值为 0.7。

表 2　不同 P 值下汉明损失的变化

Table 2　Changing of HanmingLoss under different P values

数据集	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Birds	0.0533	0.0529	0.0529	0.0526	0.0529	0.0531	0.0538	0.0539	0.0539
CAL500	0.1782	0.1762	0.1737	0.1744	0.1722	0.1718	0.1719	0.1701	0.1697
genbase	0.0011	0.0011	0.0011	0.0011	0.0011	0.0011	0.0011	0.0011	0.0011
Medical	0.0104	0.0103	0.0104	0.0104	0.0104	0.0104	0.0104	0.0104	0.0104
yeast	0.2591	0.2620	0.2644	0.2677	0.2711	0.2703	0.2702	0.2710	0.2702
enron	0.0575	0.0573	0.0573	0.0574	0.0574	0.0574	0.0573	0.0572	0.0573

对类属性保留 0.7，即保留原来属性的 70%，基分类器与元分类器均使用默认的 J48 分类器，实验测试了 birds 等 9 个数据集。实验结果的评价方式主要分为两种：基于实例的评价方式和基于标记的评价方法。其中，基于实例的评价指标主要包括子集精确度、汉明损失；基于标记的评价指标主要包括精确度、准确度、F1 值、召回率。

[1] http://www.cs.waikato.ac.nz/ml/weka/

[2] http://mulan.sourceforge.net/

由表 3 可以看出，TreeStacking 算法与原来的 Stacking 算法相比，其汉明缺失并没有明显下降，在 birds，flags 和 medical 数据集中的表现比原来更好；除了 emotions 数据集，其在其他数据集上的分类效果并没有显著的区别。在大多数数据集上，使用所有的父节点与只使用一个节点相比并没有显著的差别。

表 3 不同数据集中的汉明损失

Table 3 HammingLoss in different datasets

数据集	MultiLabel Stacking	TreeStacking (All Pa)	TreeStacking (One Pa)
birds	0.0538	0.0533	**0.0531**
CAL500	**0.1546**	0.1624	0.1560
emotions	**0.2589**	0.2960	0.2884
flags	0.3045	0.3233	**0.3029**
genbase	**0.0020**	0.0025	0.0025
medical	0.0251	**0.0249**	0.0251
yeast	**0.2237**	0.2686	0.2594
enron	**0.0572**	0.0604	0.0608
bibtex	**0.0141**	0.0144	0.0143

由表 4 可以看出，与原来相比，TreeStacking 算法在数据集(如 emotions，medicial，enron 等)上的子集精确度都有提升；对于数据集 birds，flags，genbase，yeast 和 bibtex，其子集精度表现出了与原来相当的水平。

表 4 不同数据集中的子集精确度

Table 4 Subset accuracy in different datasets

数据集	MultiLabel Stacking	TreeStacking (All Pa)	TreeStacking (One Pa)
birds	**0.4656**	0.4625	0.4625
CAL500	0.0000	0.0000	0.0000
emotions	0.1398	**0.1517**	0.1468
flags	**0.0982**	0.0818	0.0921
genbase	**0.9577**	0.9472	0.9472
medical	0.1012	**0.1084**	0.1033
yeast	**0.0579**	0.0199	0.0302
enron	0.0118	0.0200	**0.0235**
bibtex	**0.0569**	0.0505	0.0497

由表 5 可以看出，与 Stacking 相比，TreeStacking 在大多数的数据集上的召回率都有改善。

表 5 不同数据集中的召回率

Table 5 Recall in different datasets

数据集	MultiLabel Stacking	TreeStacking (All Pa)	TreeStacking (One Pa)
birds	**0.4980**	0.4943	0.4927
CAL500	0.2580	**0.2708**	0.2381
emotions	0.3590	**0.4017**	0.3998
flags	0.5939	0.5950	**0.6019**
genbase	**0.9769**	0.9731	0.9731
medical	0.1246	**0.1307**	0.1251
yeast	**0.4701**	0.3663	0.3688
enron	0.1856	**0.1954**	0.1883
bibtex	**0.1376**	0.1183	0.1189

由表 6 可以看出，TreeStacking 算法的 F 值在大多数数据集上与原始的 Stacking 算法相当，在 emotions 与 enron 上其 F 值好于原始算法。

表 6 不同数据集中的 F-Measure

Table 6 F-Measure in different datasets

数据集	MultiLabel Stacking	TreeStacking (All Pa)	TreeStacking (One Pa)
birds	**0.5061**	0.5037	0.5016
CAL500	**0.3274**	0.3260	0.3077
emotions	0.3703	0.3867	**0.3878**
flags	**0.6386**	0.6205	0.6380
genbase	**0.9794**	0.9771	0.9771
medical	0.1281	**0.1346**	0.1295
yeast	**0.5375**	0.4221	0.4295
enron	0.2253	**0.2281**	0.2201
bibtex	**0.1640**	0.1402	0.1406

由表 7 可以看出，与原始算法相比，TreeStacking 算法在过半的数据集上的 F-Measure 有所提升；对于 One Pa 与 All Pa 两种策略，除在 CAL 数据集中的结果相差比较大外，在其他数据集上两种策略的效果相当。

表 7 不同数据集中的准确率

Table 7 Accuracy in different datasets

数据集	MultiLabel Stacking	TreeStacking (All Pa)	TreeStacking (One Pa)
birds	**0.4943**	0.4915	0.4900
CAL500	0.1997	**0.2008**	0.1868
emotions	0.3134	**0.3256**	0.3250
flags	0.5091	0.4915	**0.5092**
genbase	**0.9744**	0.9708	0.9708
medical	0.1213	**0.1280**	0.1229
yeast	**0.4172**	0.3040	0.3131
enron	0.1644	**0.1663**	0.1611
bibtex	**0.1311**	0.1128	0.1128

表 8 不同数据集中的平均精度

Table 8 Average-precision in different datasets

数据集	MultiLabel Stacking	TreeStacking (All Pa)	TreeStacking (One Pa)
birds	0.4383	0.4466	**0.4486**
CAL500	**0.3997**	0.3623	0.3839
emotions	**0.6883**	0.6458	0.6585
flags	**0.8006**	0.7816	0.7820
genbase	**0.9856**	0.9840	0.9840
medical	0.4975	**0.5029**	0.4987
yeast	**0.7156**	0.6162	0.6275
enron	**0.5370**	0.5153	0.5166
bibtex	**0.2624**	0.2417	0.2405

通过上面的结果可以看到，TreeStacking 依据标记之间的依赖关系形成生成树，进而根据树的拓扑排序来确定预测顺序，并通过实验得出以下结论：结合表 3 和表 5 来看，本算法与之前相比提高了召回率，并且其在汉明损失和精确度两个指标方面与原算法相当，而在有些数据集中，其汉明损失指标相比以前变低，并且在确定标记关系时并没有提高算法复杂度，只是通过频率估计概率这种方式得到了标记的分布，然后利用式(5)计算出了各个标记之间的互信息度量。表 9 列出了各分类器测试数据集的耗时，在 ML Stacking 与 TreeStacking 中 One Pa 策略相差并不大，基本持平甚至在一些数据集中后者优于前者；而对于 All Pa 策略来说，其所用时间基本多于其他两种策略，这主要是受元层训练集的属性数目影响。而且对于一个数据集来说，整个过程只进行一次，而此算法的复杂度相对于标记个数 n 来说是 $O(n)$，而 Prim 算法的复杂度是 $O(n^2)$，这些时间成本相对于算法本身而言可以忽略不计。

表 9　TreeStacking 的两种策略与 ML Stacking 策略的效率对比

Table 9　Comparasion of efficiency between TreeStacking and ML Stacking

（单位：s）

数据集	MultiLabel Stacking	TreeStacking (All Pa)	TreeStacking (One Pa)
birds	53.489	67.766	58.174
CAL500	356.293	591.465	369.908
emotions	30.065	33.316	29.985
flags	1.092	1.111	1.133
genbase	38.003	39.470	38.671
medical	643.642	716.986	673.233
yeast	651.425	524.803	471.710
enron	10870.297	7750.272	7455.359
bibtex	34570.297	33937.206	32173.933

由表 3－表 8 可以看出，若与自身相比，TreeStacking 算法在标记个数较多（大于 20 个）的数据集上的效果比标记个数较少（少于 20 个）的数据集上的效果好。出现这种情况的原因在于：对于标记较少的数据集，各个标记之间的依赖程度较小，并不适合预测一个标记时将其他标记加入进来以拓展特征集合这种方法。例如：对于 emotions，flags 和 yeast，这 3 个数据集的汉明损失有所上升，且都达到了 10^{-2} 数量级。对照表 1 中的数据集的特点可以看到，这 3 个数据集的共同特点是它们的标记数都较少，分别为 6，7 和 14，而其他数据集大多都在 20 以上，对于标记较少的数据集，分析标记之间的关系反而会使得分类汉明损失增大，例如：emotions 数据集的标记只有 6 个，而且就实际情况来说，人类的表情是比较复杂的，惊讶可能会存在于高兴的情绪之中，也有可能存在于难过之中，因此惊讶情绪对于判断是高兴还是难过并没有太大的帮助，只是会导致单纯地通过树型网络得到依赖关系可能并不符合实际的情况。

结束语　本文分析了多标记分类技术的应用，并对几种经典的分类算法、相关理论基础、研究背景以及它们各自的特点与使用范围进行了说明。针对 BR^{+} 算法基层属性依赖问题，结合 TCC 算法中对标记依赖建模的思想，提出了 TreeStacking 算法，提升了对数据集的分类效果。

TreeStacking 确定树的根节点时，只是简单地选取互信息值最小的结点，这样选取的原因是从理论上看互信息过小，其依赖程度相比其他标记来说较小，因此被优先预测，未来还需要进一步探索是否有更好的确定树根节点的方法。

参 考 文 献

[1] TSOUMAKAS G, KATAKIS I, VLAHAVAS I. Mining Multi-label Data[M]// Data Mining and Knowledge Discovery Handbook. Boston: Springer, 2009: 667-685.

[2] ZHANG M L, ZHOU Z H. A review on multi-label learning algorithms[J]. IEEE Transactions on Knowledge & Data Engineering, 2014, 26(8): 1819-1837.

[3] UEDA N, SAITO K. Parametric mixture model for multitopic-text[J]. Systems and Computers in Japan, 2006, 37(2): 56-66.

[4] TSOURAKAKIS C. Provably fast inference of latent features from networks: with applications to learning social circles and multilabel classification[C]// Proceedings of the 24th International Conference on World Wide Web. ACM, 2015: 1111-1121.

[5] LUO Y, LIU T, TAO D, et al. Multiview matrix completion for multilabel image classification[J]. IEEE Transactions on Image Processing, 2015, 24(8): 2355-2368.

[6] ZHANG M L, ZHOU Z H. ML-KNN: A lazy learning approach to multi-label learning[J]. Pattern Recognition, 2007, 40(7): 2038-2048.

[7] CLARE A, KING R D. Knowledge Discovery in Multi-label Phenotype Data[J]. Lecture Notes in Computer Science, 2002, 2168(2168): 42-53.

[8] ELISSEEFF A E, WESTON J. A Kernel Method for Multi-Labelled Classification[C]// Advances in Neural Information Processing Systems. 2002: 681-687.

[9] GHAMRAWI N, MCCALLUM A. Collective multi-label classification[C]// Proceedings of the 14th ACM International Conference on Information and Knowledge Management. ACM, 2005: 195-200.

[10] RNKRANZ J, LLERMEIER E, LOZAMENC, et al. Multilabel classification via calibrated label ranking[J]. Machine Learning, 2008, 73(2): 133-153.

[11] ALVARES-CHERMAN E, METZ J, MONARD M C. Incorporating label dependency into the binary relevance framework for multi-label classification[J]. Expert Systems with Applications, 2012, 39(2): 1647-1655.

[12] TSOUMAKAS G, VLAHAVAS I. Random k-Labelsets: An Ensemble Method for Multilabel Classification[C]// European Conference on Machine Learning. Springer, Berlin, Heidelberg, 2007: 406-417.

[13] READ J, PFAHRINGER B, HOLMES G, et al. Classifier chains for multi-label classification[J]. Machine Learning, 2011, 85(3): 254-269.

[14] TSOUMAKAS G, DIMOU A, SPYROMITROS E, et al. Correlation-based pruning of stacked binary relevance models for multi-label learning[C]// Proceedings of the 1st International Workshop on Learning from Multi-Label Data. 2009: 101-116.

[15] DEMBCZYŃSKI K, CHENG W, HÜLLERMEIER E. Bayes optimal multilabel classification via Probabilistic Classifier Chains[C]// International Conference on Machine Learning. 2010: 279-286.

[16] ZHANG M L, ZHOU Z H. Multilabel neural networks with applications to functional genomics and text categorization[J]. IEEE Transactions on Knowledge and Data Engineering, 2006, 18(10): 1338-1351.

[17] CHENG W, HÜLLERMEIER E. Combining instance-based learning and log is ticre gression for multilabel classification[J]. Machine Learning, 2009, 76(2-3): 211-225.

[18] ALVARES-CHERMAN E, METZ J, MONARD M C. Incorporating label dependency into the binary relevance framework for multi-label classification[J]. Expert Systems with Applications, 2012, 39(2): 1647-1655.

[19] BIELZA C, LI G, LARRAÑAGA P. Multi-dimensional classification with Bayesian networks[J]. International Journal of Approximate Reasoning, 2011, 52(46): 705-727.

[20] FU B, WANG Z, XU G, et al. Multi-label learning based on iterative label propagation over graph[J]. Pattern Recognition Letters, 2014, 42(1): 85-90.

[21] DE CAMPOS L M. A Scoring Function for Learning Bayesian Networks based on Mutual Information and Conditional Independence Tests[J]. Journal of Machine Learning Research, 2006, 7(7): 2149-2187.

[22] FU B, WANG Z H. Multi-Label Classification Method Based on Tree Structure of Label Dependency[J]. Pattern Recognition and Artificial Intelligence, 2012, 25(4): 573-580. (in Chinese)
付彬，王志海. 基于树型依赖结构的多标记分类算法[J]. 模式识别与人工智能，2012，25(4)：573-580.