ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.20, No.7, July 2009, pp.1735−1745
doi: 10.3724/SP.J.1001.2009.03319

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

潜在属性空间树分类器*

何 萍[1], 徐晓华[2], 陈 崚[1,2+]

[1](南京航空航天大学 信息科学与技术学院 计算机科学与工程系,江苏 南京 210016)

[2](扬州大学 信息工程学院 计算机科学与工程系,江苏 扬州 225009)

Latent Attribute Space Tree Classifiers

HE Ping[1], XU Xiao-Hua[2], CHEN Ling[1,2+]

[1](Department of Computer Science and Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)

[2](Department of Computer Science and Engineering, Yangzhou University, Yangzhou 225009, China)

+ Corresponding author: E-mail: yzulchen@gmail.com

He P, Xu XH, Chen L. Latent attribute space tree classifiers. *Journal of Software*, 2009,20(7):1735−1745.
http://www.jos.org.cn/1000-9825/3319.htm

Abstract: A framework of latent attribute space tree classifier (LAST) is proposed in this paper. LAST transforms data from the original attribute space into the latent attribute space, which is easier for data separation or more suitable for tree classifier, so that the decision boundary of the traditional decision tree can be extended and its generalization ability can be improved. This paper presents two SVD (singular value decomposition) oblique decision tree (SODT) algorithms based on the LAST framework. SODT first performs SVD on global and/or local data to construct orthogonal latent attribute space. Then, traditional decision tree or tree nodes are built in that space. Finally, SODT obtains the approximately optimal oblique decision tree of the original space. SODT can not only handle datasets with similar or different distribution between global and local data, but also can make full use of the structure information of the labelled and unlabelled data and produce the same classification results no matter how the observations are arranged. Besides, the time complexity of SODT is identical to that of the univariate decision tree. Experimental results show that compared with the traditional univariate decision tree algorithm C4.5 and the oblique decision tree algorithms OC1 and CART-LC, SODT gives higher classification accuracy, more stable decision tree size and comparable tree-construction time as C4.5, which is much less than that of OC1 and CART-LC.

Key words: classification; decision tree; latent attribute space; singular value decomposition

摘 要: 提出一种潜在属性空间树分类器(latent attribute space tree classifier,简称 LAST)框架,通过将原属性空间变换到更容易分离数据或更符合决策树分类特点的潜在属性空间,突破传统决策树算法的决策面局限,改善树分类器的泛化性能.在 LAST 框架下,提出了两种奇异值分解斜决策树(SVD (singular value decomposition) oblique decision tree,简称 SODT)算法,通过对全局或局部数据进行奇异值分解,构建正交的潜在属性空间,然后在潜在属性

* Supported by the National Natural Science Foundation of China under Grant No.60673060 (国家自然科学基金); the Natural Science Foundation of Jiangsu Province of China under Grant No.BK2008206 (江苏省自然科学基金)

Received 2007-05-28; Accepted 2008-03-06

空间内构建传统的单变量决策树或树节点,从而间接获得原空间内近似最优的斜决策树.SODT 算法既能够处理整体数据与局部数据分布相同或不同的数据集,又可以充分利用有标签和无标签数据的结构信息,分类结果不受样本随机重排的影响,而且时间复杂度还与单变量决策树算法相同.在复杂数据集上的实验结果表明,与传统的单变量决策树算法和其他斜决策树算法相比,SODT 算法的分类准确率更高,构建的决策树大小更稳定,整体分类性能更鲁棒,决策树构建时间与 C4.5 算法相近,而远小于其他斜决策树算法.

关键词: 分类;决策树;潜在属性空间;奇异值分解

中图法分类号: TP18 **文献标识码**: A

分类是机器学习领域中被广泛研究的一个重要问题.树分类器,也称为决策树,作为机器学习中众多分类器的一种而有着特别的优势:首先,决策树具有内嵌的降维系统,可以自动筛选出对分类有利的判别属性;其次,决策树是与生俱来的多分类分类器,分类过程不受类别个数增加的影响;再次,决策树具有高可理解性和归纳推理能力,目前已被成功应用于从学习医疗诊断到学习评估贷款申请的信用风险等广阔领域[1,2].但传统的决策树算法由于在每个树节点上只对单个属性进行测试,所以决策面仅限于平行于坐标轴的超矩形,对于需要用斜线或曲线分割的数据,只能用大量的折线来近似划分,这样产生的决策树不仅庞大,对数据敏感,且泛化性能也不强.

近年来,人们为了提高决策树分类器的泛化性能,提出了多变量决策树的概念[3],即是在决策树节点上用关于多个变量(属性)的测试代替传统的单变量(属性)测试的决策树.目前,多变量决策树算法主要采用以下几种多变量测试的形式:单变量测试的布尔表达式组合[4]、多变量代数运算(+,−,×,÷)组合的测试[5]以及多变量线性组合的测试[6].它们之间的共同点是通过修改决策树算法本身来寻找满足假设形式的最佳分类测试,使构建的决策树尽可能地小而准确率却更高.其中,采用多变量线性组合测试的多变量决策树称为斜决策树[6],因为它产生的决策边界是特征空间中的斜线.斜决策树是单变量决策树的泛化形式,但 Heath 等人已证明,确定最小化分类误差的斜超平面是一个 NP-Hard 问题.所以,当前的斜决策树算法大都使用一些优化算法,如进化算法[7]、模拟退火算法[8]、爬山和随机算法[6]等来寻找近似最优的斜超平面.尽管这些方法在如减少决策树大小等方面取得了一定的成功,但却存在着计算代价过高以及分类性能不稳定的缺点,这使得斜决策树算法的应用广泛性反而远远不如传统的单变量决策树算法.

本文提出了一种潜在属性空间树分类器(latent attribute space tree classifier,简称 LAST)框架.LAST 着眼于分类系统的另一个要素——分类空间,通过将全局或局部数据无损信息地从原属性空间变换到更容易分离数据或更符合决策树分类特点的潜在属性空间中,使传统的决策树算法无须过多修改,就可以突破超矩形决策面的局限,改善树分类器的泛化性能.在 LAST 框架下,本文提出了两种奇异值分解斜决策树(SVD(singular value decomposition) oblique decision tree,简称 SODT)算法.SODT 算法选择了一个不同于其他斜决策树算法的视角:多变量决策树之所以可以改善分类的准确率,是因为它可以挖掘出具有判别信息的属性间相关性;换言之,如果属性间本来就相互无关,则单变量决策树的分类准确率应该并不亚于任何多变量决策树.因此,SODT 算法通过对全局或局部数据进行奇异值分解,显式地构建全局或局部的正交潜在属性空间,然后在该潜在属性空间内构建传统的单变量决策树或树节点,最终间接获得原属性空间内近似最优的斜决策树.根据用于构建正交潜在属性空间的数据范围不同,我们将只对整体数据构建正交潜在属性空间的 SODT 算法称为 GSODT(global SVD oblique decision tree)算法,而将对若干层或所有层的局部数据构建正交潜在属性空间的 SODT 算法称为 LSODT(local SVD oblique decision tree)算法.这两种 SODT 算法适用于不同的分类问题,GSODT 算法主要针对整体数据与局部数据分布相同的数据集,而 LSODT 算法则主要针对整体数据与局部数据分布不同的数据集.GSODT 算法构建斜决策树的时间复杂度等同于传统的单变量决策树,为 $O(mn^2)$,LSODT 算法构建斜决策树的时间复杂度为 $O(dmn^2)$,其中,d 表示构建局部正交潜在属性空间的层数.由于我们通常都会指定 LSODT 的层数 d 为一常数,所以 LSODT 算法的时间复杂度也为 $O(mn^2)$.此外, SODT 算法还具有很好的鲁棒性,它可以充分利用无标签数据的结构信息,而且分类结果不受数据样本重排的影响(有的斜决策树算法分类结果会受数据样本重排的影响,如 OC1).本文中所做实验从分类错误率、构建决策树大小以及决策树构建时间 3 个方面将 SODT

算法与经典的 C4.5 单变量决策树算法、著名的 OC1 和 CART-LC 斜决策树算法在 12 个测试数据集上进行了全面的比较.实验结果显示,SODT 算法在绝大多数数据集上的平均分类错误率低于其他算法;SODT 构建的决策树大小比单变量决策树更紧致,比其他斜决策树算法更稳定;SODT 构建决策树的运行效率远远高于其他斜决策树算法,并接近于单变量决策树 C4.5 算法.

本文第 1 节提出潜在属性空间树分类器框架.第 2 节介绍在 LAST 框架下提出的两种奇异值分解斜决策树算法 GSODT 和 LSODT.第 3 节对 SODT 算法进行全面的性能评价.第 4 节对全文做出总结.

1 潜在属性空间树分类器

本节我们提出了一种潜在属性空间树分类器框架.首先我们简单回顾基本的决策树算法,然后再介绍 LAST 框架的提出动机、具体内容和不同之处.

1.1 决策树算法

决策树是一类应用广泛的树分类器.决策树的构建一般包括两个阶段:生长阶段和剪枝阶段.在生长阶段,所有训练样本被自顶向下、分而治之地用于构建一棵完全生长的决策树.在这个阶段,每个决策树节点的构建过程如图 1 所示:首先判断分配至待建树节点的数据集 X 是否满足叶节点的条件,若满足,则构建叶节点并返回(Step 1);否则,选取最佳测试 t^*(Step 2)用于构建内部节点 Ψ(Step 3);然后,对于 t^*每一个可能的分枝(Step 4),将 X 中满足该分枝条件的数据样本(Step 5)分配给对应的子树节点用于其递归构建(Step 6);最后,返回构建的内部节点(Step 8).

PROCEDURE *TREENODECONSTRUCTION*(X)
Input: The allocated data X on the tree node;
Output: The constructed decision tree Ψ.
1. **if** *isLeaf*(X) **return** $\Psi \leftarrow$ *leaf*(X)
2. $t^* \leftarrow$ *selectBestTest*(X)
3. $\Psi \leftarrow$ *internalNode*(X,t^*)
4. **for each** $i \in$ *branchCount*(t^*)
5. $X_i \leftarrow$ *partition*(t^*,X,i)
6. $\Psi.child_i \leftarrow$ *TREENODECONSTRUCTION*(X_i)
7. **endfor**
8. **return** Ψ

Fig.1 Decision tree node construction process

图 1 决策树节点构建过程

最常用的几种用于评价测试 t^*的函数包括信息增益比(GainRatio)[9]、基尼指数(Gini)[10]、χ^2 测验[11]和 G 系数[11]等.本文采用的是著名的 C4.5 算法的 GainRatio 函数,其内容如下:令 D^{Ψ}表示分配至树节点 Ψ上的数据集,D_i^{Ψ} 表示 D^{Ψ}根据测试 t^*分配至第 i 个分支的子数据集,D_c^{Ψ} 表示 D^{Ψ}中类别标签为 c 的子数据集,则信息增益比定义为

$$GainRatio(D^{\Psi},t^*) \equiv GainInfo(D^{\Psi},t^*)/SplitInfo(D^{\Psi},t^*) \tag{1}$$

其中,

$$GainInfo(D^{\Psi},t^*) \equiv E(D^{\Psi}) - \Sigma_i |D_i^{\Psi}|/|D^{\Psi}| \times E(D_i^{\Psi}) \tag{2}$$

$$SplitInfo(D^{\Psi},t^*) \equiv -\Sigma_i (|D_i^{\Psi}|/|D^{\Psi}|) \times \log_2(|D_i^{\Psi}|/|D^{\Psi}|) \tag{3}$$

$$E(D^{\Psi}) \equiv -\Sigma_c (|D_c^{\Psi}|/|D^{\Psi}|) \times \log_2(|D_c^{\Psi}|/|D^{\Psi}|) \tag{4}$$

决策树构建的第 2 阶段为剪枝阶段,就是剪去一些统计无效、由噪声造成树节点和分支,目的是防止过拟合.常用的剪枝方法包括代价-复杂度剪枝(cost complexity pruning,简称 CCP)[10]、减少错误剪枝(reduced error pruning,简称 REP)[12]、悲观错误剪枝(pessimistic error pruning,简称 PEP)[12]、基于错误剪枝(error based pruning,简称 EBP)[12]以及基于最小描述长度的剪枝[13]等.本文采用的是 C4.5 算法的基于错误剪枝方法.EBP 假设决策树在训练数据上的错误率呈正态分布,悲观估计各个树节点在置信度(默认为 0.25)范围内的测试错误率,从而确定是否需要子树替代(subtree replacement)或子树上升(subtree raising)剪枝操作.EBP 的测试错误率估计公式:

$$e(\Psi) = [f + z^2/2n + z\sqrt{f/n - f^2/n + z^2/4n^2}]/(1 + z^2/n) \tag{5}$$

其中,f 是树节点 Ψ的训练错误率,n 是分配至 Ψ的样本数,即$|D^{\Psi}|$,z 是对应指定置信度的标准差(可查表获得).

决策树对新数据的分类是将新数据从树的根节点开始,根据树节点的测试选择对应的分枝向下移动,直至到达某个叶节点为止,则该叶节点所带的类别标签就是新数据的分类结果.

1.2 LAST框架

传统的决策树算法在构建树节点时,只选择一个分类错误率最低的单变量测试作为树节点的测试.这样做的优点是速度快,满足大多数简单数据集的分类需求;但缺点是无法挖掘出具有判别信息的属性相关性,对一些复杂的数据集分类的泛化性能较差.

我们提出的潜在属性空间树分类器框架,从决策树算法的作用对象分类空间入手,通过将原属性空间变换为某种更容易分离数据或更符合决策树分类特点的潜在属性空间,将原始数据无损信息地以一种更有利于分类的形式进行重新表达,从而使传统的决策树算法无须过多修改,就可以摆脱决策面仅限于平行于坐标轴的超矩形的束缚,扬长避短地在潜在属性空间内对数据做出泛化性能更好的分类.不难发现,LAST 框架与近年来兴起的核方法[14]有一个共同点,就是都借助于分类空间的变换,使在原空间中较难分类的数据在新空间中变得更容易分类,从而提高分类器的泛化能力.但是,两者的不同之处主要在于,核方法是通过改变数据样本间的两两距离来实现分类空间的变换,而 LAST 框架则是通过改变属性间的两两关系来实现这一目的的.此外,LAST 框架对潜在属性空间的构建也不像核方法那样有严格固定的模式,而是可以从领域知识、数据的先验信息以及分类器的分类特点等多角度出发,选择不同的方法和技术,为不同的数据集或同一数据集的不同部分构建最合适的潜在属性空间.例如,以 PCA(principle component analysis)[15]和 ICA(independent component analysis)[16]等技术作为决策树预处理的算法就属于 LAST 框架.这些算法尽管只对整体数据进行了预处理,而且有时还会造成信息的损失(如 PCA 去除可能的噪声属性),但它们都较好地体现了 LAST 框架通过变换分类空间来改善分类器泛化性能的思想.

2 奇异值分解斜决策树

本节我们在 LAST 框架下提出了两种奇异值分解斜决策树(SVD(singular value decomposition) oblique decision tree,简称 SODT)算法,分别是全局奇异值分解斜决策树算法和局部奇异值分解斜决策树算法.我们首先介绍奇异值分解技术及其在正交潜在属性空间构建中的应用,然后简单概括 SODT 算法的基本思想,并给出 GSODT 和 LSODT 的算法详细描述,最后对两种 SODT 算法的时间复杂度进行分析和讨论.

2.1 奇异值分解

奇异值分解是现代数值分析尤其是数值计算的最基本和最重要的工具之一.具体的对任意复长方矩阵的奇异值分解定理称为 Autonee-Eckart-Young 定理[17],其内容如下:

矩阵的奇异值分解:令 $A\in\mathbb{R}^{m\times n}$(或 $\mathbb{C}^{m\times n}$),则存在正交(或酉)矩阵 $U\in\mathbb{R}^{m\times m}$(或 $\mathbb{C}^{m\times m}$)和 $V\in\mathbb{R}^{n\times n}$(或 $\mathbb{C}^{n\times n}$)使得

$$A=U\Sigma V^T(\text{或 } U\Sigma V^H) \tag{6}$$

式中,

$$\Sigma=\begin{bmatrix}\Sigma_r & O\\ O & O\end{bmatrix} \tag{7}$$

且 $\Sigma_r=diag(\sigma_1,\sigma_2,...,\sigma_r)$,其对角元素按照顺序排列:

$$\sigma_1\geq\sigma_2\geq...\geq\sigma_r, r=rank(A) \tag{8}$$

本文的 SODT 算法主要利用 SVD 对外积矩阵的对角化来构建正交的潜在属性空间.首先,我们令矩阵 $X_{m\times n}$ 表示原始的输入数据(不包含类别数据),m 为条件属性个数,n 为数据样本个数,则 X 的外积矩阵表示了 X 的各个属性间的两两线性相关性:

$$C=XX^T=(x_i\cdot x_j^T)_{m\times m}, i,j\in\{1,...,m\} \tag{9}$$

如果 C 是一个对角阵,就表示 X 的特征空间是一个属性间两两线性无关的正交空间,而我们的目标就是构建这样一个关于 X 的正交潜在属性空间.根据奇异值分解定理,我们已知

$$X=U\Sigma V^T \tag{10}$$

故 X 的外积矩阵可表示为

$$C=XX^T=U\Sigma V^T V\Sigma U^T=U\Sigma^2 U^T \tag{11}$$

对等式(11)进行变化,可得等式(12)为对角阵.

$$U^T CU=(U^T X)(U^T X)^T=\Sigma^2 \tag{12}$$

若令矩阵 $Y=U^T X$,则等式(12)等价于

$$YY^T=\Sigma^2 \tag{13}$$

这表示 Y 就是我们所要构建的 X 的正交潜在属性空间,且这些潜在属性就是 X 原属性集的线性组合.

2.2 SODT算法

SODT 算法从传统决策树算法基于的属性间相互无关假设出发,结合斜决策树算法挖掘具有判别信息的属性相关性优势,通过将原属性空间变换为正交的潜在属性空间,既确保了潜在属性之间的两两正交性,又以潜在属性的形式隐式构造了若干种可能具有判别信息的原属性线性组合,从而使传统的决策树算法能够在潜在属性空间内间接地构建原属性空间的近似最优斜决策树或树节点.

SODT 算法使用奇异值分解来构建正交的潜在属性空间,但奇异值分解只能处理数值型属性,所以,SODT 算法需要事先将名词型属性转化为数值型属性.文献[10]等提供了名词型属性和数值型属性之间的相互转化方法,可作为 SODT 算法的数据预处理.本文为了描述简单,假设所有的属性都是数值型属性.

2.3 全局奇异值分解斜决策树

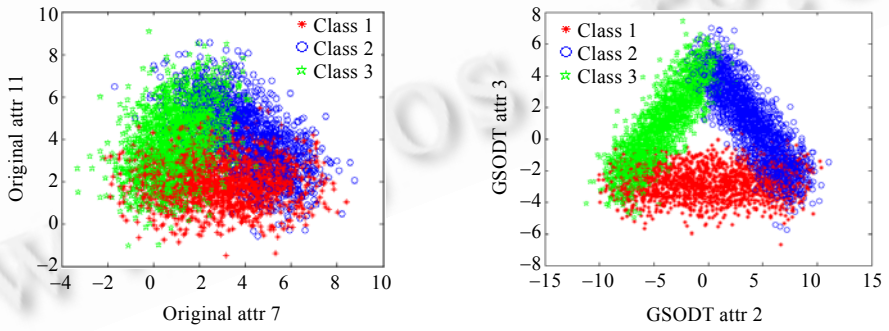
GSODT 算法假设整体数据与局部数据(如不同类别的数据)结构分布基本相同,通过对整体数据进行 SVD,构建全局的正交潜在属性空间,则在此潜在属性空间内构建的单变量决策树就等价于在原属性空间内构建的斜决策树.GSODT 算法产生的几何决策面是旋转了一定角度的超矩形决策面.

图 2 给出了 GSODT 的算法描述:首先,GSODT 将训练数据 X_{train} 和测试数据 X_{test} 合并成整体数据 X(Step 1)进行 SVD(Step 2),然后用求到的左奇异向量阵 U 构建全局正交潜在属性空间 Y(Step 3),最后在 Y 的训练数据部分 Y_{train}(Step 4)上构建潜在属性空间内的传统单变量决策树(Step 5)并返回(Step 6),则该决策树就是原属性空间内的斜决策树.

ALGORITHM $GSODT(X_{train},X_{test})$
Input: The training data X_{train} and the test data X_{test};
Output: The oblique decision tree T.
1. $X=[X_{train},X_{test}]$
2. $[U,\Sigma,V]=SVD(X)$
3. $Y=U^T X$
4. $Y_{train}=Y(:,1{:}size(X_{train},2))$
5. $T=buildDecisionTree(Y_{train})$
6. **return** T

Fig.2 GSODT construction algorithm
图 2 GSODT 算法描述

GSODT 算法虽然看似简单,但在实际应用中却显得非常有效.例如,Waveform 数据集由 21 个包含噪声的条件属性、5 000 个数据样本以及 3 个类别构成,每个类别又由 2 个基波组合而成.图 3(a)是 Waveform 数据集在原属性空间内的最佳 2 维投影,而图 3(b)则是它在 GSODT 全局正交潜在属性空间内的最佳 2 维投影.



(a) Original attribute space (a) 原属性空间
(b) GSODT global orthogonal latent attribute space (b) GSODT 全局正交潜在属性空间

Fig.3 Best 2D projections of Waveform dataset in two different attribute spaces
图 3 Waveform 数据集在两种属性空间内的最佳 2 维投影

很明显,GSODT 将对 Waveform 数据集做出更准确的分类.这充分体现了构建全局正交潜在属性空间对改善决策树分类准确率的巨大帮助.

2.4 局部奇异值分解斜决策树

LSODT 算法是将 GSODT 对整体数据构建的正交潜在属性空间扩展到局部数据上的泛化,因为多个不同结构分布的子空间可能会合成新的结构和分布的整体空间.LSODT 算法的解决方法是对开始若干层或所有层树节点上的局部数据构建局部的正交潜在属性空间,以此保证整棵决策树的构建都满足潜在属性之间的正交性.

图 4 描述了 LSODT 算法对需要局部正交潜在属性空间的决策树节点的构建过程:首先,LSODT 判断分配至该树节点的训练数据是否满足叶节点的条件,如果满足,则构建叶节点并返回(Step 1);否则,将分配至该树节点的训据数据和测试数据合并(Step 2),进行 SVD(Step 3),并用左奇异向量阵 U 构建局部的正交潜在属性空间(Step 4).然后,LSODT 提取出该潜在属性空间内的训练数据部分(Step 5)用于内部节点的构建(Step 6,Step 7),并记录剩下的测试数据部分(Step 5),以便于最后在对子树节点进行递归构建时(Step 10),根据当前内部节点的测试将其和训练数据一起分配到合适的子树节点上(Step 9).最后,LSODT 返回构建好的内部节点(Step 12).

PROCEDURE *LSODTTREENODECONSTRUCTION*(X_{train},X_{test})
Input: The allocated training data X_{train} and the allocated test data X_{test} on the tree node;
Output: The constructed tree node Ψ.
1. if *isLeaf*(X_{train}) return $\Psi \leftarrow$ *leaf*(X_{train})
2. $X=[X_{train},X_{test}]$
3. $[U,\Sigma,V]=SVD(X)$
4. $Y=U^TX$
5. $Y_{train}=Y(:,1:\text{size}(X_{train},2))$, $Y_{test}=Y-Y_{train}$
6. $t^* \leftarrow$ *selectBestTest*(Y_{train})
7. $\Psi \leftarrow$ *internalNode*(Y_{train},t^*)
8. **for each** $i \in$ *branchCount*(t^*)
9. $[Y^i_{train}, Y^i_{test}]$=*partition*($t^*$,$Y_{train}$,$Y_{test}$,$i$)
10. $\Psi.child_i \leftarrow$ *LSODTTREENODECONSTRUCTION*(Y^i_{train}, Y^i_{test})
11. **endfor**
12. **return** Ψ

Fig.4 LSODT tree node construction procedure

图 4 LSODT 算法构建决策树节点的过程

LSODT 算法与 Zhou 等人提出的 HDT(hybrid decision tree)算法[18]的主要不同之处在于,前者保留了传统决策树节点的基本构建过程,而后者则在树节点的最佳测试选择中融入了前向神经网络技术.LSODT 算法的优越之处在于,它同时保证了整体空间和局部子空间的潜在属性正交性,比 GSODT 算法可以更好地处理整体数据和局部数据不同分布的分类问题,而且产生的决策面是更为泛化的超多边形决策面;而 LSODT 的缺点则是比 GSODT 算法更复杂,并增加了额外的计算时间.

2.5 时间复杂度分析

GSODT 算法对整体数据进行 SVD 的时间复杂度为 $O(\min\{m^2n,mn^2\})$,根据求得的左奇异向量阵构建全局正交潜在属性空间的时间复杂度为 $O(mn^2)$,然后在该全局正交潜在属性空间内构建单变量决策树的时间复杂度为 $O(mn^2)$,而其他操作均可在 $O(1)$时间内完成.所以,当 $m \le n$,即属性个数少于或等于数据样本个数时,GSODT 的时间复杂度是 $O(m^2n)+O(mn^2)+O(mn^2)+O(1)=O(mn^2)$;而当 $m>n$,即属性个数大于数据样本个数时,GSODT 的时间复杂度是 $O(mn^2)+O(mn^2)+O(mn^2)+O(1)=O(mn^2)$.由此可知,GSODT 的时间复杂度是 $O(mn^2)$,与传统单变量决策树算法的时间复杂度相同.

LSODT 算法与 GSODT 算法在时间复杂度分析上的主要不同之处在于,LSODT 算法要对若干层树节点的局部数据构建局部的正交潜在属性空间,可是这些局部数据的样本容量却是未知的,因此我们要从对一层决策树节点的时间复杂度分析入手.现假设决策树一层有 k 个树节点,且第 i 个树节点上有 n_i 个局部数据样本,则

LSODT 对该层树节点构建局部正交潜在属性空间的时间复杂度为

$$O_L=(O(\min\{m^2n_1,m\,n_1^2\})+O(\min\{m^2n_2,m\,n_2^2\})+\ldots+O(\min\{m^2n_k,m\,n_k^2\}))+(O(m\,n_1^2)+O(m\,n_2^2)+\ldots+O(m\,n_k^2))$$
$$<O(mn^2),$$

其中,公式前半部分是对 k 个局部数据进行 SVD 的时间复杂度,后半部分则是利用求得的 k 个左奇异向量阵构建局部正交潜在属性空间的乘法复杂度.所以,如果 LSODT 要对 d 层决策树节点构建局部正交潜在属性空间,所需的时间复杂度为 $O(dmn^2)$.另外,再加上 LSODT 构建所有单变量决策树节点的 $O(mn^2)$时间复杂度,LSODT 算法的总时间复杂度为 $O(dmn^2)+O(mn^2)=O(dmn^2)$,即随着构建局部正交潜在属性空间的树节点层数增加而线性增长.由于我们通常都指定 LSODT 的层数 d 为一常数,所以 LSODT 算法的时间复杂度也为 $O(mn^2)$.

3 实 验

本节我们将GSODT和LSODT两种SODT算法与经典的C4.5单变量决策树算法、著名的OC1和CART-LC斜决策树算法,从测试错误率、决策树大小以及决策树构建时间 3 个方面,在 12 个测试数据集上进行了全面的比较.首先我们给出具体的实验方案和测试数据集,然后又从 3 个方面对实验结果进行了比较和讨论,最后对实验结果做出了总结.

3.1 实验设计

我们使用 Java 语言实现了 GSODT 算法和 3 种 LSODT 算法 LSODT-L1,LSODT-L2 和 LSODT-L3,其中,L1,L2 和 L3 分别表示 LSODT 算法为决策树顶端的 1 层、2 层和 3 层树节点构建局部的正交潜在属性空间.实现这 3 种 LSODT 算法的目的是观察随着构建局部正交潜在属性空间的树节点层数的增加,LSODT 算法的分类性能受到的影响.所有的 SODT 算法都采用 Fast C4.5[19]构建单变量决策树或树节点.

在算法的比较方面,我们选择了用 C 语言实现的 C4.5 单变量决策树算法和 OC1[6]及 CART-LC[10]两种斜决策树算法与 SODT 算法进行比较.选择 C4.5 算法的目的是为了验证是否斜决策树算法一定优于传统的单变量决策树算法,而选择 OC1 和 CART-LC 算法的目的则是为了给 SODT 算法提供优秀的斜决策树算法参照.由于原始的 CART-LC 算法的源代码不能免费获得,我们使用的是由 OC1 系统实现的 CART-LC 算法.

表 1 总结了用于比较 C4.5,OC1,CART-LC,GSODT,LSODT-L1,LSODT-L2 和 LSODT-L3 共 7 种算法的 12 个测试数据集.其中,数据集 spectf,bupa,pima-indian-diabetes,vehicle,vowel,satellite,waveform 和 waveformnoise 来自于 UCI 机器学习数据库[20],twonorm 和 ringnorm 来自于 DELVE 环境[21],threenorm 是由改自于文献[22]的代码直接生成的,而 gauss_2D 则是 Elena 项目[23]中的一个基准人工数据集,其目的是测试不同分类器对具有高度较叠且非线性可分数据集的分类性能.这些测试数据集的一个共同特点是传统的单变量决策树对它们的分类准确率较低.

Table 1 Experimental data sets

表 1 测试数据集

Data sets	Number of cases	Number of attributes	Number of classes
spectf	266	44	2
bupa	345	6	2
pima-indian-diabetes	768	8	2
vehicle	846	18	4
vowel	989	10	11
satellite	4 435	36	6
gauss_2D	5 000	2	2
waveform	5 000	21	3
waveformnoise	5 000	40	3
twonorm	7 400	20	2
ringnorm	7 400	20	2
threenorm	7 400	20	2

本文所有的实验都在一台装有 SuSE Linux Enterprise Desktop 10 操作系统的 Intel Pentium IV 3.40Ghz、双 CPU、1 024M 内存的 PC 控制台模式下运行.其中,Java 运行环境为 JRE1.5.0,虚拟机在客户端模式下运行,运行

时最大堆空间设为 1 024M,C/C++的编译环境为 GCC 4.2,代码优化为-O2.

3.2 测试错误率

表 2 列出了 C4.5,OC1,CART-LC,GSODT,LSODT-L1,LSODT-L2 和 LSODT-L3 共 7 种算法在 12 个测试数据集上 10 折交叉验证的平均测试错误率及其标准差.

Table 2 Comparison of averaged test error ratios and their corresponding deviations

表 2 平均测试错误率及其标准差比较

Algorithm	Spectf (%)	Bupa (%)	Pima (%)	Vehicle (%)	Vowel (%)	Satellite (%)
C4.5	25.1±7.2	36.2±8.2	26.1±4.4	**27.2±5.8**	20.5±4.8	13.9±1.7
OC1	27.8±9.3	**30.9±6.4**	26.5±7.4	35.1±3.9	20.8±4.1	13.2±2.1
CART-LC	27.7±9.4	33.7±8.3	28.3±4.3	31.2±4.2	20.5±4.5	13.4±1.7
GSODT	23.2±6.5	33.0±8.6	**25.4±5.4**	27.3±5.6	19.1±4.2	**13.0±1.8**
LSODT-L1	25.1±7.2	35.1±7.9	25.6±4.9	27.6±5.4	18.4±4.4	13.6±2.1
LSODT-L2	**21.0±6.2**	34.8±5.1	25.9±5.0	28.4±4.0	**18.2±3.2**	**13.0±1.5**
LSODT-L3	24.7±4.6	33.6±7.2	25.9±3.7	28.1±6.8	20.3±3.7	13.2±1.4
Algorithm	gauss_2D (%)	Waveform (%)	Waveformnoise (%)	Twonorm (%)	Ringnorm (%)	Threenorm (%)
C4.5	27.6±1.6	25.1±7.2	24.6±1.8	15.4±1.3	9.2±0.8	25.9±1.8
OC1	28.8±2.6	18.5±2.1	20.7±2.1	18.8±1.8	17.8±2.1	18.1±1.1
CART-LC	39.8±10.5	35.5±6.2	56.9±10.4	18.4±6.0	43.1±6.9	42.7±8.2
GSODT	27.1±1.8	**15.1±1.5**	16.8±2.0	**2.2±0.4**	**8.4±0.8**	11.1±0.8
LSODT-L1	27.1±1.4	15.5±1.9	**16.7±1.3**	2.3±0.7	**8.4±0.6**	**10.9±1.1**
LSODT-L2	27.2±2.3	15.7±1.6	17.2±2.2	2.6±0.6	8.6±0.7	11.7±0.7
LSODT-L3	**27.0±1.8**	17.0±1.4	18.9±1.9	2.3±0.4	**8.4±0.8**	11.7±0.8

从表 2 中我们可以得到以下结论:

首先,4 种 SODT 算法在大多数数据集上的平均测试错误率都远低于其他 3 种算法,这充分说明了 SODT 算法的高准确性.在测试错误率的标准差上,4 种 SODT 算法也在绝大多数数据集上都低于其他 3 种算法,所以,SODT 算法同时也具有较好的稳定性.

其次,斜决策树算法的平均测试错误率不一定比单变量决策树算法要低.实验中,C4.5 算法在 12 个测试数据集中的 7 个数据集上平均测试错误率都低于或等于 OC1 和 CART-LC 两种斜决策树算法,甚至还在 1 个数据集上比 4 种 SODT 算法的最优结果还要低 0.1%.这说明,尽管在理论上斜决策树是单变量决策树的泛化形式,但并不表示斜决策树算法的分类结果就一定优于单变量决策树算法的分类结果.OC1 算法和 CART-LC 算法由于缺少理论性指导,只用爬山和随机化方法在指数级增长的庞大空间内搜索一个近似最优的斜决策面;而 SODT 算法则紧扣单变量决策树基于的属性间相互无关假设,通过在正交的潜在属性空间内构建近似最优的单变量决策树,间接地获得在原属性空间内的近似最优斜决策树,显得更为合理,但仍在不能去除属性间的非线性相关关系上略显不足.

再次,构建更为多个局部的正交潜在属性空间并不一定比构建一个全局的正交潜在属性空间更优.在表 2 中,只有 7 个测试数据集显示至少有 1 个 LSODT 算法的平均测试错误率低于或等于 GSODT 算法,并且其中 LSODT-L1 算法占了 2 个.这说明,大多数测试数据集的数据分布结构都比较纯粹,全局数据和局部数据的分布都基本相同,所以,只要对整体数据构建全局的正交潜在属性空间就已经可以解决大部分的问题.但同时也不可忽视的是,LSODT 算法的确在有些测试数据集如 spectf 和 vowel 上起到了较明显的平均测试错误率改善作用.这说明全局数据和局部数据不同分布的情况的确存在,但可能并不普遍.

最后,SODT 算法的分类结果不受数据样本重排的影响.假设原始数据样本为 X,则对 X 构建的正交潜在属性空间为 $Y=U^TX$,其中,U 为 X 的左奇异向量阵.SODT 算法在该潜在属性空间 Y 内构建单变量决策树或决策树节点.现用置换阵 P 对 X 进行数据样本(即列)的重排,得到新数据矩阵 $X'=XP$,且根据奇异值分解定理有

$$X'=XP=(U\Sigma V^T)P=U\Sigma(P^TV)^T \tag{14}$$

若令 $V'=P^TV$,则有

$$(V')^TV'=(P^TV)^T(P^TV)=V^TPP^TV=V^TV=I \tag{15}$$

$$V'(V')^T=(P^TV)(P^TV)^T=P^TVV^TP=P^TP=I \tag{16}$$

所以 $X'=U\Sigma V'$,且 $V'=P^TV$.这表示 X'的右奇异向量阵 V'等于 X 的右奇异向量阵 V 的行重排,而 X'的左奇异向量阵 U 及对角阵 Σ则与 X 的左奇异向量阵和对角阵都相同.由此可得 X'的正交潜在属性空间为

$$Y'=U^TX'=U^TXP=YP \tag{17}$$

即 X 的正交潜在属性空间 Y 的列重排.由于我们已知单变量决策树算法的分类结果不受数据样本重排的影响,即 SODT 算法在 Y 和 Y'正交潜在属性空间内构建的单变量决策树或树节点是完全相同的,所以,SODT 算法的分类结果也不受数据样本重排的影响.

3.3 决策树大小

表 3 列出了 C4.5,OC1,CART-LC,GSODT,LSODT-L1,LSODT-L2 和 LSODT-L3 共 7 种算法在 12 个测试数据集上 10 折交叉验证的平均决策树大小及其标准差.

Table 3 Comparison of averaged decision tree size and their corresponding deviations

表 3 平均决策树大小及其标准差比较

Algorithm	spectf	Bupa	pima	vehicle	vowel	Satellite
C4.5	27.0±4.2	47.4±11.3	45.8±10.3	141.4±14.9	184.2±9.8	392.8±21.9
OC1	18.2±13.0	13.6±20.0	41.8±44.2	45.6±39.1	101.0±30.2	130.8±78.2
CART-LC	18.4±10.9	16.6±16.0	40.8±54.4	81.0±68.0	161.2±23.9	161.8±76.8
GSODT	31.2±3.2	27.2±7.9	46.2±9.4	131.0±14.5	183.4±6.5	349.8±24.4
LSODT-L1	27.0±4.2	25.0±6.7	47.8±10.6	127.8±14.4	172.8±8.2	342.2±15.0
LSODT-L2	30.8±2.2	22.6±10.2	48.0±13.2	142.8±10.9	174.4±6.9	362.6±20.9
LSODT-L3	28.4±5.9	17.2±7.1	44.4±17.0	145.0±9.6	183.2±5.3	355.4±15.0
Algorithm	gauss_2D	waveform	waveformnoise	twonorm	ringnorm	threenorm
C4.5	9.8±1.7	541.8±41.9	585.6±28.0	567.0±11.2	373.0±15.1	921.6±26.0
OC1	119.8±158.8	105.0±57.0	125.4±95.1	731.2±418.1	644.8±238.2	16.4±14.6
CART-LC	18.6±15.4	23.4±22.7	32.6±49.2	127.6±89.6	3.6±1.4	20.6±28.2
GSODT	9.2±0.6	220.0±43.8	311.4±41.5	26.0±9.8	341.4±12.8	147.0±40.4
LSODT-L1	9.6±1.4	211.0±40.5	331.6±49.9	36.6±16.6	343.4±17.0	128.8±34.5
LSODT-L2	9.4±1.3	282.8±30.0	324.2±56.9	50.2±18.9	362.2±20.2	192.0±43.2
LSODT-L3	10.0±1.1	305.0±34.9	375.6±39.9	25.2±14.1	354.8±12.4	178.6±53.6

在表 3 中,CART-LC 算法[**]是 4 种算法中决策树构建得最小的一个,OC1 算法排在其次,SODT 算法总体排在第三,而 C4.5 算法最末.但是,CART-LC 算法构建的小决策树并不完全是泛化性能好的表现:在 ringnorm 数据集上,CART-LC 算法构建的决策树平均只有 3.6 个节点,但对应的平均测试错误率却高达 43.1%,这只能理解为是无法对数据集进行正确分类的表现而已.而 OC1 算法也存在着稳定性较差的隐患:在大多数数据集上,OC1 算法都较好地将构建的决策树大小控制在 200 个节点以内,但是在 twonorm 和 ringnorm 两个数据集上,OC1 算法构建的平均决策树大小却突然飙升到 731.2 和 644.8 个节点,不但超过了 C4.5 算法,而且还高达 SODT 算法的 30 几倍,这让人不得不怀疑其算法的稳定性.SODT 算法虽然不是决策树构建得最小的一个,但却能在保证平均测试错误率较低的前提下,将决策树的大小稳定地控制在一个合理的范围内(<370 个节点),这种适度的平衡恰如其分地体现了决策树分类器应有的泛化性能.

在决策树大小标准差方面,SODT 算法在高达 9 个数据集上达到最低;而相反地,OC1 算法却在 9 个数据集上达到最高.这一方面从决策树大小的角度体现了 SODT 算法的稳定性,另一方面也验证了我们前面在评价 OC1 算法平均决策树大小时对其稳定性的怀疑.

3.4 决策树构建时间

表 4 列出了 C4.5,OC1,CART-LC,GSODT,LSODT-L1,LSODT-L2 和 LSODT-L3 共 7 种算法在 12 个测试数据集上构建 10 次决策树分类器的平均时间.

** 注意,OC1 系统为了保证构建的决策树不至于过大,默认决策树的层数不得超过 50 层,否则强制停止其生长.

表4中,4种SODT算法的平均决策树构建时间远低于其他两种斜决策树算法,并与单变量决策树算法C4.5的平均决策树构建时间接近.这种效率上的巨大差异可以用算法的时间复杂度来解释:CART-LC算法可能会用任意长的时间来构建一个树节点[6],因此它没有运行时间的上限;OC1算法的时间复杂度为 $O(mn^2\log n)$[6];传统的单变量决策树算法C4.5的时间复杂度为 $O(mn^2)$;GSODT算法的时间复杂度为 $O(mn^2)$,若事先指定构建局部正交潜在属性空间的层数(如LSODT-L1,LSODT-L2和LSODT-L3),则LSODT算法的时间复杂度也为 $O(mn^2)$,因此,SODT算法的时间复杂度为 $O(mn^2)$,与单变量决策树算法相同.

Table 4 Comparison of averaged tree construction time (milliseconds)

表4 平均决策树构建时间比较(单位:毫秒)

Algorithm	spectf	bupa	pima	vehicle	vowel	satellite
C4.5	28	6	20	64	168	672
OC1	200	214	478	2 568	2 240	57 484
CART-LC	34	20	42	130	2 392	2 075
GSODT	104	20	73	287	318	4 259
LSODT-L1	114	21	77	302	325	4 380
LSODT-L2	146	20	81	326	314	4 679
LSODT-L3	199	21	96	373	355	5 392
Algorithm	gauss_2D	waveform	w.f.noise	twonorm	ringnorm	threenorm
C4.5	118	1 008	1 944	2 300	4 056	2 036
OC1	202 106	56 122	74 658	33 452	40 770	27 745
CART-LC	12 894	242 140	2 117 580	56 478	44 524	37 857
GSODT	153	1 676	3 584	1 236	6 246	2 144
LSODT-L1	214	1 762	3 703	1 372	5 938	2 213
LSODT-L2	226	2 003	4 239	1 599	5 423	2 459
LSODT-L3	241	2 384	5 170	1 680	5 919	2 459

4 结束语

本文提出了一种潜在属性空间树分类器框架,并在此框架下提出了两种奇异值分解斜决策树算法.由理论分析可知,SODT算法能够处理整体数据与局部数据分布相同或不同的数据集,可以充分利用有标签和无标签数据的结构信息,分类结果不受数据样本重排的影响,而且时间复杂度还与单变量决策树算法相同.大量的实验结果表明,与传统的单变量决策树算法C4.5以及斜决策树算法OC1和CART-LC相比, SODT算法的分类准确率更高,构建的决策树大小更稳定,整体分类性能更鲁棒,决策树构建时间接近于C4.5算法而远小于其他斜决策树算法.

SODT算法的优秀分类性能很好地证明了LAST框架的可行性和优越性.因此,我们的下一步工作是进一步充实LAST框架的内容,使其更丰富、更具体.另外,在LAST框架下设计更多优秀的决策树算法,深入研究和验证潜在属性空间的最佳构建方案.

References:

[1] Mitchell T. Machine Learning. New York: McGraw-Hill Publisher, 1997.

[2] Liu YQ, Zhang M, Ma SP. Web key resource page judgment based on improved decision tree algorithm. Journal of Software, 2005, 16(11):1958−1966 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/16/1958.htm

[3] Brodley CE, Utgoff PE. Multivariate decision trees. Machine Learning, 1995,19(1):45−77.

[4] Sahami M. Learning non-linearly separable boolean functions with linear threshold unit trees and madaline-style networks. In: David ST, Michael CM, Michael EH, eds. Proc. of the 11th National Conf. on Artificial Intelligence. Washington: AAAI Press, 1993. 335−341.

[5] Muharram M, Smith GD. Evolutionary constructive induction. IEEE Trans. on Knowledge and Data Engineering, 2005,17(11): 1518−1528.

[6] Sreerama KM, Simon K, Steven S. A system for induction of oblique decision trees. Journal of Artificial Intelligence Research, 1994,2:1−32. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.44.6304

[7] Kretowski M. An evolutionary algorithm for oblique decision tree induction. In: Rutkowski L, *et al*., eds. Proc. of the Artificial Intelligence and Soft Computing—ICAISC 2004. LNAI 3070, Heidelberg: Springer-Verlag, 2004. 1053−1060.

[8] Heath D, Kasif S, Salzberg S. Induction of oblique decision trees. In: Ruzena B, ed. Proc. of the 13th Int'l Joint Conf. on Artificial Intelligence. Chambery: Morgan Kaufmann Publishers, 1993. 1002−1007.

[9] Quinlan J. C4.5 Programs for Machine Learning. San Mateo: Morgan Kaufmann Publishers, 1993.

[10] Breiman L, Friedman J, Olshen RA, Stone CJ. Classification and Regression Trees. Belmont: Wadsworth, 1984.

[11] Mingers J. Expert systems—Experiments with rule induction. Journal of the Operational Research Society, 1986,37(11):39−47.

[12] Quinlan JR. Simplifying decision trees. Int'l Journal of Man-Machine Studies, 1987,27(3):221−234.

[13] Mehta M, Rissanen J, Agrawal R. MDL-Based decision tree pruning. In: Usama MF, Ramasamy U, eds. Proc. of the 1st Int'l Conf. on Discovery and Data Mining (KDD'95). Montreal: AAAI Press, 1995. 216−221.

[14] John ST, Nello C. Kernel Methods for Pattern Analysis. Cambridge: Cambridge University Press, 2004.

[15] Luboš P, Pavel B. The principal components method as a pre-processing stage for decision tree learning. In: Djamel AZ, Henryk JK, Jan MZ, eds. Proc. of the Data Mining and Knowledge Discovery 4th European Conf. Lyon, 2000. 13−16. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.17.5501

[16] Sanchez-Poblador V, Monte-Moreno E, Casals JS. ICA as a preprocessing technique for classification. In: Carlos GP, Alberto P, eds. Proc. of the 5th Int'l Conf. on ICA. LNCS 3195, Heidelberg: Springer-Verlag, 2004. 1165−1172.

[17] Eckart C, Young G. A principal axis transformation for non-Hermitian matrices. Bulletin of the American Mathematical Society, 1939,45(2):118−121.

[18] Zhou ZH, Chen ZQ. Hybrid decision tree. Knowledge-Based Systems, 2002,15(8):515−528.

[19] He P, Chen L, Xu XH. Fast C4.5. In: Proc. of the Int'l Conf. on Machine Learning and Cybernetics 2007. 2007. 2841−2846. http://code.google.com/p/fastc45/downloads/list

[20] Newman D, Hettich S, Blake C, Merz C. UCI repository of machine learning databases. 2003. http://www.ics.uci.edu/~mlearn/MLRepository.html

[21] Rasmussen CE, Neal RM, Hinton GE, van Camp D, Revow M, Ghahramani Z, Kustra R, Tibshirani R. Data for evaluating learning in valid experiments. 1996. http://www.cs.toronto.edu/~delve/

[22] Ling L. Data used in machine learning research. 2005. http://www.work.caltech.edu/ling/data/index.html

[23] Jutten C, Guérin-Dugué A, Aviles-Cruz C, Voz JL, Cappel DV. Databases of enhanced learning for evolutive neural architecture project. 1994. ftp://ftpftp.dice.ucl.ac.be/pub/neural-nets/ELENA/databases

附中文参考文献:

[2] 刘奕群,张敏,马少平.基于改进决策树算法的网络关键资源页面判定.软件学报,2005,16(11):1958−1966. http://www.jos.org.cn/1000-9825/16/1958.htm

何萍(1983－),女,江苏太仓人,博士生,主要研究领域为机器学习,数据挖掘.

徐晓华(1979－),男,博士,主要研究领域为机器学习,数据挖掘,并行计算,生物信息学.

陈崚(1951－),男,教授,博士生导师,CCF高级会员,主要研究领域为人工智能,数据挖掘,系统优化.