Improved Binary Tree Support Vector Machines for Multi-class Classification

Yuqi Pan[1] Yanwei Zheng[1]

[1]School of Information Science and Engineering University of Jinan, Jinan 250022, China

Abstract: The binary tree SVMs(Support Vector Machines) algorithm for multi-class classification can solve the unclassifiable regions that exist in conventional multi-class SVMs. At present, the most of binary tree SVMs algorithm create a skew binary tree, so the sample training time and the testing speed is slow. In this paper, an improved binary tree SVMs algorithm for multi-class classification is proposed. This algorithm can create a symmetric binary tree and increase training and testing speed. The results of experiment have proved that this algorithm is feasible and effective.

Keywords: Support Vector Machine, Binary Tree, Muti-class Classification

改进的二叉树支持向量机多类分类算法

潘玉奇[1] 郑艳伟[1]

[1]济南大学 信息科学与工程学院 济南 250022

摘要: 二叉树支持向量机多类分类算法可以解决传统多类支持向量机存在的不可分区域的问题。目前，大多数的二叉树支持向量机算法生成的二叉树是偏态树，使样本训练时间和测试速度都比较慢。本文提出了一种改进的二叉树支持向量机多类分类算法，该算法可以生成一棵正态二叉树，从而提高训练和测试速度。实验结果证明了该算法的可行性与有效性。

关键词: 支持向量机, 二叉树, 多类分类

1. 引言

支持向量机(Support Vector Machine, SVM)是在统计学习理论的基础上发展的一种新的模式识别方法，在解决小样本、非线性及高维模式识别问题中表现出许多特有的优势。最初的 SVM 是用来解决两类分类问题，如何将其有效地推广到多类别分类是当前 SVM 研究的重要内容之一。

现有的多类支持向量机算法大致分为两类：(1) 构造多个 SVM 二值分类器，每个分类器用于识别其中两个类别，并将它们的判别结果以某种方式组合起来实现多类分类。(2) 将多个分类面的参数求解合并到一个最优化问题中，通过求解该最优化问题一次性地实现多类分类。

第二种方法在最优化问题求解过程中未知变量远远多于第一类方法，往往给计算带来困难，且实验证明这

类方法在分类精度上也不占优势。因此现有的大多数方法均属于第一类。例如，一对多方法(one-versus-rest SVMs)[1]，一对一方法(one-versus-one SVMs)[2]，有向无环图支持向量机(Directed Acyclic Graph SVMs)[3]和二叉树支持向量机(Binary Tree SVMs)[4]。

通过对这些多类 SVM 的性能对比[5]，发现将二叉树引入多类分类问题，可以有效地解决“一对多”及“一对一”方法存在的不可分区域。但不同的二叉树结构对 SVM 多类分类器的性能有很大的影响。目前提出的算法所生成的二叉树多是“偏态树”，这种树结构使样本训练时间和测试速度都比较慢，如果能生成“正态树”，则 SVM 分类器将具有更理想的训练和测试速度。

本文提出了一种基于类距离的正态二叉树支持向量机多类分类算法，该方法以聚类分析中的类间距离作为相似性度量函数，以此生成一棵正态二叉树。实验数据证明了该算法的可行性与有效性。

2. 基于二叉树的多类支持向量机

基于二叉树的多类 SVM 首先将所有类别分成两个子类，再将子类进一步划分成两个次级子类，如此循环下去，直到所有的结点都只包含一个单独的类别为止。该方法将原有的多类问题分解成一系列的两分类问题，在二叉树中的所有非叶子结点上使用二值 SVM 进行分类。对于 N 类问题，BT-SVMs 只需要 N-1 个分类器，分类时也不需要遍历所有的分类器就能得到分类结果，因此分类的速度相当快。

二叉树的结构有两种：一种是在每个内结点处，由一个类与其余的类构造分割面；另一种是在内结点处，是多个类与多个类的分割。文献[6,7]提出的算法是基于第一种结构的，这些算法的主要思想是让最易分割的类最早分割出来，即在二叉树的上层节点处分割，但是这样生成的二叉树都是如图 1 所示的偏态二叉树。文献[5,8] 提出的算法中类层次的定义采用自下而上的归并策略，首先，将每个类别视为一个子类,然后根据某种度量将两个子类合并成一个子类，如此循环直到所有的类别合并成一类。当识别一个未知样本时，先从根结点分类器开始，根据输出值选择下一级左结点或右结点分类器，直到抵达某个叶子结点，此时该样本的类型被确定。这种方法生成的二叉树的形态跟具体的测试数据集有关，不能保证所生成的二叉树一定是正态的。

本文提出的算法是采用第二种二叉树结构，即在内结点处是多个类与多个类的分割，各结点的左右子类所包含的类别数之差小于等于 1，这样形成的二叉树是正态的。

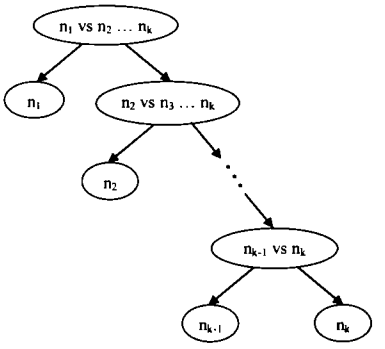


Fig.1: slanting binary tree structure
图 1 偏态二叉树结构

3. 基于类距离的正态二叉树多类支持向量机

3.1. 相似性度量函数

欧氏距离是广泛采用的度量类间相似性的方法,本文以类中心欧氏距离作为

相似性度量函数。假设类 S 有 n 个样本：x_i（$i=1$，...，n），每个样本有 m 个属性因子，第 i 个样本的第 j 个属性因子记为：x_{ij}，则 n 个样本可排成样本数据矩阵：

$$X=\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{bmatrix} \quad (1)$$

一个类的类心是该类别中的所有样本向量的平均值：

$$\bar{x}=\frac{1}{n}\sum_{i=1}^{n}x_i \quad (2)$$

假设训练样本的类别数为 N，用式(2)计算出各类的类心，用式(3)计算任意两类之间的欧氏距离，第 i 类的类心与第 j 类的类心之间的欧氏距离为：

$$d_{ij}=\sqrt{\sum_{k=1}^{m}(\bar{x}_{ik}-\bar{x}_{jk})^2} \quad (3)$$

3.2. 构造正态二叉树

构造正态二叉树的步骤如下：

(1) 对于 N 类问题，对每个类进行编号，按编号从小到大顺序放入集合 S 中，令 S1，S2 为空集；

(2) 如果 S 中只有 2 类，则这 2 类分别作为左、右子树，结束。

(3) 在集合 S 中选出最小的 d_{ij} 值对应的第 i 类和第 j 类，将它们添加到集合 S1 中，S=S-S1；

(4) 在集合 S 中找出与集合 S1 中各类距离和最大的类，将其添加到集合 S2 中，S=S-S2；

(5) 在集合 S 中找出与集合 S2 中各类距离和最小的类，将其添加到集合 S2 中，S=S-S2；

(6) 在集合 S 中找出与集合 S1 中各类距离和最小的类，将其添加到集合 S1 中，S=S-S1；

(7) 重复第(5)、(6)步，直至 S 为空集；至此，二叉树根结点的左右子树已形成，左子树包含 S1 中所有的类，右子树包含 S2 中所有的类；

(8)令 S=S1，重复第(2)—(7)步；同理，令 S=S2，重复第(2)—(7)步；直到每个类成为一个叶子结点为止。

3.3. SVM 分类器

SVM 是从线性可分情况下寻求最优分类面发展而来的。对非线性问题，SVM 通过某种事先选择的非线性映射，将输入向量 x 映射到一个高维的特征空间 H 中。在特征空间H中构造最优分类超平面。训练算法仅使用空间中的点积，即 $\Phi(x_i)\cdot\Phi(x_j)$，而无单独 $\Phi(x_i)$ 出现。因此，若能找到一个函数 K，使 $K(x_i, x_j)=\Phi(x_i)\cdot\Phi(x_j)$，则在高维空间中只需进行内积运算，又因该运算可用原空间中的函数实现，所以也不必知道变换 Φ 的形式。根据泛函的有关理论，只要一种核函数 $K(x_i, x_j)$ 满足 Mercer 条件，它就对应某一变换空间中的内积。常用的核函数有：线性核函数、多项式核函数、径向基核函数和 Sigmoid 核函数。

在已构造好的正态二叉树中的每个非叶子结点设置一个二值 SVM 分类器，对于 N 类问题需要 N-1 个二值 SVM 分类器，所有的 SVM 均采用径向基核函数。SVM 的分类步骤如下：

(1) 已知训练集为：

$$T=\{(x_1,y_1),...,(x_l,y_l)\}\in\{X\times Y\}^l$$
$$x_i\in X=R^n, y_i\in Y=\{-1,1\} \quad (4)$$
$$i=1,...,l$$

(2) 构造并求解以下优化问题，得最优解 $\alpha^* = (\alpha_1^*,...,\alpha_l^*)^T$

$$\max_{\alpha}\{\sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum y_i y_j \alpha_i \alpha_j (x_i \bullet x_j)\} \quad (5)$$

$$s.t.\sum_{i=1}^{l} y_i \alpha_i = 0,\ C \geq \alpha_i \geq 0, i = 1,...,l$$

其中，C 为惩罚系数。

(3) 计算 w^*，选择一个正分量 α_i^*，并据此计算 b^*。

$$w^* = \sum_{i=1}^{l} \alpha_i^* y_i x_i \quad (6)$$

$$b^* = y_i - \sum_{i=1}^{l} \alpha_i^* y_i (x_i \bullet x_j) \quad (7)$$

(4) 构造分类超平面

$$\alpha^*(y_i((w^* \bullet x_i) + b^*) - 1) = 0 \quad (8)$$

由此求得分类函数：

$$f(x) = Sgn\{\sum_{i=1}^{l} \alpha_i^* y_i K(x, x_i) + b^*\} \quad (9)$$

$$K(x, x_i) = \exp(-\frac{(x - x_i)^2}{2\delta^2}) \quad (10)$$

其中，δ 为核参数。

4. 实验

本文的实验数据来自 UCI 数据库(http://www.ics.uci.edu/~mlearn/ML)，选择数据集有：Glass，Shuttle 和 E.coli。表 1 中列出了各个数据集样本数，属性数和类别数。

Table1. experiment dataset
表 1 实验数据集

数据集	Glass	Shuttle	E.coli
样本数	214	43500	336
属性数	9	9	8
类别数	6	7	8

在这 3 个数据集中，Shuttle 有单独的测试集，样本数为 14500，而 Glass 和 E.coli 没有单独的测试集，所以采用 5 倍交叉验证来计算识别率。5 倍交叉验证是指将数据集分成近似相等的 5 份，每次取其中的 4 份作为训练样本，剩下的 1 份作为测试样本。当 5 份数据轮流作为测试样本进行实验后取 5 次的平均识别率作为最后结果。

对 3 个数据集分别采用文献[6]、文献[8]和本文算法进行实验，文献[6]生成的都是形如图 1 所示的偏态二叉树。对 Glass 数据集，文献[8]和本文算法生成的二叉树结构是一样的，如图 2 所示。对 Shuttle 和 E.coli 数据集，文献[8]和本文算法所生成的二叉树结构不同，具体如图 3、图 4 所示。

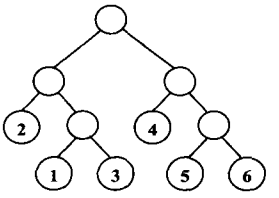
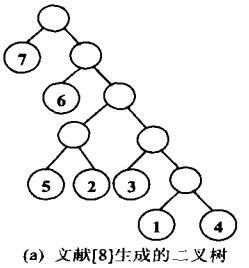
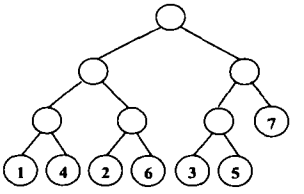


Fig.2: binary tree of Glass dataset
图 2 Glass 数据集对应的二叉树

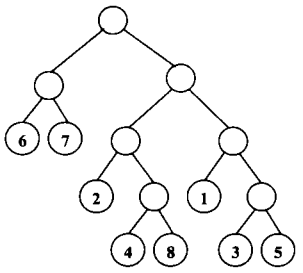


(a) 文献[8]生成的二叉树

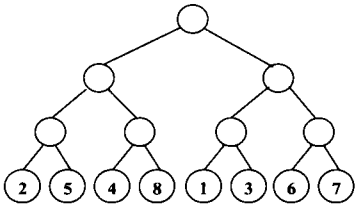


(b) 本文算法生成的二叉树

Fig.3: binary tree of Shuttle dataset
图 3 Shuttle 数据集对应的二叉树



(a) 文献[8]生成的二叉树



(b) 本文算法生成的二叉树

Fig.4: binary tree of E.coli dataset

图 4 E.coli 数据集对应的二叉树

由于 Glass 和 E.coli 数据集的样本数较少，本文算法的训练时间和测试时间与文献[6, 8]算法相比相差不大。但对于 Shuttle 数据集来说，本文算法具有明显的优势。这是因为 Shuttle 数据集的样本数较多，而且各个类别间的样本数差别很大，具体数量见表 2。

Table2. each class sample quantity of Shuttle dataset

表 2 Shuttle 数据集各类别的样本数

类别	训练样本数	测试样本数
1	34108	11478
2	37	13
3	132	39
4	6748	2155
5	2458	809
6	6	4
7	11	2
合计	43500	14500

二叉树 SVM 的训练时间主要取决于参与训练的样本数量，并受二叉树层次结构的影响，当每个类的训练样本数相同时，树的层次越多训练速度越慢。而 Shuttle 数据集各个类别间的样本数量相差很大，且样本数最多的第 1 类位于二叉树的最底层，这样就需要更多的训练时间。观察图 3(a)，类别 1 位于第 6 层，而在图 3(b)中类别 1 位于第 4 层，所以本文算法的训练时间较快。

多类 SVM 算法的分类速度主要受两个因素影响：一是对单个未知样本分类所需分类器的数量，二是分类器中支持向量的数量。对二叉树 SVM 来说，分类器的数量与树的层次结构有关，一个未知样本分类所需要的分类器数量从 1 到 N-1 不等(N 为类别数)。例如，图 3(a)所示的二叉树，若未知样本是第 6 类，则需要经过 2 个分类器，若未知样本为第 1 类，则需要经过 5 个分类器。在图 3(b)所示的二叉树中，若未知样本为第 1 类，则只需要经过 3 个分类器。Shuttle 数据集的测试样本中，约 80%的样本为第 1 类，因此本文算法的分类速度也较快，具体的实验结果见表 3。

Table 3. experiment results of Shuttle dataset

表 3 Shuttle 数据集的实验结果

	训练时间	分类时间	识别率
文献[6]	181s	2.3s	99.92%
文献[8]	165s	1.9s	99.91%
本文	106s	1.1s	99.93%

5. 结语

本文对现有的二叉树支持向量机算法进行了研究，发现基于二叉树的多类 SVM 模型的性能与二叉树的层次结构相关，正态二叉树的训练时间和分类速度都要优于偏态二叉树。于是本文提出了一种根据类距离生成正态二叉树的算法。实验表明，该算法在对大样本多分类问题中的性能表现良好。

6. References

[1] Bottou L, Cortes C, and Denker J, "Comparison of Classifier Methods: a

Case Study in Handwriting Digit Recognition," *In International Confer-ence on Pattern Recognition.[s.l.]: IEEE Computer Society Press,* pp. 77-87, 1994.
[2] Krebel U, "Pairwise Classification and Support Vector Machines," *In Advances in Kernel Methods: Support Vector Learning. MA: The MIT Press*, pp. 255-268, 1999.
[3] Platt J, Cristianini N and Shawe-Taylor J, "Large Margin DAGs for Multiclass Classification," *In Advances in Neural information Processing Systems.[s.l.]: MITPress,* pp. 547-553, 2000.
[4] Solbing F S, "Multi-class Pattern Recognition Problems with Tree-Structured Support Vector Machines," *//LNCS2191: DAGM,* pp. 283-290, 2001.
[5] Bin Hou and Jing-Liang Li, "Comparison of Multi-class SVMs Methods," *Journal of Wuhan University of Technology(Information & Manage-ment Engineering)*, pp. 673-677, 2008. (in Chinese)
[6] Tang Faming, Wang Zhongdong and Chen Mianyun, "An Improved Multiclass Support Vector Machines Based on Binary Tree," *Computer Engineering and Application,* pp. 24-26, 2005. (in Chinese)
[7] Guo Yaqin and Wang Zhenqun, "Improved Multiclass Classification Methods for Support Vector Machine," *Modern Ele-ctronics Technique,* pp. 143-146, 2009. (in Chinese)
[8] Liu Yang and Zhang Qiu-yu, "Multiclass SVM Method Based on Huffman Tree," *Computer Engineering and Design*, pp. 1792-1793, 2008. (in Chinese)