基于多代表点学习的 RSKNN 分类算法①

余 勇, 郭躬德, 陈黎飞

(福建师范大学 数学与计算机科学学院, 福州 350007)

摘 要: RSKNN 算法是一种基于变精度粗糙集理论的 *k*-近邻改进算法, 该算法能够保证在一定分类精度的前提下, 有效地降低分类的计算量, 提高分类效率. 但由于 RSKNN 算法只是简单地将每个类中的样本划分成一个核心和边界区域, 并没有根据数据集本身的特点进行划分, 因而存在极大的局限性. 针对存在的问题, 提出一种多代表点学习算法, 运用结构风险最小化理论对影响分类模型期望风险的因素进行分析, 并使用无监督的局部聚类算法学习优化代表点集合. 在 UCI 公共数据集上的实验表明, 该算法比 RSKNN 算法具有更高的分类精度.

关键词: 近邻分类; 变精度粗糙集; 代表点; 分类模型; 上、下近似

Multi-Representatives Learning Algorithm for RSKNN Classification

YU Yong, GUO Gongde, CHEN Lifei

(School of Mathmatics and Computer Science, Fujian Normal University, Fuzhou, 350007, China)

Abstract: RSKNN is an improved *k*NN algorithm based on variable parameter rough set model. The algorithm guarantees under the premise of a certain classification accuracy, effectively reduces the computation burden of the classified samples, and improves the computation efficiency and precision of classification. But in this algorithm ,the instances of each class are simply classified into core and boundary areas. It has the limitation that it isn't classified according the features of datasets. An efficient algorithm aiming at learning multi-representatives for RSKNN is proposed. Using the theory of structural risk minimization, a few factors that determine the expected risk of new classification model are analyzed. And an unsupervised algorithm for partial clustering is used to build an optimal set of representatives. Experimental results on UCI public datasets demonstrate that the proposed method significantly improves the accuracy of the classification.

Key words: nearest neighbor classification; variable precision rough set; representative; classification model; upper and lower approximation

在数据挖掘中, 分类是一种采用有监督的机器学习方法, 已经被广泛应用于各个领域. 目前, 比较常见的分类算法有 *k*NN、决策树、支撑向量机、神经网络、贝叶斯分类等. 其中 *k*NN 算法以其实现简单和较高的分类精度在众多领域得到广泛应用, 被列为最有影响力的十大数据挖掘算法之一[1].

*k*NN 算法是一种基于实例的分类算法. 搜索训练集中每个样本, 找出最接近待分类样本的 *k* 个训练样本, 通过这 *k* 个训练样本按多数服从少数的投票原则决定出待分类样本的类别. *k*NN 算法虽然简单、有效, 但由于每次都要计算训练集中的每个样本到待分类样本的距离, 并且该过程与训练集的大小成线性关系, 所以速度很慢. 对于训练集规模庞大或维数较高的情况, 寻找 *k* 个最近邻样本计算量庞大, 直接影响分类速度. 针对这个问题, 有不少学者提出了解决的办法: 如李荣陆等[2]提出基于密度对样本进行剪裁的方法, 降低了 *k*NN 的计算量; 余鹰等[3]提出了一种基于变精度粗糙集模型的 *k*NN 改进算法(RSKNN), 该算法基于

①基金项目:国家自然科学基金(61175123)

收稿时间:2014-03-10;收到修改稿时间:2014-04-21

变精度粗糙集理论, 用上、下近似域来刻画各类的分布区域, 有效地降低分类样本的计算量, 并且提高计算效率, 但该算法只适用于球状数据, 有一定的局限性. 此外, *k*NN 算法还存在参数 *k* 难以确定的问题, 针对这个问题, 已提出了多种 kNN 的改进算法. 其中, Guo 等[4-5]提出基于模型的 *k*NN 算法(*k*NNModel), 该算法通过选择代表点建立分类模型, 并且能够在学习过程中自动确定 *k* 的取值; 陈黎飞等[6]提出了一种基于多代表点的学习算法(MEC), 运用结构风险最小化理论对影响分类模型的期望风险的因素进行分析, 并使用无监督的局部聚类算法学习优化代表点集合, 有效解决了 *k* 值难以确定的问题, 并提高了分类效率.

本文提出一种基于多代表点学习的 RSKNN 算法(Multi-Representatives for RSKNN, MRSKNN), 是在 RSKNN 算法的基础上, 沿用变精度粗糙集中上、下近似的概念来刻画每个类的区域范围, 并通过局部聚类算法将每个类划分为若干个簇, 再运用结构风险最小化理论对新分类模型中簇的数目进行优化, 最终得到若干个较适宜的模型簇. 与 RSKNN 算法相比, 将一个类模型划分成多个代表点簇模型, 同时能够自动确定代表点的数目, 能够更好的适应复杂数据集的分类.

1 相关工作

1.1 RSKNN 算法思想

经典粗糙集理论是 Pawlak[7]于 1982 年提出的一种刻画不完整性和不确定性的数学工具, 能有效地分析不精确、不一致、不完整等各种不完备的信息, 还可以对数据进行分析和推理, 从中发现隐含的知识, 揭示潜在的规律. 但由于经典粗糙集会对存在噪声干扰的数据集产生过拟合的现象, 从而降低预测能力. 为了尽量避免这种情况, Ziarko[8]于 1993 年提出了一种变精度粗糙集模型, 他利用相对错误分类率, 在模型中引入一个精度 β, 将集合的上、下近似推广到任意一个精度水平, 这样就使得粗糙集的抗干扰能力大大增强.

RSKNN 算法是就是利用这种变精度粗糙集模型的上、下近似概念分别描述每个类的上、下近似区域, 依据变精度粗糙集理论, 某类的下近似区域(正域)是该类的位置和大致形状的核心部分; 而落在上近似边界区域内的样本实例的类别归属是不确定的[2]. RSKNN 算法对待测样本进行分类时, 就是判断待该测试样本在模型中的位置: 若该待测样本位于某个类的下近似区域中, 则直接判定该待测样本属于该类; 若该待测样本位于某些类的边界区域中, 则只需在这些类的边界区域(即上近似区域)中寻找 *k* 个最近邻来判定该待测样本的类别归属; 若该待测样本位于所有类的边界域之外, 则该待测样本属于距离上近似边界最近的那个类.

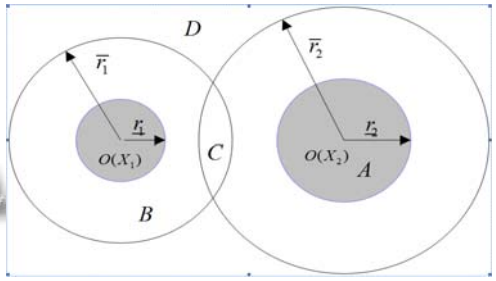

图 1 两类样本上近似区域相交

如图 1, 两类训练样本集合 X_1 和 X_2, 中心点分别为 *O(X1)*和 *O(X2)*, 类 X_1 的上、下近似半径分别为 $\overline{r_1}$ 和 $\underline{r_1}$, 类 X_2 的上、下近似半径分别为 $\overline{r_2}$ 和 $\underline{r_2}$. 图中阴影部分为各类的下近似区域, 其中有 β*n 个异类样本(n 为某类别中样本实例数). 图中所示为两类样本的上近似区域相交的情况. RSKNN 算法中对于类 X_i 的上、下近似半径的计算如下[3]:

Step1 计算类 X_i 的中心点 $O(X_i)$, 并计算类 X_i 的所有训练样本到中心点 $O(X_i)$ 的距离, 按升序排列存放在 D_i 中;

Step2 将 D_i 中的最大值作为类 X_i 的上近似半径, 即 $\overline{r_i} = \max(D_i)$;

Step3 查找其他类中位于类 X_i 上近似区域内的所有样本, 并根据与 $O(X_i)$ 的距离插入到 D_i 中;

Step4 从 D_i 中按序找出前 k 个不同于类 X_i 的样本, 若 $\frac{Num(k)}{n} \le \beta$, $\underline{r_i} = dist(v_k, O(X_i))$, 否则停止下近似半径的计算.

RSKNN 算法根据测试样本在模型中的位置确定其类别, 其分类步骤如下[3]:

Step1 计算每个类的上、下近似半径;

Step2 根据欧氏距离公式计算待分类样本 v 与各类中心点的距离, 确定待分类样本在模型中的位置;

Step3 如果 v 在某类的正域中, 那么就直接判定 v

属于此类, 算法结束;

Step4 如果 v 在某些类的边界区域中, 那么只需要在这些类的上近似区域内寻找 k 个最近邻. 计算完毕后, 依据投票原则确定 v 的类别归属, 算法结束;

Step5 如果 v 位于所有类的边界域之外, 则 v 属于距离上近似边界最近的那个类, 算法结束.

2 基于多代表点学习的RSKNN分类算法

基于多代表点学习的 RSKNN 分类算法, 简称 MRSKNN, 它是在多代表点学习和 RSKNN 算法的基础上提出的一种改进算法. 主要解决 RSKNN 算法对球状数据集的依赖性, 算法依旧沿用 RSKNN 算法中利用变精度粗糙集理论的上、下近似的概念, 对于每个类模型簇的划分采用了 *k*-Means 局部聚类[9]的方法, 并结合风险结构最小化理论对模型进行优化学习, 最终得到每个类下较适宜的多个代表点的模型簇.

2.1 分类模型

MRSKNN 的分类模型是从给定训练集 $Tr=\{(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)\}$ 学习得到的优化的分类模型 $\{M_1,M_2,\ldots,M_k\}$, 其中, x_i 表示第 i 个训练样本; y_i 是 x_i 的类别标号, K 是类别数目; $M_i=\{p_{i1},p_{i2},\ldots,p_{i\alpha}\}$, M_i 表示类 X_i 优化得到模型簇集合, 其每个元素 p_{il} 表示类 X_i 下的第 l 个模型簇, α 是类 X_i 所划分的模型簇的数目, 模型簇 p_{il} 中的类 X_i 样本点的集合记为 C_{il}.

模型簇 p_{il} 用一个四元组表示: $p_{il}=(O(X_{il}),\underline{r}_{il},\overline{r}_{il},class_{il})$, $l=1,2,\ldots,\alpha$, 其中, $O(X_{il})$ 为 p_{il} 的代表点; $\underline{r}_{il}$ 为 p_{il} 的下近似半径; $\overline{r}_{il}$ 为 p_{il} 的上近似半径; $class_{il}$ 为 p_{il} 的类别标号.

每个模型簇 p_{il} 的代表点 $O(X_{il})$ 是该模型簇覆盖范围内所有样本点的中心, 又称为中心点, 该中心点是一个 “虚拟点”, 可以采用如下公式进行计算:

$$O(X_{il})=\frac{1}{|C_{il}|}\sum_{x\in c_{il}}x \tag{1}$$

其中, $|C_{il}|$ 表示 C_{il} 集合包含的样本数.

每个模型簇 p_{il} 的上近似半径 $\overline{r}_{il}$ 和下近似半径 $\underline{r}_{il}$ 的计算方法依然采用 RSKNN 算法.

MRSKNN 分类算法步骤如下:

分类算法 MRSKNNTesting

输入 分类模型 $\{M_1,M_2,\ldots,M_k\}$, 待分类样本 x_t, 参数 *K*, 精度 β

输出 x_t 的类别

Begin

Step1 计算待分类样本 x_t 到分类模型各个簇的中心点的距离, 确定待分类样本在模型簇区域中的位置.

Step2 如果 x_t 在某个簇的正域中, 则直接判定 x_t 属于此类, 算法结束.

Step3 如果 x_t 在某些簇的边界域中, 则在这些簇的上近似区域内寻找 k 个最近邻, 依据投票原则判定 x_t 的类别归属, 算法结束.

Step4 如果 x_t 位于所有类的边界域之外, 则将 x_t 归属于距离上近似边界最近的那个簇, 算法结束.

End

MRSKNNTesting 分类阶段对待分类样本进行分类时, 只需要计算待分类样本到各模型簇的代表点的距离, 并依此判定待分类样本在模型簇中的位置, 进行分类. 相对于RSKNN算法, 该算法模型簇的数目会有所增加, 而每个模型簇的相对区域会有所减小, 尤其是在上近似区域中进行 *k*-近邻分类时, 区域中样本点数量会有所减少, 计算量必然会有所下降. 而对于传统的 *k*NN 算法, 需要计算 n 次的距离并排序. 因此相对于传统的算法, 距离计算的工作量以及排序的工作量均会减少. 由于在下近似区域中的样本点尽包含极少量的异类样本点, 故在下近似区域的分类准确率是非常高的; 在分类之前就已经根据数据集的特点, 将各个类的样本点进行了局部聚类, 并进行优化, 这就减少了在上近似区域中的异类样本点, 从而使得上近似区域分类准确率的提高.

2.2 结构风险

文献 [6] 提出了一种 *k*-分类模型, 即 $M_k=\{p_l\,|\,l=1,2,...,\alpha\ and\ class(l)=k\}$, 对于给定的一个分类样本 x_t, 应用 $M_k(k=1,2,...,K)$ 对其进行 K 次二类分类, 其第 k 次分类的目的是判断 x_t 是否属于类别 k. 因而对 x_t 分类的结果只有两种: 分类正确, 值为 k; 分类错误, 值为 0. 即

$$\Pr ediction(x_t,M_k)=\begin{cases}k, & x_t\text{正确分类}\\ 0, & otherwise\end{cases} \tag{2}$$

本文提出的分类模型 M_k 对未知样本进行分类其实就是一个二类分类过程, 可以利用 *VC* 维理论[10]对

分类模型进行结构风险分析. 文献[10]给出了二类分类模型期望风险的上界, 即:

$$R(M_k) \le R_{emp}(M_k) + VC\_confidence(h_k)$$

其中, $R_{emp}(M_k)$ 表示经验风险, 是分类模型在训练时产生的平均误差; h_k 表示 M_k 的 *VC* 维; *VC_confidence(h_k)*表示 *VC* 置信度. 在 MRSKNN 算法中, M_k 的经验风险可以表示为:

$$R_{emp}(M_k) = \frac{1}{n}\left(\sum_{(x,y)\in Tr, y=k} I(k \ne \Pr ediction(x, M_k)) + \sum_{(x,y)\in Tr, y\ne k} I(k = \Pr ediction(x, M_k))\right) \quad (3)$$

由上式可知, 为了提高分类性能, 就应当尽量降低分类模型的期望风险. 当精度 β 一定时, 对于给定的训练集 , M_k 的经验风险与模型簇数目 $|M_k|$ 有关. 考虑两种极端情况: 当 $|M_k|=1$ 时, MRSKNN 其实就是 RSKNN 算法, 此时分类模型具有最大的经验风险; 当 $|M_k|$ 取最大值时, 即训练集中类别标号为 *k* 的每个样本点都构成一个模型簇, 此时分类模型的经验风险可降到最小值 0. 可以看出, 一个很大的 $|M_k|$ 取值将导致分类模型预测风险的增加, 而模型的经验风险会随着 $|M_k|$ 的增大呈下降趋势. 但 $R_{emp}(M_k)$ 不一定是关于 $|M_k|$ 单调递减函数, 在分类模型的经验风险从最大值变化到最小值 0 的过程中, 一定存在若干个局部极小点[6]. 因此在模型簇数目变化的过程中, 应找出 M_k 经验风险的局部极小点.

对学习矢量量化(LVQ)算法的 *VC* 维研究可知, LVQ 的 *VC* 维是其 “原型”数目的单调递增函数[11]. 结合本文中对模型簇的定义, M_k 可以看做是 LVQ 中“原型”的一种扩展. 由文献[10] 中对 *VC* 置信度的定义可知, *VC* 置信度是 *VC* 维的递增函数, 因此, M_k 的 *VC* 置信度是 $|M_k|$ 的单调递增函数.

综上所述, $R(M_k)$ 的上限由经验风险和 *VC* 置信度共同决定. 因而, MRSKNN 模型训练算法就是要找出一个优化的模型 M_k 以尽量降低模型的期望风险, 取得二者之间的某种平衡.

2.3 模型训练算法 MRSKNNTraining

假定训练集中的实例有 *K* 个类别, 模型训练过程中调用 *K* 次训练算法 MRSKNNTraining, 为每个类别建立优化的 *k*-分类模型. MRSKNN 算法步骤如下:

训练算法 MRSKNNTraining

输入 训练集 *Tr*, 待生成分类模型的类别标号 $k(k=1,2,\ldots,K)$

输出 分类模型 M_k

Begin

Step1 构造初始模型 $M_k=\{p_{k1}\}$, $|M_k|=1$. 令 C_1={*Tr* 中类别标号为 *k* 的样本}, 利用 RSKNN 算法中计算上、下近似半径的方法求出模型簇的上、下近似半径, 并构造初始模型簇 $p_{k1}=\{O(X_{k1}), \underline{r}_{k1}, \overline{r}_{k1}, k\}$, 使用式(3)计算经验风险 $R_{emp}(M_k)$.

Step2 如果 $R_{emp}(M_k)=0$ 或者 $|M_k|$ 取最大值, 返回 M_k, 算法结束.

Step3 使用 *k*-Means 聚类算法[9]对 C_1 的训练样本进行局部聚类, 划分成 $|M_k|+1$ 个簇.

Step4 构造模型 $M_k' = \{p_{kj} \mid j=1,2,...,|M_k|+1\}$ 和模型簇 $p_{kl}=\{O(X_{kl}), \underline{r}_{kl}, \overline{r}_{kl}, k\}$.

Step5 使用式(3)计算经验风险 $R_{emp}(M_k')$. 如果 $R_{emp}(M_k) \le R_{emp}(M_k')$, 返回 M_k, 算法结束; 否则, $M_k = M_k'$, 重复执行 Step3 ~ Step5.

End

MRSKNNTraining 算法的原理是通过 *k*-Means 聚类算法[9]对每个类别 $k(k=1,2,\ldots,K)$ 下的实例进行局部聚类, 构造模型簇, 并估算分类模型的经验风险, 直到经验风险不再下降或者降为 0 为止. 这不同于 RSKNN 算法, RSKNN 算法只是在每个类别下根据粗糙集理论划分成一个簇区域, 并未考虑整个模型的经验风险. 相比较 RSKNN 算法, 本文提出的算法在数据集的处理上将投入更多的计算成本. 上文中对分类模型经验风险的分析, 模型的经验风险会随着模型簇 $|M_k|$ 的数目增大而呈现减小的趋势. 为取得某种平衡, 本文采用文献[6]提出的策略: 每个类别选择对应经验风险第一个极小值.

MRSKNNTraining 算法每次执行 *k*-Means 聚类[9]的时间复杂度为 $O(n_k \mid M_k \mid)$, 假设算法结束时模型簇的平均数目为 α^k, 则共进行了 α^k -1 次模型构造, 总的时间复杂度为 $O(n_k {\alpha_k}^2)$, 每次进行模型构造的时间复杂度平均为 $\alpha_k O(nlgn)$. 对于给定 *K* 个类别的训练集, MRSKNNTraining 将执行 *K* 次, 故训练过程总的时间复杂度为 $K\alpha_k O(nlgn)$. 由于在训练集 *Tr* 中, 类别数 *K* 和每个类别 *k* 下所划分的模型簇的数目 α_k, 都是独立于 *n* 的常数, $K << n$, $\alpha_k << n$.

3 实验与分析

3.1 实验数据和实验环境

实验使用了 10 个数据集进行测试, 均来自 weka 官网的 UCI 数据集(http://www.cs.waikato.ac.nz/ml/weka), 数据均为 arff 格式, 为了保证数据集的多样性, 数据集中的实例个数, 属性数目, 类别数目以及类分布情况都有所甄选, 具体如表 1 所示.

表 1　数据集相关信息

数据集	实例个数	属性数目	类别数目	类分布
Iris	150	4	3	50:50:50
Wine	178	13	3	59:71:48
Ecoli	336	7	8	143:77:52:35:20:5:2:2
Glass	214	9	7	70:76:17:0:13:9:29
Breast-w	683	9	2	444:239
Ionosphere	351	34	2	116:225
Liver-disorders	345	6	2	145:200
Segment	2310	19	7	330:330:330:330:330:330:330
Tae	151	5	3	49:50:52
Sonar	208	60	2	97:111

实验中所采用的电脑配置为: CPU 为 Pentium(R) Dual-Core ,CPU E5300 2.60GHz, 内存 2GB, Windows XP 操作系统, Eclipse 开发平台, JDK1.7, WEKA 应用程序接口. 为了验证算法的有效性, 实验中引入 *k*NN 算法和 RSKNN 算法进行对照.

3.2 实验结果和分析

实验中采用十折交叉验证方法, 即通过随机抽样的方法将数据集等分成十份, 每次轮流选择其中的一份作为测试集, 其余九份作为训练集, 每次都以这样的测试集和训练集进行试验, 进行十次后算出算法的平均准确率. 对比算法中每次都保证了相同的测试集和训练集.

实验过程中, k(上近似区域中测试样本的最近邻数)取值为 10、20、30, 精度 β 取值为 0.1、0.05、0. 表 2 为不同 k 和 β 时三种算法在 10 个数据集上的分类准确率; 表 3 为不同分类器的平均训练时间和分类时间.

表 2　k 取不同值时三种算法的准确率对比

k	数据集	kNN 算法	RSKNN 算法			MRSKNN 算法		
			β=0.1	β=0.05	β=0	β=0.1	β=0.05	β=0
10	iris	96.67	92.67	94.00	95.33	95.33	96.67	96.67
	wine	71.60	65.81	65.83	66.47	66.47	74.98	76.88
	ecoli	86.01	82.41	83.66	83.61	83.61	84.55	84.55
	glass	63.16	64.83	65.08	65.08	65.08	68.79	68.79
	breast-w	96.92	96.50	96.50	97.37	97.37	97.37	97.37
	ionosphere	83.19	81.57	82.66	82.66	82.66	85.19	85.19
	liver-disorders	65.96	64.70	65.96	65.96	65.96	68.09	68.09
	segment	92.64	91.05	91.93	92.06	92.06	95.80	95.80
	tae	39.83	33.71	33.00	34.50	34.50	56.92	57.58
	sonar	73.07	62.90	65.90	65.90	65.90	79.09	80.76
20	iris	94.67	92.67	92.67	93.33	95.33	95.33	95.33
	wine	70.20	71.80	71.86	72.42	71.86	74.64	74.64
	ecoli	84.79	82.10	83.87	83.01	83.96	84.77	85.11
	glass	64.00	64.46	64.46	64.60	66.36	66.39	67.84
	breast-w	96.94	96.35	96.49	96.49	96.63	96.63	97.07
	ionosphere	83.48	82.57	82.94	82.94	84.05	84.05	84.05
	liver-disorders	65.71	65.62	65.05	66.53	66.97	68.57	68.67
	segment	91.13	90.02	90.02	91.13	95.41	95.50	95.50
	tae	37.75	35.75	35.75	35.75	54.33	54.33	54.33
	sonar	68.33	65.50	64.33	66.40	71.67	72.48	72.07

30	iris	94.00	92.67	93.33	93.33	94.00	94.00	95.33
	wine	70.10	71.90	71.27	72.39	72.42	73.56	73.50
	ecoli	82.74	82.13	83.32	83.06	82.14	82.14	85.44
	glass	61.23	63.39	63.39	63.39	64.96	64.87	65.48
	breast-w	96.49	95.74	96.49	96.49	96.34	96.78	97.22
	ionosphere	81.22	81.78	82.63	82.63	83.18	84.02	84.02
	liver-disorders	67.82	66.02	66.02	66.23	66.97	68.72	69.53
	segment	89.65	87.94	88.49	89.06	94.37	94.55	94.55
	tae	36.46	31.17	32.46	33.21	47.67	47.08	47.08
	sonar	67.24	64.93	65.91	66.43	69.21	70.81	70.24

表 3 三种分类器的平均训练/分类时间对比

分类器名称	iris	wine	ecoli	glass	breast-w	ionosphere	Liver-disorders	segment	tae	sonar
*k*NN	0/13.5	0/15.3	0/16	0/16	0/37	0/34	0/17	0/542	0/16	0/27
RSKNN	121/1.3	152/1.6	210/6.2	241/5.5	321/2.8	224/6.9	196/4.0	4085/83.6	142/2.0	193/1.9
MRSKNN	140/0.4	234/0.8	339/2.4	305/2.1	403/1.2	396/2.7	278/1.9	6468/35.6	203/0.6	314/0.5

从表 2 的数据可以看出, 10 个数据集的实验中, MRSKNN 算法在 8 个数据集上的分类准确率要明显高于 RSKNN 和 *k*NN 算法, 尤其在 tae 和 sonar 两个数据集上, 当 RSKNN 算法的分类准确率低于 *k*NN 算法时, MRSKNN 算法依然保持很好的效果. 这说明文中提出的基于多代表点的改进思想是有效的. 对于高维数据集, 如 ionosphere、segment 和 sonar, 维数都是在几十维以上. 从实验结果可以看出, MRSKNN 算法在这些高维数据集上也有很好的表现.

RSKNN 算法只适用于球状数据集[3]. 针对不规则的数据集(非球状), 这些数据集各个类间的样本可能存在严重的交互重叠现象, 这就使得 RSKNN 算法的分类准确率很差. 如数据集 tae 和 sonar, RSKNN 算法的准确率明显低于 *k*NN 算法, 而 MRSKNN 算法却有很好的表现. MRSKNN 算法以降低分类器的期望风险为目标, 通过训练过程学习优化的模型簇集合, 使之可有效处理这样具有复杂类别结构的数据. 从实验结果可以看出, MRSKNN 算法对这种复杂类别结构的数据集有很好的分类准确率.

从表 2 还可以看出, 当精度 β 从 0.1 逐渐减小为 0 时, MRSKNN 和 RSKNN 算法的下近似区域逐渐缩小, 算法的准确率逐渐提高. 对于 RSKNN 算法, 选择合适的 β 时, 算法的准确率基本等同于 *k*NN; 而对于 MRSKNN 算法, 选择合适的 β 时, 算法的准确率要明显高于 *k*NN. 原因在于, 对于各个类划分较适宜的类簇模型区域后, 相似性比较只在相似的类簇模型区域中进行, 从而排除了其他非相似类的干扰. 对于某些数据集的分类准确率低于 *k*NN 算法, 是因为在簇模型中引入了精度 β, 下近似区域中包含了不属于该类的样本, 这与 RSKNN 算法的理论依据是一致的.

表 3 显示不同分类器训练阶段和分类阶段使用的 CPU 时间. 从表三中可以看出, *k*NN 算法在分类阶段需要更多的时间, 这是因为 *k*NN 是一种“懒”分类器, 没有训练阶段, 在分类时需要在所有的训练样本中搜索最近邻. 比较 MRSKNN 和 RSKNN 两种算法, MRSKNN 算法在训练阶段需要更多的时间开销, 而在分类阶段使用的 CPU 时间相对较少. MRSKNN 算法在模型构造时需要对类样本进行局部聚类, 还要对模型进行经验风险分析, 加大了对 CPU 的开销; 在分类时, 由于每个类别划分成多个代表点的模型簇, 每个模型簇的近似区域相对减小, 而在上近似区域进行 *k*-近邻分类时, 样本数量减少, 计算量下降. 故在进行批量分类时, MRSKNN 算法效率远好于传统算法.

4 结语

本文提出一种基于多代表点学习的 RSKNN 改进算法, 利用局部聚类和变精度粗糙集上、下近似的概念来刻画每个类多个簇模型区域, 依据结构风险最小化理论, 以降低模型簇的期望风险为目标, 得出了一个较适宜的类模型簇数目. 在 UCI 数据集上的实验结果表明算法的可行性, 并且与理论分析是一致的. 下一步的工作重点是如何围绕粗糙集来更好的刻画各类

的分布, 以及降低在训练模型中刻画多代表点的成本.

参考文献

1 Yang Q, Wu X. 10 challenging problems in data mining research. International Journal of Information Technology & Decision Making, 2006, 5(4): 597–604.

2 李荣陆,胡运发.基于密度的kNN文本分类器训练样本裁剪方法.计算机研究与发展,2004,41(4):539–545.

3 余鹰,苗夺谦,刘财辉,等.基于变精度粗糙集的 KNN 分类改进算法.模式识别与人工智能,2012,25(4):617–623.

4 Guo G, Wang H, Bell D, et al. KNN model-based approach in classification. On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE. Springer Berlin Heidelberg, 2003: 986–996.

5 Guo G, Wang H, Bell D, et al. Using kNN model for automatic text categorization. Soft Computing, 2006, 10(5): 423–430.

6 陈黎飞,郭躬德.最近邻分类的多代表点学习算法.模式识别与人工智能,2012,24(6):882–888.

7 Pawlak Z. Imprecise Categories,Approximations and Rough Sets. Springer Netherlands, 1991.

8 Ziarko W. Variable precision rough set model. Journal of Computer and System Sciences, 1993, 46(1): 39–59.

9 Kotsiantis S, Pintelas P. Recent advances in clustering: A brief survey. WSEAS Trans. on Information Science and Applications, 2004, 1(1): 73–81.

10 Burges CJC. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 1998, 2(2): 121–167.

11 Crammer K, Gilad-Bachrach R, Navot A, et al. Margin analysis of the LVQ algorithm. NIPS. 2002,2: 462–469.