支持向量机分类算法研究

周宽久,张世荣
ZHOU Kuan-jiu,ZHANG Shi-rong

大连理工大学 软件学院,辽宁 大连 116620
Software School,Dalian University of Technology,Dalian,Liaoning 116620,China
E-mail:Zhoukj@dlut.edu.cn

ZHOU Kuan-jiu,ZHANG Shi-rong.Support vector machines based classification algorithm.Computer Engineering and Applications,2009,45(1):159-162.

Abstract: The accuracy of classification of SVM in a two-class classification problem would be decreased because of those promiscuous samples.KCNN-SVM is proposed in this paper as an improved NN-SVM algorithm,which prunes a sample according to their nearest neighbor's class label as well as the average distance in kernel space between it and its k congener nearest neighbors.Experimental results show that KCNN-SVM algorithm is better than both SVM and NN-SVM in accuracy of classification and the total training and testing time is comparative to that of NN-SVM.

Key words: Support Vector Machine;kernel space;text categorization

摘 要:支持向量机在处理两类分类问题时,当两类样本混杂严重时会降低分类精度。在NN-SVM分类算法的基础上,通过计算样本点与其最近邻点类别的异同以及该点与其k个同类近邻点在核空间的平均距离修剪混淆点,进而提出了一种改进的NN-SVM算法——KCNN-SVM。实验数据表明,KCNN-SVM算法与SVM以及NN-SVM相比,有着更高的分类精度和更快的训练、分类时间。

关键词:支持向量机;核空间;文本分类
DOI:10.3778/j.issn.1002-8331.2009.01.050 **文章编号**:1002-8331(2009)01-0159-04 **文献标识码**:A **中图分类号**:TP391

1 引言

支持向量机((Support Vector Machine,SVM)是20世纪90年代中期在统计学习理论基础上发展起来的一种新型机器学习方法[1]。支持向量机采用结构风险最小化准则(Structural Risk Minimization,SRM)训练学习机器,其建立在严格的理论基础之上,较好地解决了非线性、高维数、局部极小点等问题,成为继神经网络研究之后机器学习领域新的研究热点[2]。支持向量机从提出、被广泛重视到现在只有几年的时间,其中还有很多尚未解决或尚未充分解决的问题,在应用方面还具有很大的潜力。因此,支持向量机是一个十分值得大力研究的领域。

尽管支持向量机追求的目标是较强的泛化能力,但相对于具体的样本集,也可能出现神经网络遭遇的过学习问题。例如,当两类样本集混叠严重时,SVM的决策面可能由于构造的过分复杂反而降低了他的泛化能力。文献[3]提出了一种ESVM(Editing Supporting Vector Machines),其基本做法是:首先用SVM对训练集学习得到决策边界,去掉决策边界附近一定区域内的样本以及错分的样本,然后再对新训练样本集重新用SVM学习得到新的决策边界。在必要的情况下,对最初的训练样本集用新决策边界编辑,去掉错分的样本,得到另一个新的训练集再对他训练得到更新的决策边界。文献[4]的做法较为复杂,需要反复使用SVM训练;文献[5]提出了另一种改进的SVM——NN-SVM(Nearest Neighbor-Support Vector Machine):他先对训练集进行修剪,根据每个样本与其最近邻(nearest neighbor)类标的异同决定其取舍,然后再用SVM训练得到分类器。相比文献[4]的做法,NN-SVM提高了训练及分类时间,但该修剪方法过于简单,容易造成支持向量的误删除,从而降低分类精度。因此,寻找一种更有效的混淆点修剪算法对于提高支持向量机分类精度与速度都是有重要意义的。

2 基于k个同类近邻的SVM分类算法(KCNN-SVM)

2.1 NN-SVM算法

在训练分类器时,SVM算法的着眼点在于两类的交界部分,那些混杂在另一类中的点往往无助于提高分类器的性能,反而会大大增加训练器的计算负担,同时他们的存在还可能造成过学习使泛化能力减弱。文献[5]提出了一种改进的SVM——NN-SVM:他先对训练集进行修剪,根据每个样本点与其最近邻点类标的异同决定其取舍,然后再用SVM训练得到分类器。其修剪策略为:首先找出每一个样本点的最近邻点,然后对每一个样本点,如果该点与其最近邻属于同类,则保留此点;如果该点与其最近邻点属于异类,将该点删除。采用欧式距离作为

基金项目:国家自然科学基金(the National Natural Science Foundation of China under Grant No.70431001)。
作者简介:周宽久(1966-),博士,研究兴趣包括:人工智能,机器学习等。
收稿日期:2007-12-26 **修回日期**:2008-03-18

两个向量之间的距离,即设 $x_i=(x_i^1,x_i^2,\cdots,x_i^n)$,$x_j=(x_j^1,x_j^2,\cdots,x_j^n)$,则 x_i 与 x_j 之间的距离定义为:

$$D(x_i,x_j)=\sqrt{\sum_{k=1}^{n}(x_i^k-x_j^k)^2} \tag{1}$$

经过实验对比分析发现,NN-SVM 算法的修剪策略对于不同的数据集并不是完全适合,例如数据集中的样本点分布如图 1 所示。

由于 NN-SVM 算法仅仅根据样本点与其最近邻点类别的异同删除混淆点,对于图 1 的情形,负类点 x_1 与 x_2 的最近邻点都是正类点 x_3,由于他们的类标号相异,因此 x_1 与 x_2 被修剪算法认为是混杂点而删除。但是通过观察可以发现,样本点 x_1 与 x_2 不仅不是混淆点,相反他们是构造最优分类超平面的支持向量点,如果将他们删除,则新建立的分类超平面与原来的分类超平面有较大的误差,因此,新建立的学习机对分类的精度会产生负面的影响,结果如图 2 所示。

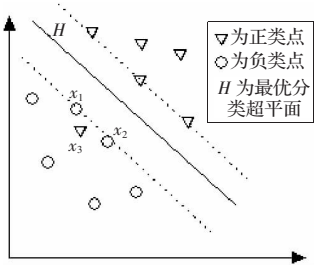

图 1 不适合 NN-SVM 修剪的情形

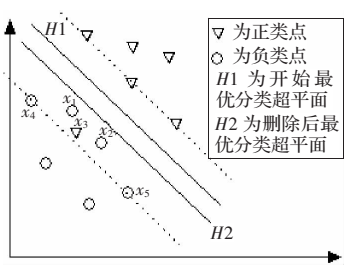

图 2 NN-SVM 修剪后的情形

从图 2 中可以看到将本来是支持向量的样本点 x_1 与 x_2 删除后,最优分类超平面向负类点方向移动了,此时负类的支持向量变为了样本点 x_4 与 x_5,显然此时的最优分类超平面并不能正确反映该数据集原始的几何分布。这时通过 NN-SVM 分类算法训练得到的学习机容易将靠近原来最优分类超平面的负类样本点错误地分类为正类样本点。与此类似,如果 NN-SVM 分类算法误删除了位于正类点中的支持向量,那么分类时将把靠近原来最优分类超平面的正类样本点错误地分类为负类样本点。其原因就在于 NN-SVM 修剪算法可能误删除支持向量,构造了不合理的分类超平面。

通过图 1 和图 2 可以得出结论:NN-SVM 算法的修剪策略对于不同的数据集并不是完全适合。例如,在一个正负两类分类问题中,一个正类点周围都是分布密集的负类点而且他们都以该正类点为最近邻点,那么通过 NN-SVM 算法进行修剪时,就会将这些负类点删除。而此时,显然孤立位于负类点中的该正类点作为混淆点应该被删除,而其周围密集的负类点作为可能对构造分类面有贡献的点不应该被删除。此时,NN-SVM 算法就不适合修剪混淆点了。

2.2 KCNN-SVM 算法提出

通过上一节的分析,必须改进 NN-SVM 的修剪方法才能避免误删除的情形。由于 NN-SVM 删除混淆点的规则过于简单,本文充分考虑了同类点之间分布的关系提出了以下的修剪策略:

首先找出每个样本点 x_i 的最近邻点 x_{ne},判断 x_i 与 x_{ne} 的类别是否一致:如果一致则不做处理;如果不一致,则计算 x_i 到其最近 k 个同类点的平均距离 d_i 以及 x_{ne} 到其最近 k 个同类点的平均距离 d_{ne},如果 $d_i>d_{ne}$,删除 x_i,反之删除 x_{ne}。

通过选择不同的 k 值,新的样本点修剪策略在不同程度上考虑了每个样本点的同类点对其分布的影响,可以有效地防止类似于上边所分析的误删除样本点的情形。例如在图 2 中取 $k=1$,首先判断出 x_1 的最近邻点是 x_3 后,并不是接着删除样本点 x_1,而是计算 x_1 与 x_3 分别到其最近邻同类点的距离,经过比较发现 x_3 到其同类点的距离要大于 x_1 到其同类点的距离,所以将 x_3 删除;对于 x_2 也是同样的处理。显然新的修剪策略比 NN-SVM 中的修剪算法更加准确。

在 NN-SVM 算法中是用欧式距离作为两个样本向量之间的距离,而利用 SVM 训练得到的最优分类超平面是建立在核空间(即映射后的特征空间)之上的。因此,不像 NN-SVM 算法用原始空间的欧式距离度量两个样本点向量之间的距离,而是采用核空间的欧式距离度量两个样本点向量之间的距离。为了计算样本点在核空间的欧式距离,引入核函数[6]的定义:

定义 1(核或正定核) 设 χ 是 R^n 中的一个子集。称定义在 $\chi\times\chi$ 上的函数 $K(x,y)$ 是核函数(核或正定核),如果存在着从 χ 到某一个 Hilbert 空间 H 的影射:

$$\chi\rightarrow H$$
$$x\rightarrow\phi(x)$$

使得 $K(x,y)=\phi(x)\cdot\phi(y)$,其中 $\phi(x)\cdot\phi(y)$ 表示空间 H 中向量的内积。

核技巧是支持向量机的重要组成部分,他把高维 Hilbert 空间中两个点的内积计算,用原来输入空间的两个模式的简单函数的求值来代替,而所利用的简单函数就是定义 1 中的核或正定核。从而,在 KCNN-SVM 算法中计算样本点向量的内积运算可以利用核函数来求得。通过定义 1 可以求出原始空间中两个样本点 x,y 在核空间 H 中的距离 $D(x,y)$,如式(2)所示:

$$\begin{aligned}D(x,y)&=\sqrt{\|\Phi(x)-\Phi(y)\|^2}=\\&\sqrt{[\Phi(x)-\Phi(y)]\cdot[\Phi(x)-\Phi(y)]}=\\&\sqrt{\Phi(x)\cdot\Phi(x)+\Phi(y)\cdot\Phi(y)-2\Phi(x)\cdot\Phi(y)}=\\&\sqrt{K(x,x)+K(y,y)-2K(x,y)}\end{aligned} \tag{2}$$

通过公式(2)可以利用原始样本空间中向量的内积运算代入核函数求得映射后核空间中样本点间的距离,其中的核函数可以采用核函数如公式(3)~(5)所示:

多项式核函数:$k(x,y)=[x\cdot y+1]^d$ (3)

径向基核函数:$k(x,y)=\exp\{-\gamma\|x-y\|^2\}$ (4)

Sigmoid 核函数:$k(x,y)=\tanh(\gamma x\cdot y-\theta)$ (5)

在本节介绍的 KCNN-SVM 修剪策略中,样本点的同类近邻点是通过计算原始空间中的欧式距离获得的,而我们需要的是通过映射后的核空间中的欧式距离度量向量之间的距离。因此,介绍定义 2 描述训练集中一个样本点在核空间中的 k 个同类近邻样本点:

定义 2(KNN(x)) x 为原始训练集中的一个样本点,定义 KNN(x)为包含 x 在核空间中的 k 个同类近邻的集合,集合 KNN(x)是通过式(1)计算核空间中样本点间的欧式距离得到的。

通过定义(1)和定义(2)分别求出样本点在核空间中的距离以及样本点在核空间的近邻集合,由此给出 KCNN-SVM 分类算法如下:

算法输入:给定训练样本集$(x_1,y_1),(x_2,y_2),(x_3,y_3),\cdots,(x_m,y_m)$,$x\in R^n$,$y\in\{1,-1\}$设初始训练集为 T,将训练集 T 表示

为矩阵 $\boldsymbol{TR}_{m\times(n+1)}=[\boldsymbol{XY}]$,其中 $\boldsymbol{X}=\begin{bmatrix}x_1\\ \vdots\\ x_m\end{bmatrix}$,$\boldsymbol{Y}=\begin{bmatrix}y_1\\ \vdots\\ y_m\end{bmatrix}$。

算法输出:修剪后的训练集 S 以及用 SVM 训练 S 得到的分类模型。

算法步骤:

(1)通过公式(1)计算训练集 T 中每个样本点 x_p 与其他样本点间的距离,记为 $z_{pq},p,q=1,2,\cdots,m$,每个点与自身的距离定义为 ∞;

for T 中的每个样本点 $x_p(p=1,2,\cdots,m)$

{

建立核空间距离矩阵 $\boldsymbol{Z}_{m\times m}=(z_{ij}),z_{ij}=\infty,i,j=1,2,\cdots,m$;

for T 中的每个样本点 $x_q(q=1,2,\cdots,m)$

if $q\neq p,z_{pq}=D(x_p,x_q)$;

}

(2)对于训练集 T 中每个样本点 x_p 利用 z_{pq} 寻找其最近邻点并记为 x_{ne};

建立核空间样本点近邻矩阵 $\boldsymbol{NN}_{m\times1}=(nn_i),i=1,2,\cdots,m;t=1;value=z_{p1}$;

for T 中的每个样本点 $x_q(q=1,2,\cdots,m)$

{

if $z_{pq}<value,\{value=z_{pq};t=q;\}$

$nn_i=t$;

}

$x_{ne}=x_{nn_i}$;

(3)判断 x_p 与 x_{ne} 的类别标号是否一致$(p,ne=1,2,\cdots,m)$,如果类别标号一致则不进行任何处理;如果 x_p 与 x_{ne} 的类别标号相异,则同样利用公式(2)分别计算核空间距离 $dis_p=1/k\sum_{x_i\in knn(x_p)}D(x_i,x_p)$,以及核空间距离 $dis_{ne}=1/k\sum_{x_i\in knn(x_{ne})}D(x_{ne},x_i)$;如果 $dis_p>dis_{ne}$,则标记 x_p 为删除,反之则标记 x_{ne} 为删除;

建立类别标记矩阵 $\boldsymbol{L}_{m\times1}=(l_i),l_i=1,i=1,2,\cdots,m$;

for T 中的每个样本点 $x_p(p=1,2,\cdots,m)$

{

if $y_p\neq y_{nn_p},\{dis_p=1/k\sum_{x_i\in knn(x_p)}D(x_i,x_p)$;

$dis_{ne}=1/k\sum_{x_i\in knn(x_{ne})}D(x_{ne},x_i)\}$

if $dis_i>dis_{ne},l_i=-1$;

else,$l_{ne}=-1$;

}

(4)将原始训练集中标记为删除的样本点从样本集 T 中删除,得到修剪后的样本集 TR,即 S;

for T 中的每个样本点 $x_p(p=1,2,\cdots,m)$

{

if $l_p=-1$

删除矩阵 $\boldsymbol{TR}$ 及 $\boldsymbol{L}$ 的第 p 行,新矩阵仍设为 $\boldsymbol{TR}$ 及 $\boldsymbol{L}$;

}

(4)用标准 SVM 训练 S 得到分类模型。

以上算法就是基于 k 个同类核空间近邻点的改进 NN-SVM 分类算法(KCNN-SVM),下一节对其进行相关分析。

2.3 KCNN-SVM 算法性能分析

在 KCNN-SVM 算法中,当样本点 x_i 与 x_{ne} 的类别相异时,需要分别计算 x_i 与 x_{ne} 在核空间的 k 近邻点 KNN(x_i)与 KNN(x_{ne})。相对于 NN-SVM 算法,改进算法 KCNN-SVM 的时间复杂度会增加 O($s\cdot m$)(其中 $s\cdot m$ 为判别 x_i 与 x_{ne} 类别相异的次数,m 为样本点的个数),但是其远小于 NN-SVM 的 O(m^2)。虽然改进算法在训练时间方面略有增加,但他相对于 NN-SVM 算法仍有以下的优点:

(1)通过 KCNN-SVM 算法的步骤 3 计算样本点与其 k 个同类近邻点之间的距离,能够判别该样本点与其最近邻点分别在另一类样本点中的混淆程度。因此,通过比较样本点间的混淆程度,KCNN-SVM 算法可以更准确地删除那些混淆严重的样本点而避免删除那些原本可能对分类有贡献的样本点,进而提高分类精度,后边的实验证明了我们的推断。随着 k 值的增加,KCNN-SVM 可以更准确地进行修剪,然而过大的 k 值并不会更显著地提高分类精度,反而会增加算法在寻找样本点 k 个同类近邻点的计算时间,所以针对不同的训练集应该进行实验选择合适的 k 值,通过实验发现对一般的数据集选取 k=2 或 k=3 可以取得较高的分类精度。

(2)KCNN-SVM 算法中把输入空间的样本点映射到高维特征空间,经过非线性映射能够较好地分辨、提取并放大有用的特征,从而可以更好地在变换后的特征空间中利用高维距离对训练数据进行测量、修剪,从而获得更能准确反映原始数据集几何特征的训练子集。

(3)作为标准 SVM 算法分类之前的修剪算法,KCNN-SVM 算法大量的计算用在了计算高维距离上,但是他为后面的 SVM 分类提供了完全可以利用的信息,如在第二章中介绍的 SMO 算法,他将工作样本集的规模减到了两个样本,满足等式线性约束的存在使得同时至少有两个 Lagrange 乘数发生变化,在每步更新 Lagrange 乘数的时候,需要利用 $K(x,y)=\phi(x)\cdot\phi(y)$ 来计算参数,这样就可以利用公式(2)在 KCNN-SVM 算法中计算距离时获得的数据。所以,KCNN-SVM 算法在计算上也为后续的标准 SVM 分类算法提供了准备。

此外,可以看到当 KCNN-SVM 算法步骤 3 中的 k=0 时,改进算法就退化为原始的 NN-SVM 算法,所以可以将 NN-SVM 算法看成是改进的 KCNN-SVM 算法的一种 k=0 时的特例。

3 实验

3.1 实验环境

在 PC 机(P4 1.5 G,640 M 内存)上,利用文献[4]中的 libsvm 以及本文所编写的修剪程序进行实验,实验中所用到的数据集如下(数据集 1 和数据集 2 可以从网站 http://ida.first.fraunhofer.de/projects/bench 上获得,数据集 3 可以从网站 http://www.csie.ntu.edu.tw 上获得):

数据 1:banana,该数据集的训练集包含 400 个样本点,测试集包括 4 900 个样本点。数据 1 是两类分类问题,每个样本点是一个二维特征向量,训练集均值为 0,标准偏差为 1。

数据 2:ringnorm,该数据集是用于两类分类的样本集,每一类都是取自一个可以产生任意维的多变量正态分布(在实验中分别选用 10 维和 20 维)。类别 1 的均值为 0,协方差为单位元的 4 倍;类别2 的均值为$(\alpha,\alpha,\cdots,\alpha)$,协方差为单位元,其中

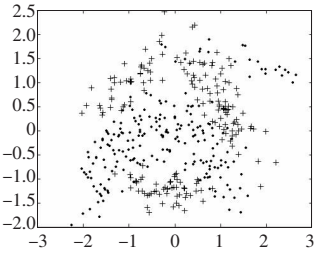

图3 修剪前的数据1

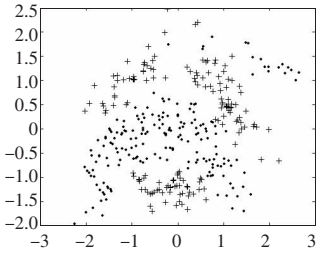

图4 KNN-SVM修剪后的数据1

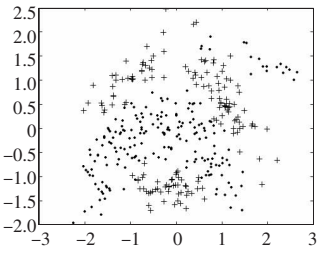

图5 KCNN-SVM修剪后的数据1(k=1)

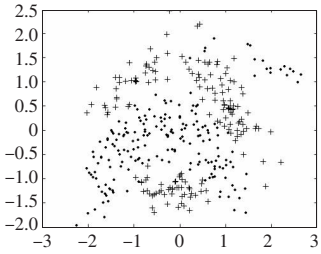

图6 KCNN-SVM修剪后的数据1(k=3)

表1 数据1上的比较

	样本点数	训练时间/ms	分类时间/ms	正确分类数	正确率/(%)
SVM	400	371	891	4 340	88.57
NN-SVM	344	200	591	4 324	88.24
KCNN-SVM(k=1)	353	213	611	4 333	88.43
KCNN-SVM(k=2)	360	220	630	4 349	88.76
KCNN-SVM(k=3)	359	215	620	4 351	88.80

表2 数据2上的比较(10维)

	样本点数	训练时间/ms	分类时间/ms	正确分类数	正确率/(%)
SVM	400	482	1 862	6 388	91.25
NN-SVM	311	280	1 082	6 316	90.23
KCNN-SVM(k=1)	321	300	1 172	6 397	91.39
KCNN-SVM(k=2)	317	296	1 131	6 420	91.71

表3 数据2上的比较(20维)

	样本点数	训练时间/ms	分类时间/ms	正确分类数	正确率/(%)
SVM	400	607	2 483	6 856	97.94
NN-SVM	251	370	1 362	6 672	95.31
KCNN-SVM(k=1)	266	391	1 513	6 832	97.60
KCNN-SVM(k=2)	260	382	1 453	6 860	98.00

表4 数据3上的比较

	样本点数	训练时间/ms	分类时间/ms	正确分类数	正确率/(%)
SVM	3 185	13 797	50 541	24 621	83.81
NN-SVM	2 441	3 315	17 707	24 564	83.62
KCNN-SVM(k=1)	2 640	4 998	23 855	24 587	83.70
KCNN-SVM(k=2)	2 615	4 507	21 393	24 619	83.80
KCNN-SVM(k=3)	2 611	4 457	19 787	24 635	83.86

α=1/$sqrt(d)$(其中 d 是向量所选取的维数)。数据2有400个样本点的训练集,7 000个样本点的测试集。

实验中,采用高斯核(即公式(4))做为距离度量以及SVM训练中的核函数,此时KCNN-SVM中使用的核空间距离公式(2)可以简化为:

$$D(x,y)=\sqrt{K(x,x)+K(y,y)-2K(x,y)}=\sqrt{\exp\{-\gamma\|x-x\|^2\}+\exp\{-\gamma\|y-y\|^2\}-2\exp\{-\gamma\|x-y\|^2\}}=\sqrt{2-2\exp\{-\gamma\|x-y\|^2\}} \quad (6)$$

3.2 实验结果分析

对于数据1(高斯核的参数采取默认参数,c=10),通过图3和图4,比较原始数据1和经过NN-SVM算法修剪后的数据1的分布情况。通过比较图3和图4发现,NN-SVM算法对于数据1的修剪并不完全适合。虽然NN-SVM算法可以删除样本集中大量的混淆样本点,特别是其删除了两类样本点交界处大量重叠混杂的样本点,但是有很多可能对最后构造最优分类超平面有贡献的样本点,即那些可能成为支持向量的样本点都被删除了,这样虽然样本点的数量减少了,但剩下的样本点并不能真实地反映原始训练集的几何分布,因此,在此基础上构造的学习机的分类正确率必然受到影响。

接下来,通过图5至图6比较选取不同 k 值的KCNN-SVM分类算法在数据1上的修剪效果。

通过比较修剪前后训练集样本点的变化情况可以发现,NN-SVM与KCNN-SVM都可以删除掉两类交界区域重叠严重的样本点,但是SVM存在错误删除样本点的情况,而KCNN-SVM删除混淆点更加合理。通过表1比较了三者在数据1上的分类效果。在时间方面,由于修剪数据集减少了样本点的个数,特别是难于分类的混淆严重的样本点被删除,NN-SVM及KCNN-SVM在训练时间上都低于SVM;分类时用更少的SV(Support Vector)构造分类函数,因此分类时间也有所减少。在分类准确率方面,NN-SVM存在误删除样本点的情况,所以其正确率相比SVM有所下降,而KCNN-SVM算法随着 k 的增大,通过样本点 k 个同类近邻比较他们在另一类中的混淆程度,能够更准确地删除混淆点,分类正确率比SVM更高。在数据1中 k=3时,KCNN-SVM算法取得了最好的分类效果。

对于数据2(高斯核的参数采取默认参数,c=10),由于其本身是随机排列的,向量的各分量之间是独立的,可以选取不同的维数(10,20)进行实验,实验结果如表2和表3。

对于数据2不同维数的测试得到了与数据1类似的结果。KCNN-SVM与原始的NN-SVM与SVM相比有着更高的分类精度;在训练以及分类时间上都优于SVM。数据2上改进算法 k=2取得了较好的分类效果。数据1和数据2属于低维的训练集,为了对更高维的数据集进行测试,对数据3进行实验(高斯核的参数采取默认参数,c=100),分类结果如表4。

通过表4发现改进算法的分类精度在高维数据集3中并没有像在数据集1和数据集2中一样得到明显的提高,只是在 k=3时略高于标准SVM的分类精度,具体原因是对于数据集3使用欧式距离测量高维空间向量距离时会产生误差,由此可能造成训练集中样本点的错误删除。尽管如此,KCNN-SVM算法在分类精度上还是优于NN-SVM算法;此外,对于样本点较多的高维数据集3,KCNN-SVM算法在训练以及分类时间上都远优于标准的SVM算法。

4 结论

针对现有知识向量机分类算法存在的不足,提出一种用于

(下转182页)