文章编号: 1001-0920 (2013) 07-1073-05

一种 k-NN 分类器 k 值自动选取方法

杜 磊, 杜 星, 宋擒豹

(西安交通大学 电子与信息工程学院，西安 710049)

摘 要: k-NN 分类算法已广泛应用于文本挖掘和模式识别等领域, 其近邻数 k 直接影响着分类精度, k 值过小时 k-NN 会受到噪声的影响, k 值过大时同样会降低分类精度, 为此提出一种快速选取 k 值的方法. 首先给出 k 值的候选集, 然后在候选集上快速地选取 k 值. 在 100 个公开数据集上的实验结果表明, 所提出的算法能够选取一个有效的近邻数 k, 是一种效果好、有潜力的方法.

关键词: 分类; k-NN 算法; 近邻数; 近邻数选取

中图分类号: TP274 **文献标志码:** A

An automatic selection method of k in k-NN classifier

DU Lei, DU Xing, SONG Qin-bao

(School of Electronics and Information Engineering，Xi'an Jiaotong University，Xi'an 710049，China．Correspondent: DU Lei，E-mail：dulei.323@stu.xjtu.edu.cn)

Abstract: The k-NN classification algorithm has been broadly applied to text mining, pattern recognition and so on. Its nearest neighbor k has directly effect on its classification accuracy. If the value of k is too small, k-NN is sensitive to noise. On the other hand, if k is too large, its accuracy is also low. On this account, a fast k value selection method is proposed. Firstly, a candidate set of k is calculated. Then an appropriate k is found quickly in this candidate set. Experiment results on 100 publicly available data sets show that, the proposed method can find an effective nearest neighbor successfully, which is a method of good effect and potential.

Key words: classification; k-NN; nearest neighbor; nearest neighbor selection

0 引 言

k-NN 分类算法[1]是机器学习领域的一个经典算法, 该算法简单、易于实现, 从而引起了人们的广泛关注[2]. k-NN 是一种基于实例的学习法, 它根据被预测对象与已知对象之间的相似度进行分类, 并不会显式地构建模型. 当给定一个未知实例时, k-NN 根据与该实例最近的 k 个实例进行分类, 因此 k 值的优劣将直接决定着 k-NN 分类的效果. 显然, k-NN 算法的关键在于选择一个合适的 k 值[2-3], 也就是近邻数. 如果 k 值取得过小, 则算法易受噪声的影响, 使分类结果不稳定; k 值取得过大, 则近邻集中含有太多其他类别的实例, 导致分类错误. 同时, 过大的 k 值也增加了算法的时间开销. 理论上可以采用穷举法, 选择所有可能的 k 值, 然后选择最好的 k 值作为近邻数. 但在实际中, 穷举法带来的时间开销通常是难以接受的.

目前, 已经有许多文献研究了如何选择 k 值. 文献 [4] 指出, 为了确保分类正确, 近邻集中至少有 $n+1$ 个实例来自正确的类别, 因此建议 $k=2n+1$. 文献 [5] 提出根据数据分布使用不同的 k 值, 显然这种动态求 k 值的方法使算法的时间开销增大. 文献 [6] 首先从训练集中选择一个子集, 然后在该子集上应用 k-NN, 这相当于缩减了训练集, 因此没有充分利用训练集的信息. 文献 [7] 先在数据集上进行聚类, 进而对不同的簇使用不同的 k 值, 但需要提前指定 k 值. 文献 [8] 指出了最优 k 值的取值范围为 $1\sim\sqrt{N}$, 但未给出进一步确定 k 值的方法.

综上, 本文提出一种新的选取 k 值的方法, 主要包含以下两部分: 1) 针对穷举法 k 值搜索空间过大的问题, 提出了确定近邻数 k 的候选集的方法; 2) 如果候选集仍然过大, 则对其进行采样进一步压缩候选集, 然后据此快速确定 k 值.

收稿日期: 2012-02-28; **修回日期:** 2012-08-28.

基金项目: 国家自然科学基金项目(61070006).

作者简介: 杜磊(1985−), 男, 博士生, 从事数据挖掘的研究; 宋擒豹(1966−), 男, 教授, 博士生导师, 从事数据挖掘、软件缺陷预测等研究.

1 相关背景

1.1 k-NN 算法

k-NN 算法已广泛应用于文本挖掘、模式识别等领域, 并不预先显式地建立分类模型, 只有分类任务被触发时才会构建模型. k-NN 是基于实例的学习, 它依赖待分类实例与训练集中最近的 k 个邻居. 下面给出 k-NN 的分类过程.

给定训练集 $D=\{(x_i,y_i)|i=1\sim n\}$ 和待分类实例 $z=(x',y')$, 其中 y_i 和 y' 分别为训练和待分类实例的类标签, y'未知. 分类时, k-NN 首先计算 z 和 D 中所有实例的距离, 然后根据 k 值确定出 k 近邻集 D_z, 最后按下面的规则进行分类:

$$\text{Majority Voting: } y'=\arg\max_{v}\sum_{(x_i,y_i)\in D_z} I(v=y_i).$$

其中: v 为类标签; y_i 为第 i 个近邻的类标签; $I(\cdot)$ 为一个指示函数, 当条件为真时返回 1, 当条件为假时返回 0.

1.2 分类精度与 k 值的关系

在实际中, 数据集通常包含噪声, 而且不同类别所占的比例各不相同, 同时数据集类别之间也没有清晰的界限. 因此, 当 k 较小时, k-NN 对噪声过于敏感; 相反, 当 k 取较大值时, 类别中占多数的那一类将主导 k-NN 的结果, 同样使得算法的性能低下. 那么, 在 k 值由小增大的过程中, k-NN 的精度是怎样变化的呢? 作者任意选取 5 个数据集 (sick、labor、pima、iono 和 vechicle)[9], 对它们应用不同的 k 值得到了对应的精度 (k-NN 算法的 k 值值域为 $[1,n-1]$, 因而分别取遍各个数据集的 k 值得到其对应的精度). 为了说明分类精度与 k 值的关系, 分别绘出它们的 k-精度曲线如图 1 所示.

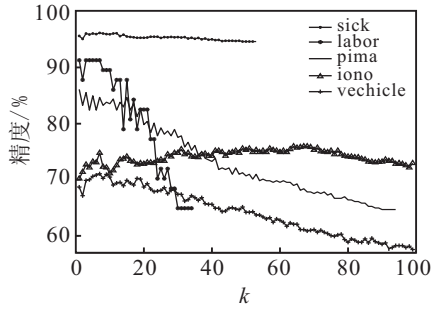

图 1 k-精度曲线

从图 1 中可以看到, 对于不同的数据集, k-精度曲线也不相同, 但它们都有一个共同的现象: k 值较小时分类精度忽高忽低, 极不稳定; 随着 k 值的增加, 分类精度逐渐稳定; 当 k 增大到一定程度时, 精度开始逐步下降并趋于平稳. 就整个变化趋势而言, 精度随着 k 值的变化是先增加, 后稳步下降并趋于平稳. 通过分析可知: 这种现象并不是偶然的, 正如本节开始所述, 当 k 较小时, k-NN 对噪声比较敏感, 算法必然不稳定; 随着 k 值增大, 噪声的影响变小, 此时分类精度开始上升; 而后, 当 k 大于最优 k 时, 训练集中实例最多的类别将在近邻集中占绝对优势, 此时任何未知实例都会被划分为该类别, 从而精度下降.

根据以上分析, 本文提出求近邻数 k 的方法, 称为 find$K$ 算法.

2 k-NN 近邻数选取算法

2.1 算法简介

对于一个给定的数据集, 总有一个或多个 k 值对应着 k-NN 在其上的最高精度, 这里称其为最优精度, 该最优精度对应的 k 值称为最优 k (最优 k 可能不止一个, 因为可能有不同的 k 值同时对应着最优精度).

根据第 1.2 节的分析, 分类精度随着 k 值的增加先增加、后稳步减小, 从而 k-精度曲线的最高点分别对应着最优精度和最优 k. 如果能找到 k-精度 曲线的最高点或者其附近点对应的 k 值, 则推荐的 k 值是非常有效的, 因此可将确定 k 值的问题转化为求 k-精度曲线上的最大值问题.

本文提出的 k 值选取方法包含以下两部分:

1) 确定 k 值的候选集. 压缩 k 值的搜素空间, 得到 k 值的候选集.

2) 选择 k 值. 如果候选集仍然过大, 则先对候选集进行采样, 然后在候选集中查找合适的 k 值.

2.2 确定 k 的候选集

本文的目的是为数据集找到最优 k, 并依此进行分类. 因为没有先验知识, 直接指定 k 值具有很大的盲目性, 同时, 穷举法又由于时间开销过大而不可取.

为了解决以上两个问题, 设想如果能找到一个全体 k 值的子集作为 k 的候选集, 并尽可能使该候选集包含最优 k, 再在该候选集中查找合适的 k值, 则能在保证 k-NN 精度的前提下大大提高其时间效率. 根据该思想, 本文首先提出下面的方法来确定 k 的候选集.

对于一个数据集, 假设不含有噪声, 而且不同类别之间的距离都远远大于类别内实例间的距离, 则此时 k 取 1 或者很小的值便可得到好的效果. 在这种情况下, k 值的候选范围可以取为 $1\sim M$(M 为很小的常数). 然而, 实际中的数据集大都不会这么理想. 不管是噪声、还是类别之间的界线不明确 (类别之间有重叠)时, 增大 k 都能提高分类精度, 所以此时 k 值的候选集应适当扩大, 以确保该候选集包含较好的 k.

本文所给出的 k 的候选集为 $1\sim M$, 其中 M 的计算方法如下.

设数据集中最少类别包含的实例数目为 Count_1, 次少类别实例数目为 Count_2, 根据下面两种情况计

算 M 值: 1) 如果 Count_1 足够大, 则 $M=1\sim 2\,\mathrm{Count}_1-1$, 从而使属于最小类别的实例有足够多的近邻来保证其被正确分类; 2) 如果 Count_1 太小, 则此时属于最小类别的实例有被误分的风险, 因为近邻数太少. 因此

$$M=1\sim\max(2\,\mathrm{Count}_1-1,(\mathrm{Count}_1+\mathrm{Count}_2)/2).$$

2.3 采　样

由第2.2节的结论可以看到, 当 M 很大时, k 的候选集依然很大, 所以要找到合适的 k 值仍然需要较大的时间消耗. 此外, 曲线的局部震荡对求 k 值也非常不利. 为此, 本文对 k-精度曲线进行采样, 以进一步压缩 k 值候选集的大小, 同时, 采样也能够削弱 k-精度曲线的局部震荡.

为了使采样后的候选集包含较好的 k 值, 采样间隔不宜过大, 这是因为大的采样间隔很可能会遗漏最优 k, 而太小的采样间隔则达不到采样的目的. 通过实验可以发现, 采样间隔为 3 时便可以得到较好的效果, 因此本文的采样间隔取为 3, 即 $k=1,4,\cdots,3\times i+1$. 如果 k 的候选集本身较小, 则无需对其进行采样.

在应用中, 采样策略及采样间隔可以根据实际需要灵活选择.

2.4 find K 算法

下面给出 find K 算法的详细描述, 主要包括以下步骤.

Step 1: 对数据集按类别所含实例进行升序排列, 得到候选集的上界 M; 若 M 较大, 则采样; 初始化 k 为 $[1,M]$ 上的一个随机值, 初始化搜索步长为 step.

Step 2: 计算当前 k 值对应的斜率 (斜率的计算如下: 设 k_1 和 $k_2(k_1<k_2)$ 为 2 个 k 值, 如果 k_1 对应的精度小于 k_2 对应的精度, 则 k-精度曲线上升, 此时斜率为正; 如果 k_1 和 k_2 对应的精度相等, 则斜率为 0, 否则斜率为负). 如果斜率大于 0, 则精度随着 k 的增大而增大, 说明当前 k 值小于最优 k, 因此增大 $k(k=k+\mathrm{step})$; 如果当前 k 值对应的斜率小于 0, 则精度随着 k 的增大而减小, 说明当前 k 值大于最优 k, 因此减小 $k(k=k-\mathrm{step})$; 如果当前斜率等于 0, 则收缩候选集, 为 k 重新选取初值; 如果当前 k 值对应的斜率符号与前一次不一致, 且 $\mathrm{step}>1$, 则搜索步长减半 ($\mathrm{step}=\mathrm{step}/2$).

Step 3: 如果 k 值对应的斜率符号开始正负交替且 $\mathrm{step}=1$, 或者 k 到了候选集的边界, 则算法以当前 k 值作为推荐的近邻数, 否则重复以上步骤.

算法实现的伪代码如图 2 所示. 在图 2 中, 第 1 行和第 2 行对应于 Step 1, find K 算法的 Step 1 消耗的时间为常数; 整个循环 (第 3~15 行) 用来查找合适的 k 值, 对应于 Step 2 和 Step 3. 循环过程不断地选择不同的 k 值, 因此 find K 对待分类实例与训练集之间的距离进行了排序, 从而时间复杂度依赖于排序算法的时间复杂度, 为 $O(N\log N)$ (本文使用快速排序). 在最好情况下, find K 需要 3 次迭代便可找到 k 值, 这时算法时间复杂度仅依赖于排序算法; 在最坏的情况下, find K 迭代 M 次才能给出 k 值, 这时时间复杂度为 $O(MN\log N)$. 因为 M 本身远小于 N, 且算法在 M 较大时对候选集进行采样, 即 $M\ll N$, 所以该步骤的时间复杂度为 $O(N\log N)$.

综上, find K 算法的时间复杂度为 $O(N\log N)$.

```
输入：T—训练集
      step—初始步长
      recurMax—最大迭代次数
      flag—k摆动标志,初始为false
输出：近邻数 k
1  M=2*Count1−1; //Count1是数据集T的最小分类实例个数
2  k=random(M);  /random为随机数函数
3  while(!flag && recurNum<recurMax)
4     p=kNN(k,T),p1=kNN(k+1,T);
5     h=(p1−p)/1;
6     if h*h1<0
7        if step!=1
8           step=step/2;
9        else
10          flag=true
11       end if
12    end if
13    if h>0
14       k=k+step;
15    else if h<0
16       k=k−step;
17    else
18       k=Random(M/2),M=M/2;
19    end if
20 end while
```

图 2　find K 算法

3 实　验

3.1 数 据 集

实验采用的数据集来自加州大学欧文分校公开的机器学习数据库[9]. 其中的数据集常被学者和研究人员广泛使用. 本文选取其中 100 个分类型数据集作为本实验的数据来源, 分别涉及计算机工程、生命科学、物理科学、社会生活、电子商务以及游戏等各个领域. 表 1 给出了 100 个数据集的基本情况, 这些数据集中实例最少的数据集有 10 个实例, 实例最多的有 28 056 个; 属性最少的数据集有 4 个属性, 最多的有 70 个; 类别个数最少的数据集含有 2 个类标签, 最多的含有 28 个. 数据集的特征足够多样, 使得在此基础上得到的 k 值选取方法具有足够的适用性和普遍性.

3.2 实验设置

3.2.1 实验方法

实验采用十折交叉验证 (10-fold cross-validation) 的方法, 该方法可以使得到的结果更加可靠稳定. 为了证明 find K 算法的性能, 本文采用穷举法求出每个

表 1 数据集

id	实例	属性	class	id	实例	属性	class	id	实例	属性	class	id	实例	属性	class
1	898	39	6	26	368	23	2	51	958	10	2	76	345	7	2
2	226	70	24	27	3 163	26	2	52	846	19	4	77	32	57	2
3	690	15	2	28	3 772	30	4	53	435	17	2	78	106	59	4
4	205	26	7	29	351	35	2	54	5 000	41	3	79	432	7	2
5	625	5	3	30	150	5	3	55	5 000	22	3	80	124	7	2
6	16	5	2	31	3 196	37	2	56	178	14	3	81	432	7	2
7	286	10	2	32	28 056	7	18	57	1 484	9	10	82	169	7	2
8	699	11	2	33	57	17	2	58	101	17	17	83	432	7	2
9	699	10	2	34	3 200	8	10	59	898	39	6	84	122	7	2
10	1 728	7	4	35	24	5	3	60	205	26	7	85	5 473	11	5
11	303	14	2	36	20 000	17	26	61	699	10	2	86	36	23	3
12	1 473	10	3	37	345	7	2	62	107	13	6	87	90	9	3
13	368	28	2	38	148	19	4	63	107	13	6	88	15	7	2
14	690	16	2	39	8 124	23	2	64	368	23	2	89	323	13	2
15	1 000	21	2	40	12 960	9	5	65	540	40	2	90	1 066	13	3
16	690	16	2	41	768	9	2	66	366	35	6	91	269	45	2
17	768	9	2	42	90	9	3	67	736	20	5	92	80	45	2
18	132	11	2	43	339	18	22	68	194	30	8	93	187	23	2
19	336	8	8	44	2 310	20	7	69	155	9	4	94	80	23	2
20	1 000	21	2	45	15	7	2	70	132	5	4	95	76	46	3
21	214	10	7	46	2 800	30	2	71	303	14	5	96	52	25	3
22	306	4	2	47	208	61	2	72	270	14	2	97	52	24	3
23	294	14	5	48	683	36	19	73	155	20	2	98	10	33	2
24	270	14	2	49	3 190	62	3	74	351	35	2	99	435	17	2
25	155	20	2	50	151	6	3	75	600	62	6	100	63	32	4

数据集上的最优 k, 并用该最优 k 对应的分类精度作为参照, 这样可以清晰地展现出 find K 的效果. 为保证公平, 两种方法所采用的数据和实验方法完全相同.

3.2.2 评价标准

因为所给出的 find K 算法推荐的 k 值无法直观地体现算法的性能, 同时, k-NN 算法的最终目的是进行分类, 故本文通过 k 值对应的精度来评价 find$K$ 的性能.

本文采用 3 种指标来评估算法的性能, 即 MAE、MRE 和 PRED$(\cdot)$. 通常选择其中一种或几种, 本文为了充分评价 find K 算法, 在实验中采用全部 3 种指标进行评价.

MAE (magnitude of absolute error) 表示推荐的 k 值对应的分类精度 (预测精度) 与最优 k 对应的分类精度 (最优精度) 的绝对误差, 即

$$\mathrm{MAE} = (\text{最优精度} - \text{预测精度}) \times 100\%.$$

MRE (magnitude of relative error) 表示预测精度与最优精度之间的相对误差, 即

$$\mathrm{MRE} = \frac{\text{最优精度} - \text{预测精度}}{\text{最优精度}} \times 100\%.$$

对于 MAE 和 MRE, 它们的值越小表示预测精度越接近最优精度, 意味着算法效果越好.

PRED $(\cdot)$ 是另一个常用的评价指标. PRED (m) 表示 MAE (MRE) 小于 $m\%$ 的预测个数占总预测个数的百分比, 即

$$\mathrm{PRED}(m) = \frac{n_0}{n} \times 100\%.$$

其中: n_0 表示 MAE $\leqslant m\%$ (或者 MRE $\leqslant m\%$) 的数据集的个数; n 表示全部数据集的个数, 在本文中为 $n = 100$. PRED$(\cdot)$ 的值越大表明预测精度高的数据集所占的比例越大, 算法效果越好. 本文中取 $m = 5$, 则 PRED(5) 能够筛选出精度误差在 5% 以内的数据集在全体数据集上所占的百分比.

3.3 实验结果与分析

根据第 2 节所述, 本文使用 find K 算法在 100 个数据集上用 10 折交叉验证方法进行实验, 得到了推荐的 k 值. 以此 k 值作为 k-NN 算法的近邻数进行分类, 从而得到了各个数据集上的预测精度. 本文分别计算了 find K 算法的精度在各个数据集上的 MAE 和 MRE, 如表 2 所示.

从表 2 可以看出, 整体上 find K 算法推荐的 k 值取得了较好的精度, 除 id 为 43、68、78、86、96 和 100 的 6 个数据集, find K 算法得到的 MAE 均小于 5%. 这意味着占全部 94% 的数据集都得到了较高的精度. 然而, 从表 2 中无法清晰直观地看出 find K 的效果. 为此, 本文分别计算了 100 个数据集上 MAE 和 MRE 的最小值 (min)、最大值 (max)、中位数 (median)、众数 (mode)、平均值 (mean) 和标准差 (std.), 同时还给出了各自的 PRED(5), 如表 3 所示.

表 2　find K 在全体数据集上推荐的 k 值、MAE (%) 和 MRE(%)

id	k 值	MAE	MRE	id	k 值	MAE	MRE	id	k 值	MAE	MRE	id	k 值	MAE	MRE
1	1	0.00	0.00	26	71	0.28	0.33	51	6	0.00	0.00	76	66	0.29	0.44
2	1	0.00	0.00	27	1	0.32	0.33	52	1	2.36	3.32	77	3	0.00	0.00
3	6	0.44	0.51	28	46	0.00	0.00	53	1	0.46	0.50	78	7	8.49	18.75
4	1	0.00	0.00	29	1	0.00	0.00	54	366	4.04	4.99	79	91	3.47	6.97
5	46	0.00	0.00	30	36	1.33	1.37	55	121	0.42	0.49	80	6	2.42	4.41
6	3	0.00	0.00	31	1	0.22	0.23	56	21	0.00	0.00	81	86	0.00	0.00
7	26	2.10	2.82	32	11	0.00	0.00	57	7	1.62	2.71	82	11	1.19	2.05
8	26	0.86	0.89	33	7	0.00	0.00	58	1	0.00	0.00	83	86	0.93	1.98
9	1	0.28	0.29	34	151	0.18	0.24	59	1	0.00	0.00	84	16	1.64	2.78
10	6	0.00	0.00	35	3	0.00	0.00	60	1	0.00	0.00	85	1	0.00	0.00
11	36	0.33	0.39	36	1	0.00	0.00	61	16	0.28	0.29	86	4	8.34	10.35
12	196	0.54	1.06	37	21	1.45	2.19	62	1	0.00	0.00	87	5	0.00	0.00
13	76	0.00	0.00	38	4	0.68	0.83	63	13	3.81	5.88	88	1	0.00	0.00
14	21	0.44	0.51	39	1	0.00	0.00	64	91	0.27	0.32	89	9	0.00	0.00
15	11	0.60	0.80	40	6	0.00	0.00	65	1	0.00	0.00	90	2	0.00	0.00
16	11	1.45	1.66	41	66	0.13	0.17	66	16	0.00	0.00	91	16	1.48	1.81
17	61	0.00	0.00	42	7	2.22	3.07	67	31	2.44	4.37	92	1	5.00	6.90
18	31	0.00	0.00	43	2	7.37	15.14	68	1	5.67	9.09	93	26	1.07	1.49
19	3	1.78	2.04	44	1	0.00	0.00	69	16	0.64	1.34	94	6	1.25	1.75
20	11	0.00	0.00	45	9	0.00	0.00	70	1	0.00	0.00	95	5	0.00	0.00
21	1	0.00	0.00	46	11	0.11	0.11	71	16	0.99	1.17	96	15	15.39	21.06
22	26	1.31	1.74	47	1	0.00	0.00	72	46	0.37	0.44	97	3	3.85	5.27
23	46	0.00	0.00	48	1	0.43	0.47	73	36	4.52	5.27	98	5	0.00	0.00
24	51	0.74	0.88	49	521	0.03	0.03	74	1	2.28	2.57	99	11	0.46	0.50
25	6	0.64	0.75	50	1	0.00	0	75	11	1.50	1.53	100	2	6.35	8.89

从表 3 可以看到, MAE 和 MRE 的最小值都为 0, 说明 find K 算法能够成功地找到最优 k. 众数为 0, 则说明推荐到最优 k 的次数在全体数据集上出现的次数是最多的. 同时还可以看到, MAE 和 MRE 的中位数也是非常小的 (0.28% 和 0.32%), 这说明 find$K$ 推荐的 k 值得到的精度在一半数据集上都非常接近最优精度, 而均值和方差都较小更证明了 find K 在多数数据集上的精度都非常有效. 因此, 即使 MAE 和 MRE 的最大值较大 (15.38% 和 21.05%), 但这仅是在非常少的数据集上的结果, 其中 MAE 和 MRE 的 PRED(5) 分别为 94% 和 89% 也验证了这一点. 综上, find K 算法在整体上取得了非常好的效果.

表 3　find K 在所有数据集上 MAE 和 MRE 的统计结果　%

	min	max	median	mode	mean	std.	PRED(5)
MAE	0.00	15.38	0.28	0.00	1.20	2.31	94
MRE	0.00	21.05	0.32	0.00	1.79	3.65	89

find K 算法并不能每次都推荐出最优的近邻数, 但通常是非常接近最优 k 的. 因为最终推荐的 k 值与迭代的初值有关, 所以可能会使算法陷入局部最优. 因此, 为了尽可能地避免陷入局部最优, 可以采用多次随机初值的策略. 在采用多次随机初值方法时, 每次可以使用上一次保存的结果, 这样预测所需的时间并不会线性增长.

本文分别进行了一次初值、二次初值和三次初值实验, 并将它们的结果列在表 4 中, 其中平均迭代次数表示在求解 k 值的过程中算法尝试的次数. 从表 4 可以看出, 平均迭代次数随着初值次数的增加而增加, 这是显然的. 同时可以看到, 平均 MAE 随着初值次数的增加明显减少, 其中三次初值的平均 MAE 比一次初值的平均 MAE 减少了 40%. 因此, 在实际应用中, 为了增加 find K 算法的健壮性, 消除初值带来的影响, 避免陷入局部最优, 可以使用多次初值的方法. 实验表明, 三次初值往往能够达到较高的精度.

表 4　一次初值、二次初值、三次初值实验结果　%

	一次初值	二次初值	三次初值
平均迭代次数	7.74	11.68	15.25
平均 MAE	1.20	1.12	0.72

4　结　论

本文针对 k-NN 算法中近邻数 k 的确定问题进行了研究. 不同于已有的研究成果[8], 本文提出了一种快速选取 k 值的方法 find K. 它根据 k 值与精度的关系, 首先大大地压缩了 k 值的搜索范围, 然后在 k 值的候选集中进行快速查找, 迅速地定位到 k 值. 在 100 个公开数据集上的实验结果表明了 find K 算法的优异性能, 表明了能推荐出较好的 k 值, 从而得到较高的精度. 在实际应用中, 为了消除初值带来的影响, 可以采用多次初值的方法. 实验表明, 三次初值往往可以取得较高的精度. 本文所提出的算法较好地解决了 k-NN 算法中参数 k 的取值问题.

参考文献(References)

[1] Cover T, Hart P. Nearest neighbor pattern classification[J]. IEEE Trans on Information Theory, 1967, 13(1): 21-27.

(下转第1082页)