基于自然邻居和最小生成树的原型选择算法

朱庆生 段浪军 杨力军
(重庆大学计算机学院 重庆 400044)

摘 要 K最近邻居是最流行的有监督分类算法之一。然而,传统的K最近邻居有两个主要的问题:参数K的选择以及在大规模数据集下过高的时间和空间复杂度需求。为了解决这些问题,提出了一种新的原型选择算法,它保留了一些对分类贡献很大的关键原型点,同时移除噪声点和大多数对分类贡献较小的点。不同于其他原型选择算法,该算法使用了自然邻居这个新的邻居概念来做数据预处理,然后基于设定的终止条件构建若干个最小生成树。基于最小生成树,保留边界原型,同时生成一些具有代表性的内部原型。基于UCI基准数据集进行实验,结果表明提出的算法有效地约简了原型的数量,同时保持了与传统KNN相同水平的分类准确率;而且,该算法在分类准确率和原型保留率上优于其他原型选择算法。

关键词 K最近邻居,原型选择,自然邻居,最小生成树,分类
中图法分类号 TP391 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.04.051

Prototype Selection Algorithm Based on Natural Neighbor and MST

ZHU Qing-sheng DUAN Lang-jun YANG Li-jun
(College of Computer Science, Chongqing University, Chongqing 400044, China)

Abstract K-nearest neighbor (KNN) is one of the most popular algorithms for supervised classification. However, the traditional KNN classification has two limitations that the option of parameter K and prohibitive computational and storage demand for large datasets. To overcome these limitations, a new prototype selection algorithm was proposed, which retains some key prototypes that make a large contribution to classification and removes the most of other prototypes with little contribution for classification. Differing from other prototype selection algorithms, the proposal uses a novel neighbor concept natural neighbor to preprocess the dataset and builds minimum spanning tree based on the specific terminal conditions. According to the MST, the prototypes close to boundaries and some internal prototypes are preserved. Experimental results show that the proposed algorithm effectively reduces the number of prototypes while maintaining the same level of classification accuracy as the traditional KNN classification algorithm. Moreover, it is a little bit superior to other prototype selection algorithms in classification accuracy and retention ratio.

Keywords K-nearest neighbor, Prototype selection, Natural neighbor, Minimum spanning tree, Classification

1 引言

目前,最近邻居概念及其相关的算法已经在模式识别、图像处理、数据挖掘、智能推荐等领域得到了广泛的应用。K最近邻居分类是最著名同时被广泛应用的无参数模式分类方法之一[1]。由于简单、高效的优点,KNN被广泛应用于模式分类领域,包括欺诈检测、文本分类、信息检索等[2]。然而,传统的KNN算法存在两个主要的问题:1)参数K(邻居数量)的选择,因为不知道在应用KNN时应该选择什么参数才能得到最好的结果;2)计算的时间复杂度和空间复杂度高,特别是在数据集的规模很大、维度很高时,对于一个未知标签的对象,KNN的使用可能会产生很长的运行时间和很重的存储空间负担。通常解决第一个问题的方法是经验性地选择或者研究人员通过大量实验验证来选择一个效果最好的合适的K值。对于第二个问题,最主要的有效解决方案是原型选择,这种类型的算法主要是通过移除初始训练集中的冗余数量以及噪声数据,在保证不降低甚至提高分类精度的基础上,约简原始训练集的大小和降低分类时计算的时间复杂度以及空间复杂度。这两个问题虽然已经被研究多年,但是在实际情况中依然没有得到很好的解决,参数K的选择依然是困难的,许多原型选择算法总是获得较低的分类准确率和较高的原型保留率。

为了解决前面提到的问题,提出一个新的无参数选择的邻居概念,并用它来做原型选择的预处理操作。在一个训练集中,作为类的边沿区域,类的边界原型为分类提供了有用的信息,而类的内部原型对于分类器则相对不那么有用[3]。所

到稿日期:2016-03-10 返修日期:2016-05-08 本文受国家自然科学基金(61272194)资助。
朱庆生(1956-),男,博士,教授,博士生导师,主要研究方向为数据挖掘、服务计算、图像处理,E-mail:qszhu@cqu.edu.cn;段浪军(1989-),男,硕士,主要研究方向为数据挖掘;杨力军(1984-),男,博士,主要研究方向为数据挖掘。

提算法旨在使用最小生成树来寻找靠近边界的关键原型，同时生成一些内部原型。

本文第 2 节介绍一些原型选择的相关工作；第 3 节介绍提出的新的邻居概念；第 4 节介绍提出的原型选择算法；第 5 节介绍实验及结果；最后进行总结并讨论未来的研究内容。

2 相关工作

为了解决大规模数据集下 KNN 分类算法的时间复杂度和空间复杂度过高的问题，原型约简技术被广泛应用在 KNN 分类中。通常的约简技术主要有两种：原型选择和原型生成。目前已经有许多原型选择算法被提出，压缩和剪辑策略是其中两个最著名的方法。Hart[4] 于 1968 年提出了压缩最近邻居算法(Condensing Nearest Neighbor，CNN)，该算法的主要思想是通过只保留靠近类边界的点来尽可能多地约简数据集的大小。虽然该算法通常能使训练集的约简率较高，但是分类精度却有所降低；同时，CNN 算法对数据集的扫描顺序和噪声点很敏感。之后，一些基于 CNN 的改进算法被提了出来，例如：快速压缩最近邻居(Fast CNN，FCNN)[5] 和广义压缩最近邻(Generalized CNN，GCNN)[6]。Wilson 于 1972 年提出了剪辑最近邻居(Edited Nearest Neighbor，ENN)[7]，该方法排除了噪音点和一些导致类重叠的原型点，如非常靠近决策边界的原型点，从而达到原型选择的目的。同时，基于 ENN 的一系列的改进算法被提出，例如：RENN(Repeated ENN)，DROP1，…，DROP5(Decremental Reduction Optimization Procedure)[8] 等。然而，这些基于剪辑策略的算法并没有移除那些对分类决策没有重要贡献的内部原型点，因此其数据约简率比较低，但整体的分类准确率比较高。

近年来，一些原型选择的新技术被提出。Fayed 和 Atiya 于 2009 年提出了一种新的压缩算法——KNN 模板约简算法(the Template Reduction for KNN，TRKNN)[1]。TRKNN 的基本思想是定义一个最近邻居“链”，对于链上的每一段距离，设置一个阈值来将其划分成保留的压缩原型集和要移除的原型集。Arturo 等于 2010 年提出了一种基于聚类的原型选择算法 PSC (Prototype Selection by Clustering)[3]，PSC 首先使用 K 均值聚类算法将数据集划分成不同的簇(即使该区域中包含不同类标签的原型)，然后在每一个簇中选择原型。如果一个簇中的所有原型点都是属于同一类别，那么最靠近簇中心的原型被保留下来；如果一个簇中存在不同类别的原型，则将不同类边界的原型保留下来。Li J 和 Wang Y P 于 2014 年提出了一种基于二叉最近邻树的原型选择算法 BNNT(the Binary Nearest Neighbor Tree)[2]，该算法保留了原数据集中的一些边界原型，同时生成了一些重要的内部原型点。BNNT 算法首先会建立一棵二叉最近邻树，当该二叉树位于一个类的内部时，树的所有原型节点都会被一个生成的中心原型替代；当二叉树位于多个类的边界时，那些拥有不同类标签同时在树中直接相连的原型被保留下来。

大多数已有的原型选择算法都具有局限性，其中最主要的两个问题是原型选择的约简率不够高以及用于分类的准确率相对变低。因此，本文提出了一种新的基于自然邻居和最小生成树的原型选择算法(Prototype Selection Algorithm Based on Natural Neighbor and MST，2NMST)。该算法基于自然邻居(Natural Neighbor，2N)[9-11] 这个新的邻居概念进行预处理，然后构建最小生成树来进行原型选择。通过实验表明，相比前面提到的算法，2NMST 算法获得了更高的准确率和约简率。所提算法与 BNNT 算法有相似的处理策略和时间复杂度。为了展示本文算法的性能，基于 UCI 数据集将 2NMST 与其他算法做了比较，重点比较了算法的约简率和分类准确率，同时也分析了算法的时间复杂度。

3 自然邻居

不同于传统的 K 最近邻居，自然邻居是一个新的邻居概念，是一种无尺度的最近邻居，其邻居搜索过程没有任何参数。如果 KNN 被看作是一个主动的邻居搜索过程，自然邻居则是完全被动的邻居查找过程。自然邻居的主要思想是每个点都可能有不同数量的自然邻居，越是密集的点拥有越多的自然邻居，相反，越稀疏的点则包含越少的自然邻居。

考虑到实际情况中噪声数据点的存在会对自然邻居的搜索过程产生干扰，针对邻居搜索的终止条件进行改进，提出一种新的自然邻居搜索算法。算法 1 描述了详细的搜索过程，此过程有效排除了离群点的影响，以便利用自然邻居对数据集做预处理。

算法 1 自然邻居搜索算法(2N_Search)

输入：训练集 X

输出：每个原型的自然邻域 NN；

每个原型的自然邻居数量 nb；

自然邻居的平均数量 sup_k

步骤：

```
1. r=1,flag=0,∀i∈X,nb(i)=0,NN(i)=∅
2. While flag==0
3.     For ∀i∈X
4.         k=getNN_r(i)
5.         nb(k)=nb(k)+1
6.     NN(k)=NN(k)∪{i}
7.     End For
8.     r=r+1
9.     cnt=count(nb(i)==0)
10.    If all(nb(i))≠0 ‖ isSametoLast(cnt)
11.    Then falg=1
12. End While
13. sup_k=r-1
```

在算法 1 中，函数 $getNN_r(i)$ 取得数据点 i 的第 r 个最近邻居，函数 $count(nb(i)==0)$ 统计了 $nb(i)$ 为 0 时的所有数据点数目，函数 $isSametoLast(cnt)$ 判断变量 cnt 的当前值是否与上次一样。

4 基于自然邻居和最小生成树的原型选择算法

众所周知，越靠近类边界的原型对分类有越大的贡献，而其他大量的内部原型和离群点对分类的贡献不大。本文所提算法的主要目标是排除噪声数据，同时移除大量的远离边界且对分类准确率影响不大的原型数据。因此，本文算法的第一步是使用自然邻居来排除噪声数据，包括那些没有自然邻

居的原型和与自然邻域内原型点主要类别不一样的原型数据。然后在剩余点的完备图上建立最小生成树来选择离边界近的关键原型和一些内部原型。本文算法设置了两个选择策略：当树的位置在多个不同类的边界上时，保留在该最小生成树上直接相连同时具有不同类标签的原型数据；当树的所有节点类标签都相同时，直接生成一个中心原型点取代整个树并保留下来。

接下来介绍 2NMST 算法，其分为如下 3 步：

1）基于自然邻居做数据预处理；

2）基于完备图建立若干棵最小生成树；

3）分析每一棵最小生成树，选择关键原型并保留下来。

4.1　数据预处理

在建立最小生成树之前，需要预处理数据集。这一步的目标是排除噪声数据。噪声数据指没有自然邻居的原型点或者类标签与其自然邻居内主要类标签不同的点。算法 2 给出数据集预处理的完整过程，由于使用了自然邻居来取代 K 最近邻居，因此本文提出的预处理算法没有参数，这使得算法具有更强的灵活性。

算法 2　数据集预处理

输入：原始训练集 X

输出：预处理后的训练集 X′

步骤：

```
1. 2N_Search(X)
2. X′=∅
3. For ∀i∈X
4.     If nb(i)≠0 && class(i)==getMajorClass(NN(i))
5.     Then X′=X′∪{i}
6. End For
```

在算法 2 中，函数可以获取数据点 i 的自然邻域内主要的类标签。

4.2　最小生成树的建立

在预处理完数据集之后，基于剩余点的完备图构建最小生成树。本文所提算法采用 Prim 算法来构建最小生成树，与传统的 Prim 算法不同，所提算法添加了额外的终止条件。图 1 示出了在一个包含两个类的数据集中一些原型构建最小生成树的过程。

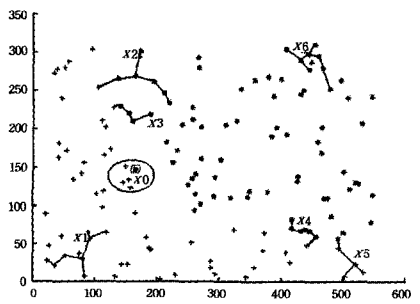

图 1　最小生成树建树示例

构建最小生成树的步骤如下：1）选择从一个随机的原型点 x 开始，在原始训练集中寻找 x 的最近邻居 x'，并将其与 x 连接起来形成最小生成树的一个新的分支；2）如果还存在原型没有在任何树中，则继续重复选择离当前 MST 最近的原型点并生成新的分支；3）当满足任一终止条件时，最小生成树停止生长。本文所提算法在构建 MST 的过程中使用了两个终止策略，一方面，通过设置最小生成树的节点阈值来控制树的规模，若一个正在生长的最小生成树的节点总数超过了节点阈值，则该树的生长必须终止；另一方面，当一个 MST 没有生长能力时，就需要停止，即当 MST 的所有节点都处于终止状态 stop 时，该树的生长必须终止。在 MST 生长过程中，首先要找到离正在生长的最小生成树最近的原型点，那些已经存在于 MST 中的原型点将不再被考虑，其余点中最近的原型点将会被插入到当前的最小生成树中，并且连接到树中最近的节点；如果最近的原型点的类标签不同于其在 MST 中连接的树节点的类标签，则在这个节点插入树中时，将这个新加入的节点和与其相连的另一节点的状态都设置成 stop，当一个节点的所有相连节点的状态都是 stop 时，该节点的状态也变为 stop。

基于上述最小生成树的生长控制策略，从一个随机的没有被插入到任何一个树的原型点开始构建 MST，在建树过程中，数据集被当作一个整体考虑。当不同类标签的原型点加入进来时，则表明最小生成树生长到了剩余原型集的边界位置，并且一些节点的状态开始变为 stop 以便后续选择边界原型。当 MST 的生长总是遇到相同类标签的原型时，表明节点阈值能够有效地分割数据集为许多不同的部分，这有利于生成关键的内部原型。根据建树的终止条件，原型选择和原型生成会不断持续直到所有原型都被搜索到。以上给出了本文算法做约简的基本思想，算法 3 详细描述了建树的过程。

算法 3　构建最小生成树

输入：预处理后的训练集 X′；

　　节点阈值 tree_capacity

输出：构建的最小生成树集 msts

步骤：

```
1. 初始化：顶点集 vertexs=X′，最小生成树集 msts=∅
2. While isEmpty(vertexs)==false
3.     从 vertexs 中任选一点 i 开始建树 tree，并将 i 从 vertexs 中移除
4.     While isEmpty(vertexs)==false && num(tree)<tree_capacity && allStop(tree)==false
5.         在 vertexs 中寻找离 tree 最近的点 j
6.         将 j 连接到 tree 中最近的邻居点 k，并将 j 从 vertexs 中移除
7.         If class(j) !=class(k)
8.             status(j)=status(k)=stop
9.             For each node n connect to k
10.                If all nodes connect to n are in stop
11.                Then status(n)=stop
12.            End For
13.        End If
14.    End While
15. msts=msts { tree }
16. End While
```

在算法 3 中，函数 *isEmpty*()判断数据集是否为空，函数 *num*(*tree*)获取最小生成树 *tree* 中的节点总数，函数 *allStop*(*tree*)判断树中所有节点状态是否都为 stop。

4.3　原型选择

依据前面的建树过程，可获得若干最小生成树，原型选择

就是基于这些树的。这些 MST 有两种类型：1)树中每个节点原型的类标签都相同，这说明该最小生成树完全位于一个簇的中间，算法 2NMST 生成一个新的中心原型来取代整个 MST 上的节点并保留了下来；2)树中的节点包含有不同类标签原型，也就是说，MST 位于不同簇的边界处，2NMST 算法选择那些直接相连同时类标签也不一样的原型点进行保留。这两个原型选择策略可以约简大部分远离边界的内部原型，只保留边界原型，同时生成一些具有代表性的内部原型保留下来。算法 4 详细描述了原型选择的过程。

算法 4 原型选择

输入：构建的最小生成树集 msts

输出：最终保留的原型集 PS

步骤：

```
1. PS=∅
2. For ∀tree ∈msts
3. flag=0
4.     For ∀branch(x,y)∈tree
5.         If class(x) !=class(y)
6.             Then PS=PS∪{x,y},flag=1
7.     End For
8.     If flag ==0
9.         Then PS=PS∪{centriodPoint(tree)}
10. End For
```

在算法 4 中，函数 $branch(x,y)$ 表示树中连接节点 x 和 y 的边，函数 $centriodPoint(tree)$ 生成了最小生成树 $tree$ 的中心原型点。

4.4 算法时间复杂度分析

假设训练集 X 的规模为 n，算法 1 用于计算 X 中所有数据点之间的距离，计算距离的时间复杂度是 $O(n^2)$，在算法 1 中的邻居搜索时间复杂度是 $O(n*sup_k)$，算法 1 的终止条件将使得 sup_k 变得相对更小。算法 2 的计算时间复杂度主要来自于邻居搜索，即 $O(n*sup_k)$。算法 3 中使用的距离数据都是前面已经计算的距离矩阵，同时使用 Prim 算法建树，其计算时间复杂度是 $O(n^2d)$。算法 4 的计算时间复杂度是 $O(n)$。由前面的算法说明可以看出，2NMST 与 BNNT 算法的时间复杂度是相同的，都是 $O(n^2d)$。

5 实验及分析

本节基于 15 个 UCI 数据集对 2NMST 和其他相关算法进行实验。所有的实验都是使用五折交叉验证，即每个 UCI 数据集被划分成 5 个互不相同的数据子集，同时每个原型选择算法都会用其中的 4 个数据子集的数据来做训练，然后用余下的一个数据子集来做测试，每一个数据子集都将做一次测试集，5 次的平均结果作为最终的结果。分类准确率(Acc)和保留率(Str)是两个主要的考查指标。对每种算法来说，保留率就是原始数据集经过该算法处理之后保留下来的数据子集所占的比例，可以通过 $Str=100|PS|/|X|$ 计算得出，其中 PS 是通过原型选择算法从原始数据集 X 中保留下来的数据子集，$||$ 表示一个数据集的原型数量；分类准确率的检测使用了 KNN 算法，准确率的计算可以通过 $Acc=100|TS'|/|TS|$ 计算得出，其中 TS 表示测试集，TS' 表示测试集中被正确识别的测试子集。在所有的实验中，距离度量都是使用欧氏距离，最后的分类使用 KNN 算法，选择 $k=3,5,7$ 中分类准确率最高的值作为最终的 k 值进行实验。使用的数据集在表 1 和表 2 中进行了展示，包括数据集的名称、规模、维度和类标签数量。

本文所提算法使用了自然邻居，因此没有参数选择，但是在最小生成树的构建过程中使用了节点阈值参数 tree_capacity 来控制 MST 的生长规模。为了检验本文算法对参数 tree_capacity的敏感度，选择了不同值的 tree_capacity 在 2NMST 中进行实验。Tree_capacity 具体选择了 $n/40$，$n/35$，$n/30$，$n/25$，$n/20$，$n/15$，$n/10$ 和 $\sqrt{n}$，其中 n 是训练集的原型数量。基于以上定义的不同的参数值，实验选择了相同的 10 个 UCI 数据集来比较 2NMST 算法的实验效果，实验结果展示在表 1 和图 2 中。当 tree_capacity 的值较小时，原型集中包含的内部原型的数量就会较大，从而会得到更大的保留率和更高的准确率。相反，如果 tree_capacity 值的较大，则最后原型集包含的内部原型数量就会更少，导致保留率也会更小。但是当 tree_capacity 的值变大时，由于第二个终止条件的控制，即类边界的影响，最后保留的原型集也不会突然变得很小。实验结果表明，随着 tree_capacity 值的不断增大，保留率变化缓慢，虽然保留率在变小，但是分类的准确率却没有显著降低。因此，对于不同的保留率，分类准确率并没有明显差异。根据理论分析和实验评估可以得出结论：2NMST 算法对于节点阈值参数 tree_capacity 并不敏感，它是一个比较独立的原型选择算法。实验发现，在 tree_capacity 取 $\sqrt{n}$ 时，针对不同规模的数据集，通常都能得到较低的保留率和较高的准确率，因此在后面的实验中，算法 2NMST 都取 tree_capacity 为 $\sqrt{n}$。

表 1 算法 2NMST 中参数 threshold 取不同值时准确率和保留率的比较结果

Dataset	$n/40$		$n/35$		$n/30$		$n/25$		$n/20$		$n/15$		$n/10$		$\sqrt{n}$	
	Acc	Str	Acc	Str	Acc	Str	Acc	Str	Acc	Str	Acc	Str	Acc	Str	Acc	Str
Iris (150,4,3)	97.33	34.33	97.33	34.00	96.67	27.17	**98.00**	23.17	96.00	20.67	96.67	19.83	94.67	**13.33**	95.33	19.33
Wine (178,13,3)	97.16	29.21	97.17	28.93	96.11	24.44	96.63	23.88	96.08	19.10	**97.27**	19.10	97.19	**17.41**	96.65	18.68
Sonar (208,60,2)	**68.82**	29.69	67.34	**26.68**	65.78	27.29	65.31	29.22	68.29	31.50	66.32	31.37	65.38	32.69	67.33	30.76
Glass (214,9,6)	65.93	24.76	64.95	23.25	62.54	21.73	**66.38**	22.43	64.50	23.13	63.09	21.83	61.90	**20.30**	64.01	21.61
Liver (345,6,5)	67.54	22.25	66.09	20.58	64.06	21.16	65.80	20.07	65.80	18.70	65.22	20.07	65.22	20.80	**68.12**	**18.41**
Yeast (1484,8,10)	56.62	19.27	56.07	**18.62**	56.60	20.57	56.06	20.97	56.87	21.88	56.87	21.65	**58.36**	26.13	56.81	19.68
Segmentation (2100,19,7)	84.57	21.14	83.95	**20.45**	83.14	22.69	85.38	26.86	85.52	24.50	82.00	25.57	**87.29**	28.30	85.43	21.49
Spambase (4601,57,10)	74.96	9.90	75.09	10.20	74.79	11.07	74.79	11.40	74.70	11.73	**75.33**	13.21	75.09	13.43	74.68	8.92
Pendigits (10992,16,10)	94.13	13.56	93.77	13.38	92.72	14.69	92.62	17.87	**96.45**	18.61	95.41	24.49	95.86	25.90	95.00	8.53
Letter (20000,16,26)	**89.22**	45.84	87.23	48.27	88.47	42.58	87.00	38.34	85.57	37.70	85.43	29.96	85.12	**26.50**	85.64	32.01
Average	**79.63**	24.99	78.90	24.44	78.09	23.34	78.80	23.42	78.98	22.75	78.36	22.71	78.61	22.48	78.90	**19.94**

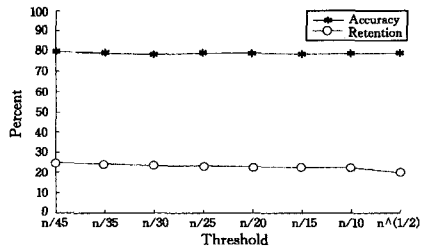

图 2　表 1 中平均准确率和保留率的折线图

为了展示本文所提算法的性能，重点将 2NMST 与最新的 BNNT 算法进行了比较，同时也与其他几种算法进行了比较，包括 CNN，ENN，TRKNN 和 PSC。在提出 BNNT 算法的论文基础之上，我们做了很多相似的比较实验，由于 TRKNN，PSC 和 BNNT 算法都有参数存在，因此选择了提出该算法的论文中最理想的参数值进行比较实验。除 2NMST 的实验数据之外，其他算法的实验数据均是参考提出 BNNT 算法的文献[2]。

首先，对 BNNT 和 2NMST 算法基于 14 个相同数据集进行了实验。对于 BNNT 算法，设置 $tree\_level=\sqrt{n}$。KNN 算法也被用来作比较，其由于没有约简，因此保留率是 100%。表 2 列出了准确率和约简率的结果。同样，图 3 示出了准确率和保留率之间的比较。从这些实验结果中可以看出，2NMST 的准确率最高或非常接近最高，同时保留率几乎都是最低的。对于中小型数据集，相比 BNNT 算法，2NMST 算法在准确率和保留率上有明显的优势；而对于大数据集时，如 Spambase，Pendigits 和 Letter，2NMST 不仅可以保持好的准确率，而且相比其他算法在保留率上还有一定优势，并且 2NMST 在平均的情况下总是表现得最好，拥有最小的保留率，同时准确率还比 KNN 略高。

表 2　算法 2NMST 与 KNN，BNNT 的准确率和保留率的比较结果

Dataset	KNN		BNNT($\sqrt{n}$)		2NMST($\sqrt{n}$)	
	Acc	Str	Acc	Str	Acc	Str
Iris	95.84	100	93.33	21.67	**95.33**	**19.33**
Wine	67.61	100	55.54	23.88	**96.65**	**18.68**
Sonar	78.70	100	65.63	42.07	**67.33**	**30.76**
Glass	62.47	100	55.13	26.87	**64.01**	**21.61**
Liver	54.92	100	47.68	52.54	**68.12**	**18.41**
Yeast	55.88	100	48.11	**18.62**	**56.81**	19.68
Segmentation	94.03	100	**86.15**	**19.76**	85.43	21.49
Spambase	73.51	100	68.37	34.34	**74.68**	**8.92**
Pendigits	98.29	100	**95.21**	17.79	95.00	**8.53**
Letter	95.24	100	84.06	**25.49**	**85.64**	32.01
Breast cancer(699,9,2)	96.71	100	90.85	33.35	**97.28**	**8.30**
Pima Indians(768,8,2)	71.92	100	69.75	35.86	**70.19**	**15.46**
Balance scale(625,4,3)	83.84	100	79.58	38.47	**83.99**	**10.76**
Landsat(6435,36,6)	90.32	100	**89.94**	15.96	88.22	**11.60**
Average	79.95	100	73.52	29.05	**80.62**	**17.54**

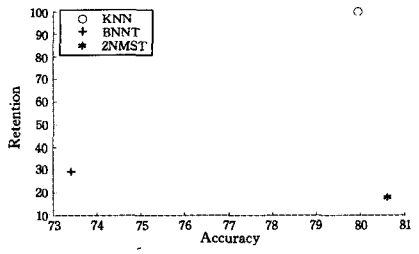

图 3　表 2 中平均准确率和保留率的散点图

其次，对 2NMST 和 CNN，ENN，TRKNN，PSC 和 BNNT 算法进行了实验比较，实验中选择的算法参数都是使该算法运行效果最好的参数值。实验结果如表 3 和图 4 所示，可以看出：对于 Iris，Wine，Glass，Liver 和 Letter 数据集，2NMST 算法的准确率比其他算法更高；而对于其他数据集，2NMST 也可以获得较好的准确率。整体上看来，2NMST 的保留率更小，而且数据集越大，保留率越小；同时，2NMST 算法的平均性能依然是最好的。

表 3　算法 2NMST 与 CNN，ENN，PSC，TRKNN，BNNT 的准确率和保留率的比较结果

Dataset	CNN		ENN		PSC		TRKNN		BNNT		2NMST	
	Acc	Str	Acc	Str	Acc	Str	Acc	Str	Acc	Str	Acc	Str
Iris	88.67	51.67	88.34	70.83	92.72	31.67	92.08	58.83	94.67	22.08	**96.00**	**19.17**
Wine	67.43	67.42	69.40	67.77	51.93	36.52	63.23	57.94	55.56	23.17	**97.76**	**17.69**
Sonar	61.08	46.27	**74.76**	58.59	63.98	55.59	56.47	56.19	62.98	40.56	65.35	**28.48**
Glass	58.87	65.42	53.52	49.36	57.37	48.78	54.10	63.08	46.49	26.29	**63.13**	**23.46**
Liver	45.75	58.33	46.87	91.49	46.22	42.39	48.02	43.84	46.47	49.82	**66.09**	**19.42**
Yeast	46.09	37.82	44.51	26.28	48.42	27.21	**75.96**	41.23	47.17	19.12	56.48	**18.97**
Segmentation	**94.42**	74.70	77.06	**16.43**	81.71	50.09	58.49	39.73	86.87	19.67	85.14	20.92
Spambase	55.31	22.69	46.04	40.25	68.63	42.21	**76.54**	35.79	68.76	35.36	74.59	**8.88**
Pendigits	**98.43**	32.06	84.33	21.24	82.67	28.75	80.05	39.53	95.39	17.12	96.32	9.28
Letter	57.63	29.15	54.41	43.73	73.73	**23.51**	68.32	24.19	83.91	24.92	**84.80**	29.10
Average	67.37	48.55	63.92	48.60	66.74	38.67	67.33	46.03	68.83	27.81	**78.57**	**19.54**

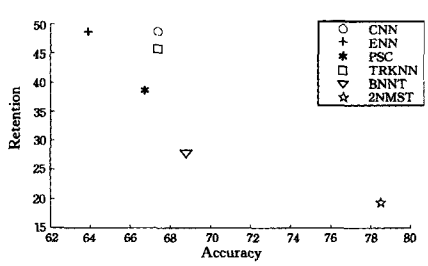

图 4　表 3 中平均准确率和保留率的散点图

此外，为了验证 2NMST 算法处理实际应用的能力，选择了著名的光手写字数据集来测试该算法的性能。表 4 所列的实验结果表明 2NMST 有更好的优势，可以获得较高的分类准确率和最小的保留率，在处理实际工作时具有更好的效果。总体上，从分类准确率和数据集保留率角度考虑，2NMST 是一个有效的原型选择算法，并且相比其他算法拥有明显的优势。

表 4　不同算法基于光手写字数据集的比较结果

Algorithm	KNN	CNN	ENN	PSC	TRKNN	BNNT	2NMST
Classification accuracy	97.99	92.07	95.57	93.25	94.48	**97.08**	97.00
Retention ratio	100.00	41.34	65.72	33.94	21.86	24.27	**10.03**

结束语　KNN 分类是最常见、最简单的有监督分类算法之一，然而参数 K 的选择和大规模数据集下过高的时间和空

（下转第 268 页）

[J]. 电子学报,2007,35(2):261-264.

[3] WAN X L. Unary Operator Logic Theory of three Agents—Special Function Relativity under the View of Modern Modal Logic [J]. Journal of Huazhong University of Science and Technology (Social Science Edition),2012,26(3):38-47. (in Chinese)
万小龙. 一元算符逻辑理论三探——狭义函数相对论视野下的现代模态逻辑[J]. 华中科技大学学报(社会科学版),2012,26(3):33-39.

[4] ZHAO X. Research Progress of Multi Modal Logic[J]. Philosophical Trends,2013(2):92-96. (in Chinese)
赵贤. 多模态逻辑研究进展[J]. 哲学动态,2013(2):92-96.

[5] ZHU W J,XIAO X. An Extension of the Propositional Calculus System of Medium Logic (I)[J]. Journal of Nanjing University (Natural Sciences Edition),1990,26(4):564-578.

[6] ZHU W J,XIAO X. Propositional Calculus System of Medium logic (II)[J]. J Math Res & Exposition,1988,8(3):457-466.

[7] ZOU J,QUI W. Medium Modal Logic Formal System and Semantics[C]// Proc 19th Intern Symp Multiple-Valued Logic. 1989.

[8] GONG N S,ZHANG D M,ZHU W J. Theory and Implementation of the Medium Automatic Reasoning (IV) - a kind of Modal Logic System based on Medium Logic[J]. Pattern Recognition and Artificial Intelligence,1995,8(1):1-8. (in Chinese)
宫宁生,张东摩,朱梧槚. 中介自动推理的理论与实现(IV)一类基于中介逻辑的模态逻辑系统[J]. 模式识别与人工智能,1995,8(1):1-8.

[9] ZHANG D M,GONG N S. Theory and Implementation of the Medium Automatic Reasoning (V)-the table deduction system of the medium modal logic MK[J]. Pattern Recognition and Artificial Intelligence,1995,8(2):114-120. (in Chinese)
张东摩,宫宁生. 中介自动推理的理论与实现(V)—中介模态逻辑 MK 的表推演系统[J]. 模式识别与人工智能,1995,8(2):114-120.

[10] PAN Z H. A Logical Description of Different Negative Relation in Knowledge[J]. Progress in Natural Science,2008,18(11):66-74.

[11] PAN Z H. Three Kinds of Negation of Fuzzy Knowledge and Their Base of Set[J]. Chinese Journal of Computers,2012(7):1421-1428.

[12] PAN Z H. Fuzzy Propositional Logic System of distinguish between three kinds of Negation and its Application[J]. Journal of Software,2014,25(6):1255-1272. (in Chinese)
潘正华. 区分 3 种否定的模糊命题逻辑系统及其应用[J]. 软件学报,2014,25(6):1255-1272.

(上接第 245 页)

间复杂度限制了其应用。为了解决这些问题,本文提出了一种新的原型选择算法 2NMST,该算法可以保证分类准确率不降低甚至更高的情况下,约简原始数据集,降低数据集的保留率。2NMST 算法使用了自然邻居(2N),它是一个全新的邻居概念。2NMST 在数据集预处理阶段使用自然邻居来移除噪声数据,然后使用最小生成树来保留关键原型,包括边界原型和生成的内部原型等,从而有效约简数据集。在数据集预处理阶段使用自然邻居使得处理结果明显优于 KNN,因为自然邻居是一个无参数的最近邻居概念,每个数据原型的邻居数都是自动计算的。与其他不同的原型选择算法如 TRKNN,PSC 和 BNNT 的实验比较结果表明,2NMST 的性能优于其他算法。

在中小规模数据集的实验中,2NMST 总是有最高的分类准确率和最低的原型保留率;而对于大规模数据集,2NMST 也可以获得与 BNNT 一样好的准确率,而且 2NMST 的保留率更低,性能更好。所以,可以得出结论:2NMST 相比其他原型选择算法,是具有竞争力的,在相比较的原型选择算法中,2NMST 无疑是一个很好的选择。

不过,本文所提算法存在与 BNNT 相似的问题,就是需要一个节点阈值参数来控制最小生成树的生长,我们发现,在 2NMST 中取节点阈值为 tree_capacity 是一个较好的选择,它是一个无尺度的,对任意规模的数据集都可以自动产生一个合适的参数值。但是,这个参数值的选择在理论上还没有得到证明,所以我们会在未来的工作中重点研究算法的优化和理论分析。

参 考 文 献

[1] FAYED H,ATIYA A F. A Novel Template Reduction Approach for the-Nearest Neighbor Method[J]. IEEE Transactions on Neural Networks,2009,20(5):890-896.

[2] LI J,WANG Y. A new fast reduction technique based on binary nearest neighbor tree [J]. Neurocomputing,2015,149:1647-1657.

[3] OLVERA-LÓPEZ J A,CARRASCO-OCHOA J A,MARTÍNEZ-TRINIDAD J F. A new fast prototype selection method based on clustering[J]. Pattern Analysis and Applications,2010,13(2):131-141.

[4] HART B P E. The condensed nearest neighbor rule[J]. IEEE Trans Information Theory,1968,14:515-516.

[5] CHOU C H,KUO B H,CHANG F. The generalized condensed nearest neighbor rule as a data reduction method[C]// 18th International Conference on Pattern Recognition, 2006 (ICPR 2006). IEEE,2006:556-559.

[6] ANGIULLI F. Fast nearest neighbor condensation for large data sets classification[J]. IEEE Transactions on Knowledge and Data Engineering,2007,19(11):1450-1464.

[7] WILSON D L. Asymptotic properties of nearest neighbor rules using edited data[J]. IEEE Transactions on Systems Man and Cybernetics,1972,SMC-2(3):408-421.

[8] WILSON D R,MARTÍNEZ T R. Reduction techniques for instance-based learning algorithms[J]. Mach Learn,2000,38(3):257-286

[9] ZOU X,ZHU Q,JIN Y. An adaptive neighborhood graph for LLE algorithm without free-parameter[J]. International Journal of Computer Applications,2011,16(2):20-23.

[10] ZOU X L,ZHU Q S. Abnormal structure in regular data revealed by Isomap with natural nearest neighbor[M]// Advances in Computer Science, Environment, Ecoinformatics, and Education. Springer Berlin Heidelberg,2011:538-544.

[11] ZHU Q S,ZHANG Y,LIU H J. Classification Algorithm Based on Natural Nearest Neighbor[J]. Journal of Information and Computetional Science,2015,12(2):573- 580.