**针对模糊K-近邻的样例选择算法**

翟俊海1 齐家兴1 张素芳2

1（河北省机器学习与计算智能重点实验室，河北大学数学与信息科学学院 保定 071002）

2（中国气象局气象干部培训学院河北分院 保定 071000）

mczsf@126.com

摘 要 压缩近邻（Condensed Nearest Neighbor, CNN）算法是一种针对K-近邻（K-Nearest Neighbor, K-NN）的样例选择算法。受CNN算法的启发，提出了一种针对模糊K-近邻的样例选择算法，并于CNN算法进行了实验比较。该算法从一个初始化的样例集合S开始，以递增方式从数据集T中选择样例。具体地，提出的算法分为3步：首先，对于T中的每一个样例***x***，在S中寻找***x***的K个近邻，并计算这K个近邻的模糊隶属度；然后，用模糊K-近邻算法，计算样例***x***的类别隶属度；最后，计算样例***x***的信息熵，并根据信息熵的大小，确定是否选择该样例。本文提出的算法有如下几个优点：(1) 速度快；（2）选择样例更具代表性；（3）易于实现。虽然提出的算法思想简单，但是实验结果显示提出的算法非常有效。此外，还得出了一些有价值的结论。

关键词 K-近邻，模糊K-近邻，模糊隶属度，样例选择，信息熵

中图法分类号：TP181 文献标识码：A

An Instance Selection Algorithm for Fuzzy K-Nearest Neighbor

ZHAI Jun-hai1 QI Jia-xing1 ZHANG Su-fang2

1(Hebei Key Laboratory of Machine Learning and Computational Intelligence, College of Mathematics and Information Science, Hebei University, Baoding 071002, China)

2(Hebei Branch of China Meteorological Administration Training Centre, China Meteorological Administration, Baoding 071000, China)

**Abstract** Condensed nearest neighbor (CNN）is an instance selection algorithm for K-nearest neighbor (K-NN). Motivated by the idea of CNN, an instance selection algorithm for fuzzy K-nearest neighbor was proposed in this paper. Starting from an initialized instance set S, the proposed algorithm incrementally selects informative instances from instance set T. Specifically, the proposed algorithm consists of three steps. Firstly, for each instance *x*∈S, find its K-nearest neighbors in S, and calculate the fuzzy membership degrees of the K-nearest neighbors. Secondly, compute the fuzzy membership degrees of *x* using fuzzy K-nearest neighbor algorithm. Finally, calculate the information entropy of *x*, and according to the value of the information entropy to determine whether to select this instance. The proposed algorithm has three advantages: (1) fast speed; (2) the selected samples are more representative; (3) easy to implement. Although the proposed algorithm is simple, experimental results show that it is very effective. In addition, some valuable conclusions have been drawn from the study.

**Key words** K-nearest neighbor, Fuzzy K-nearest neighbor, Fuzzy membership degree, Instance selection, Information entropy

# 1引言

K-近邻（K-Nearest Neighbor, K-NN）[1]是一种简单易用的分类算法，其简单易用性体现在它不需要训练分类器模型，只需要计算测试样例与训练样例之间的距离。虽然K-NN广泛应用于许多领域，但是它有如下两个缺点[2]：

（1）计算复杂度高。为了分类测试样例，需要计算测试样例与训练集中所有样例之间的距离。此外，还需要将整个训练集加载到内存中。

（2）对噪声敏感。为了分类测试样例，需要所有的训练样例，即便是噪声或错误的样例也不例外。

为了克服K-NN的第一个缺点，研究人员提出了基于样例选择的解决方案。样例选择是从训练集T中选择一个子集S，用S代替T训练分类器。历史上，第一个样例选择算法是Hart针对1-近邻提出的压缩近邻算（Condensed Nearest Neighbor, CNN）[3]。在CNN算法的基础上，研究人员提出了许多针对K-近邻的样例选择算法。根据比较在S和在T上训练的分类器性能的差异，这些算法可以分为3类[4, 5]：能力保持型、能力增强型和混合型。根据样例选择方式的不同，这些算法也可以分为另外三类[2]：递增型、递减型和混合型。

递增算法从初始化的集合S开始，按着某种样例选择准则，逐渐从集合T中选择重要的样例加入到S中，直到满足预定义的停止条件。集合S可以初始化为空集，也可以从T中随机选择若干样例初始化S。在CNN算法中，是从T中随机选择一个样例初始化S。样例选择准则是T中的样例被S中的样例用1-近邻错误分类的样例是重要的样例，将T中这样的样例逐渐加入到S中，直到T为空集，或T中的样例都能用S中的样例用1-近邻正确分类。如果子集S是T的最小子集，那么称S为T的最小一致子集。在CNN算法的基础上，Tomek提出了两个CNN的改进算法[6]。第一个改进算法和CNN类似，不同点是当一个样例***x***被S用1-近邻错误分类时，并不是将***x***加入S中，而是寻找其异类最近邻***y***的最近邻*z*，而且要求***z***和***x***属于同一个类。然后，将***z***加入到S中。第二个改进算法不是从原训练集T中选择样例，而是从T的一个子集F中选择样例，而F中的样例与其最近邻属于相同的类。Devi和Murty提出了一种称为MCNN（Modified CNN）的样例选择算法[7]。当一个样例被错误分类时，MCNN算法并不是直接将其加入S，而是对所错误分类的样例先打上标记。当T中的所有样例都测试完后，从每一类中选一个代表性的样例（类中心样例）加入S中。Chang等人提出的GCNN（Generalized CNN, GCNN）是CNN的另一种改进[8]。GCNN用每一类中得票最多的样例初始化S，然后运用CNN规则从T中选择样例。不同的是，被S正确分类的样例***x***，不仅要求和它在S中的最近邻属于相同的类，而且还要求他们之间的距离要小于***x***到其异类最近邻之间的距离。基于Voronoi划分和Voronoi异类最近邻的概念，Angiulli提出了FCNN（Fast CNN）算法[9]。

递减算法初始化S=T，然后按着某种样例选择准则，从S中将不重要的样例逐渐删除，直到满足预定义的停止条件。代表性的递减算法包括RNN、Shrink、MCS和DROP系列算法等。RNN（Reduced Nearest Neighbor）[10]初始化S=T，然后从S中删除每一个样例。如果删除一个样例，没有引起T中的样例被错误分类，那么保留该样例在S中。Shrink算法[11]和RNN算法的思想是类似的，不同点是Shrink仅考虑删除的样例是否被正确分类。用CNN算法得到的子集S未必是最小一致子集，为了确保S是最小一致子集，Dasarathy提出了MCS（Minimal Consistent Subset）算法[12]。基于异类最近邻和关联样例的概念，Wilson和Martinez提出了DROP（Decremental Reduction Optimization Procedure）[13]系列算法。这些算法都是从S=T开始，按着不同的准则从S中样例。例如，DROP1删除样例的准则是：如果没有S中的样例***x***参与分类，它的关联样例也能被正确分类，则删除该样例***x***。

在递增算法中，S中的样例只增不减。而在减量算法中，S中的样例只减不增。混合算法是递增算法和减量算法的组合，在样例选择的过程中，S中的样例既可以增加，也可以删除。从直观上看，混合算法的性能应该优于递增算法和减量算法，但由于这种算法需要额外度量S中样例的重要性，其计算量大，计算复杂度高。因此，这种算法研究的相对较少。

针对K-NN对噪声敏感的问题，Keller等人提出了模糊K-NN[14]。但是模糊K-NN没能解决K-NN的第一个缺点，其计算复杂度依然比较高。实际上，模糊K-NN的计算机复杂度比K-NN的还高。目前，针对模糊K-NN的样例选择只有Zhai等人[15]做了初步探讨。受CNN算法的启发，本文提出了一种针对模糊K-近邻的样例选择算法。虽然提出的算法思想简单，但是实验结果证明该算法非常有效。

# 2 **K-NN与CNN**

# **2.1 K-NN**

K-NN的思想非常简单。设T是训练集，***x***是测试样例。在T中寻找距离***x***最近的K个样例，这K个样例所属类别最多的类就是***x***的类别。K-NN算法的伪代码如算法1所示。

**算法1. K-NN**

输入：训练集T，测试样例***x***，参数K

输出：***x***的类别

1. for (each ***x***'∈T)

2. 计算***x***与***x***'之间的距离；

3. 寻找距离***x***最近的K个样例，设这K个样例构成的集合为N；

4. 计算；

5. //其中，*I*( )是特征函数，L是类别集合；

5. 输出***x***的类别。

# **2.2 CNN**

CNN是针对K-NN的样例选择算法，它认为被错误分类的样例比较重要。开始时，从训练集T中随机选择一个样例加入到S中，然后对T中的其他样例***x***，在S中寻找其最近邻，如果***x***与其最近邻类别不同（即***x***被错误分类），那么将***x***加入到S中；如果***x***与其最近邻类别相同（即***x***被正确分类），那么不将***x***加入到T中。当T变成空集，或T中剩余的样例都能被S用1-近邻正确分类时，算法结束。CNN算法的伪代码如算法2所示。

**算法2. CNN**

输入：训练集T

输出：

1. 从T中随机选择一个样例加入到S中；

2. do

3. for (each ***x***∈T)

4. 在S中选择***x***的最近邻***x***'；

5. if (class(***x***)≠class(***x***'))

6. ；

7. until (T为空集或T中的样例都能被S用1-近邻正确分类。

# 3 模糊K-NN与CFNN

为了内容的自包含性，本节首先介绍模糊K-NN[14]，然后介绍提出的样例选择算法CFNN（Condensed Fuzzy Nearest Neighbor）。

# 3.1 模糊K-NN

为了克服K-NN的第二个缺点，Keller等人提出了模糊K-NN。对于待分类的测试样例***x***，模糊K-NN通过***x***的K个近邻的模糊隶属度来确定***x***的模糊隶属度。具体地，***x***的模糊隶属度用下面的公式(1)计算。在公式(1)中，表示***x***的第*i*个近邻***x****i*隶属于第*j*类的隶属度，，*p*是训练样例的类别数。而用下面的公式(2)计算，模糊K-NN算法的伪代码如算法3所示。

 (1)

 (2)

在公式(2)中，表示样例***x****i*到第*j*类的中心***c****j*的距离。

**算法3. 模糊K-NN**

输入：训练集T，测试样例***x***，参数K

输出：***x***的模糊隶属度，

1. for (each ***x***'∈T)

2. 计算***x***与***x***'之间的距离；

3. 寻找距离***x***最近的K个样例，；

4. 用公式(2)计算；

5. 用公式(1)计算

6. 输出

# 3.2 CFNN

下面介绍提出的针对模糊K-近邻的样例选择算法，称之为压缩模糊近邻（Condensed Fuzzy Nearest Neighbor, CFNN）算法，CFNN也可以看作是CNN在模糊环境下的推广，如图1所示。



图1 CFNN与CNN之间的关系

Fig. 1 The relationship between CFNN and CNN

与CNN算法不同，在CFNN中，子集S用每一类随机选取的K个样例进行初始化；此外，测试样例***x***的K个近邻的模糊隶属度的计算不是用原始训练集T进行计算，而是用子集S进行计算。CFNN算法的伪代码如算法4所示。

**算法4. CFNN**

输入：训练集T，参数K，阈值λ

输出：

1. 从T的每一类样例中，随机选择K个样例初始化子集S，并将选择的样例从T中删除；

2. for (each ***x***∈T)

3. 在S中寻找***x***的K个近邻；

4. 用公式(2)计算这K个近邻的隶属度；

5. 用公式(1)计算***x***的模糊类别隶属度；

6. 计算***x***的信息熵；

7. if (E(***x***)>*λ*)

8. ；

6. 输出S。

根据模糊K-NN，为了确定测试样例的模糊隶属度，只需要测试样例的K个近邻的模糊隶属度。在我们之前的工作中，所有样例的模糊隶属度都需要计算，而且用原始训练集T进行计算。在本文提出的方法中，只需要用S计算测试样例的K个近邻的模糊隶属度。这种计算策略不仅可以大幅度降低计算复杂度，而且计算得到的模糊隶属度更精确，因为S中的样例是选出的重要的样例。

# 4 实验结果及分析

我们在1个人工数据集、2个真实数据集和7个UCI数据集上进行了实验，并与CNN算法进行了比较。实验所用的10个数据集的基本信息列于表1中。人工数据集（记为Gaussian）是一个2维2类包括20000个样例点的数据集，由高斯分布产生，两类数据服从的高斯分布的参数列于表2中。2个真实数据集分别是CT数据集和RenRu数据集，关于这两个数据集的详细介绍见参考文献[15]。在实验中，分别取K=3和K=5。实验环境是Win10操作系统，CPU为Intel(R) Core(TM) i3-3120M CPU @ 2.50GHz，内存8G，Python3.6.3

在人工数据集上，一方面用于说明提出的方法的可行性，另一方面用可视化的方法展示选出的样例的分布情况。人工数据集Gaussian中样例的分布如图2所示，对应阈值*λ*=0.5和*λ*=0.6从人工数据集Gaussian中选出的样例的分布如图3和图4所示。从图3和图4可以看出，和大多数样例选择算法一样，利用提出的算法选出的样例都分布在分类边界附近。

表1 实验所用数据集的基本信息

Table 1 The basic information of data sets used in experiments

|  |  |  |  |
| --- | --- | --- | --- |
| 数据集 | 训练样例 | 测试样例 | 类别数 |
| Gaussian | 13333 | 6667 | 2 |
| CT | 154 | 48 | 2 |
| RenRu | 103 | 45 | 2 |
| WDBC | 388 | 167 | 2 |
| Parkinsons | 136 | 59 | 2 |
| Pima | 537 | 231 | 2 |
| Skin | 171539 | 73518 | 2 |
| Iris | 105 | 45 | 3 |
| Glass | 112 | 48 | 6 |
| Image | 135 | 57 | 7 |

表2 人工数据集的高斯分布参数

Table 2 The parameters of Gaussian distribution of the artificial data set

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 1 |  |  |
| 2 |  |  |

图2 数据集Gaussian中样例的分布

Fig. 2 The distribution of instances in Gaussian data set

图3 从数据集Gaussian中按阈值*λ*=0.5选出的样例的分布

Fig. 3 The distribution of instances selected from Gaussian data set with *λ*=0.5

图4 从数据集Gaussian中按阈值*λ*=0.6选出的样例的分布

Fig. 4 The distribution of instances selected from Gaussian data set with *λ*=0.6

对于2类的数据集，样例信息熵的最大值为1.0，在实验中，取阈值*λ*≥0.5。对于3类的数据集，样例信息熵的最大值为1.58，在实验中，取阈值*λ*≥0.7。对于6类的数据集，样例信息熵的最大值为2.58，在实验中，取阈值*λ*≥1.0。对于7类的数据集，样例信息熵的最大值为2.81，在实验中，取阈值*λ*≥1.0。在10个数据集上的实验结果列于表3-表12中。其中，表的第一行是在原训练集上分类精度，作为基准便于实验比较。最后一行是CNN算法的分类精度。

表3 在数据集Gaussian上的实验结果

Table 3 The experimental results on data set Gaussian

|  |  |  |  |
| --- | --- | --- | --- |
| 阈值*λ* | 测试精度K=3 | 测试精度K=5 | 选出的样例数 |
| 0.00 | 0.923804 | 0.9385031 | 13333 |
| 0.50 | 0.926354 | 0.927404 | 2100 |
| 0.55 | 0.927104 | 0.928154 | 2118 |
| 0.60 | 0.926204 | 0.927554 | 2051 |
| 0.65 | 0.926354 | 0.927254 | 2094 |
| 0.70 | **0.927854** | 0.926354 | 1966 |
| 0.75 | 0.926204 | 0.927404 | 1970 |
| 0.80 | 0.925004 | 0.928454 | 1924 |
| 0.85 | 0.927254 | 0.929204 | 1670 |
| 0.90 | 0.927254 | 0.929204 | 1510 |
| 0.95 | 0.927104 | **0.930103** | 1254 |
| CNN | 0.916454 | 0.926204 | 2468 |

表4 在数据集CT上的实验结果

Table 4 The experimental results on data set CT

|  |  |  |  |
| --- | --- | --- | --- |
| 阈值*λ* | 测试精度K=3 | 测试精度K=5 | 选出的样例数 |
| 0.00 | 0.940299 | 0.940299 | 154 |
| 0.50 | **0.955224** | 0.925373 | 98 |
| 0.55 | 0.940299 | 0.925373 | 82 |
| 0.60 | 0.940299 | 0.925373 | 83 |
| 0.65 | 0.940299 | 0.895522 | 78 |
| 0.70 | 0.925373 | 0.895522 | 77 |
| 0.75 | 0.880597 | 0.910448 | 72 |
| 0.80 | 0.880597 | 0.910448 | 54 |
| 0.85 | 0.865672 | 0.895522 | 43 |
| 0.90 | 0.880597 | 0.910448 | 40 |
| 0.95 | 0.910448 | **0.925373** | 31 |
| CNN | 0.865672 | 0.925373 | 52 |

表5 在数据集RenRu上的实验结果

Table 5 The experimental results on data set RenRu

|  |  |  |  |
| --- | --- | --- | --- |
| 阈值*λ* | 测试精度K=3 | 测试精度K=5 | 选出的样例数 |
| 0.00 | 0.866667 | 0.844444 | 0.00 |
| 0.50 | **0.888889** | **0.844444** | 77 |
| 0.55 | 0.844444 | 0.777778 | 61 |
| 0.60 | 0.866667 | 0.800000 | 58 |
| 0.65 | 0.822222 | 0.800000 | 51 |
| 0.70 | 0.844444 | 0.800000 | 50 |
| 0.75 | 0.777778 | 0.711111 | 43 |
| 0.80 | 0.800000 | 0.755556 | 31 |
| 0.85 | 0.800000 | 0.822222 | 28 |
| 0.90 | 0.644444 | 0.666667 | 18 |
| 0.95 | 0.644444 | 0.533333 | 12 |
| CNN | 0.711111 | 0.733333 | 27 |

表6 在数据集WDBC上的实验结果

Table 6 The experimental results on data set WDBC

|  |  |  |  |
| --- | --- | --- | --- |
| 阈值*λ* | 测试精度K=3 | 测试精度K=5 | 选出的样例数 |
| 0.00 | 0.934132 | 0.934132 | 388 |
| 0.50 | **0.940120** | 0.934132 | 108 |
| 0.55 | **0.940120** | 0.928144 | 104 |
| 0.60 | 0.910180 | 0.904192 | 86 |
| 0.65 | 0.922156 | 0.916168 | 73 |
| 0.70 | 0.910180 | 0.904192 | 83 |
| 0.75 | 0.922156 | 0.916168 | 62 |
| 0.80 | 0.922156 | 0.916168 | 75 |
| 0.85 | 0.940120 | **0.946108** | 73 |
| 0.90 | 0.916168 | 0.910180 | 71 |
| 0.95 | 0.928144 | 0.910180 | 33 |
| CNN | 0.91018 | 0.91018 | 64 |

表7 在数据集Parkinsons上的实验结果

Table 7 The experimental results on data set Parkinsons

|  |  |  |  |
| --- | --- | --- | --- |
| 阈值*λ* | 测试精度K=3 | 测试精度K=5 | 选出的样例数 |
| 0.00 | 0.898305 | 0.915254 | 136 |
| 0.50 | 0.898305 | 0.898305 | 52 |
| 0.55 | 0.898305 | **0.915254** | 64 |
| 0.60 | 0.830508 | 0.779661 | 25 |
| 0.65 | 0.898305 | 0.881356 | 73 |
| 0.70 | 0.779661 | 0.779661 | 34 |
| 0.75 | **0.915254** | 0.847458 | 44 |
| 0.80 | 0.881356 | 0.847458 | 53 |
| 0.85 | 0.864407 | 0.864407 | 31 |
| 0.90 | 0.847458 | 0.864407 | 24 |
| 0.95 | 0.881356 | 0.864407 | 20 |
| CNN | 0.898305 | 0.847458 | 48 |

表8 在数据集Pima上的实验结果

Table 8 The experimental results on data set Pima

|  |  |  |  |
| --- | --- | --- | --- |
| 阈值*λ* | 测试精度K=3 | 测试精度K=5 | 选出的样例数 |
| 0.00 | 0.692641 | 0.748918 | 537 |
| 0.50 | 0.714286 | 0.757576 | 380 |
| 0.55 | 0.714286 | 0.744589 | 368 |
| 0.60 | 0.705628 | 0.740260 | 361 |
| 0.65 | 0.679654 | 0.705628 | 207 |
| 0.70 | 0.688312 | 0.748918 | 275 |
| 0.75 | 0.714286 | 0.753247 | 325 |
| 0.80 | 0.709957 | 0.744589 | 307 |
| 0.85 | 0.705628 | 0.740260 | 315 |
| 0.90 | **0.740260** | 0.761905 | 265 |
| 0.95 | 0.735931 | **0.779221** | 248 |
| CNN | 0.692641 | 0.74026 | 282 |

表9 在数据集Skin上的实验结果

Table 9 The experimental results on data set Skin

|  |  |  |  |
| --- | --- | --- | --- |
| 阈值*λ* | 测试精度K=3 | 测试精度K=5 | 选出的样例数 |
| 0.00 | 0.999551 | 0.999483 | 171539 |
| 0.50 | 0.999538 | 0.999510 | 808 |
| 0.55 | 0.999524 | 0.999510 | 805 |
| 0.60 | 0.999510 | 0.999510 | 795 |
| 0.65 | 0.999510 | 0.999470 | 767 |
| 0.70 | 0.999524 | 0.999483 | 791 |
| 0.75 | **0.999578** | **0.999524** | 692 |
| 0.80 | 0.999538 | 0.999497 | 667 |
| 0.85 | 0.999470 | 0.999442 | 602 |
| 0.90 | 0.999565 | 0.999320 | 544 |
| 0.95 | 0.999497 | 0.999061 | 433 |
| CNN | 0.999197 | 0.997062 | 418 |

表10 在数据集Iris上的实验结果

Table 10 The experimental results on data set Iris

|  |  |  |  |
| --- | --- | --- | --- |
| 阈值*λ* | 测试精度K=3 | 测试精度K=5 | 选出的样例数 |
| 0.00 | 0.977778 | 0.977778 | 105 |
| 0.70 | 0.955556 | 0.866667 | 33 |
| 0.75 | 0.955556 | 0.955556 | 26 |
| 0.80 | 0.933333 | 0.955556 | 26 |
| 0.85 | 0.955556 | 0.977778 | 25 |
| 0.90 | **0.977778** | **0.977778** | 22 |
| 0.95 | 0.933333 | 0.955556 | 23 |
| 1.00 | 0.955556 | 0.955556 | 15 |
| CNN | 0.622222 | 0.511111 | 13 |

表11 在数据集Glass上的实验结果

Table 11 The experimental results on data set Glass

|  |  |  |  |
| --- | --- | --- | --- |
| 阈值*λ* | 测试精度K=3 | 测试精度K=5 | 选出的样例数 |
| 0.00 | 0.625000 | 0.625000 | 112 |
| 1.00 | 0.604167 | 0.604167 | 64 |
| 1.10 | 0.562500 | 0.583333 | 61 |
| 1.20 | 0.562500 | 0.583333 | 63 |
| 1.30 | 0.562500 | 0.562500 | 58 |
| 1.40 | 0.541667 | 0.625000 | 54 |
| 1.50 | **0.666667** | 0.625000 | 47 |
| 1.60 | 0.583333 | **0.645833** | 17 |
| CNN | 0.479167 | 0.500000 | 57 |

表12 在数据集Image上的实验结果

Table 12 The experimental results on data set Image

|  |  |  |  |
| --- | --- | --- | --- |
| 阈值*λ* | 测试精度K=3 | 测试精度K=5 | 选出的样例数 |
| 0.00 | 0.779661 | 0.762712 | 135 |
| 1.00 | **0.830508** | **0.813559** | 76 |
| 1.10 | 0.779661 | 0.762712 | 68 |
| 1.20 | 0.728814 | 0.762712 | 67 |
| 1.30 | 0.711864 | 0.796610 | 63 |
| 1.40 | 0.677966 | 0.644068 | 53 |
| 1.50 | 0.644068 | 0.694915 | 46 |
| 1.60 | 0.677966 | 0.542373 | 30 |
| CNN | 0.661017 | 0.508475 | 56 |

从列于表3-表12的实验结果可以看出，对K=3和K=5两种情况，即便都是2类分类问题，最优的阈值*λ*（测试精度用粗体显示的结果）也不相同。而且，阈值*λ*取大于等于0到信息熵最大值区间的中值这一结论不一定正确。对K=3的情况，通过用提出的算法进行样例选择，在所有10个数据集上，在选出的子集上训练出的分类器的性能都有所提高。对K=5的情况，在WDBC、Pima、Skin、Glass和Image5个数据集上的实验结果是能力增强的，在其他5个数据集上的实验结果是能力保持的。另外，从表3-表12的最后一列还可以看出，提出的算法压缩比也比较高。

将FKNN算法与CNN算法的实验结果进行比较可以看出，两个算法在测试集分类精度的比较下，FKNN选择的样例更少，且精度损失小，证明选择的样例比CNN更具有代表性。在运行时间上，本文提出的算法在每次迭代计算量与CNN基本相同的情况下，只需要遍历一遍数据集，而CNN算法可能会多次遍历数据集，所以运行速度更快。

## 结束语

受CNN算法的启发，提出了一种针对模糊K-近邻的样例选择算法。提出的算法既可以看作CNN算法在模糊环境下的推广，也可以看作我们之前工作[15]的一种改进。提出的算法具有如下优点：（1）CNN算法仅适用于K-NN中K=1的情况，而本文提出的算法适用于K取不同的值。（2）计算机复杂度更低，这主要体现在模糊类别隶属度的计算上，提出的算法用S进行计算，而不是用T进行计算。（3）具有动态阈值的特点，从实验结果也可以看出，对于不同的K和不同的数据集，最优的阈值*λ*是不同的。本文后续的工作将探讨在大数据环境中的可扩展性，以及对于不同的数据集，如何选择最优的阈值*λ*。

参 考 文 献

1. T. Cover, P. Hart. Nearest neighbor pattern classification[J]. IEEE Transactions on Information Theory, 1967, 13(1):21-27.
2. G. Salvador, D. Joaquin, R. C. Jose, et al. Prototype selection for nearest neighbor classification: taxonomy and empirical study[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2012, 34(3):417-435.
3. P. Hart, The condensed nearest neighbor rule[J]. IEEE Transaction on Information Theory, 1967, 14(5):515-516.
4. D. R. Wilson, T.R. Martinez. Reduction techniques for instance-based learning algorithms[J]. Machine Learning, 2000, 38(3): 257-286.
5. B. Henry, M. Chris. Advances in instance selection for instance-based learning algorithms[J]. Data Mining and Knowledge Discovery, 2002, 6:153-172.
6. I. Tomek. Two Modifications of CNN[J]. IEEE Trans. Systems, Man and Cybernetics, 1976, 6(11):769-772.
7. V. S. Devi, M. N. Murty. An incremental prototype set building technique[J]. Pattern Recognition, 2002, 35(2):505-513.
8. F. Chang, C. C. Lin, C. J. Lu, et al. Adaptive prototype learning algorithms: theoretical and experimental studies[J]. Journal of Machine Learning Research, 2006, 7(4):2125-2148.
9. F. Angiulli, Fast nearest neighbor condensation for large data sets classification[J]. IEEE Transactions on Knowledge and Data Engineering, 2007, 19(11):1450-1464.
10. G. W. Gates. The reduced nearest neighbor rule[J]. IEEE Transactions on Information Theory, 1972, 18(3): 431-433.
11. D. Kibler, D. W. Aha. Learning representative exemplars of concepts: an initial case study[C]. Proceedings of the Fourth International Workshop on Machine Learning, 1987:24-30.
12. B. V. Dasarathy. Minimal consistent set identification for optimal nearest neighbor decision systems design[J]. IEEE Transactions on Systems, Man, and Cybernetics, 1994, 24(1):511-517.
13. D. R. Wilson, T. R. Martínez. Improved Heterogeneous Distance Functions[J]. Journal of Artificial Intelligence Research, 1997, 11(1):1-34.
14. J. R. Keller, M. R. Gray, J. A. Givens. A fuzzy k-nearest neighbor algorithm[J]. IEEE Transactions on Systems, Man, and Cybernetics, 1985, 15(4):580-585.
15. J. H. Zhai, N. Li, M. Y. Zhai. The condensed fuzzy k-nearest neighbor rule based on sample fuzzy entropy[C]. Proceedings of the 2011 International Conference on Machine Learning and Cybernetics, Guilin, 10-13 July, 2011, Vol. 1, pp:282-286.