KNN Model-Based Approach in Classification

Article · August 2004

Source: CiteSeer

CITATIONS
164

READS
4,621

4 authors:

Gongde Guo
Fujian Normal University
97 PUBLICATIONS **1,042** CITATIONS
SEE PROFILE

Hui Wang
Yancheng Bioengineering Higher Vocational and Technical School
1,031 PUBLICATIONS **16,178** CITATIONS
SEE PROFILE

David A. Bell
Queen's University Belfast
299 PUBLICATIONS **4,872** CITATIONS
SEE PROFILE

Yaxin Bi
Ulster University
91 PUBLICATIONS **1,412** CITATIONS
SEE PROFILE

KNN Model-Based Approach in Classification

Gongde Guo[1], Hui Wang [1], David Bell [2], Yaxin Bi [2], and Kieran Greer [1]

School of Computing and Mathematics, University of Ulster
Newtownabbey, BT37 0QB, Northern Ireland, UK [1]
{G.Guo, H.Wang, Krc.Greer}@ulst.ac.uk

School of Computer Science, Queen's University Belfast
Belfast, BT7 1NN, UK [2]
{DA.Bell, Y.Bi} @qub.ac.uk

Abstract. The *k*-Nearest-Neighbours (*k*NN) is a simple but effective method for classification. The major drawbacks with respect to *k*NN are (1) its low efficiency - being a lazy learning method prohibits it in many applications such as dynamic web mining for a large repository, and (2) its dependency on the selection of a "good value" for *k*. In this paper, we propose a novel *k*NN type method for classification that is aimed at overcoming these shortcomings. Our method constructs a *k*NN model for the data, which replaces the data to serve as the basis of classification. The value of *k* is automatically determined, is varied for different data, and is optimal in terms of classification accuracy. The construction of the model reduces the dependency on *k* and makes classification faster. Experiments were carried out on some public datasets collected from the UCI machine learning repository in order to test our method. The experimental results show that the *k*NN based model compares well with C5.0 and *k*NN in terms of classification accuracy, but is more efficient than the standard *k*NN.

1 Introduction

The *k*-Nearest-Neighbours (*k*NN) is a non-parametric classification method, which is simple but effective in many cases [1]. For a data record *t* to be classified, its *k* nearest neighbours are retrieved, and this forms a *neighbourhood of t*. Majority voting among the data records in the neighbourhood is usually used to decide the classification for *t* with or without consideration of distance-based weighting. However, to apply *k*NN we need to choose an appropriate value for *k*, and the success of classification is very much dependent on this value. In a sense, the *k*NN method is biased by *k*. There are many ways of choosing the *k* value, but a simple one is to run the algorithm many times with different *k* values and choose the one with the best performance.

In order for *k*NN to be less dependent on the choice of *k*, Wang [2] proposed to look at multiple sets of nearest neighbours rather than just one set of *k*-nearest neighbours. The proposed formalism is based on contextual probability, and the idea is to aggregate the support of multiple sets of nearest neighbours for various classes to give a more reliable support value, which better reveals the true class of *t*. However, in its basic form the method is relatively slow, which needs $O(n^2)$ to classify a new

instance, though it is indeed less dependent on *k* and is able to achieve classification performance close to that for the best *k*.

*k*NN has a high cost of classifying new instances. This is single-handedly due to the fact that nearly all computation takes place at classification time rather than when the training examples are first encountered. Though *k*NN has been applied to text categorization since the early days of its research [3] and is shown to be one of the most effective methods on Reuters corpus of newswire stories – a benchmark corpus in text categorization. Its efficiency as being a lazy learning method without pre-modelling prohibits it from being applied to areas where dynamic classification is needed for large repository. There are techniques [10, 11] that are used to significantly reduce the computation required at query time, such as indexing training examples, but it is out of our concerns in this paper.

We attempt to solve these problems and present in this paper a *k*NN type classification method called *k*NNModel. The method constructs a model from the data and classifies new data using the model. The model is a set of representatives of the training data, as regions in the data space.

The rest of the paper is organised as follows. Section 2 discusses related research in this area. Section 3 introduces the basic idea of the proposed modelling and classification algorithm, where the modelling and classification processes are illustrated by an example with the help of some graphs. The experimental results are described and discussed in Section 4. Section 5 ends the paper with a discussion on existing problems and addresses further research directions.

2 Related Work

This work is a subsequent research based on our previous research on data reduction (DR) [4]. The advantage of DR is that raw data and reduced data can be both represented by hyper relations. The collection of hyper relations can be made into a complete Boolean algebra in a natural way, and so for any collection of hyper tuples its unique least upper bound (lub) can be found, as a reduction. The experimental result shows that DR can obtain relatively higher reduction rate whilst preserving its classification accuracy. However, it is relatively slow in its basic form of model construction, since much time is spent in trying probable merge.

As the *k*-Nearest-Neighbours classifier requires storing the whole training set and may be too costly when this set is large, many researchers have attempted to get rid of the redundancy of the training set to alleviate this problem [5,6,7,8]. Hart [5] proposed a computationally simple local search method as Condensed Nearest Neighbour (CNN) by minimizing the number of stored patterns and storing only a subset of the training set for classification. The basic idea is that patterns in the training set may be very similar and some do not add extra information and thus may be discarded. Gate [6] proposed the Reduced Nearest Neighbour (RNN) rule that aims to further reduce the stored subset after having applied CNN. It simply removes those elements from the subset which will not cause an error. Alpaydin [7] investigated some voting schemes over multiple learners in order to improve classification accuracy, and Kubat *et al* [8] addressed an approach that selects three very small groups of examples such that, when used as 1-NN subclassifiers, each tends to error

in a different part of the instance space. Simple voting then corrects many failures of individual subclassifiers. The experimental results of those methods conducted on some public datasets are reported in [9].

The proposed *k*NN model-based approach is different from the DR and other condensed nearest neighbour methods. It constructs a model by finding a set of representatives with some extra information from the training data based on similarity principle. The created representatives can be seen as regions in the data space and will be used for further classification.

3 Modeling and Classification Algorithm

3.1 Basic Idea of *k*NNModel

*k*NN is a case-based learning method, which keeps all the training data for classification. Being a lazy learning method prohibits it in many applications such as dynamic web mining for a large repository. One way to improve its efficiency is to find some representatives to represent the whole training data for classification, viz. building an inductive learning model from the training dataset and using this model (representatives) for classification. There are many existing algorithms such as decision trees or neural networks initially designed to build such a model. One of the evaluation standards for different algorithms is their performance. As *k*NN is a simple but effective method for classification and it is convincing as one of the most effective methods on Reuters corpus of newswire stories in text categorization, it motivates us to build a model for *k*NN to improve its efficiency whilst preserving its classification accuracy as well.

Looking at Figure 1, a training dataset including 36 data points with two classes {square, circle} is distributed in 2-dimensional data space.

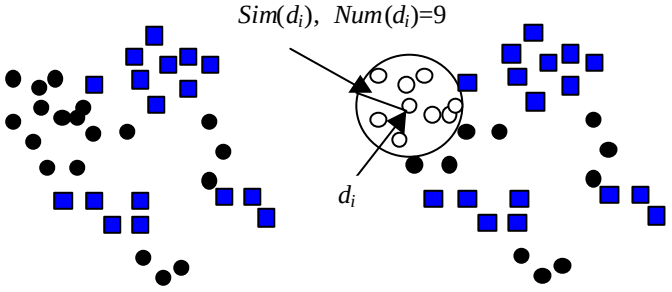

Fig. 1. The distribution of data points.

Fig. 2. The first obtained representative.

If we use Euclidean distance as our similarity measure, it is clear that many data points with the same class label are close to each other according to distance measure in many local areas. In each local region, the central data point d_i looking at Figure 2

for example, with some extra information such as $Num(d_i)$ - the number of data points inside the local region and $Sim(d_i)$ - the similarity of the most distant data point inside the local region to d_i, might be an ideal representative of this local region. If we take these representatives as a model to represent the whole training dataset, it will significantly reduce the number of data points for classification, thereby to improve its efficiency. Obviously, if a new data point is covered by a representative it will be classified by the class label of this representative. If not, we calculate the distance of the new data point to each representative's nearest boundary and take each representative's nearest boundary as a data point, then classify the new data point in the spirit of KNN.

In model construction process, each data point has its largest local neighbourhood which covers the maximal number of data points with the same class label. Based on these local neighbourhoods, the largest local neighbourhood (called largest global neighbourhood) can be obtained in each cycle. This largest global neighbourhood can be seen as a representative to represent all the data points covered by it. For data points not covered by any representatives, we repeat the above operation until all the data points have been covered by chosen representatives. Obviously, we needn't choose a specific k for our method in the model construction process, the number of data points covered by a representative can be seen as an optimal k but it is different in different representatives. The k is generated automatically in the model construction process. Further, using a list of chosen representatives as a model for classification not only reduces the number of data for classification, also significantly improves its efficiency. From this point of view, our proposed method overcomes the two shortcomings inherited in the kNN method.

3.2 Modeling and Classification Algorithm

Let $\boldsymbol{D}$ be a collection of n class-known data tuples $\{d_1, d_2,\ldots, d_n\}$. $d_i \in \boldsymbol{D}$ could be a document represented in a form of space vector $d_i=<w_{i1}, w_{i2}, \ldots, w_{im}>$, where w_{ij} could be the normalised TF-IDF weighting representation in text categorization as an example. For the reason of generalisation from now on, we use the term 'data tuple' to represent all kinds of data in different applications in this paper in order to avoid limiting our algorithm to specific applications. Also the term 'similarity measure' could be any similarity measure such as Euclidean distance or Cosine similarity only if it is suitable for a concrete application. For simplicity, from now on, we use Euclidean distance as default similarity measure to describe following algorithms.

The detailed model construction algorithm is described as follows:

(1) Select a similarity measure and create a similarity matrix from the given training dataset.
(2) Set to 'ungrouped' the tag of all data tuples.
(3) For each 'ungrouped' data tuple, find its largest local neighbourhood which covers the largest number of neighbours with the same category.
(4) Find the data tuple d_i with a largest global neighbourhood N_i among all the local neighbourhoods, create a representative $<Cls(d_i), Sim(d_i), Num(d_i),$

$Rep(d_i)$> into $\boldsymbol{M}$ to represent all the data tuples covered by N_i, and then set to 'grouped' the tag of all the data tuples covered by N_i.

(5) Repeat step 3 and step 4 until all the data tuples in the training dataset have been set to 'grouped'.

(6) Model $\boldsymbol{M}$ consists of all the representatives collected from the above learning process.

In the above algorithm, $\boldsymbol{M}$ represents the created model. The representative <$Cls(d_i)$, $Sim(d_i)$, $Num(d_i)$, $Rep(d_i)$> respectively represents the class label of d_i, the lowest similarity to d_i among the data tuples covered by N_i; the number of data tuples covered by N_i, and a representation of d_i itself. In step (4), if there are more than one neighbourhoods having the same maximal number of neighbours, we choose the one with minimal value of $Sim(d_i)$, viz. the one with highest density, as representative.

The classification algorithm is described as follows:

(1) For a new data tuple d_t to be classified, calculate its similarity to all representatives in the model $\boldsymbol{M}$.

(2) If d_t is covered only by one representative <$Cls(d_j)$, $Sim(d_j)$, $Num(d_j)$, $Rep(d_j)$>, viz the Euclidean distance of d_t to d_j is smaller than $Sim(d_j)$, d_t is classified as the category of d_j.

(3) If d_t is covered by at least two representatives with different category, classify d_t as the category of the representative with largest $Num(d_j)$, viz. the neighbourhood covers the largest number of data tuples in the training dataset.

(4) If no representative in the model $\boldsymbol{M}$ covers d_t, classify d_t as the category of a representative which boundary is closest to d_t.

The Euclidean distance of d_t to a representative d_i's nearest boundary equals to the difference of the Euclidean distance of d_i to d_t minus $Sim(d_i)$.

To improve the classification accuracy for *k*NNModel, we implemented two different pruning methods in our *k*NNModel. One method is by removing the representatives from the model $\boldsymbol{M}$ that only covers a few data tuples and the relevant data tuples covered by these representatives from the training dataset, and then constructing the model again from the revised training dataset. The second method is by modifying the step 3 in the model construction algorithm to allow each largest local neighbourhood cover r (called error tolerant degree) data tuples with different categories to the majority category in this neighbourhood. This modification integrates the pruning work into the process of model construction. Experimental results will be reported in the next section.

3.3 An Example of Model Construction and Classification Process

To grasp the idea here, the best way is by means of an example, so we graphically illustrate the model construction and classification process.

A training dataset including 36 data tuples is divided into two classes denoted as square and circle. The distribution of data tuples in 2-dimensional data space is shown in Figure 3.

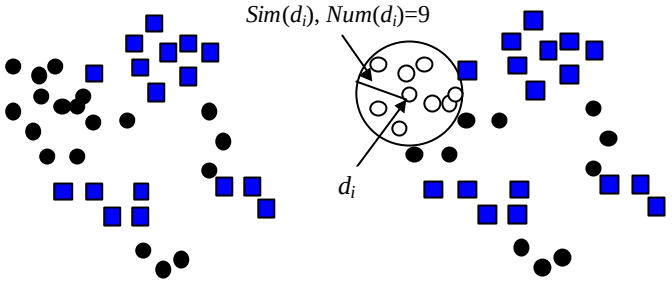

Fig. 3. The distribution of data tuples.

Fig. 4. The first obtained representative.

In Figure 4, the fine line circle covers 9 ($Num(d_i)$=9) data tuples with the same class label of d_i – circle (in this example, we use the first pruning method, viz. we assign 0 to r). The representative covers the maximal number of neighbours with the same class label at the first cycle. The $Sim(d_i)$ represents the Euclidean distance of d_i to the most distant data tuple from d_i in N_i.

After the first cycle, we obtain the first representative <$Cls(d_i)$, $Sim(d_i)$, $Num(d_i)$, $Rep(d_i)$>, add it into the model ***M***, and then turn to the next cycle. At the end of the second cycle we add another representative <$Cls(d_j)$, $Sim(d_j)$, $Num(d_j)$, $Rep(d_j)$> into the model ***M*** shown in Figure 5. Repeat this process until all the data tuples in the training dataset have been set to 'grouped' (represented by a empty circle or square). At the end, ten representatives shown in Figure 6 are obtained from the training dataset and stored in the model ***M***, where seven in ten representatives cover more than 2 data tuples denoted by a fine line circle and the other three representatives, each of them covers only one data tuple denoted by a bold line circle.

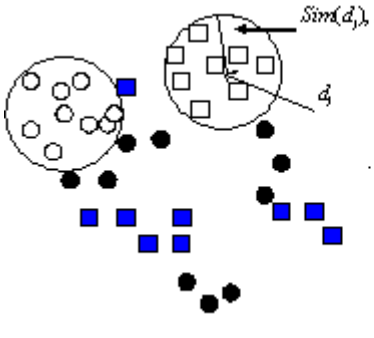

Fig. 5. The second obtained representative.

Fig. 6. The model before pruning.

In this situation, pruning work can be done by removing the representatives from the model ***M*** which only cover a few data tuples (for example, $Num(d_i)<2$). All the data tuples covered by these representatives will be removed as well from the training dataset. After that, we construct the model again from the revised training dataset. After pruning and model construction, we obtain the final model ***M***, looking at Figure 7 for a graphical illustration.

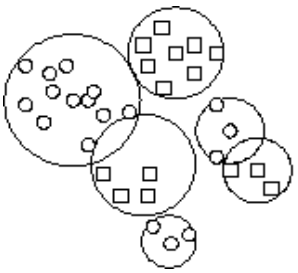

Fig. 7. The model after pruning.

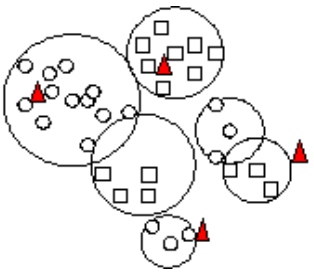

Fig. 8. The distribution of test data tuples.

In Figure 8, there are four triangles which represent the test data tuples. According to the classification algorithm described before, these four test data tuples are classified as a label of circle, square, circle, square from left to right respectively.

If we use the second pruning method and assign 1 to *r*, the model construction process is shown as follows:

Fig. 9. The first representative.

Fig. 10. The second representative.

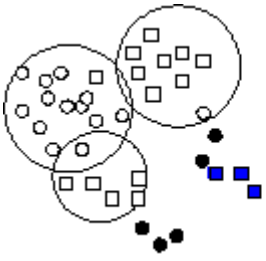

Fig. 11. The third representative.

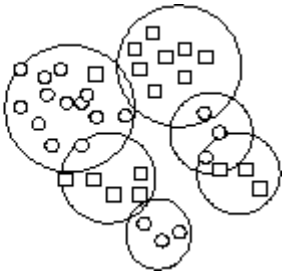

Fig. 12. The final model.

4 Experiment and Evaluation

Experiment using the 5-fold cross validation method has been carried out to evaluate the prediction accuracy of *k*NNModel, and to compare the experimental results with C5.0 and *k*NN as our benchmarks. The C5.0 is implemented in the Clementine' software package.

Six public datasets were chosen from the UCI machine learning repository. Some information about these datasets is listed in Table 1.

Table 1. Some information about the datasets.

Dataset	NA	NN	NO	NB	NE	CD
Glass	9	0	9	0	214	70:17:76:0:13:9:29
Iris	4	0	4	0	150	50:50:50
Heart	13	3	7	3	270	120:150
Wine	13	0	13	0	178	59:71:48
Diabetes	8	0	8	0	768	268:500
Aust	14	4	6	4	690	383:307

In Table 1, the meaning of the title in each column is follows: NA-Number of attributes, NN-Number of Nominal attributes, NO-Number of Ordinal attributes, NB-Number of Binary attributes, NE-Number of Examples, CD-Class Distribution.

The comparison of C5.0, *k*NN, and *k*NNModel in testing accuracy using the 5-fold cross validation method is listed in Table 2. The data reduction rate in the final model of the *k*NNModel is listed in Table 3. As we use Euclidean distance as a similarity measure in the experiment for *k*NN and *k*NNModel, six datasets were pre-processed including normalization and feature selection before conducting the classification. In experiments, we assign 1 to *r* and use information gain as our feature selection measure.

Table 2. A comparison of C5.0, *k*NN, and *k*NNModel.

Dataset	C5.0	Classification Accuracy (%)							
		*k*NNModel (*r* = 1)					*k*NN		
		N>1	N>2	N>3	N>4	N>5	K=1	K=3	K=5
Glass	66.3	70.95	70.95	70.00	68.57	67.62	67.14	70.48	68.10
Iris	92.0	96.00	96.00	96.00	96.00	96.00	95.33	94.67	95.33
Heart	75.6	80.37	80.37	80.74	80.37	81.11	75.93	80.74	82.96
Wine	92.1	96.00	96.00	96.00	96.00	96.00	96.57	95.43	94.86
Diabetes	76.6	73.59	73.59	73.86	74.25	75.42	68.24	73.59	74.38
Aust	85.5	85.65	85.65	85.07	84.93	84.93	82.17	86.23	86.38
Average	81.35	83.76	83.76	83.61	83.35	83.51	80.90	83.52	83.67

Table 3. The number of representatives and the average reduction rate in the final model.

Dataset	The number of representatives in the model					
	*K*NNModel (*r* = 1)					*k*NN
	N>1	N>2	N>3	N>4	N>5	
Glass	31	31	24	18	13	214
Iris	5	5	4	4	4	150
Heart	26	26	22	21	19	270
Wine	8	8	8	7	7	178
Diabetes	106	106	94	78	59	768
Aust	54	54	49	44	41	690
	The average reduction rate (%) of *k*NNModel					
	89.87	89.87	91.15	92.42	93.70	0

Note that in Table 2 and Table 3, *N*>*i* means each representative in the final model of the *k*NNModel at least covers *i*+1 data tuples of the training dataset. It is not an integrant parameter. It can be removed from the *k*NNModel algorithm by pruning process. The experimental results of different *N* listed here are to demonstrate the relationship between classification accuracy and reduction rate of the *k*NNModel algorithm.

We also carried out an experiment by assigning different value to *r* and N to find the best classification accuracy for knnModel, and to see the influence of the *r* and N play to knnModel's classification accuracy. In experiment, *r* and N varied from 1 to 10. The best classification accuracy of *k*NNModel for each dataset is obtained and listed in table 4. It shows us that the best accuracy for each dataset can be obtained via tuning *r* and N with rather small value. The best accuracy of knnModel outperforms C5.0 and KNN for all datasets with an exception of 1-NN on Wine.

Table 4. The influence of *r* and N to *k*NNModel.

Dataset	C5.0	Classification Accuracy (%)					
		*k*NNModel			*k*NN		
		Best Accuracy	*r*	N	K=1	K=3	K=5
Glass	66.3	70.95	1	1	67.14	70.48	68.10
Iris	92.0	96.00	1	1	95.33	94.67	95.33
Heart	75.6	83.70	3	3	75.93	80.74	82.96
Wine	92.1	96.00	1	1	96.57	95.43	94.86
Diabetes	76.6	77.12	5	3	68.24	73.59	74.38
Aust	85.5	87.10	2	1	82.17	86.23	86.38
Average	81.35	85.15			80.90	83.52	83.67

With N=1, the influence of different *r* (0~15) plays to the classification accuracy of knnModel is shown in Figure 13 and Figure 14 when do test on Aust and Diabetes datasets.

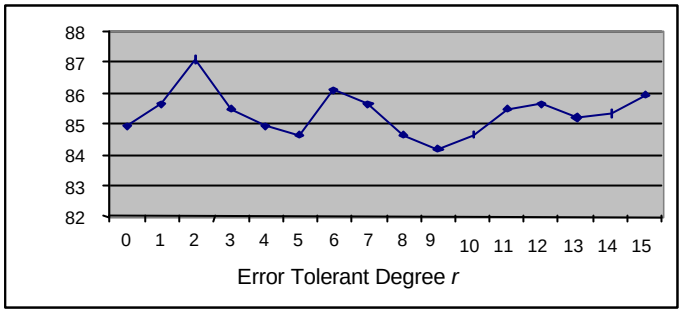

Fig.13. The accuracy of knnModel testing on Aust dataset with different *r*

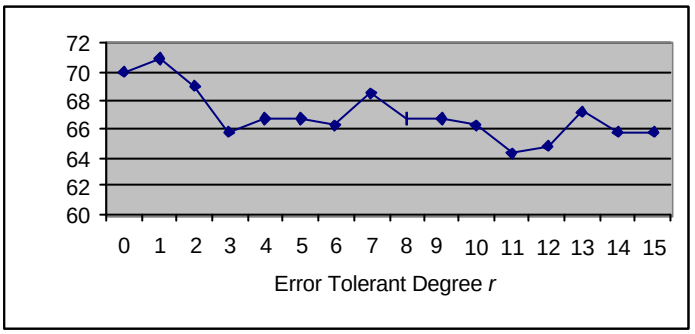

Fig.14. The accuracy of knnModel testing on Glass dataset with different *r*

The experimental results of the *k*NNModel without *N* on six datasets are listed in Table 5.

Table 5. A comparison of C5.0, *k*NN, and *k*NNModel with model pruning.

Dataset	C5.0	*k*NNModel (*r*=0)		*k*NN	
		CA%	RR%	CA%	RR%
Glass	66.3	68.57	82.71	68.57	0
Iris	92.0	95.33	95.33	95.11	0
Heart	75.6	80.74	89.26	79.88	0
Wine	92.1	95.43	94.94	95.62	0
Diabetes	76.6	74.77	86.32	72.07	0
Aust	85.5	86.09	93.91	84.93	0
Average	81.35	83.49	90.41	82.70	0

In table 5, CA means classification accuracy, RR means reduction rate. For *k*NN, the CA is the average classification accuracy of *k*=1, 3, 5.

From the experimental results, it is clear that the average classification accuracy of our proposed *k*NNModel method on six datasets is better than C5.0 in 5-fold cross validation and is comparable to *k*NN. But the *k*NNModel significantly improves the efficiency of *k*NN by keeping only a few representatives for classification. The experimental results show that the average reduction rate is 90.41%.

5 Conclusions

In this paper we have presented a novel solution for dealing with the shortcomings of *k*NN. To overcome the problems of low effeciency and dependency on *k*, we select a few representatives from training dataset with some extra information to represent the whole training dataset. In the selection of each representative we use the optimal but different *k* decided by dataset itself to eliminate the dependency on *k* without user's intervention. Experimental results carried out on six public datasets show that the *k*NNModel is a quite competitive method for classification. Its average classification accuracy on six public datasets is comparable with C5.0 and *k*NN. Also the *k*NNModel significantly reduces the number of the data tuples in the final model for clasification with a 90.41% reduction rate on average. It could be a good replacement for *k*NN in many applications such as dynamic web mining for a large repository. Further research is required into how to improve the classification accuracy of marginal data which fall outside the regions of representatives.

Acknowledgements

This work was partly supported by the European Commission project ICONS, project no. IST-2001-32429.

References

1. D. Hand, H. Mannila, P. Smyth.: Principles of Data Mining. The MIT Press. (2001)
2. H. Wang.: Nearest Neighbours without k: A Classification Formalism based on Probability, technical report, Faculty of Informatics, University of Ulster, N.Ireland, UK (2002)
3. F.Sebastiani.: Machine Learning in Automated Text Categorization. In ACM Computing Surveys, Vol. 34, No. 1, March (2002) pp.1-47.
4. H. Wang, I. Duntsch, D. Bell.: Data Reduction Based on Hyper Relations. In proceedings of KDD98, New York, pages 349-353 (1998)
5. P. Hart.: The Condensed Nearest Neighbour Rule, IEEE Transactions on Information Theory, 14, 515-516, (1968)

6. G. Gates.: The Reduced Nearest Neighbour Rule. IEEE Transactions on Information Theory, 18, 431-433, (1972)
7. E. Alpaydin.: Voting Over Multiple Condensed Nearest Neoghbors. Artificial Intelligence Review 11:115-132, (1997) ©1997 Kluwer Academic Publishers.
8. M. Kubat, M. Jr.: Voting Nearest-Neighbour Subclassifiers. Proceedings of the 17th International Conference on Machine Learning, ICML-2000, pp.503-510, Stanford, CA, June 29-July 2, (2000)
9. D. R. Wilson, T. R. Martinez.: Reduction Techniques for Exemplar-Based Learning Algorithms. Machine learning, 38-3, pp.257-286, (2000)
10. T. Mitchell.: Machine Learning. MITPress and McGraw-Hill (1997)
11. C.M.Bishop.: Neural Networks for Pattern Recognition. Oxford University Press, UK (1995)