Similarity-Based Data Reduction Techniques

Gongde Guo

School of Computing and Mathematics, University of Ulster, BT37 0QB, Northern Ireland, UK
Department of Computing, University of Bradford, Bradford, BD7 1DP, UK
Email: G.Guo@Bradford.ac.uk

Hui Wang

School of Computing and Mathematics, University of Ulster, BT37 0QB, Northern Ireland, UK
Email: H.Wang@ulster.ac.uk

David Bell

School of Computer Science, Queen's University Belfast, Belfast, BT7 1NN, UK
Email: DA.Bell @qub.ac.uk

The k-nearest neighbours (kNN) is a simple but effective method for classification. Its major drawbacks are (1) low efficiency, and (2) dependency on the selection of a "good value" for k. In this paper, we propose a novel similarity-based data reduction method (SBModel) together with three variants aimed at overcoming these shortcomings. Our method constructs a similarity-based model for the data, which replaces the data to serve as the basis of classification. The value of k is automatically determined, is varied in terms of local data distribution, and is optimal in terms of classification accuracy. The construction of the model significantly reduces the amount of data needed for classification, thus making classification faster. Experiments conducted on some public data sets show that SBModel and its variants compare well with C5.0, kNN, wkNN, and other data reduction methods in both efficiency and effectiveness.

ACM Classification: I.2.4 (Computing Methodologies – Artificial Intelligence – Knowledge Representation Formalisms And Methods); I.2.6 (Computing Methodologies – Artificial Intelligence – Learning); I.5.2 (Computing Methodologies – Pattern Recognition – Design Methodology)

1. INTRODUCTION

The *k*-nearest neighbours (*k*NN) is a non-parametric classification method which is simple but effective in many cases (Hand *et al*, 2001). For an instance d_t to be classified, its *k* nearest neighbours are retrieved, and this forms a *neighbourhood of* d_t. Majority voting among the instances in the neighbourhood is commonly used to decide the classification for d_t, with or without consideration of distance-based weighting. Despite its conceptual simplicity, the *k*NN method performs as well as any other possible classifier when applied to non-trivial problems. Over the last 50 years, this simple classification method has been extensively used in a broad range of applications such as medical diagnosis, text categorization (Sebastiani, 2002), pattern recognition, data mining, and e-commerce. 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, and a simple one is to run the algorithm many times with different *k* values and choose the one with the best performance, but this is not a pragmatic method in real applications.

Manuscript received: 22 March 2004
Communicating Editor: John Yearwood

In order for *k*NN to be less dependent on the choice of k, we look at multiple sets of nearest neighbours rather than just one set of k nearest neighbours in Wang (2003). 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 d_t. As it is aimed at improving classification accuracy and alleviating the dependency on k, the efficiency of the method in its basic form is expected to be worse than *k*NN. However, it is indeed less dependent on k and is able to achieve classification performance close to that for the best k.

From the point of view of implementation, the *k*NN method consists of a search of pre-labelled instances given a particular distance definition to find k nearest neighbours for each new instance. If the number of instances available is very large, the computational burden for *k*NN is unbearable. This drawback prohibits its use in many applications, such as dynamic web mining from a large document repository.

These drawbacks of *k*NN motivate us to find a way of instance reduction which chooses a few representatives to be stored for use for classification in order to improve efficiency whilst both preserving its classification accuracy and alleviating the dependency on k.

2. RELATED WORK

Many researchers in automatic classification have devoted extensive resources to solving either the computational problem and/or the performance problem during the past 50 years. Among those methods, *instance selection* aims at modifying a given set of instances in order to reduce its size as well as to improve classification performance. From the point of view of their goal, those instance selection methods can be divided into two different kinds of methods: the *editing* and *condensing* methods (Devijver *et al*, 1982). The editing methods aim at removing outliers and instances which are placed at the overlap among classes. These types of methods e.g. *ENN* (Wilson, 1972), *RENN* (Wilson, 1972), and *AllK-NN* (Tomek, 1976), do not generally entail substantial reductions in size, but they usually produce well-clustered groups of homogeneous instances that lead to optimal *k*NN classification results (Wilson *et al*, 2000). On the other hand, condensing methods, e.g. *CNN* (Hart, 1968), *SNN* (Ritter *et al*, 1975), *IB1~IB5* (Aha *et al*, 1991; 1992), *DROP1~DROP5* and *DEL* (Wilson *et al*, 2000), try to find a significantly reduced set of instances whose *k*NN classification results are as close as possible to those obtained using all original instances.

From the way in which representatives are obtained and represented in these methods, there is a separation between *instance selection methods*, in which the resulting representatives are taken from the original set, and *instance replacement methods* in which the resulting representatives are built and may be different from any instances in the original set. Representatives obtained by either of these methods can be referred to as *S*- and *R*-representatives (Liudmila *et al*, 1998) respectively.

2.1 Instance Selection Methods

Many researchers have addressed the problem of training set size reduction. Hart (1968) made one of the first attempts to reduce the size of the training set with his *Condensed Nearest Neighbour Rule* (*CNN*). His algorithm finds a subset S of the training set T such that every instance of T is closer to an instance of S of the same class than to an instance of S of a different class. In this way, the subset S can be used to classify all the instances in T correctly.

Ritter *et al* (1975) extended the condensed NN method in their *Selective Nearest Neighbour Rule* (*SNN*) such that every instance of T must be closer to an instance of S of the same class than to any instance of T (instead of S) of a different class. Further, the method ensures a minimal subset satisfying these conditions.

Gate (1972) introduced the *Reduced Nearest Neighbour Rule* (*RNN*). The *RNN* algorithm starts with $S=T$ and removes each instance from S if such a removal does not cause any other instances in T to be misclassified by the instances remaining in S.

Wilson (1972) developed the *Edited Nearest Neighbour* (*ENN*) algorithm in which S starts out the same as T, and then each instance in S is removed if it does not agree with the majority of its k nearest neighbours. The *Repeated ENN* (*RENN*) applies the *ENN* algorithm repeatedly until all instances remaining have a majority of their neighbours with the same class, which continues to widen the gap between classes and smooth the decision boundary.

Tomek (1976) extends the *ENN* with his *Allk-NN* method of editing. This algorithm works as follows: for $i=1$ to k, flag as "bad" any instance not classified correctly by its i nearest neighbours. After completing the loop all k times, remove any instances from S flagged as "bad".

Cameron-Jones (1995) proposed the use of an encoding length heuristic with his *ELGrow* to determine how good the subset S is in describing T. ELGrow is based on a cost function used to quantify an instance-based model. It begins with a growing phase that takes each instance i in T and adds it to S if that results in a lower cost than not adding it. Cameron-Jones also extended ELGrow with his *Explore* method. It starts by growing and reducing S using the ELGrow method, and then performs a predefined number of mutations to try to improve the classifier. Each mutation tries adding an instance to S, removing one from S, or swapping one in S with one in T-S. It keeps the change if it does not increase the cost of the classifier.

Other S-representatives methods include IB1~IB5 by Aha *et al* (1991; 1992), DROP1~DROP5, and DEL by Wilson *et al* (2000).

2.2 Instance Replacement Methods

Following the basic idea of replacing a group of instances by a representative, some researchers have proposed iterative solutions to build consistent sets of representatives via merging. Chang's condensing method (Chang, 1974) was based on this strategy. It begins with a training set T, considering all the instances in T as initial representatives, and then successively merges any two closest representatives (p, q) of the same class (by replacing p and q with a new representative p^*) if the merging does not degrade the classification of instances in T. The new representative p^* is computed as a weighted average between p and q. This process is stopped when no new merge is possible in any class.

The *modified Chang Algorithm* (*MCA*) presented by Bezdek *et al* (1998) constitutes a slight improvement of the previous algorithm. Compared to the original algorithm of Chang (1974), the MCA changes the algorithm in the way in which pairs of prototypes are considered, introducing a slightly different way of merging prototypes. A computational improvement is also reached based on storing cross distances among prototypes in the same class only. The simple arithmetic mean between p and q is used to compute p^* without weights. Nevertheless, the strategy behind the original idea remains unchanged.

The RISE 2.0 system presented by Domingos (1995) treats each instance in T as a rule in R. For each rule r in R, the nearest instance t in T of the same class as r is found that is not yet covered by r. The rule r is then minimally generalized to cover t, unless that reduces accuracy. This process is repeated until no rules are generalized during an entire pass through all the rules in R. During the generalization, the nearest rule to a new instance is used to provide the output class. If two rules are equally close, the one with higher generalization accuracy on the training set is used.

Wettschereck *et al* (1995) introduced a hybrid nearest-neighbour and nearest-hyperrectangle algorithm that uses hyperrectangles to classify new instances if they fall inside the hyperrectangle, and

*k*NN to classify instances which fall outside any hyperrectangle. This algorithm must store the entire training set *T*, but accelerates classification by using relatively few hyperrectangles whenever possible.

A generalized condensing scheme based on class-conditional hierarchical clustering (GMCA) is proposed by Mollineda *et al* (2002). The basic idea is to replace a group of instances by a representative while keeping the consistency property. The algorithm improves and generalizes previous works by explicitly introducing the concept of cluster consistency. The use of geometric cluster properties produces a merging scheme based on local consistency verification, guaranteeing the whole system's consistency while minimizing the computation needed.

In the next section, we introduce a hybrid *S*-representatives and *R*-representatives method called SBModel (Similarity-Based Model). The method constructs a similarity-based model by selecting a subset *S* (*S*-representatives) with some extra information (*R*-representatives) from the training set *T*, to replace the data and serve as the basis of classification. The model consists of a set of representatives of the training data, as regions in the data space. Based on SBModel, three variants of SBModel: ε-SBModel, *p*-SBModel, SBModel-*i* are also presented, which aim at further improving both the efficiency and effectiveness of SBModel. The experimental results and a comparison with other data reduction methods in the literature are presented in Section 4.

3. SIMILARITY-BASED DATA REDUCTION

3.1 The SBModel Concept

Looking at Figure 1, the Iris data with two features, petal length and petal width, is used for demonstration. It contains 150 instances with three classes represented as square, triangle, and diamond respectively, and is plotted in 2-dimensional data space. In Figure 1, the horizontal axis is for petal length and the vertical axis is for petal width.

If we use distance function as our similarity measure, many instances with the same class label are close to each other in many local areas. Consider Figure 2, for example. In each local region, the central instance d_i, with some extra information such as $Cls(d_i)$ – the class label of instance d_i; $Num(d_i)$ – the number of instances inside the local region; $Sim(d_i)$ – the distance of the furthest instance inside the local region to d_i, and $Rep(d_i)$ – a representation of d_i, might be a good representative of this local region. If we take these representatives as a model to represent the whole training set, it will significantly reduce the number of instances for classification, thereby improving efficiency.

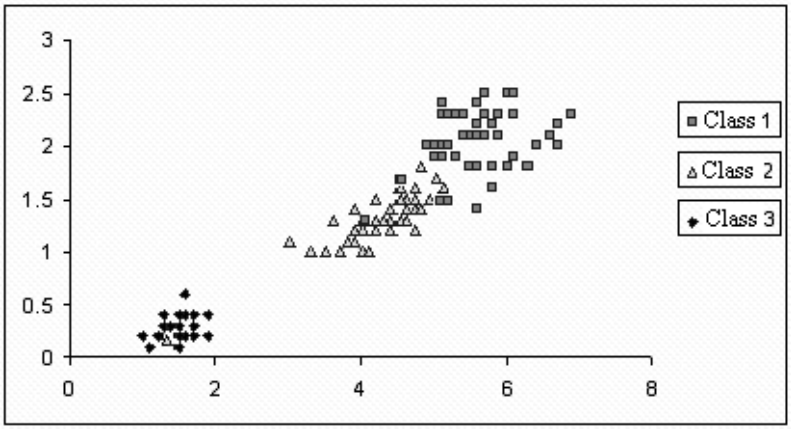

Figure 1: Data distribution in 2-dimensional data space

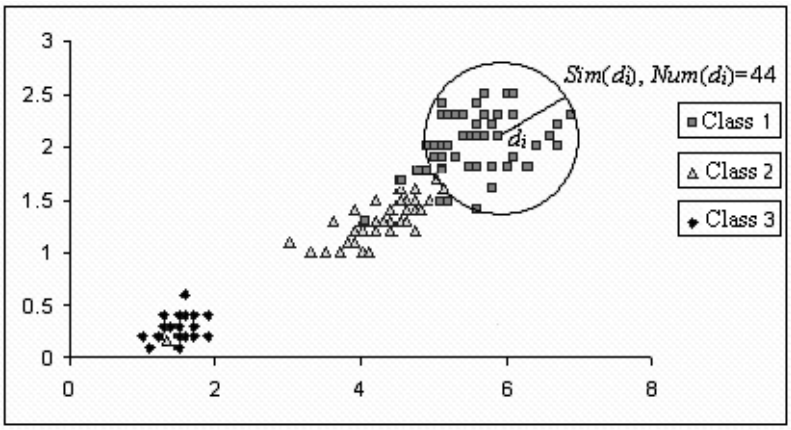

Figure 2: The first obtained representative <$Cls(d_i)$ $Sim(d_i)$, $Num(d_i)$, $Rep(d_i)$>

In addition, the extra information kept in each representative is used to represent those instances that are covered by this representative and have been removed. The goal of keeping such information is an attempt to reduce the information loss during the process of instances removal. The extra information kept in each representative is obtained by inductively learning from the training set. Thereby it is expected to preserve or possibly improve the effectiveness of *k*NN.

Consider a new instance to be classified at the classification stage. If it 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 instance to each representative's nearest boundary and take each representative's nearest boundary as an instance, then classify the new instance in the spirit of *k*NN.

3.2 Terminology and Definitions

Before we give more details about the designs of the proposed algorithms, some important terms (or concepts) need to be explicitly defined first.

Definition 1. Neighbourhood

A *neighbourhood* is a contextual term with respect to a given instance in data space. A neighbourhood of a given instance is defined to be a set of nearest neighbours around this instance.

Definition 2. Local Neighbourhood

A *local neighbourhood* is a neighbourhood which covers the maximal number of instances with the same class label.

Definition 3. Local ε-Neighbourhood

A local *ε-neighbourhood* is a neighbourhood which covers the maximal number of instances with the same class label except for allowing ε exceptions.

Definition 4. Local *i*-Consistent Neighbourhood

A *local i-consistent neighbourhood* is a neighbourhood which covers the maximal number of instances, each of which is correctly classified by the same class label using *k*NN (where $k=i$).

Definition 5. Global Neighbourhood

A *global neighbourhood* is a local neighbourhood which covers the largest number of instances among a set of obtained local neighbourhoods.

Definition 6. Global ε-Neighbourhood

A *global ε-neighbourhood* is a local ε-neighbourhood which covers the largest number of instances among a set of obtained local ε-neighbourhoods.

Definition 7. Global *i*-Consistent Neighbourhood

A *global i-consistent neighbourhood* is a local *i*-consistent neighbourhood which covers the largest number of instances among a set of obtained local *i*-consistent neighbourhoods.

With these definitions, given a training set, each instance has a local neighbourhood. Based on these local neighbourhoods, the global neighbourhood can be obtained. This global neighbourhood can be seen as a representative of all the instances covered by it. For instances not covered by any representative we repeat the above operation until all the instances have been covered by chosen representatives. At the end, all representatives obtained in the model construction process are used for replacing the data and serving as the basis for classification. There are three obvious advantages: (1) we needn't choose a specific k in the sense of kNN for our method in the model construction process; the number of instances covered by a representative can be seen as an optimal k, which is generated automatically in the model construction process, but it is different for different representatives; (2) using a set of chosen representatives as a model reduces the number of instances for classification, thus improving the efficiency of classification; (3) keeping some important information for the removed instances in each representative can minimize the information loss during the process of instances removal, thus preserving or possibly improving the classification accuracy. From this point of view, the proposed method overcomes the two shortcomings of kNN whilst preserving its effectiveness.

3.3 Modelling and Classification Algorithm

Let D be a collection of n class-known instances $\{d_1, d_2, \ldots, d_n\}$. Given a similarity measure, the detailed model construction algorithm of SBModel is described as follows:

1. For a given similarity measure create a similarity matrix from a given training set D
2. Set to "ungrouped" the tag of all instances for training and set model $M=\emptyset$
3. For each "ungrouped" instance, find its local neighbourhood
4. Among all the local neighbourhoods obtained in step 3, find its global neighbourhood N_i. Create a representative $(Cls(d_i), Sim(d_i), Num(d_i), Rep(d_i))$ into M to represent all the instances covered by N_i, and then set to "grouped" the tag of all the instances covered by N_i
5. Repeat step 3 and step 4 until all the instances in the training set have been set to "grouped"
6. Model M consists of all the representatives collected from the above learning process.

In this algorithm, symbol M represents a set used to store the created model (a set of representatives). The elements of representative $<Cls(d_i), Sim(d_i), Num(d_i), Rep(d_i)>$ respectively represent the class label of d_i; the similarity of d_i to the furthest instance among the instances covered by N_i; the number of instances covered by N_i, and a representation of instance d_i. In step 4, if there are more than one local neighbourhood having the same maximal number of neighbours, we choose the one with minimal value of $Sim(d_i)$, i.e. the one with highest density, as a representative.

The detailed classification algorithm of SBModel is described as follows:

1. For a new instance d_t to be classified, calculate its similarity to all representatives in the model M
2. If d_t is covered by only one representative $<Cls(d_i), Sim(d_i), Num(d_i), Rep(d_i)>$, d_t is classified as $Cls(d_i)$

3. If d_t is covered by more than one representative with different class labels, classify d_t as the class label of the representative covering the largest number of instances
4. If no representative in the model M covers d_t, classify d_t as the class label of the representative whose boundary is closest to d_t.

In the above algorithm, an instance d_t is said to be *covered* by a representative $<Cls(d_i), Sim(d_i), Num(d_i), Rep(d_i)>$ if the similarity of d_t to d_i is equal to or smaller than $Sim(d_i)$. The similarity of d_t to a representative d_i's nearest boundary is equal to the difference of the similarity of d_i to d_t, minus $Sim(d_i)$.

In order to improve the classification accuracy of SBModel, we implemented three different pruning methods for SBModel.

The first pruning method removes both the representatives from the model M created by SBModel that only cover a few instances and the instances covered by these representatives from the training set. Then set $M=\emptyset$ and construct the M again using SBModel from the revised training set. The SBModel algorithm based on this pruning method is called *p*-SBModel.

The model construction process of *p*-SBModel is described as follows:

1. For each representative in the model M created by SBModel that only covers a few (a predefined threshold) instances, remove all the instances covered by this representative from the training set D, then go to step 2
2. Set model $M=\emptyset$ and construct model M from the revised training set D again by using SBModel
3. The final model M consists of all the representatives collected from the above pruning process.

The second pruning method modifies step 3 in the model construction algorithm of SBModel to allow each local neighbourhood to cover ε (called error tolerance rate) instances with different class labels to the majority class label of this neighbourhood. This modification integrates the pruning work into the model construction process of SBModel. The SBModel algorithm based on this pruning method is called ε-SBModel. The detailed modelling process of ε-SBModel is described as follows:

1. For a given similarity measure, create a similarity matrix from a given training set D
2. Set to "ungrouped" the tag of all instances for training and set model $M=\emptyset$
3. For each "ungrouped" instance, find its local ε-neighbourhood
4. Among all the local ε-neighbourhoods obtained in step 3, find its global ε-neighbourhood N_i. Create a representative $<Cls(d_i), Sim(d_i), Num(d_i), Rep(d_i)>$ into M to represent all the instances covered by N_i, and then set to "grouped" the tag of all the instances covered by N_i
5. Repeat step 3 and step 4 until all the instances in the training set have been set to "grouped"
6. Model M consists of all the representatives collected from the above learning process.

The last pruning method uses local *i*-consistent neighbourhood ($i>0$) to construct the model, i.e. in the modelling process, all the instances in a representative must be classified correctly by *i*NN. The SBModel algorithm based on this pruning method is called SBModel-*i*. The detailed modelling process of SBModel-*i* is described as follows:

1. For a given similarity measure, create a similarity matrix from a given training set D
2. Set to "ungrouped" the tag of all instances for training and set model $M=\emptyset$
3. For each "ungrouped" instance, find its local *i*-consistent neighbourhood
4. Among all the local *i*-consistent neighbourhoods obtained in step 3, find its global *i*-consistent neighbourhood N_i. Create a representative $<Cls(d_i), Sim(d_i), Num(d_i), Rep(d_i)>$ into M to

represent all the instances covered by N_i, and then set to "grouped" the tag of all the instances covered by N_i

5. Repeat step 3 and step 4 until all the instances in the training set have been set to "grouped"
6. Model *M* consists of all the representatives collected from the above learning process.

The SBModel algorithm is a basic algorithm with ε = 0 (error tolerance rate), *i*=0 (*i*-consistent neighbourhood), and without pruning (*p*=0). The experimental results of SBModel, ε-SBModel, *p*-SBModel, and SBModel-*i* conducted on some public data sets are reported in the Section 4.

3.4 An Example of ε-SBModel

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

A training data set including 150 instances is divided into three classes denoted as diamond, square, and triangle respectively. The distribution of instances in 2-dimensional data space is plotted in Figure 1.

The error tolerance rate ε is set to 1, i.e. in the model construction process, each representative is allowed to cover one instance with different class label from the majority one. The first three representatives obtained from the model construction process are shown in Figure 3, and the final model obtained from the data by employing ε-SBModel is shown in Figure 4.

In Figure 3, after the first cycle, we obtain the first representative <$Cls(d_k)$, $Sim(d_k)$, $Num(d_k)$, $Rep(d_k)$>, add it into the model *M*, and then turn to the next cycle. At the end of the second cycle we add another obtained representative <$Cls(d_j)$, $Sim(d_j)$, $Num(d_j)$, $Rep(d_j)$> into the model *M*. Repeat this process until all the instances in the training set have been set to "grouped" (covered by any circle). At the end, eight representatives shown in Figure 4 are obtained from the training set. Five of eight representatives cover at least two instances each; these are stored in the model *M* of ε-SBModel. The other three representatives which cover only one instance each can be discarded as noise.

In Figure 5, there are five *crosses* "+" which represent the new instances to be classified. According to the classification algorithm described in Section 3.3, these five instances are respectively classified as diamond, triangle, triangle, square, and square from left to right.

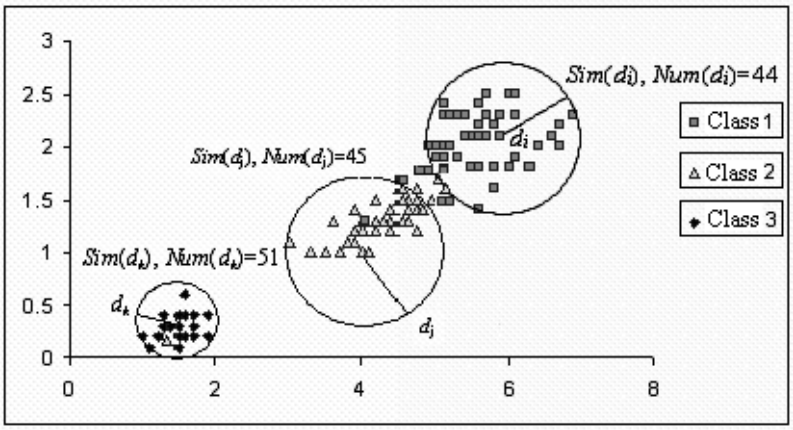

Figure 3: The first three obtained representatives in the model construction process of ε-SBModel

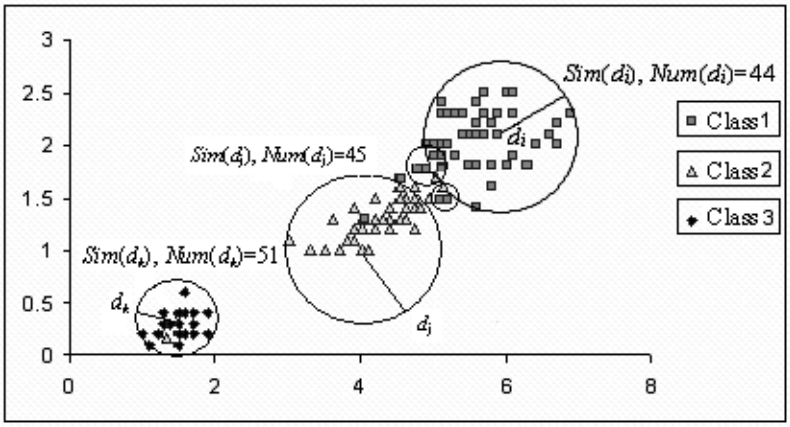

Figure 4: The final model created by ε-SBModel

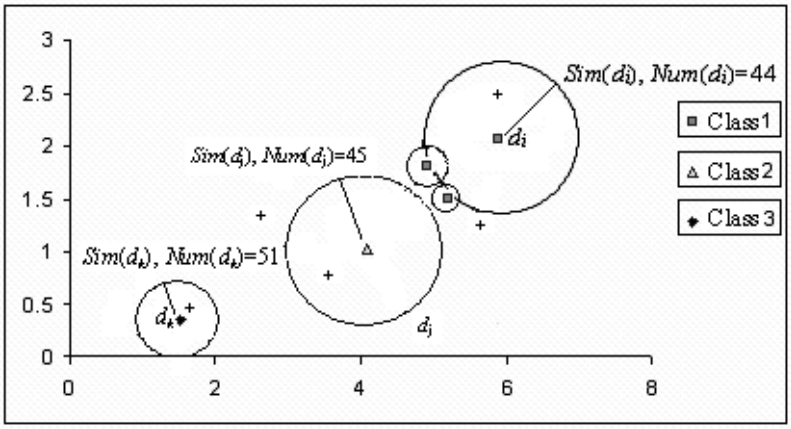

Figure 5: The distribution of five new instances denoted as "+"

4. EXPERIMENT AND EVALUATION

4.1 Data Sets

To evalutate the SBModel method and its variants, fifteen public data sets have been collected from the UCI machine learning repository for training and testing. Some information about these data sets is given in Table 1.

In Table 1, the meaning of the title in each column is follows: NF-Number of Features, NN-Number of Nominal features, NO-Number of Ordinal features, NB-Number of Binary features, NI-Number of Instances, CD-Class Distribution.

4.2 Experimental Environment

Experiments use the 10-fold cross validation method to evaluate the performance of SBModel and its variants and to compare them with C5.0, *k*NN (voting *k*NN), and *wk*NN (distance weighted *k*NN). We implemented SBModel and its variants, *k*NN and *wk*NN in our prototype. The C5.0 algorithm used in our experiment is implemented in the Clementine software package. The

Data set	NF	NN	NO	NB	NI	CD
Australia	14	4	6	4	690	383:307
Colic	23	16	7	0	368	232:136
Diabetes	8	0	8	0	768	268:500
Glass	9	0	9	0	214	70:17:76:0:13:9:29
HCleveland	13	3	7	3	303	164:139
Heart	13	3	7	3	270	120:150
Hepatitis	19	6	1	12	155	32:123
Ionosphere	34	0	34	0	351	126:225
Iris	4	0	4	0	150	50:50:50
LiverBupa	6	0	6	0	345	145:200
Sonar	60	0	60	0	208	97:111
Vehicle	18	0	18	0	846	212:217:218:199
Vote	16	0	0	16	435	267:168
Wine	13	0	13	0	178	59:71:48
Zoo	16	16	0	0	90	37:18:3:12:4:7:9

Table 1: Some information about the data sets

experimental results for other editing and condensing algorithms to be compared here are obtained from Wilson's reported experiments (Wilson *et al*, 2000).

In voting *k*NN, the *k* neighbours are implicitly assumed to have equal weight in decisions, regardless of their distances to an instance *d* to be classified. It is intuitively appealing to give different weights to the *k* neighbours based on their distances to *d*, with closer neighbours having greater weights.

In *wk*NN, the *k* neighbours are assigned different weights. Let Δ be a distance function, and d_1, d_2, …, d_k be the *k* nearest neighbours of *d* arranged in increasing order of $\Delta(d_i, d)$. So d_1 is the first nearest neighbour of *d*. The distance weight w_i for *i*-th neighbour d_i is defined as follows:

$$w_i = \begin{cases} \dfrac{\Delta(d_k,d) - \Delta(d_i,d)}{\Delta(d_k,d) - \Delta(d_1,d)} & if \quad \Delta(d_k,d) \neq \Delta(d_1,d) \\ 1 & if \quad \Delta(d_k,d) = \Delta(d_1,d) \end{cases} \tag{1}$$

Instance *d* is assigned to the class for which the weights of the representatives among the *k* nearest neighbours sum to the greatest value.

In order to handle heterogeneous applications – those with both ordinal and nominal features – we use a heterogeneous distance function *HVDM* (Wilson *et al*, 1997) as the distance function in the experiments, which is defined as:

$$HVDM(\vec{x}, \vec{y}) = \sqrt{\sum_{k=1}^{m} \Delta^2(x_k, y_k)} \tag{2}$$

In formula (2), instance $\vec{x}$ is represented as a vector of values $(x_1, x_2, \ldots, x_m)$, where x_i is a value occurring in the i^{th} feature and *m* is the number of features; $\Delta(x_k, y_k)$ is the distance function of two values x_k, y_k occurring in the k^{th} feature and is defined (Wilson *et al*, 1997) as:

$$\Delta(x_k, y_k) = \begin{cases} 1 & \text{If } x_k \text{ or } y_k \text{ is unknown, otherwise …} \\ VDM(x_k, y_k) & \text{If the } k^{\text{th}} \text{ feature is nominal, else} \\ |x_k - y_k| / 4\sigma_k & \text{If the } k^{\text{th}} \text{ feature is ordinal} \end{cases} \quad (3)$$

In the above distance function, σ_k is the standard deviation of the values occurring in the k^{th} feature of the instances in the training set D, and $VDM(x_k, y_k)$ is the distance function for nominal feature called *Value Difference Metric* (Stanfill *et al*, 1986). Using *VDM*, the distance between two value x_k and y_k occurring in the k^{th} feature is given as:

$$VDM(x_k, y_k) = \sum_{i=1}^{\#C} \left(\frac{N_{x_k, c_i}}{N_{x_k}} - \frac{N_{y_k, c_i}}{N_{y_k}} \right)^2 \quad (4)$$

where N_{x_k} is the number of times the k^{th} feature has value x_k; N_{x_k, c_i} is the number of times the k^{th} feature has value x_k and the output class is c_i; C is the set of output classes, and #C is the number of output classes.

4.3 Experimental Results

[Experiment 1] In this experiment, our goal is to evaluate SBModel, and to compare its experimental results with C5.0, *k*NN, and *wk*NN. A summary of parameter settings is given in Table 2. Under these settings, the comparison of C5.0, SBModel, *k*NN, and *wk*NN in classification accuracy using 10-fold cross validation is given in Table 3. The number of representatives in the final model of SBModel on each data set is given in Table 4, and the reduction rate of SBModel on each data set is given in Table 5.

Note that in Table 3, Table 4, and Table 5, $N = i$ means each representative in the final model of SBModel covers at least i instances of that training set. The parameter N can be removed from SBModel by a pruning process (more details will be presented in experiment 4 below). The experimental results of different N listed here are to demonstrate the relationship between the classification accuracy and reduction rate of SBModel on each data set. The reduction rate used in Table 5 is defined as follows:

$$Reduction\ Rate = \frac{NumOfIs - NumOfRs}{NumOfIs} \times 100 \quad (5)$$

where *NumOfIs* is the number of instances in the training set and *NumOfRs* is the number of representatives in the final model.

Parameter	Explanation	Setting
i	Local i-consistent neighbourhood	0
ε	Error tolerance rate	0
N	The allowed minimal number of instances covered by a representative in the final model	2 ~ 5
k	The value of k for *k*NN	1, 3, 5
k	The value of k for *wk*NN	5

Table 2: A summary of parameter settings

From the experimental results, it is clear that the average classification accuracy of SBModel on fifteen training sets is better than C5.0 in 10-fold cross validation, and is comparable to *k*NN and *wk*NN. But SBModel significantly improves the efficiency of *k*NN by keeping only 11.37 percent

		SBModel				*k*NN			*wk*NN
Data set	**C5.0**	**N=2**	**N=3**	**N=4**	**N=5**	**k=1**	**k=3**	**k=5**	**k=5**
Australian	85.5	84.20	84.64	84.78	84.63	79.42	82.75	85.22	82.46
Colic	80.9	81.67	82.50	83.06	82.50	78.89	83.89	83.06	81.94
Diabetes	76.6	75.00	74.08	74.21	74.74	70.92	73.03	74.21	72.37
Glass	66.3	65.24	65.24	61.43	55.71	68.10	66.67	67.62	67.62
HCleveland	74.9	82.67	80.33	80.33	78.00	78.33	82.33	81.00	81.33
Heart	75.6	80.37	80.74	80.37	77.78	76.30	80.37	80.37	77.41
Hepatitis	80.7	83.33	85.33	87.33	87.33	80.67	80.67	83.33	83.33
Ionosphere	84.5	94.29	93.71	92.57	91.43	87.14	85.14	84.00	87.14
Iris	92.0	96.00	96.00	96.00	96.00	95.33	94.67	96.67	95.33
LiverBupa	65.8	63.53	64.41	63.82	61.76	60.00	66.47	66.47	66.47
Sonar	69.4	84.00	82.50	80.00	79.50	88.00	83.50	85.00	86.50
Vehicle	67.9	65.83	65.36	63.69	62.26	68.57	71.79	69.29	71.43
Vote	96.1	88.70	88.70	88.70	88.70	91.30	92.17	92.17	90.87
Wine	92.1	95.29	94.71	94.12	94.12	95.88	94.71	94.71	95.29
Zoo	91.1	92.22	92.22	88.89	88.89	96.67	95.56	95.56	95.56
Average	79.96	82.16	82.03	81.29	80.22	81.03	82.25	82.58	82.34

Table 3: A comparison of C5.0, *k*NN, SBModel, and *wk*NN

	SBModel				*k*NN/*wk*NN
Data set	**N=2**	**N=3**	**N=4**	**N=5**	
Australian	91	66	54	52	690
Colic	80	58	46	41	368
Diabetes	150	100	80	64	768
Glass	44	25	21	13	214
HCleveland	48	37	26	22	303
Heart	41	31	27	23	270
Hepatitis	22	18	15	13	155
Ionosphere	65	52	40	36	351
Iris	7	6	6	6	150
LiverBupa	91	57	39	25	345
Sonar	38	29	26	19	208
Vehicle	166	103	68	55	846
Vote	20	15	14	13	232
Wine	17	17	14	13	178
Zoo	8	7	7	6	90

Table 4: The number of representatives in the final model of SBModel

Data set	SBModel N=2	N=3	N=4	N=5	*k*NN/*wk*NN
Australian	86.81	90.43	92.17	92.46	0
Colic	78.26	84.24	87.50	88.86	0
Diabetes	80.47	86.98	89.58	91.67	0
Glass	79.44	88.32	90.19	93.93	0
HCleveland	84.16	87.79	91.42	92.74	0
Heart	84.81	88.52	90.00	91.48	0
Hepatitis	85.81	88.39	90.32	91.61	0
Ionosphere	81.48	85.19	88.60	89.74	0
Iris	95.33	96.00	96.00	96.00	0
LiverBupa	73.62	83.48	88.70	92.75	0
Sonar	81.73	86.06	87.50	90.87	0
Vehicle	80.38	87.83	91.96	93.50	0
Vote	91.38	93.53	93.97	94.40	0
Wine	90.45	90.45	92.13	92.70	0
Zoo	91.11	92.22	92.22	93.33	0
Average	84.35	88.63	90.81	92.40	0

Table 5: The reduction rate of SBModel on each data set

instances of the original data sets on average for classification without much degradation in classification accuracy (82.03%, N=3) in comparison with kNN (82.58%, k=5 – the highest average classification accuracy among 1-NN, 3-NN, 5-NN) and wkNN (82.34%, k=5).

The relationship between N and the average classification accuracy of SBModel on fifteen data sets is shown in Figure 6. The relationship between N and the average reduction rate of SBModel on fifteen data sets is shown in Figure 7. In both Figure 6 and Figure 7 N ranges from 1 to 20.

With increasing N, the average reduction rate of SBModel increases, but the average classification accuracy of SBModel decreases. A tradeoff should be made between efficiency and effectiveness.

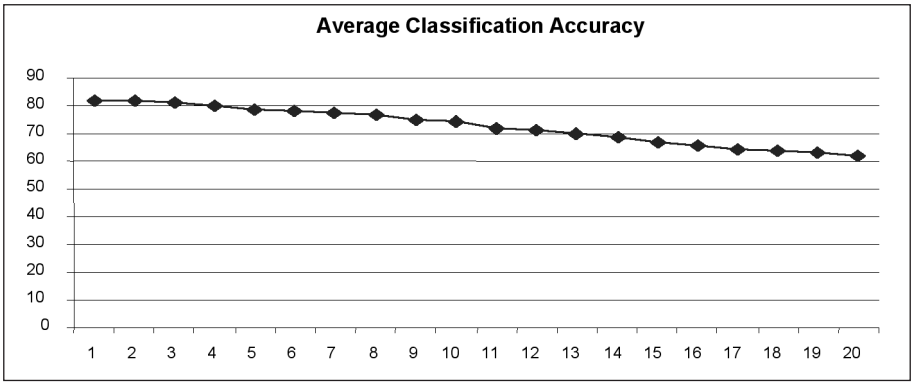

Figure 6: SBModel: relationship between N and the average classification accuracy

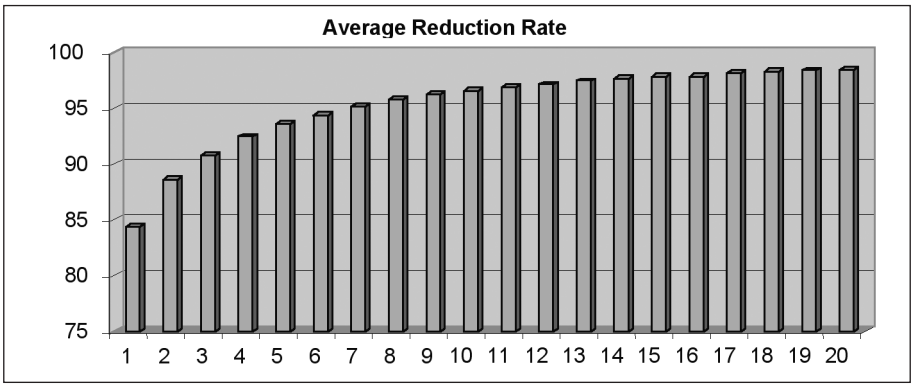

Figure 7: SBModel: relationship between *N* and the average reduction rate

[Experiment 2] In an attempt to improve the classification accuracy of SBModel, the second pruning method is used in this experiment. We tune the error tolerance rate ε in a small range (0~4) (the SBModel with ε is denoted as ε-SBModel) for each training set, and choose the ε for obtaining relatively high classification accuracy in the experiment. A summary of parameter settings for this experiment is given in Table 6, and the experimental results are presented in Table 7, Table 8, and Table 9 respectively.

From the experimental results presented in Table 7, ε-SBModel obtains better performance than C5.0, *k*NN, and *wk*NN. Even when *N*=5, ε-SBModel still obtains 82.93% average classification accuracy which is higher than the 79.96% of C5.0, 82.58% of *k*NN, and 82.34% of *wk*NN. In this situation, ε-SBModel keeps only 7.67 percent instances of the fifteen training sets on average for classification, thus significantly improving the efficiency whilst preserving the classification accuracy of *k*NN.

In this experiment, we also assign different values to ε and *N* to find the best classification accuracy on each data set for ε-SBModel, and to see the influence of ε and *N* on the classification accuracy of ε-SBModel. When ε varies from 0 to 15 and *N* varies from 2 to 15, the best classification accuracy of ε-SBModel on each data set is obtained and presented in Table 10. It shows us that the best classification accuracy of ε-SBModel on each data set can be obtained by tuning the values of ε and *N* in a small range. The best classification accuracy of ε-SBModel outperforms C5.0, *k*NN, and *wk*NN. Moreover, ε-SBModel obtains 89.78% reduction rate on average.

In Table 10, the abbreviation BCA means "Best Classification Accuracy", and RR means "Reduction Rate". With *N*=6, the influence of different ε (1~15) on the classification accuracy of ε-SBModel is shown in Figure 8 and Figure 9 respectively for Australian and Diabetes data sets.

Parameter	Explanation	Setting
i	Local *i*-consistent neighbourhood	0
ε	Error tolerance rate	0 ~ 4
N	The allowed minimal number of instances covered by a representative in the final model	2 ~ 5
k	The value of *k* for *k*NN and *wk*NN	5

Table 6: A summary of parameter settings

Data set	C5.0	SBModel ε	SBModel N=2	SBModel N=3	SBModel N=4	SBModel N=5	*k*NN k=5	*wk*NN k=5
Australian	85.5	2	84.93	84.93	85.22	85.51	85.22	82.46
Colic	80.9	1	83.06	83.06	82.78	83.61	83.06	81.94
Diabetes	76.6	1	74.34	74.47	75.13	75.53	74.21	72.37
Glass	66.3	3	69.52	69.52	69.52	69.05	67.62	67.62
HCleveland	74.9	4	81.67	81.67	81.67	81.67	81.00	81.33
Heart	75.6	1	80.74	81.11	81.85	81.11	80.37	77.41
Hepatitis	80.7	1	88.00	89.33	88.67	88.67	83.33	83.33
Ionosphere	84.5	1	93.71	93.71	92.86	92.57	84.00	87.14
Iris	92.0	0	96.00	96.00	96.00	96.00	96.67	95.33
LiverBupa	65.8	2	68.53	68.53	68.24	67.94	66.47	66.47
Sonar	69.4	2	82.50	82.50	82.50	81.50	85.00	86.50
Vehicle	67.9	2	66.43	66.43	66.55	66.07	69.29	71.43
Vote	96.1	4	91.74	91.74	91.74	91.74	92.17	90.87
Wine	92.1	0	95.29	94.71	94.12	94.12	94.71	95.29
Zoo	91.1	0	92.22	92.22	88.89	88.89	95.56	95.56
Average	79.96		83.25	83.33	83.05	82.93	82.58	82.34

Table 7: A comparison of C5.0, ε-SBModel, *k*NN, and *wk*NN in classification accuracy

Data set	ε-SBModel N=2	ε-SBModel N=3	ε-SBModel N=4	ε-SBModel N=5
Australian	66	66	54	52
Colic	80	58	46	41
Diabetes	150	100	80	64
Glass	21	21	21	13
HCleveland	24	24	24	24
Heart	41	31	27	23
Hepatitis	22	18	15	13
Ionosphere	65	52	40	36
Iris	7	6	6	6
LiverBupa	57	57	39	25
Sonar	28	28	23	20
Vehicle	103	103	68	55
Vote	13	13	13	13
Wine	17	17	14	13
Zoo	8	7	7	6

Table 8: The number of representatives in the final model of ε-SBModel

Data set	ε-SBModel N=2	N=3	N=4	N=5
Australian	90.43	90.43	92.17	92.46
Colic	78.26	84.24	87.50	88.86
Diabetes	80.47	86.98	89.58	91.67
Glass	90.19	90.19	90.19	93.93
HCleveland	92.08	92.08	92.08	92.08
Heart	84.81	88.52	90.00	91.48
Hepatitis	85.81	88.39	90.32	91.61
Ionosphere	81.48	85.19	88.60	89.74
Iris	95.33	96.00	96.00	96.00
LiverBupa	83.48	83.48	88.70	92.75
Sonar	86.54	86.54	88.94	90.38
Vehicle	87.83	87.83	91.96	93.50
Vote	94.40	94.40	94.40	94.40
Wine	90.45	90.45	92.13	92.70
Zoo	91.11	92.22	92.22	93.33
Average	87.51	89.13	90.99	92.33

Table 9: The reduction rate of ε-SBModel on each data set

Data set	C5.0	ε-SB Model ε	N	BCA	RR	*k*NN k=5	*wk*NN k=5
Australian	85.5	2	6	86.09	93.62	85.22	82.46
Colic	80.9	1	5	83.61	88.86	83.06	81.94
Diabetes	76.6	1	6	75.78	93.49	74.21	72.37
Glass	66.3	3	4	69.52	90.19	67.62	67.62
HCleveland	74.9	0	2	82.67	84.16	81.00	81.33
Heart	75.6	1	4	81.85	90.00	80.37	77.41
Hepatitis	80.7	1	3	89.33	88.39	83.33	83.33
Ionosphere	84.5	0	2	94.29	81.48	84.00	87.14
Iris	92.0	0	3	96.00	96.00	96.67	95.33
LiverBupa	65.8	2	3	68.53	83.48	66.47	66.47
Sonar	69.4	0	4	84.00	87.50	85.00	86.50
Vehicle	67.9	2	4	66.55	91.96	69.29	71.43
Vote	96.1	4	6	91.74	94.83	92.17	90.87
Wine	92.1	0	2	95.29	90.45	94.71	95.29
Zoo	91.1	0	3	92.22	92.22	95.56	95.56
Average	79.96			83.83	89.78	82.58	82.34

Table 10: The best classification accuracy and reduction rate of ε-SBModel on each data set

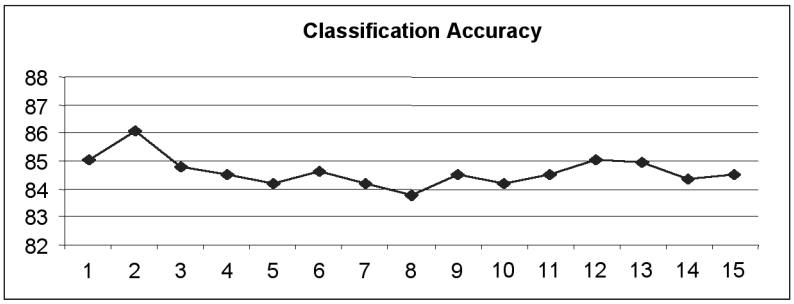

Figure 8: The classification accuracy of ε-SBModel testing on Australian data set with different ε

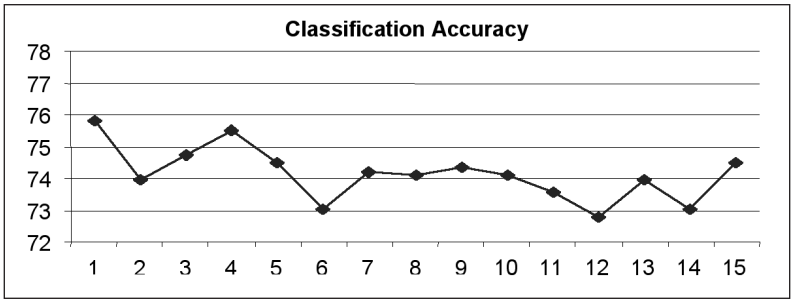

Figure 9: The classification accuracy of ε-SBModel testing on Diabetes data set with different ε

[Experiment 3] In this experiment, our goal is to evaluate SBModel-*i*. So we assign 1 and 3 to parameter *i* for local *i*-consistent neighbourhood in turn and test the classification accuracy and reduction rate of SBModel-*i*. When the value of *N* is set to 2 and 3 in turn, the comparisons of different SBModel-*i* with SBModel in classification accuracy and reduction rate using ten-fold cross validation are presented in Table 11, Table 12 respectively.

From the experimental results presented in Table 11 and Table 12, it is clear that using local *i*-consistent neighbourhood results in no significant changes in both classification accuracy and reduction rate between SBModel-*i* and SBModel. This is why many proposed editing and condensing methods in the literature that tried to keep the consistency rule as a basic principle in their editing or condensing processes did not improve their classification accuracy. The detailed experimental results will be reported in experiment 5.

[Experiment 4] In this experiment, our goal is to evaluate *p*-SBModel. This is a non-parametric pruning method, which conducts pruning by removing both the representatives from the model *M* that only cover 1 instance (this means no induction is being done for this representative) and the instances covered by these representatives from the training set, then reconstructing the model from the revised training set. The experimental results are given in Table 13.

From the experimental results presented in Table 13, it is clear that with the same classification accuracy, *p*-SBModel has a slightly higher reduction rate than SBModel on average. The main merit of *p*-SBModel is that it does not need any parameters to be set in the modelling and classification stages. However, its classification accuracy is comparable to *k*NN and *wk*NN. It keeps only 10.13 percent instances of the original training set on average for classification.

Data set	SBModel	RR	SBModel-1	RR	SBModel-3	RR
Australian	84.20	86.81	83.91	88.41	84.93	89.13
Colic	81.67	78.26	82.78	79.62	82.78	81.79
Diabetes	75.00	80.47	74.74	82.29	74.34	85.03
Glass	65.24	79.44	64.76	81.31	65.71	85.51
HCleveland	82.67	84.16	79.00	84.16	77.67	85.81
Heart	80.37	84.81	78.52	84.81	79.63	86.67
Hepatitis	83.33	85.81	86.00	86.45	87.33	87.74
Ionosphere	94.29	81.48	93.71	84.33	92.86	84.33
Iris	96.00	95.33	96.00	96.00	96.00	96.00
LiverBupa	63.53	73.62	63.24	78.26	63.82	80.00
Sonar	84.00	81.73	86.00	81.73	83.00	82.69
Vehicle	65.83	80.38	64.05	83.92	64.88	85.82
Vote	88.70	91.38	89.57	91.81	89.13	92.67
Wine	95.29	90.45	94.71	89.89	94.71	89.89
Zoo	92.22	91.11	93.33	92.22	93.33	93.33
Average	82.16	84.35	82.02	85.68	82.01	87.09

Table 11: A comparison of SBModel-*i* with SBModel in both classification accuracy and reduction rate when *N*=2

Data set	SBModel	RR	SBModel-1	RR	SBModel-3	RR
Australian	84.64	90.43	84.49	90.87	84.49	91.16
Colic	82.50	84.24	82.78	85.05	83.33	83.42
Diabetes	74.08	86.98	74.34	88.67	75.53	88.02
Glass	65.24	88.32	59.05	88.32	60.48	88.32
HCleveland	80.33	87.79	79.00	89.77	77.00	88.45
Heart	80.74	88.52	78.89	89.26	80.00	89.26
Hepatitis	85.33	88.39	87.33	88.39	86.67	88.39
Ionosphere	93.71	85.19	92.86	86.89	92.86	86.61
Iris	96.00	96.00	96.00	96.00	96.00	96.00
LiverBupa	64.41	83.48	63.24	86.67	63.82	85.80
Sonar	82.50	86.06	83.00	85.10	82.00	84.13
Vehicle	65.36	87.83	63.33	89.72	64.17	89.60
Vote	88.70	93.53	89.57	93.53	89.13	93.53
Wine	94.71	90.45	94.12	90.45	94.12	89.89
Zoo	92.22	92.22	92.22	93.33	93.33	93.33
Average	82.03	88.63	81.35	89.47	81.53	89.06

Table 12: A comparison of SBModel-*i* with SBModel in both classification accuracy and reduction rate when *N*=3

[Experiment 5] In this experiment, we compare SBModel and its variants with other algorithms in the literature in both classification accuracy and storage reduction. The algorithms to be compared in the experiment include CNN, SNN, IB3, DEL, ENN, RENN, Allk-NN, ELGrow, Explore and DROP3, each of which has been described in Section 2. The experimental results are presented in Table 14, Table 15, and Table 16 respectively.

Data Set	*k*NN (k=5)	*wk*NN (k=5)	RR	SBModel (N=3)	RR	*p*-SBModel	RR
Australian	85.22	82.46	0	84.64	90.43	86.23	95.22
Colic	83.06	81.94	0	82.50	84.24	82.78	88.59
Diabetes	74.21	72.37	0	74.08	86.98	73.16	87.11
Glass	67.62	67.62	0	65.24	88.32	65.24	84.58
HCleveland	81.00	81.33	0	80.33	87.79	80.67	89.11
Heart	80.37	77.41	0	80.74	88.52	81.85	91.11
Hepatitis	83.33	83.33	0	85.33	88.39	84.67	96.77
Ionosphere	84.00	87.14	0	93.71	85.19	92.00	87.18
Iris	96.67	95.33	0	96.00	96.00	95.33	95.33
LiverBupa	66.47	66.47	0	64.41	83.48	62.94	82.03
Sonar	85.00	86.50	0	82.50	86.06	82.50	86.54
Vehicle	69.29	71.43	0	65.36	87.83	67.26	83.69
Vote	92.17	90.87	0	88.70	93.53	90.00	96.98
Wine	94.71	95.29	0	94.71	90.45	94.71	90.45
Zoo	95.56	95.56	0	92.22	92.22	91.11	93.33
Average	82.58	82.34	0	82.03	88.63	82.03	89.87

Table 13: A comparison of *k*NN, *wk*NN, SBModel, and *p*-SBModel in both clasification accuracy and reduction rate

Data set	CNN	RR	SNN	RR	IB3	RR	DEL	RR	DROP3	RR
Australian	77.68	75.78	81.31	75.85	85.22	95.22	84.78	97.44	83.91	94.04
Colic	59.90	64.34	64.47	51.35	66.75	91.51	67.73	78.18	70.13	89.70
Diabetes	65.76	63.11	67.97	57.05	69.78	89.03	71.61	87.36	75.01	83.10
Glass	68.14	61.47	64.39	57.37	62.14	66.20	69.59	61.58	65.02	76.12
HCleveland	73.95	69.16	76.25	66.12	81.16	88.89	79.49	86.36	80.84	87.24
Heart	70.00	73.83	77.04	66.22	80.00	86.42	78.89	95.27	83.33	86.38
Hepatitis	75.50	74.70	81.92	69.04	73.08	94.91	80.00	92.41	81.87	92.20
Ionosphere	82.93	78.38	81.74	80.79	85.75	85.41	86.32	87.12	87.75	92.94
Iris	90.00	87.26	83.34	85.93	94.67	80.22	93.33	90.44	95.33	85.19
LiverBupa	56.80	59.13	57.70	47.41	58.24	89.34	61.38	87.36	78.00	75.01
Sonar	74.12	67.15	79.81	71.74	69.38	87.98	83.59	70.14	78.00	73.13
Vehicle	67.50	62.96	67.27	56.79	67.62	71.64	68.10	67.49	65.85	77.00
Vote	93.59	90.88	95.40	89.79	95.64	94.56	94.27	97.98	95.87	94.89
Wine	92.65	85.70	96.05	85.77	91.50	83.40	94.38	90.95	94.93	83.89
Zoo	91.11	87.53	76.67	89.38	92.22	70.62	90.00	81.73	90.00	80.00
Average	75.98	73.43	76.76	70.04	78.21	85.02	80.23	84.79	81.72	84.72

Table 14: A comparison of different algorithms in both classification accuracy and storage percentage (I)

From the experimental results, it is clear that the average classification accuracies and reduction rates of our proposed SBModel method and its variants on fifteen data sets are better than other data reduction methods in 10-fold cross validation with the exceptions of ELGrow and Explore in reduction rate. Though ELGrow obtains the highest reduction rate among all the algorithms, its

Data set	ENN	RR	RENN	RR	Allk-NN	RR	ELGrow	RR	Explore	RR
Australian	84.49	13.51	84.20	15.20	86.09	21.93	83.62	99.68	85.80	99.68
Colic	45.89	41.79	32.91	72.13	45.89	47.84	67.09	99.63	67.09	99.63
Diabetes	75.39	23.63	75.91	25.48	74.88	35.39	67.84	99.71	75.27	99.71
Glass	65.91	29.18	64.00	30.94	67.75	34.11	50.24	97.72	63.98	96.47
HCleveland	82.49	16.54	82.16	17.49	81.51	27.28	81.52	99.27	82.15	99.27
Heart	81.11	16.87	81.11	18.02	81.85	27.98	74.44	99.18	81.85	99.18
Hepatitis	81.25	16.27	80.58	17.20	81.33	24.80	76.67	99.00	78.67	98.71
Ionosphere	84.04	15.76	84.04	17.73	84.05	17.81	73.77	99.37	80.89	99.37
Iris	95.33	5.26	95.33	5.33	95.33	6.22	88.67	97.70	92.67	97.70
LiverBupa	61.12	31.85	58.77	36.87	60.24	47.66	56.74	99.45	57.65	99.36
Sonar	81.79	15.65	78.53	18.21	80.36	19.71	70.24	98.93	70.29	98.93
Vehicle	69.52	26.19	69.05	30.25	70.21	35.26	58.15	97.75	60.76	97.53
Vote	95.41	4.16	95.41	4.19	95.41	5.65	88.99	99.49	94.25	99.49
Wine	94.93	4.43	94.93	4.43	94.93	5.24	81.47	98.07	95.46	97.88
Zoo	91.11	7.04	91.11	7.41	93.33	5.93	94.44	92.10	95.56	91.60
Average	79.32	17.88	77.87	21.39	79.54	24.19	74.26	98.47	78.82	98.30

Table 15: A comparison of different algorithms in both classification accuracy and storage percentage (II)

Data set	SBModel (N=3)	RR	ε-SBModel (ε=0~4, N=5)	RR	*p*-SBModel (*p*=1)	RR	SBModel-*i* (*i*=3, N=2)	RR
Australian	84.64	90.43	85.51	92.46	86.23	95.22	84.93	89.13
Colic	82.50	84.24	83.61	88.86	82.78	88.59	82.78	81.79
Diabetes	74.08	86.98	75.53	91.67	73.16	87.11	74.34	85.03
Glass	65.24	88.32	69.05	93.93	65.24	84.58	65.71	85.51
HCleveland	80.33	87.79	81.67	92.08	80.67	89.11	77.67	85.81
Heart	80.74	88.52	81.11	91.48	81.85	91.11	79.63	86.67
Hepatitis	85.33	88.39	88.67	91.61	84.67	96.77	87.33	87.74
Ionosphere	93.71	85.19	92.57	89.74	92.00	87.18	92.86	84.33
Iris	96.00	96.00	96.00	96.00	95.33	95.33	96.00	96.00
LiverBupa	64.41	83.48	67.94	92.75	62.65	82.03	63.82	80.00
Sonar	82.50	86.06	81.50	90.38	82.50	86.54	83.00	82.69
Vehicle	65.36	87.83	66.07	93.50	67.14	83.68	64.88	85.82
Vote	88.70	93.53	91.74	94.40	90.00	96.98	89.13	92.67
Wine	94.71	90.45	94.12	92.70	94.71	90.45	94.71	89.89
Zoo	92.22	92.22	88.89	93.33	91.11	93.33	93.33	93.33
Average	82.03	88.63	82.93	92.33	82.03	89.87	82.01	87.09

Table 16: A comparison of different algorithms in both classification accuracy and storage percentage (III)

rather low classification accuracy conteracts this advantage. Explore seems to be a competitive algorithm with a higher reduction rate and a slightly lower classification accuracy than SBModel and its variants. Otherwise, DROP3 is closest to our algorithms in both classification accuracy and reduction rate.

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 efficiency and dependency on *k*, we select a few representatives from the training set with some extra information to represent the whole training set. In the selection of each representative we use an optimal and different *k*, chosen automatically for each data set itself, to eliminate the dependency on *k*, without the user's intervention. Experimental results carried out on fifteen public data sets show that SBModel and its variants: ε-SBModel, *p*-SBModel and SBModel-*i* are competitive for classification. Their average classification accuracies on fifteen public data sets are better than C5.0 and are comparable with *k*NN, and *wk*NN. But SBModel and its variants significantly reduce the number of instances in the final model for classification, with average reduction rates ranging from 87.09% to 92.33%. Moreover, compared with other reduction techniques, ε-SBModel obtains the best performance. It keeps only 7.67 percent instances of the original training set on average for classification whilst preserving the classification accuracy of *k*NN and *wk*NN. It is a good alternative to *k*NN in many application areas such as dynamic web mining from a large document repository.

ACKNOWLEDGEMENTS

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

REFERENCES

AHA, D.W., KIBLER, K. and ALBERT, M.K. (1991): Instance-based learning algorithms, *Machine Learning*, 6: 37–66.

AHA, D.W. (1992): Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms, *International Journal of Man-Machine Studies*, 36: 267–287.

BEZDEK, J.C., REICHHERZER, T.R., LIM, G.S. and ATTIKIOUZEL, Y. (1998): Multiple-prototype classifier design, *IEEE Transactions on Systems, Man and Cybernetics*, 28(1): 67–79.

CAMERON-JONES, R.M. (1995): Instance selection by encoding length heuristic with random mutation hill climbing, *Proc. of the 8th Australian Joint Conference on Artificial Intelligence*, 99–106.

CHANG, C.L. (1974): Finding prototypes for nearest neighbour classifiers, IEEE *Transactions on Computers*, 23(11): 1179–1184.

DEVIJVER, P. and KITTLER, J. (1982): Pattern recognition: A statistical approach, Prentice-Hall, Englewood Cliffs, NJ.

DOMINGOS, P. (1995): Rule induction and instance-based learning: A unified approach. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, Montreal, Canada: Morgan Kaufmann, 1226–1232.

GATES, G. (1972): The reduced nearest neighbour rule, *IEEE Transactions on Information Theory*, 18: 431–433.

HAND, D., MANNILA, H. and SMYTH, P. (2001): Principles of data mining, MIT Press.

HART, P. (1968): The condensed nearest neighbour rule, *IEEE Transactions on Information Theory*, 14: 515–516.

KUNCHEVA, L.I. and BEZDEK, J.C. (1998): Nearest prototype classification: Clustering, generic algorithm, or random search? *IEEE Trans. System Man Cybernet*, 28: 160–164.

MOLLINEDA, R.A., FERRI, F.J. and VIDAL, E. (2002): An efficient prototype merging strategy for the condensed 1-NN rule through class-conditional hierarchical clustering, *Pattern Recognition* 35: 2771–2782.

RITTER, G.L., WOODRUFF, H.B., LOWRY, S.R. *et al*. (1975): An algorithm for a selective nearest neighbour decision rule. *IEEE Transactions on Information Theory*, November, 665–669.

SEBASTIANI, F. (2002): Machine learning in automated text categorization. In *ACM Computing Surveys*, 34(1): 1–47.

STANFILL, C. and WALTZ, D. (1986): Toward memory-based reasoning communications of the ACM, 29: 1213–1228.

TOMEK, A. (1976): An experiment with the edited nearest-neighbour rule. *IEEE Transactions on Systems, Man, and Cybernetics*, 6(6): 448–452.

WANG, H. (2003): Contextual probability, in *Journal of Telecommunications and Information Technology*, 4(3):92–97.

WETTSCHERECK, D. and THOMAS, G.D. (1995): An experimental comparison of nearest-neighbour and nearest-hyperrectangle algorithms. *Machine Learning*, 19(1): 5–28.

WILSON, D.L. (1972): Asymptotic properties of nearest neighbour rules using edited data, *IEEE Transactions on Systems, Man, and Cybernetics*, 2(3): 408–421.

WILSON, D.R. and MARTINEZ, T.R. (1997): Improved heterogeneous distance functions, *Journal of Artificial Intelligence Research (JAIR)*, 6(1): 1–34.

WILSON, D.R. and MARTINEZ, T. R. (2000): Reduction techniques for instance-based learning algorithms, *Machine Learning*, 38(3): 257–286.

BIOGRAPHICAL NOTES

Gongde Guo

Gongde Guo is an associate professor in the School of Computer Science, Fujian Normal University, China. He received his PhD from the University of Ulster in 2004 and is currently working in the Department of Computing, University of Bradford, UK, as a research assistant. His research interests include artificial intelligence, machine learning, data mining, and text categorisation. He has authored over 30 papers in international conferences and journals.

Hui Wang

Hui Wang is a senior lecturer in the School of Computing and Mathematics, University of Ulster, UK. His research interests include artificial intelligence, machine learning, data mining, spatial reasoning, intelligent agents, and financial applications. He has authored over 70 papers in international journals and conferences. He has served on the programme committees of a number of international conferences, and has co-edited a special issue on financial data mining for an international journal.

David Bell

David Bell is a professor of computer science at Queen's University, Belfast. His research addresses the issues of functionality and performance in data and knowledge management systems, and linking these to reasoning under uncertainty, machine learning, and other artificial intelligence techniques. He has been a programme committee chairman of several international conferences, including Very Large Database (VLDB) Conference and the IEEE Conference on Data Engineering. Professor Bell is an associate editor of the Pergamon Press journal 'Information Systems' and a member of the Editorial Board of the Computer Journal. He was previously editor-in-chief of another Pergamon Press Journal, 'Database Technology'. His publication list contains more than 300 papers, articles and books. He has attracted funding from the EU for about twenty projects dating back to 1981, as well as from various other national sources.