

# A Selective Sampling Approach to Active Feature Selection

Huan Liu<sup>1</sup>, Hiroshi Motoda<sup>2</sup>, Lei Yu<sup>1</sup>

<sup>1</sup>Department of Computer Science & Engineering  
Arizona State University, Tempe, AZ 85287-8809, USA  
{hliu,leiyu}@asu.edu

<sup>2</sup>Institute of Scientific & Industrial Research  
Osaka University, Ibaraki, Osaka 567-0047, Japan  
motoda@sanken.osaka-u.ac.jp

## Abstract

Feature selection, as a preprocessing step to machine learning, has been very effective in reducing dimensionality, removing irrelevant data, increasing learning accuracy, and improving result comprehensibility. Traditional feature selection methods resort to random sampling in dealing with data sets with a huge number of instances. In this paper, we introduce the concept of active feature selection, and investigate a selective sampling approach to active feature selection in a filter model setting. We present a formalism of selective sampling based on data variance, and apply it to a widely used feature selection algorithm Relief. Further, we show how it realizes active feature selection and reduces the required number of training instances to achieve time savings without performance deterioration. We design objective evaluation measures of performance, conduct extensive experiments using both synthetic and benchmark data sets, and observe consistent and significant improvement. We suggest some further work based on our study and experiments.

**Key Words:** Dimensionality Reduction, Feature Selection and Ranking, Sampling, Learning

# 1 Introduction

Inductive learning is one of the major approaches to automatic extraction of useful patterns (or knowledge) from data. Data becomes increasingly larger in both number of features and number of instances in many applications such as genome projects, text mining, customer relationship management, and market basket analysis [2, 35, 43, 44, 55, 57]. This trend poses a severe challenge to inductive learning systems in terms of efficiency and effectiveness. Feature selection has proven to be an effective means when dealing with large dimensionality with many irrelevant features [17, 42, 57]. In particular, feature selection removes irrelevant features, increases efficiency of learning tasks, improves learning performance (e.g., predictive accuracy), and enhances comprehensibility of learned results [34, 37]. Although there exist numerous feature selection algorithms [6, 12, 23], new challenging research issues arise for feature selection: from handling a huge number of instances, large dimensionality (e.g., thousands of features), to dealing with data without class labels. This work is concerned with the number of instances in the context of feature selection.

When the number of instances is large, how can one use a portion of data to achieve the original objective without performance deterioration? Sampling is a common approach to this problem and many sampling methods are available [9, 22]. Random sampling is a method of selecting a sample of  $n$  out of  $N$  ( $n \leq N$ ) units such that every one of the  $\binom{N}{n}$  distinct samples has an equal chance of being drawn. It has proven to be an effective approach to dealing with large data sets for many machine learning tasks including classification, clustering, and association rule mining [7, 30, 52]. However, random sampling is blind because it does not exploit any data characteristic. In this work, we explore if sampling can use data characteristics and achieve better performance than random sampling in the following sense: either maintaining the performance of feature selection with much fewer instances, or improving the performance of feature selection with the same amount of instances.

This work is about feature selection on labeled data, i.e., class information is available. For feature selection on unlabeled data, various work can be found in [13, 15, 17, 18, 27, 53]. In Section 2, we briefly review the development of feature selection, its models, and active learning. In Section 3, we describe active feature selection and a formalism of selective sampling that exploits data variance. In Section 4, we demonstrate active feature selection using Relief and discuss implementation details. In Section 5, we design objective measures for performance evaluation including time savings. Section 6 is an empirical study in which we

evaluate the performance improvement obtained with active feature selection and discuss the implications of the findings. Section 7 presents a conclusion and some work to be completed in the near future.

## 2 Related Work

Feature selection refers to the study of algorithms selecting an optimal subset of the input feature set. Optimality is normally dependent on the evaluation criteria or the application's needs [37]. Major aspects of feature selection [34] include feature subset generation, search strategies, goodness evaluation, etc. Feature subset generation studies how a subset is generated following search directions. Search strategies cover exhaustive and complete search, random search, and heuristic search. A search can start from an empty feature set and add new features, or begin with a full set and remove irrelevant features. The goodness of a feature subset can be evaluated using various measures: consistency, distance, information, dependency, accuracy, etc. Feature selection algorithms fall into two broad categories, the filter model or the wrapper model [11, 31]. The filter model relies on the general characteristics of the training data to select some features independently of any learning algorithm, therefore it does not inherit any bias of a learning algorithm. The wrapper model requires one predetermined learning algorithm and uses the performance of the learning algorithm to evaluate and determine which features are selected. The wrapper model needs to learn a new hypothesis (or a classifier) [40] for each new feature subset. It tends to find features better suited to the predetermined learning algorithm resulting in good learning performance, but it also tends to be more computationally expensive than the filter model [34]. When the number of instances becomes large, the filter model is usually chosen due to its computational efficiency. In the context of this work, therefore, we focus on **the filter model** for feature selection.

The concept of active feature selection is inspired by the successful use of selected instances (or data points) in active learning [36, 49]. Active learning [49, 50] is different from traditional supervised learning: an active learner has the freedom to select which instances are added to the training set. An active learner may begin with a very small number of labeled instances, carefully select a few additional instances for which it requests labels, learn from the result of that request, and then using its newly-gained knowledge, carefully choose another few instances to request next. Several methods have been developed for active learning such as uncertainty sampling, adaptive sampling, and query-by-

committee [10, 19, 36]. The advantage of active learning is that the data requirements for some problems decrease drastically when the instances to be labeled are properly selected. In [50], they reported that when training the support vector machine (SVM) [8, 26] on a small subset of the available data chosen by their heuristic, its performance is frequently better than that of an SVM trained on all available data. Thus, an active learner provides better generalization and requires less data than a passive learner trained on the entire data set. Similar results were observed in [36] with their probabilistic classifiers.

Before we delve into the details of active feature selection, let us briefly describe the difference and commonality between active learning and active feature selection. The essence of active learning lies in its control over the choice of instances used in the *iterative* learning process [49, 54]. **Active feature selection** shares this essential characteristic with active learning in that it can influence the instances used for feature selection by exploiting some characteristics of the data. The selection process to form a representative sample data set is not iterative. As discussed earlier, we work with a filter model of feature selection and thus we do not employ a learning algorithm to actively choose instances. Therefore, the problem of active feature selection boils down to how we can employ **selective sampling** to choose representative instances for feature selection.

### 3 Active Feature Selection via Selective Sampling

Traditional feature selection methods perform dimensionality reduction using whatever training data is given to them. When the training data set is very large, random sampling is commonly used to deal with memory and performance issues. Active feature selection avoids pure random sampling and is realized by selective sampling. The idea of selective sampling stems from the fact that instances are not uniformly distributed and some instances are more representative than others [3]. If one can identify and select representative instances, fewer instances are needed to achieve similar performance. Therefore, the objective of selective sampling for feature selection is to select only those instances with a high probability to be informative in determining feature relevance. By adopting the filter model for feature selection, we have ruled out the possibility to use a learning algorithm to determine which instances are most relevant [38, 49]. In order to select representative instances for feature selection, we need to explore data characteristics: we first try to partition data according to data dissimilarity and then select representative instances from the resulting partitions.

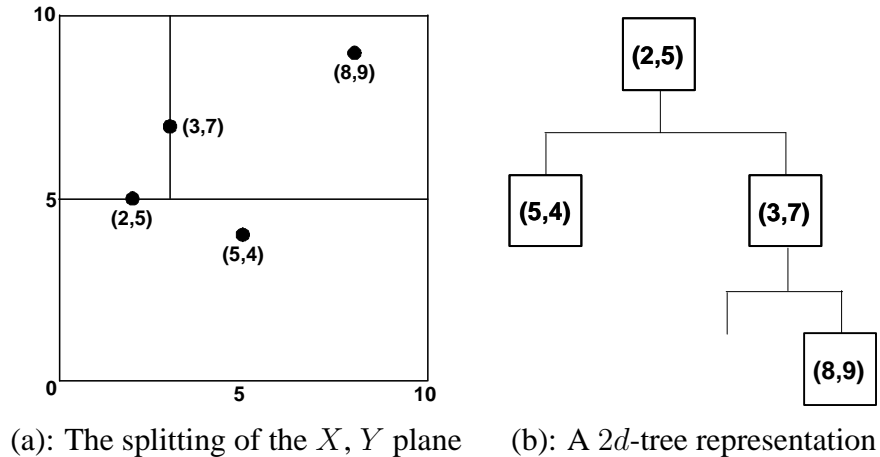


Figure 1: A typical  $kd$ -tree example.

There are many data partitioning techniques [21] in the literature on multi-dimensional indexing. We choose  $kd$ -tree [20] in this work because of its simplicity and popularity. A  $kd$ -tree is an index structure often used for fast nearest neighbor search [41]. It is a generalization of the simple binary tree which uses  $k$  dimensions (features) instead of a single dimension (feature) to split data points (instances) in a multi-dimensional space. In a  $kd$ -tree, the root presents all the instances. Each interior node has an associated splitting feature  $A_i$  and a splitting value  $V_i$  ( $1 \leq i \leq k$ ) that divide the instances into two partitions: those with  $A_i$ -values less than  $V_i$  and those with  $A_i$ -values equal to or greater than  $V_i$ . Splitting features at different levels of the tree are different, with levels rotating among the features of all dimensions. The splitting is done recursively in each of the successor nodes until the node contains no more than a predefined number of instances (called bucket size) or cannot be split further. The order in which features are chosen to split can result in different  $kd$ -trees.

Figure 1 provides a typical example of  $kd$ -tree with four instances  $(2, 5)$ ,  $(3, 7)$ ,  $(5, 4)$ ,  $(8, 9)$  in a 2-dimension space: (a) shows how the instances are partitioned in the  $X, Y$  plane, and (b) gives a tree representation of the four instances. The root node, with point  $(2, 5)$ , splits the plane along the  $Y$ -axis into two subspaces. The point  $(5, 4)$  lies in the lower subspace (i.e.,  $(x, y) \mid y < 5$ ), and thus is in the left subtree. The points  $(3, 7)$  and  $(8, 9)$  lie in the upper subspace (i.e.,  $(x, y) \mid y \geq 5$ ), and thus are in the right subtree. The point  $(3, 7)$  further splits the upper subspace into two parts along the  $X$ -axis.

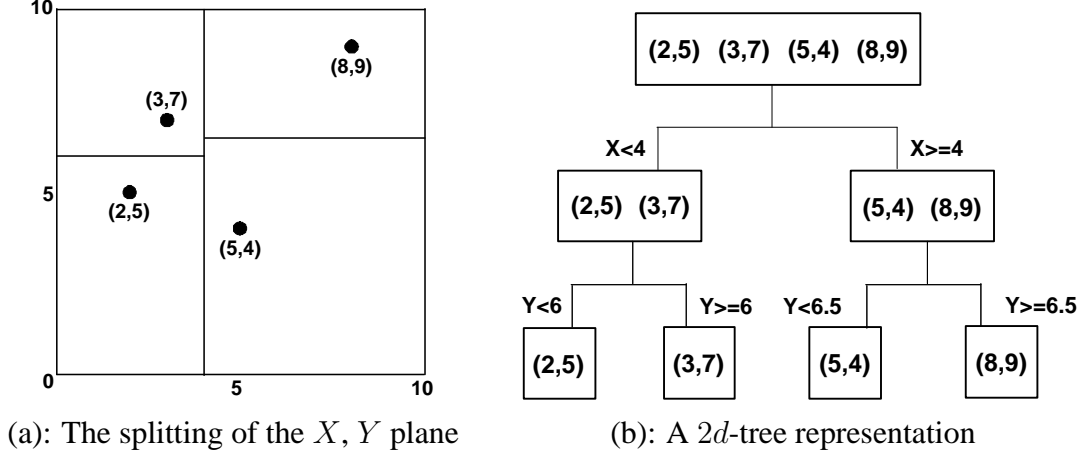


Figure 2: A modified use of  $kd$ -tree.

For the purpose of selective sampling, we want to select features that can split instances into different groups based on their dissimilarity measures as early as possible. Hence, in our building a  $kd$ -tree, a splitting feature is chosen if the data variance is maximized along the dimension associated with the feature. The variance of a feature  $A_i$  is calculated according to the formula  $Var(A_i) = \frac{\sum_{j=1}^N (V_j - V_i)^2}{N}$ , where  $V_i$  is the median value of feature  $A_i$ , and  $j$  is the index of each instance in a data set with  $N$  instances. Once feature  $A_i$  is determined, value  $V_i$  is then used to split the instances into two partitions. Figure 2 shows a modified  $kd$ -tree of the example shown in Figure 1. At the root level, since  $Var(X) = 5.5$  (with median value 4) and  $Var(Y) = 3.75$  (with median value 6), the space is split along the  $X$ -axis ( $x = 4$ ) into two subspaces. The points (2, 5) and (3, 7) lie in the left subspace (i.e.,  $(x, y) \mid x < 4$ ), and thus are in the left subtree. The points (5, 4) and (8, 9) lie in the right subspace (i.e.,  $(x, y) \mid x \geq 4$ ), and thus are in the right subtree. Each of the two subspaces are further split along the  $y$ -axis ( $y = 6$ ,  $y = 6.5$ , respectively) into two parts, producing four leaf nodes in the tree. Since the median value (instead of mean) of feature  $A_i$  is used in the calculation of variance, the resulting  $kd$ -tree is theoretically a balanced tree. The time complexity of building such an optimized  $kd$ -tree is  $O(kN \log N)$ . However, in some cases, a feature selected to split the tree may contain duplicate values, which may result in an unbalanced tree. The  $kd$ -tree can be built once if necessary, or dynamically when required.

In building a modified  $kd$ -tree (as shown in Figure 2), the leaf nodes (buckets) represent mutually exclusive small subsets of instances which collectively form a partition of the whole data set<sup>1</sup>. The size of a bucket can be an input parameter and determined *a priori*. Since instances in each bucket are relatively close to each other, we can randomly select one instance from each bucket to represent the effect of all the instances in its corresponding bucket. Therefore, a subset of the instances chosen by selective sampling is used to approximate the full sample space. Selective sampling can be summarized by a 3-step procedure:

1. Determine the size  $t$  of a bucket
2. Build a variance-based  $kd$ -tree with  $m$  buckets
3. Randomly select an instance from each bucket

We provide next details of selective sampling for feature selection.

## 4 Applying Selective Sampling to Feature Selection

Efficiency is critical for feature selection algorithms in the context of large data sets. To demonstrate selective sampling, we use a well-known and efficient algorithm Relief that can select statistically relevant features in linear time of the numbers of features and instances [28, 47]. After introducing Relief in Section 4.1, we illustrate how selective sampling can be applied to Relief and present a new algorithm of active feature selection in Section 4.2. In Section 4.3, we discuss other related work on Relief.

### 4.1 Relief algorithm

The key idea of Relief (given in Figure 3) is to estimate the quality of features according to how well their values distinguish between instances that are near to each other. For this purpose, given a randomly selected instance  $X$  from a data set  $S$  with  $k$  features, Relief searches the data set for its two nearest neighbors: one from the same class, called nearest hit  $H$ , and the other from a different class, called nearest miss  $M$ . It updates the quality estimation  $W[A_i]$  for all the features  $A_i$  based on the values of difference function  $diff()$  about  $X$ ,  $H$ , and  $M$ . The process is repeated  $m$  times, where  $m$  is a user-defined parameter [28, 32]. For

---

<sup>1</sup>Every node at each level of the tree only records indexes of the instances this node contains.

---

Given  $m$  - desired number of sampled instances, and  $k$  - number of features,

1. set all weights  $W[A_i] := 0.0$ ;
  2. for  $j := 1$  to  $m$  do begin
  3. randomly select an instance  $X$ ;
  4. find nearest hit  $H$  and nearest miss  $M$ ;
  5. for  $i := 1$  to  $k$  do begin
  6.  $W[A_i] := W[A_i] - \text{diff}(A_i, X, H)/m + \text{diff}(A_i, X, M)/m$ ;
  7. end;
  8. end;
- 

Figure 3: Original Relief algorithm.

instances  $X_1, X_2$ ,  $\text{diff}(A_i, X_1, X_2)$  calculates the difference between the values<sup>2</sup> ( $x_{1i}$  and  $x_{2i}$ ) of feature  $A_i$ :

$$\text{diff}(A_i, x_{1i}, x_{2i}) = \begin{cases} |x_{1i} - x_{2i}| & \text{if } A_i \text{ is numeric} \\ 0 & \text{if } A_i \text{ is nominal \& } x_{1i} = x_{2i} \\ 1 & \text{if } A_i \text{ is nominal \& } x_{1i} \neq x_{2i} \end{cases}$$

Normalization with  $m$  in calculation of  $W[A_i]$  (line 6 in Figure 3) guarantees that all weights are in the interval of  $[-1, 1]$ .

The time complexity of Relief for a data set with  $N$  instances is  $O(mkN)$ . Efficiency is one of the major advantages of the Relief family over other algorithms [12]. With  $m$  being a constant, the time complexity becomes  $O(kN)$ . However, since  $m$  is the number of instances used to approximate probabilities, a larger  $m$  implies more reliable approximations. When  $N$  is very large, it is often required that  $m \ll N$ .

## 4.2 Relief with selective sampling

Although random sampling of  $m$  instances for Relief reduces the time complexity from  $O(kN^2)$  to  $O(kN)$ , the optimal results of Relief are not guaranteed. When applying selective sampling to Relief, we aim to obtain results that are better than using random sampling and similar to the results using all the instances.

In our attempt to selectively sample  $m$  instances based on the modified  $kd$ -tree introduced in Section 3, we can use either sample size  $m$  or bucket size  $t$

---

<sup>2</sup>Numeric values should be normalized into the range between 0 and 1.



---

Given  $t$  - bucket size,

1. set all weights  $W[A_i] := 0.0$ ;
  2. `buildKDTree( $t$ )`;
  3.  $m :=$  number of buckets;
  4. for  $j := 1$  to  $m$  do begin
  5.   randomly select an instance  $X$  from  $Bucket[j]$ ;
  6.   find nearest hit  $H$  and nearest miss  $M$ ;
  7.   for  $i := 1$  to  $k$  do begin
  8.      $W[A_i] := W[A_i] - diff(A_i, X, H)/m + diff(A_i, X, M)/m$ ;
  9.   end;
  10. end;
- 

Figure 4: Relief with selective sampling.

to control the  $kd$ -tree splitting process. Since only one instance is selected from each bucket, one can easily establish that  $t = N/m$ , where  $N$  is the total number of instances. Therefore, if  $m$  is predetermined, the splitting process stops when it reaches the level where each bucket contains  $N/m$  or fewer instances. For example, in Figure 2 (b), if  $m = 2$ , the splitting process stops after the first split when each bucket contains two instances. On the other hand, given a  $t$ , we can estimate  $m$  based on  $t$ . In Figure 2 (b), if we choose  $t$  to be 1 (i.e., each bucket contains only one instance), all the four instances will be selected. For a bucket size  $t = N$ , the root node is not split at all, and only one instance will be selected. As we mentioned earlier, in practical, the resulting  $kd$ -tree may not be a balanced tree due to duplicate feature values. Therefore, for  $1 < t < N$ , the number of selected instances  $m$  will be within  $(\frac{1}{t}N, N)$ .

In this work, we use bucket size  $t$  to control the number of selected instances  $m$  which is equal to the number of buckets. The algorithm of Relief with selective sampling is detailed in Figure 4. One important point in `buildKDTree( $t$ )` is that each feature should be normalized [56] before the variance calculation in order to choose to split on the feature with largest variance.

In the empirical study, we use ReliefF [32, 56] which is an extension to the original Relief. It handles multiple classes and searches for several nearest neighbors to be robust to noise. We compare ReliefF with its counterpart ReliefS which applies selective sampling for instance selection in the same way as described in Figure 4. The difference between the two is illustrated in Figure 5. The top flow

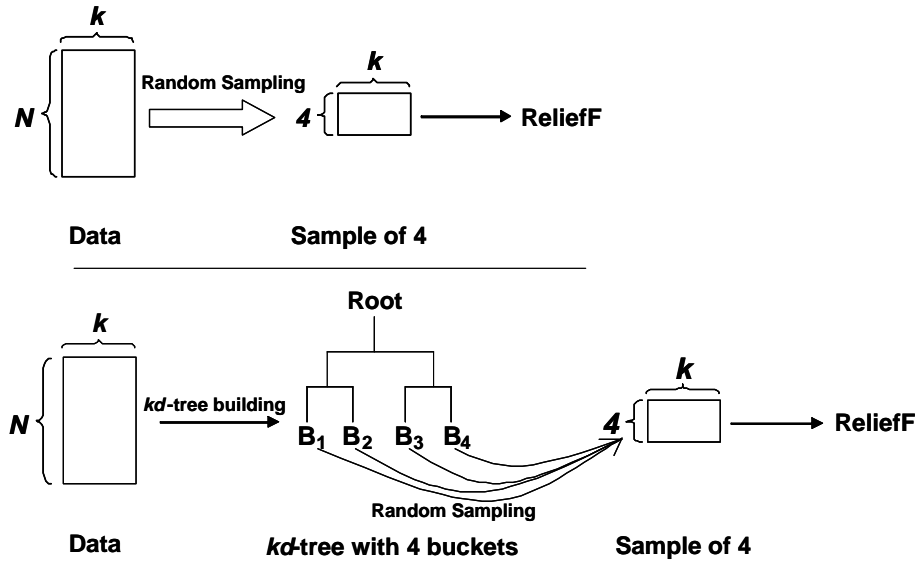


Figure 5: The difference between ReliefF and ReliefS: The top and bottom flows show how ReliefF and ReliefS work, respectively.

shows ReliefF with random sampling of 4 instances from the data. The bottom flow shows ReliefS with random sampling of one instance from each of the four buckets of the *kd*-tree.

### 4.3 Related work on Relief

There has been substantial research on the Relief family of algorithms (Relief, ReliefF, and RReliefF) [25, 28, 32, 33, 45, 46, 47, 48]. Relief (introduced in Section 4.1) was designed for feature subset selection [28, 29] and it is considered one of the best algorithms for this purpose [16]. Relief only deals with binary classes. This limitation was overcome by ReliefF [32] which handles multiple classes and incomplete and noisy data. ReliefF was further extended to RReliefF [45] in order to handle continuous classes in regression. The Relief family of algorithms are general and successful feature estimators and are especially good in detecting conditional dependencies between features [48]. In inductive learning, ReliefF was successfully employed in building decision trees for classification. The resulting decision trees achieved superior predictive accuracy over Naive Bayesian classifiers and k-NN classifiers across various artificial and real-world data sets [33]. It

was also shown in [45] that RReliefF is effective in selecting features in learning regression trees for regression problems.

Albeit a broad spectrum of successful uses of the Relief family of algorithms, little work has shown how to determine an optimal sample size<sup>3</sup>. Therefore, work regarding Relief usually circumvents the issue of optimal  $m$  by simply choosing  $m$  to be the full size of the data set [32] as the larger  $m$  leads to better performance<sup>4</sup>. Robnik-Sikonja and Kononenko showed in [48] that although feature selection results become stable after a number of iterations for simple data sets, the quality of results keeps improving as the sample size increases for more complicated data sets. The goal of this work is to show that for a given sample size  $m$ , instances obtained by selective sampling are more effective for feature selection than those selected by random sampling. Hence, in our work, we observe how selective sampling differs from random sampling by varying  $m$  from 10% to 100% of  $N$  using ReliefF with all  $N$  instances as the performance reference.

Recall that Relief relies on the search of a predefined number of nearest neighbors [32], and the  $kd$ -tree data structure is often used for fast nearest neighbor search [41]. In [51],  $kd$ -trees are applied to speed up Relief and its extensions by providing a fast way to locate nearest neighbors for each class, thus reducing the overall time complexity of the algorithms to  $O(kN\log N)$ . Additional time savings can be achieved by building the  $kd$ -tree once and using it for both bucket generation and the nearest neighbor search. However, the focus of this work is not to speed up current algorithms in the Relief family, but to investigate the effectiveness of selective sampling in partitioning data points for feature selection.

## 5 Issues of Performance Evaluation

Before we proceed to define performance measures for selective sampling, let us first discuss some criteria for a suitable performance measure:

1. It is a function of the features of the data.
2. Its value improves as  $m$  increases.
3. Its value reaches the best when  $m = N$ .

---

<sup>3</sup>One heuristic suggested in [25] is to choose  $m = \log N$  in an algorithm similar to Relief.

<sup>4</sup>Doing so increases the complexity of the algorithm to  $O(kN^2)$ .

In ReliefF, since  $m$  is the number of instances used to approximate probabilities, a larger  $m$  implies more reliable approximations. Therefore, it is reasonable to assume that the optimal result ReliefF can achieve is the features ranked according to their weights when  $m = N$ . This ranked list of features is named  $S_N$ . Given various sizes of  $m$ , the results of ReliefF and ReliefS can be compared in two aspects: (1) subset selection - we compare which resulting subset is more similar to the optimal subset obtained from  $S_N$ ; and (2) feature order - we compare the order of features determined by ReliefF or ReliefS to the order of features in  $S_N$ . In Section 5.1, we discuss the reason why it is important to consider the order of features. In Section 5.2, we present three different measures with different emphasis on feature order and check if they satisfy the above three criteria. In Section 5.3, we discuss time savings when using active feature selection.

## 5.1 Importance of feature order

Based on the output type, feature selection algorithms can be grouped into two categories, minimum subset algorithms and feature ranking algorithms [37]. Minimum subset algorithms return a minimum feature subset but do not rank features in the subset [14, 24, 53]. Features in the subset are relevant, others are irrelevant. Feature ranking algorithms assign a weight to each feature of the data set and rank the relevance of features according to their weights [15, 32, 58]. Order information is important for these algorithms because it indicates relative relevance. Order information is also important if one aims to select a feature subset from the resulting ranked list. Below, we illustrate how order information can affect the selection of relevant features for ReliefF and other feature ranking algorithms.

We first define some terms. For an optimal list of features  $S_N$ , a target subset of features  $T$  is defined as an optimal subset of features which contains the top  $n$  weighted features in  $S_N$ . For a data set with an unknown number of relevant features ( $n$ ),  $T$  contains the top  $n$  features whose weights  $\geq \gamma$ , where  $\gamma$  is a threshold equal to  $W[i]$  (the  $i$ th largest weight in  $S_N$ ) and the gap defined by  $W[i]$  and  $W[i+1]$  is sufficiently large (e.g., greater than the average gap among the  $k-1$  gaps). Let  $S_{ReliefF}$  and  $S_{ReliefS}$  be the two resulting lists obtained by ReliefF and ReliefS, respectively. To compare the performance of both ReliefF and ReliefS with different sizes of  $m$ , we can define a performance measure  $\mathcal{P}(S_N, R)$  where  $R$  can be either  $S_{ReliefF}$  or  $S_{ReliefS}$  with varying  $m$ . Figure 6 (a) illustrates the relationships among  $S_N$ ,  $T$ ,  $R$ , and  $R_n$  (the subset of the top  $n$  features in  $R$ ). Figure 6 (b) shows five cases for a set of five features. Column I shows the optimal ranking of the five features ( $S_N$ ), and each of the remaining four columns (II-V)

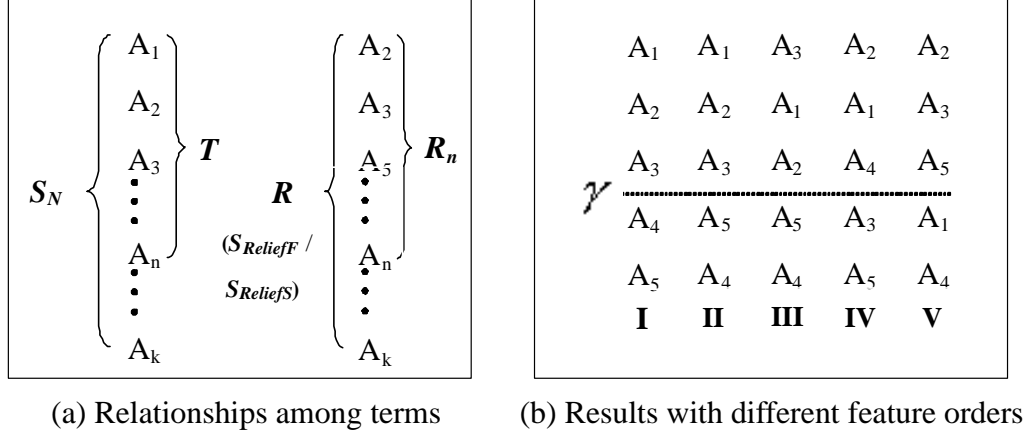


Figure 6: An example for the importance of feature order.

represents a ranking from either ReliefF or ReliefS with  $m$  instances, assuming  $m < N$ .

For a given  $n$  (either known *a priori* or determined by threshold  $\gamma$ ), the order of features in  $R$  directly affects the selection of a feature subset  $R_n$ . For example, in Figure 6 (b), the threshold  $\gamma$  is chosen to include the top three features from each resulting list. All the three relevant features ( $A_1, A_2, A_3$ ) are selected into  $R_n$  in cases II and III, while only two of them are selected in cases IV and V. However, given the same ordered lists II-V, if  $\gamma$  is chosen to include the top two features, the selected subset  $R_n$  in case III will fail to include both of the two relevant features  $A_1$  and  $A_2$ , while  $R_n$  in case IV will become an optimal subset. Therefore, it is important to develop performance measures that take order information into consideration. Below we examine sensible candidates for  $\mathcal{P}()$ .

## 5.2 Performance measures

### 5.2.1 Precision

Precision (P) is computed as the number of features that are in both  $T$  and  $R_n$ , divided by the number of features in  $T$ :

$$P = \frac{|T \cap R_n|}{|T|}.$$

P ranges from 0 to 1, where P is 1 when subsets  $T$  and  $R_n$  are equal and 0 when none of the features in  $T$  appears in  $R_n$ . In Figure 6 (b), P is 1 for cases II and III and  $2/3$  for cases IV and V.

### 5.2.2 Distance

Precision treats all features in  $T$  and  $R_n$  equally without considering the orders of the features. In order to account for the order of features in both  $S_N$  and  $R$ , Distance (D) is defined based on the sum of distances between common features in  $T$  and  $R$ . The distance of a feature between two sets is the difference between its positions in the two ranked lists. Let  $S'_N$  be  $S_N$  in reverse order. The maximum possible ranking distance between two sets that share the same features is:

$$D_{max} = \sum_{\forall A_i \in S_N} |position(A_i \in S_N) - position(A_i \in S'_N)|.$$

D is then defined as the follows:

$$D = \frac{\sum_{\forall A_i \in T} |position(A_i \in T) - position(A_i \in R)|}{D_{max}}$$

Since the subset  $R_n$  may not contain all the features in  $T$ , we use the full set  $R$  in the definition of D.  $D_{max}$  is used to normalize D so that D ranges from 0 to 1, where D is 0 when the two sets  $T$  and  $R_n$  have identical ranking (as shown in case II), otherwise, D is larger than 0. In Figure 6 (b), with  $D_{max} = 12$ , the D values for cases II, III, IV, and V are 0,  $4/12$ ,  $3/12$ , and  $5/12$ , respectively.

### 5.2.3 Raw Distance

Both Precision and Distance require choosing a threshold  $\gamma$  to decide the target set  $T$ . In some cases, it is difficult to determine an optimal threshold. If  $\gamma$  is wrongly estimated, meaning that the target set  $T$  is not an optimal subset, evaluation using these measures will be incorrect. For example, in Figure 6 (b), if  $\gamma$  is chosen between features  $A_2$  and  $A_3$  in  $S_N$ , the target set will be  $\{A_1, A_2\}$  instead of  $\{A_1, A_2, A_3\}$ . Thus, Precision for case III will become  $1/2$  instead of 1, and Precision for case IV will become  $1$  instead of  $2/3$ . A straightforward performance measure is to directly calculate the sum of the differences of weights for each of the feature pairs (same features) in the optimal list  $S_N$  and the resulting

Table 1: Summary of performance measures.

|                  | Precision | Distance | Raw Distance |
|------------------|-----------|----------|--------------|
| Complexity       | $O(n^2)$  | $O(nk)$  | $O(k^2)$     |
| Upper bound      | 1         | 1        | None         |
| Lower bound      | 0         | 0        | 0            |
| Ordering         | No        | Yes      | Yes          |
| $\gamma$ setting | Yes       | Yes      | No           |

list  $R$ . We name it Raw Distance (RD):

$$\text{RD} = \sum_{i=1}^k |W_S[A_i] - W_R[A_i]|,$$

where  $W_S[A_i]$  and  $W_R[A_i]$  are associated with  $S_N$  and  $R$ , respectively. RD considers all  $k$  features in the two results. Thus, this measure does not rely on threshold  $\gamma$ . It is designed to compare the results of ReliefF and ReliefS, but it cannot be used for measuring the performance of subset selection as it uses all the features.

#### 5.2.4 Comparison of measures

Each measure serves a unique purpose in the evaluation of feature selection results. Table 1 provides a summary of these four measures. The setting of  $\gamma$  is required in ReliefF for feature subset selection. Precision is a simple measure of subset selection, but it is not sufficient due to its insensitivity to the order of features. Both Distance and Raw Distance are order-sensitive, but Raw Distance does not require the threshold setting.

### 5.3 Measuring time savings

It is sensible to question whether the use of selective sampling in feature selection would result in any time savings overall because the initial building of  $kd$ -tree incurs certain costs. This question is best answered by measuring computation time during experiments. ReliefS and ReliefF require different numbers of instances to achieve the same level of performance. We can compare the running times required to achieve a given level of performance. Let  $T_{kd-tree}$ ,  $T_{ReliefS}$ , and

$T_{ReliefF}$  be the times for  $kd$ -tree building, running ReliefS, and running ReliefF, respectively, with a given performance. A straightforward approach is to report  $T_{kd-tree}$ ,  $T_{ReliefS}$ , and  $T_{ReliefF}$  and compare  $T_{kd-tree} + T_{ReliefS}$  with  $T_{ReliefF}$ . Their difference can be either the saving or the loss of computation time.

## 5.4 Measuring accuracy improvement

One can indirectly measure the results of feature selection using a learning algorithm to check its effect on accuracy. Likewise, the effectiveness of selective sampling for feature selection can be further verified by comparing the learning accuracy on the subsets of features chosen by ReliefF and ReliefS. For the two resulting lists produced by ReliefF and ReliefS with the same number of sampled instances of a data set, two different feature subsets of the same cardinality can be chosen from the top of the two lists and then used to obtain the learning accuracy for a certain learning algorithm. We expect an accuracy gain from the subset chosen by ReliefS over the subset chosen by ReliefF.

# 6 Empirical Study

The objective of this section is to empirically evaluate if selective sampling can do better in selecting  $m$  instances than random sampling in the context of ReliefF. We examine if the results of feature selection are consistently better when using instances sampled from  $kd$ -tree buckets than when using the same number of instances selected by random sampling. In Section 6.1, we present two groups of data sets (synthetic and benchmark) and the experimental procedures. In Section 6.2 and Section 6.3, we present and discuss results for synthetic data sets and benchmark data sets. In Section 6.4, we further examine the effectiveness of selective sampling in terms of learning accuracy.

## 6.1 Data and experimental procedures

We choose synthetic data sets in our experiments because the relevant features of these data sets are known beforehand. The use of synthetic data in our experiments serves three purposes: (1) to verify the effectiveness of ReliefF, (2) to avoid choosing the optimal threshold  $\gamma$  for subset selection, and (3) to evaluate the effectiveness of active feature selection. Since we rarely know the relevant features



Table 2: Features of the test data adapted from Agrawal et al.[1].

| Feature    | Description          | Value  |
|------------|----------------------|--|
| salary     | salary               | uniformly distributed from 20,000 to 150,000   |
| commission | commission           | if salary $\geq 75000 \rightarrow$ commission = 0<br>else uniformly distributed from 10000 to 75000.       |
| age        | age                  | uniformly distributed from 20 to 80.   |
| elevel     | education level      | uniformly distributed from $[0, 1, \dots, 4]$ .  |
| car        | make of the car      | uniformly distributed from $[1, 2, \dots, 20]$ .   |
| zipcode    | zip code of the town | uniformly chosen from 9 available zipcodes.  |
| hvalue     | value of the house   | uniformly distributed from $0.5k10000$ to $1.5k1000000$<br>where $k \in \{0 \dots 9\}$ depends on zipcode. |
| hyears     | years house owned    | uniformly distributed from $[1, 2, \dots, 30]$ .   |
| loan       | total amount of loan | uniformly distributed from 1 to 500000.  |

beforehand in practice, we also conduct experiments on benchmark data sets to evaluate selective sampling. We describe the two groups of data sets below.

### 6.1.1 Synthetic data

We use functions described in [1] to generate synthetic data sets so that we know exactly which features are relevant. The nine features are described in Table 2. Ten classification functions of Agrawal et al. [1] were used to generate classification problems with different complexities. Efforts were made to generate data sets as described in the original functions. Each data set consists of 5000 instances. The values of the features of each instance were generated randomly according to the distributions given in the table. For each instance, a class label was determined according to the rules that define the functions.

As an example, we give Function 2 that uses two features and classifies an instance into Group A if

$$\begin{aligned}
 & ((\mathbf{age} < 40) \wedge (50000 \leq \mathbf{salary} \leq 100000)) \vee \\
 & ((40 \leq \mathbf{age} < 60) \wedge (75000 \leq \mathbf{salary} \leq 125000)) \vee \\
 & ((\mathbf{age} \geq 60) \wedge (25000 \leq \mathbf{salary} \leq 75000)).
 \end{aligned}$$

Otherwise, the instance is classified into Group B.

### 6.1.2 Benchmark data

All together 23 data sets are selected from the UCI Machine Learning Data Repository [5] and the UCI KDD Archive [4]. They all have nominal classes with varied numbers of instances (from 150 to 145000), numbers of features (from 4 to 85), and numbers of classes (from 2 to 22). A summary of these data sets is presented in Table 3 which is divided into three groups: Group 1 contains only numeric data, Group 2 only nominal data, and Group 3 mixed data.

### 6.1.3 Experimental procedures

The experiments are conducted using Weka’s implementation of ReliefF, and ReliefS is also implemented in the Weka environment [56]. We use different percentages of data (varying  $m$ ). The performance of ReliefF with  $m = N$  is set as the performance reference point. The departure from the reference point is a measure of performance deterioration. In particular, we choose six increasing bucket sizes  $t_i$  ( $1 \leq i \leq 6$ ) from 1 to 6 corresponding to six percentage values  $P_i$ . For example,  $t_2$  corresponds to  $P_2 \approx 50\%$ . For each data set, the experiment is conducted as follows:

1. Run ReliefF with bucket size  $t_1$  ( $P_1 = 100\%$ ), and obtain the optimal ranked list of features ( $S_N$ ) according to their weights. The parameter for  $k$  in the  $k$ -nearest neighbor search is set to 5 (neighbors). This parameter remains the same for all experiments.
2. Run ReliefS with bucket sizes  $t_i$  ( $2 \leq i \leq 6$ ). At each  $t_i$ , run ReliefS 30 times with different seeds and calculate values of Precision, Distance, and Raw Distance for each iteration to obtain average values of these performance measures to eliminate any idiosyncrasy in a single run. A curve is plotted for each measure for comparison. Some of the curves are shown in Figures 7 and 8.
3. Run ReliefF with each  $P_i$  ( $2 \leq i \leq 6$ ) determined by corresponding  $t_i$  in step 2. For each  $P_i$ , run ReliefF 30 times and calculate Precision, Distance, and Raw Distance each time, and obtain their average values after 30 runs. A curve is plotted for each measure for comparison. Some of the curves are shown in Figures 7 and 8.

Table 3: Summary of benchmark data sets: N - number of instances, Num - numeric features, Nom - nominal features, #C - number of classes.

| Title         | N     | Num | Nom | #C |
|---------------|-------|-----|-----|----|
| Iris          | 150   | 4   | 0   | 3  |
| Glass         | 214   | 9   | 0   | 7  |
| WDBC          | 569   | 30  | 0   | 2  |
| Balance       | 625   | 4   | 0   | 3  |
| Pima-Indian   | 768   | 8   | 0   | 2  |
| Vehicle       | 846   | 18  | 0   | 4  |
| German        | 1000  | 24  | 0   | 2  |
| Segment       | 2310  | 19  | 0   | 7  |
| Abalone       | 4177  | 8   | 0   | 3  |
| Satimage      | 4435  | 36  | 0   | 6  |
| Waveform      | 5000  | 40  | 0   | 3  |
| Page-Blocks   | 5473  | 10  | 0   | 5  |
| CoIL2000      | 5822  | 85  | 0   | 2  |
| Shuttle       | 14500 | 8   | 0   | 7  |
| Breast-cancer | 286   | 0   | 9   | 2  |
| Primary-tumor | 339   | 0   | 17  | 22 |
| KRKPA7        | 3196  | 0   | 36  | 2  |
| Mushroom      | 8124  | 0   | 22  | 2  |
| Zoo           | 101   | 1   | 16  | 7  |
| Autos         | 205   | 15  | 10  | 7  |
| Colic         | 368   | 7   | 15  | 2  |
| Vowel         | 990   | 10  | 3   | 11 |
| Hypothyroid   | 3772  | 7   | 22  | 4  |

Table 4: Effectiveness of ReliefF on synthetic data. The order of features in a relevant feature set has no significance based on function definitions.

|             | Relevant Feature Set     | Result from ReliefF ( $m = N$ )     |
|-------------|--------------------------|-------------------------------------|
| Function 1  | {A3}                     | A3,  A5, A6, A8, A4, A7, A9, A2, A1 |
| Function 2  | {A1, A3}                 | A1, A3,  A2, A9, A7, A4, A8, A6, A5 |
| Function 3  | {A3, A4}                 | A4, A3,  A7, A1, A9, A2, A6, A8, A5 |
| Function 4  | {A1, A3, A4}             | A1, A4, A2,  A3, A7, A6, A5, A9, A8 |
| Function 5  | {A1, A3, A9}             | A9, A3, A1,  A2, A7, A4, A8, A6, A5 |
| Function 6  | {A1, A2, A3}             | A1, A3, A2,  A8, A4, A5, A7, A9, A6 |
| Function 7  | {A1, A2, A9}             | A9, A1, A2,  A8, A5, A7, A6, A3, A4 |
| Function 8  | {A1, A2, A4}             | A1, A4, A2,  A5, A7, A9, A6, A3, A8 |
| Function 9  | {A1, A2, A4, A9}         | A9, A1, A4, A2,  A5, A3, A7, A6, A8 |
| Function 10 | {A1, A2, A4, A7, A8, A9} | A4, A1, A2, A8, A3, A9,  A7, A6, A5 |

## 6.2 Results on synthetic data

Table 4 reports the set of relevant features used to determine the class labels in the definition of each function and the optimal ranking list obtained from running ReliefF on the whole data set of 5000 instances generated by each function. From Table 4, we observe that given the cardinality of a relevant feature set ( $n$ ), for each of these 10 functions except Function 4 and Function 10<sup>5</sup>, the set of the top  $n$  features in each ranking list matches exactly with the known relevant feature set. This verifies that ReliefF can indeed find the relevant features of a data set in most cases. Thus we can use the results obtained from the first step of the experimental procedures as a reference to evaluate the performance of ReliefF and ReliefS with varying size  $m$ .

Table 5 presents a summary of the three performance measures for each synthetic data set. For a given measure, let  $\mathcal{P}_i$  be the averaged value (over 30 runs) obtained at percentage  $P_i$  ( $2 \leq i \leq 6$ ). Each value in Table 5 is averaged over five percentage values, i.e.,

$$val_{Avg} = (\sum_{i=2}^6 \mathcal{P}_i) / 5.$$

Recall that Precision varies from 0 (worst) to 1 (best); Distance varies from 0 (best) to 1 (worst); and Raw Distance starts at 0 (best) and increases. The last

<sup>5</sup>For these two functions, one relevant feature is just outside the selected set. This could be due to the inability of ReliefF to differentiate redundant features with strong correlation.

Table 5: Average values of three different measures using ReliefF and ReliefS on **synthetic** data.

|              | Precision    |         | Distance     |         | Raw Distance  |         |
|--------------|--------------|---------|--------------|---------|---------------|---------|
|              | ReliefF      | ReliefS | ReliefF      | ReliefS | ReliefF       | ReliefS |
| Function 1   | 1.0          | 1.0     | 0.0          | 0.0     | 0.049         | 0.041   |
| Function 2   | 0.997        | 1.0     | 0.017        | 0.0     | 0.047         | 0.044   |
| Function 3   | 1.0          | 1.0     | 0.0          | 0.0     | 0.046         | 0.044   |
| Function 4   | 0.930        | 0.932   | 0.075        | 0.061   | 0.049         | 0.043   |
| Function 5   | 0.860        | 0.886   | 0.182        | 0.169   | 0.045         | 0.044   |
| Function 6   | 0.891        | 0.927   | 0.280        | 0.225   | 0.047         | 0.026   |
| Function 7   | 0.852        | 0.881   | 0.157        | 0.137   | 0.047         | 0.041   |
| Function 8   | 1.0          | 1.0     | 0.039        | 0.007   | 0.053         | 0.042   |
| Function 9   | 0.915        | 0.935   | 0.135        | 0.109   | 0.050         | 0.040   |
| Function 10  | 0.900        | 0.910   | 0.079        | 0.072   | 0.061         | 0.046   |
| <b>W/L/T</b> | <b>7/0/3</b> |         | <b>8/0/2</b> |         | <b>10/0/0</b> |         |

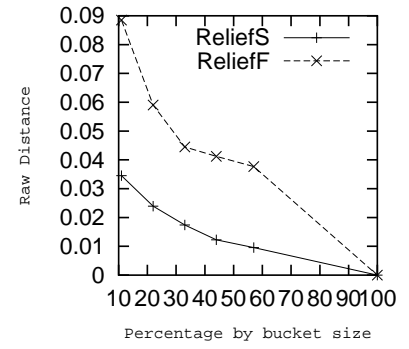
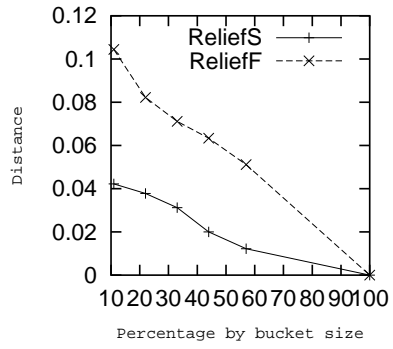
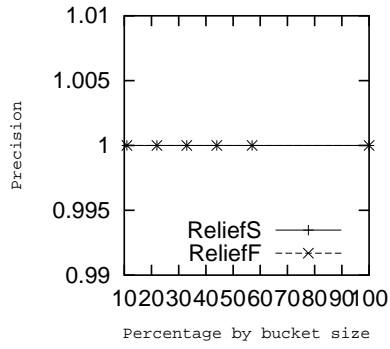
row (W/L/T) in Table 5 summarizes Win/Loss/Tie in comparing ReliefS with ReliefF for each measure. It is clear that for the ten synthetic data sets, taking all the three measures into account, ReliefS is as good as or better than ReliefF. This suggests that  $m$  instances selected using  $kd$ -trees are more effective than  $m$  instances selected at random.

### 6.3 Results on benchmark data

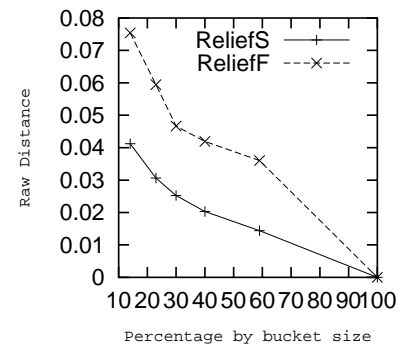
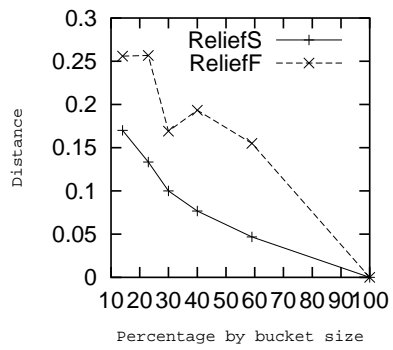
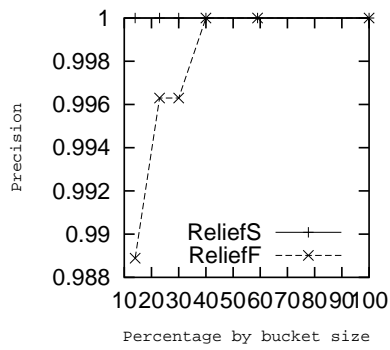
We present and discuss separately the results on numeric data and non-numeric data below.

#### 6.3.1 Results on numeric data

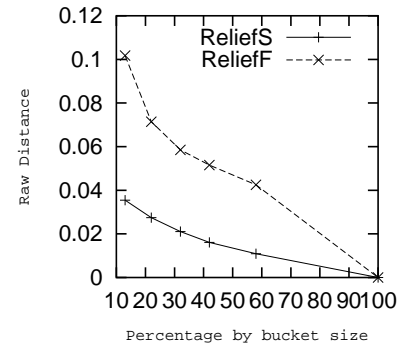
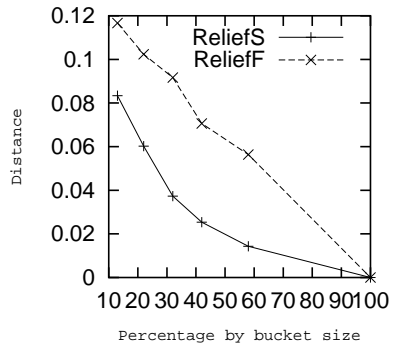
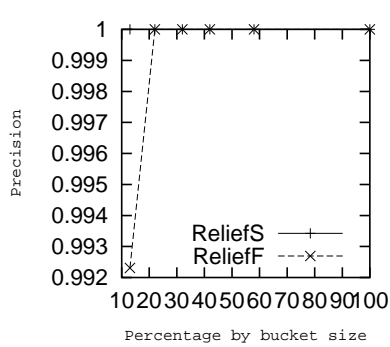
Two sets of results on numeric data sets are reported in Table 6 and Figure 7, respectively. Like Table 5, Table 6 shows that ReliefS is better than or as good as ReliefF in determining relevant instances for feature selection. Now let us look at the trends of the three performance measures when the number of instances increases for both ReliefF and ReliefS. Figure 7 shows the results of three illustrative data sets for Precision, Distance, and Raw Distance. For the Segment data,



### Segment Data



### Vehicle Data



### Satimage Data

Figure 7: Three illustrative **numeric** data sets: Average results of 30 runs in three performance measures (P, D, RD). 22

Table 6: Average values of three different measures using ReliefF and ReliefS on **numeric** data.

|              | Precision    |         | Distance      |         | Raw Distance  |         |
|--------------|--------------|---------|---------------|---------|---------------|---------|
|              | ReliefF      | ReliefS | ReliefF       | ReliefS | ReliefF       | ReliefS |
| Iris         | 1.0          | 1.0     | 0.020         | 0.013   | 0.054         | 0.019   |
| Glass        | 0.988        | 0.992   | 0.141         | 0.090   | 0.069         | 0.046   |
| WDBC         | 0.991        | 0.994   | 0.139         | 0.103   | 0.111         | 0.068   |
| Balance      | 0.864        | 0.940   | 0.468         | 0.247   | 0.037         | 0.018   |
| Pima-Indian  | 0.906        | 0.930   | 0.248         | 0.209   | 0.019         | 0.016   |
| Vehicle      | 0.996        | 1.0     | 0.206         | 0.105   | 0.052         | 0.026   |
| German       | 0.898        | 0.920   | 0.360         | 0.309   | 0.154         | 0.125   |
| Segment      | 1.0          | 1.0     | 0.074         | 0.029   | 0.054         | 0.020   |
| Abalone      | 0.947        | 0.971   | 0.257         | 0.176   | 0.003         | 0.001   |
| Satimage     | 0.998        | 1.0     | 0.088         | 0.047   | 0.065         | 0.022   |
| Waveform     | 1.0          | 1.0     | 0.080         | 0.056   | 0.047         | 0.036   |
| Page-Blocks  | 1.0          | 1.0     | 0.202         | 0.043   | 0.006         | 0.003   |
| CoIL2000     | 1.0          | 1.0     | 0.060         | 0.041   | 0.110         | 0.074   |
| Shuttle      | 1.0          | 1.0     | 0.0           | 0.0     | 0.003         | 0.001   |
| <b>W/L/T</b> | <b>8/0/6</b> |         | <b>13/0/1</b> |         | <b>14/0/0</b> |         |

we notice that both ReliefF and ReliefS perform equally well in Precision but differently in Distance and Raw Distance. Precision being 1 indicates that all feature subsets selected are the same as if we use all  $N$  instances (recall that the order of selected features is not considered by Precision). Distance and Raw Distance have values greater than 0, which suggests that the selected results are not in the same order as those obtained using all  $N$  instances. It can be observed that the more instances used for both ReliefF and ReliefS, the better the performance of feature selection. A similar trend can be observed for the Vehicle data and Satimage data as well.

Figure 7 and Table 6 indicate that active feature selection can significantly improve performance on numeric data over feature selection with random sampling with the same amount of instances. In other words, the use of  $kd$ -trees to partition the data makes the difference. However, as mentioned earlier, the building of  $kd$ -trees incurs certain costs. To actually compare the running time of ReliefS with that of ReliefF, we consider the case of bucket size 2 for ReliefF (i.e.,  $m_{ReliefF}$

Table 7: Time savings by ReliefS w.r.t. ReliefF for **numeric** data.  $m_{ReliefS}$  and  $m_{ReliefF}$  are numbers of instances used by ReliefS and ReliefF to achieve the same performance in raw distance.

|          | ReliefS               |                       |               | ReliefF               |               |
|----------|-----------------------|-----------------------|---------------|-----------------------|---------------|
|          | $T_{kd-tree}$<br>(ms) | $T_{ReliefS}$<br>(ms) | $m_{ReliefS}$ | $T_{ReliefF}$<br>(ms) | $m_{ReliefF}$ |
| Iris     | 20                    | 14                    | 18            | 60                    | 87            |
| Glass    | 61                    | 140                   | 62            | 259                   | 126           |
| WDBC     | 455                   | 1414                  | 125           | 3282                  | 313           |
| Balance  | 136                   | 223                   | 88            | 743                   | 313           |
| Pima     | 246                   | 484                   | 77            | 2684                  | 476           |
| Vehicle  | 530                   | 2874                  | 195           | 6790                  | 499           |
| German   | 697                   | 6274                  | 420           | 8310                  | 590           |
| Segment  | 1520                  | 10870                 | 277           | 53300                 | 1317          |
| Abalone  | 1359                  | 23879                 | 961           | 58228                 | 2423          |
| Satimage | 4630                  | 63901                 | 577           | 255716                | 2572          |
| Waveform | 7170                  | 359840                | 1900          | 529860                | 2950          |
| Page     | 1858                  | 43774                 | 1095          | 125709                | 3284          |
| CoIL     | 14117                 | 612203                | 1979          | 983601                | 3377          |
| Shuttle  | 6058                  | 187156                | 1885          | 802001                | 8700          |

is around 50% of the whole data set) and use the performance measured by Raw Distance to find corresponding  $m_{ReliefS}$  for ReliefS.

Table 7 records the running times  $T_{kd-tree}$ ,  $T_{ReliefS}$  and  $T_{ReliefF}$  as well as  $m_{ReliefS}$  and  $m_{ReliefF}$ . We can observe that:

1. The time savings are consistent with the time complexity analysis of ReliefF: it is usually  $O(mkN)$ , or linear in  $N$  if  $m$  and  $k$  are fixed. Now,  $k$  and  $N$  are fixed, its time complexity is  $O(m)$ . That is, the reduction of  $m$  results in direct time savings.
2. The larger the data set, the more savings in time.
3. The ratio of  $T_{kd-tree}/T_{ReliefS}$  decreases when data size increases. In other words, the time spent on  $kd$ -tree building becomes immaterial when  $T_{ReliefS}$  increases. This is consistent with earlier theoretical analysis.



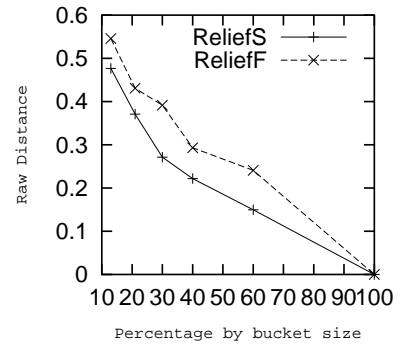
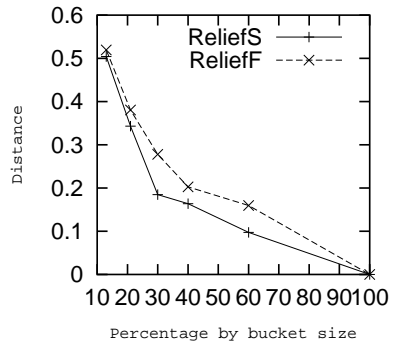
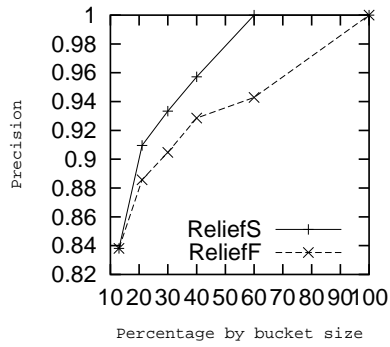
Table 8: Average values of three different measures using ReliefF and ReliefS on **non-numeric** data.

|               | Precision    |         | Distance     |         | Raw Distance |         |
|---------------|--------------|---------|--------------|---------|--------------|---------|
|               | ReliefF      | ReliefS | ReliefF      | ReliefS | ReliefF      | ReliefS |
| Breast-cancer | 0.765        | 0.821   | 0.695        | 0.601   | 0.203        | 0.162   |
| Primary-tumor | 0.957        | 0.965   | 0.259        | 0.207   | 0.296        | 0.246   |
| KRKPA7        | 0.970        | 0.977   | 0.103        | 0.070   | 0.152        | 0.101   |
| Mushroom      | 1.0          | 1.0     | 0.065        | 0.032   | 0.141        | 0.063   |
| Zoo           | 0.986        | 0.994   | 0.136        | 0.119   | 0.633        | 0.539   |
| Autos         | 0.9          | 0.928   | 0.308        | 0.259   | 0.380        | 0.298   |
| Colic         | 0.854        | 0.867   | 0.351        | 0.316   | 0.382        | 0.334   |
| Vowel         | 1.0          | 1.0     | 0.056        | 0.023   | 0.043        | 0.021   |
| Hypothyroid   | 0.965        | 0.981   | 0.110        | 0.085   | 0.065        | 0.050   |
| <b>W/L/T</b>  | <b>7/0/2</b> |         | <b>9/0/0</b> |         | <b>9/0/0</b> |         |

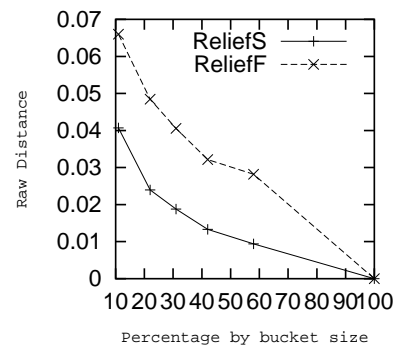
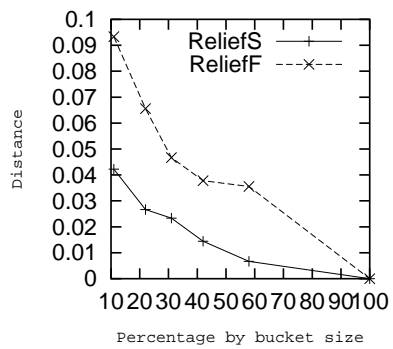
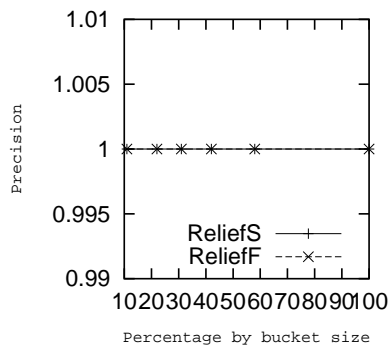
### 6.3.2 Results on non-numeric data

Based on previous results, it is clear that ReliefS works well on numeric data. We then experiment if ReliefS can be directly extended to non-numeric data (Groups 2 and 3 in Table 3). Since variance is calculated on numeric data when building the *kd*-tree, in our experiments we apply ReliefS to non-numeric data by associating a distinct number to a nominal value, e.g., assigning 1, 2 and 3 to nominal values A, B, and C, respectively.

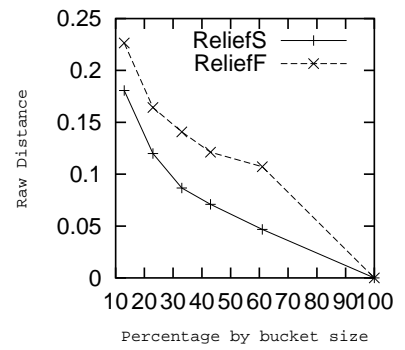
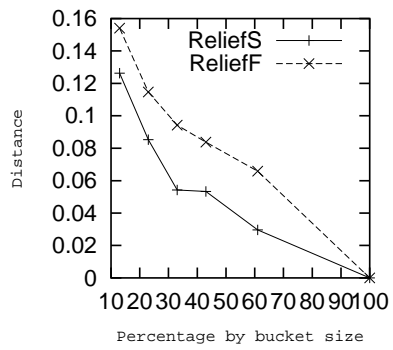
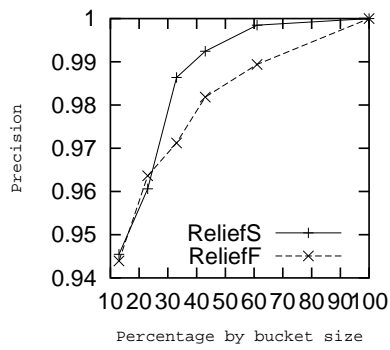
Results for the three performance measures are reported in Table 8 and Figure 8. From Table 8, we still notice that for each data set, ReliefS is better than or as good as ReliefF. This suggests that active feature selection works for non-numeric data as well. However, by comparing the results in Table 8 with Table 6, we observe that both ReliefF and ReliefS generally work better on numeric data than on non-numeric data. In addition, the performance gains obtained by ReliefS on non-numeric data are not as significant as those on numeric data, especially for small data sets (this can also be observed in Figure 8). Through three illustrative data sets from Groups 2 and 3, Figure 8 demonstrates similar trends of performance measures as those in Figure 7 when the number of instances increases for both ReliefF and ReliefS. Take Autos data for example, ReliefS performs better on all the three measures than ReliefF, but as we can see that the two curves for



**Autos Data**



**Vowel Data**



**KRKPA7 Data**

Figure 8: Three illustrative **non-numeric** data sets: Average results of 30 runs in three performance measures (P, D, RD).

Table 9: Time savings by ReliefS w.r.t. ReliefF for **non-numeric** data.  $m_{ReliefS}$  and  $m_{ReliefF}$  are numbers of instances used by ReliefS and ReliefF to achieve the same performance in raw distance.

|               | ReliefS               |                       |               | ReliefF               |               |
|---------------|-----------------------|-----------------------|---------------|-----------------------|---------------|
|               | $T_{kd-tree}$<br>(ms) | $T_{ReliefS}$<br>(ms) | $m_{ReliefS}$ | $T_{ReliefF}$<br>(ms) | $m_{ReliefF}$ |
| Breast-cancer | 90                    | 234                   | 123           | 300                   | 166           |
| Primary-tumor | 175                   | 546                   | 112           | 836                   | 176           |
| KRKPA7        | 4559                  | 84880                 | 1055          | 145520                | 1950          |
| Mushroom      | 7668                  | 248336                | 1950          | 491665                | 4062          |
| Zoo           | 50                    | 73                    | 40            | 105                   | 65            |
| Autos         | 161                   | 417                   | 82            | 568                   | 123           |
| Colic         | 303                   | 1047                  | 151           | 1518                  | 221           |
| Vowel         | 484                   | 2633                  | 218           | 6570                  | 574           |
| Hypothyroid   | 4950                  | 98094                 | 1094          | 173950                | 2112          |

each measure are very close to each other, which suggests that the performance gains obtained by selecting  $m$  instances using  $kd$ -trees are not significant.

Table 9 records the running times  $T_{kd-tree}$ ,  $T_{ReliefS}$  and  $T_{ReliefF}$  as well as  $m_{ReliefS}$  and  $m_{ReliefF}$  for data sets in Groups 2 and 3. From this table, we observe similar results: (1) the larger the data set, the more savings in time; and (2) the ratio of  $T_{kd-tree}/T_{ReliefS}$  decreases when data size increases. However, five small data sets out of the nine do not show significant time savings. This observation is consistent with what we saw in Table 8 and Figure 8. For these data sets, selective sampling using ReliefS does not significantly reduce the number of instances required to achieve a given level of performance when compared to random sampling using ReliefF. Thus the extra time spent on building a  $kd$ -tree is not compensated by the modest time savings obtained from using fewer instances.

## 6.4 Examining learning accuracy

In this section, we first show how ReliefF using all instances as suggested by [32] affects the learning accuracy, and then examine how the reduction of training instances affect the results of feature selection in terms of learning accuracy. The

Table 10: Effectiveness of ReliefF on benchmark data: 10-fold cross validation accuracy (%) of 5-NN on original features (full sets without feature selection) and target sets of features chosen by ReliefF (with  $m = N$ ). P reports the probability associated with a Student’s paired two-tailed t-Test.

|              | Full set     | Target Set<br>( $m = N$ ) | P           |
|--------------|--------------|---------------------------|-------------|
| Iris         | 96.67        | 97.33                     | 0.32        |
| Glass        | 65.80        | 71.95                     | <b>0.07</b> |
| WDBC         | 96.67        | 95.43                     | 0.15        |
| Balance      | 88.00        | 77.42                     | <b>0.00</b> |
| Pima-Indian  | 74.49        | 73.84                     | 0.69        |
| Vehicle      | 69.97        | 69.27                     | 0.66        |
| German       | 71.00        | 72.50                     | 0.19        |
| Segment      | 95.80        | 95.76                     | 0.94        |
| Abalone      | 53.72        | 52.79                     | 0.17        |
| Satimage     | 90.73        | 86.16                     | <b>0.00</b> |
| Waveform     | 79.20        | 83.74                     | <b>0.00</b> |
| Page-Blocks  | 95.87        | 93.92                     | <b>0.00</b> |
| CoIL2000     | 93.63        | 93.89                     | <b>0.07</b> |
| Shuttle      | 99.71        | 99.60                     | <b>0.01</b> |
| <b>W/L/T</b> | <b>3/4/7</b> |                           |             |

fi rst set of comparisons is shown in Table 10 and the second set of comparisons is shown in Table 11.

In Section 6.2, we have demonstrated the effectiveness of ReliefF using all instances on a group of synthetic data sets for which we know the relevant features in advance. We now examine the effectiveness of ReliefF using all instances on benchmark data through a learning algorithm. Table 10 records the 10-fold cross validation results of the 5-NN (nearest neighbor) classifier on the full sets of features and the target sets of features chosen by ReliefF with all instances (defined in Section 5.1) for the 14 numeric data sets (shown in Table 3). In order to evaluate the statistical significance of the difference between the two averaged accuracy values for a given data set, a Student’s paired two-tailed t-Test is conducted for the two underlying samples of individual accuracy values. The P value

Table 11: Comparison of ReliefF and ReliefS on benchmark data: 10-fold cross validation accuracy (%) of 5-NN on target sets of features chosen by ReliefF (with  $m = N$ ), subsets of features chosen by ReliefF (with  $m \approx \frac{1}{10}N$ ), and subsets of features chosen by ReliefS (with  $m \approx \frac{1}{10}N$ ). P reports the probability associated with a Student's paired two-tailed t-Test.

|              | A           | B                             | C                             | P            |               |              |
|--------------|-------------|-------------------------------|-------------------------------|--------------|---------------|--------------|
|              | Target Set  | ReliefF                       | ReliefS                       |              |               |              |
|              | ( $m = N$ ) | ( $m \approx \frac{1}{10}N$ ) | ( $m \approx \frac{1}{10}N$ ) | (A,B)        | (A,C)         | (B,C)        |
| Iris         | 97.33       | 97.33                         | 97.33                         | 1            | 1             | 1            |
| Glass        | 71.95       | 65.84                         | 67.23                         | <b>0.01</b>  | <b>0.04</b>   | 0.40         |
| WDBC         | 95.43       | 95.43                         | 96.71                         | 0.99         | <b>0.09</b>   | <b>0.08</b>  |
| Balance      | 77.42       | 76.97                         | 77.42                         | 0.79         | 1             | 0.79         |
| Pima-Indian  | 73.84       | 65.61                         | 72.27                         | <b>0.02</b>  | 0.21          | <b>0.04</b>  |
| Vehicle      | 69.27       | 66.01                         | 69.16                         | <b>0.04</b>  | 0.92          | <b>0.09</b>  |
| German       | 72.50       | 70.60                         | 72.60                         | <b>0.06</b>  | 0.94          | <b>0.06</b>  |
| Segment      | 95.76       | 95.84                         | 96.19                         | 0.44         | 0.25          | 0.35         |
| Abalone      | 52.79       | 52.79                         | 54.51                         | 1            | <b>0.04</b>   | <b>0.04</b>  |
| Satimage     | 86.16       | 85.75                         | 86.11                         | 0.18         | 0.84          | 0.27         |
| Waveform     | 83.74       | 83.48                         | 83.38                         | 0.49         | 0.36          | 0.83         |
| Page-Blocks  | 93.92       | 91.17                         | 89.73                         | <b>0.00</b>  | <b>0.00</b>   | <b>0.00</b>  |
| CoIL2000     | 93.89       | 93.77                         | 93.77                         | 0.15         | 0.15          | 1            |
| Shuttle      | 99.60       | 99.60                         | 99.60                         | 1            | 1             | 1            |
| <b>W/L/T</b> |             |                               |                               | <b>0/5/9</b> | <b>2/2/10</b> | <b>5/1/8</b> |

in each row of Table 10 reports the probability that the two underlying samples are different. The smaller the P value, the more significant the difference of the two average values is. The last row (W/L/T) summarizes Win/Loss/Tie in comparing the averaged accuracy values on the target sets with those on the full sets based on a significance threshold 0.1. It is clear that out of the 14 data sets, seven pairs of results are significantly different. ReliefF significantly improves the accuracy of the 5-NN classifier for 3 data sets and maintains the accuracy for 7 data sets with selected subsets of features.

To examine the effect of reduction of training instances and verify the effectiveness of ReliefS, for each data set, two feature subsets of the same cardinality are selected from the top of the two resulting lists produced by ReliefF and ReliefS with the same number of sampled instances (we choose  $m \approx 10\%N$ , corresponding to the smallest bucket size, in our experiments). Table 11 records the 10-fold cross validation results of 5-NN on the subsets of features chosen by ReliefF (column B) or ReliefS (column C). We use the results on the target sets of features chosen by ReliefF with all instances ( $m = N$ ) as the reference point (column A) in comparison of ReliefF and ReliefS. As discussed in Section 5.1, for feature ranking methods like ReliefF, the order of features in a resulting list is important for subset selection. However, learning accuracy on a selected subset may not be sensitive to the order of features; as long as two subsets contain the same features (having equal Precision values), they can result in the same learning accuracy for a given learning algorithm. Therefore, in order to show the effect of the different orders of features, we choose two subsets of different features of cardinality  $n$  from the top of the two resulting feature lists to obtain the accuracy for 5-NN.

Table 11 also contains P values resulting from pair-wised comparisons of columns A, B, and C. P (A,B) values for columns A and B suggest that there are 5 pairs of average accuracy rates that are significantly different and the subsets selected by ReliefF with  $m \approx \frac{1}{10}N$  cause accuracy decrease in all 5 cases. P (A,C) values for columns A and C show that there are 4 pairs of average accuracy rates that significantly different, and the subsets selected by ReliefS with  $m \approx \frac{1}{10}N$  result in accuracy increase for 2 data sets and decrease for 2 data sets among the 4 cases. We then further compare the 5-NN results using the feature subsets of ReliefS and ReliefF with  $m \approx \frac{1}{10}N$  directly. P (B,C) values for columns B and C indicate that there are 6 pairs of average accuracy rates that are significantly different. It is clear that the subsets selected by ReliefS with  $m \approx \frac{1}{10}N$  result in better accuracy than the subsets selected by ReliefF with  $m \approx \frac{1}{10}N$  for 5 data sets and worse accuracy for only one data set among the 6 cases. According to the above results, we conclude that with the same number of sampled instances,

ReliefS in general achieves better performance than ReliefF in terms of learning accuracy and hence selective sampling is an effective approach for active feature selection.

## 7 Conclusions and Further Work

In this paper, we present a case for active feature selection using a formalism of selective sampling. We choose an efficient feature selection algorithm ReliefF in our case study to evaluate whether selective sampling has consistent advantages over random sampling. In particular, we use the *kd*-tree to partition data and select instances from the partitions. We conduct extensive experiments to evaluate the performance of active feature selection. Significant time savings are observed using the Raw Distance performance measure. Improvement of learning accuracy is reported for the nearest neighbor classifier on numeric data.

Although the experimental study demonstrates the effectiveness of active feature selection, we plan future work along the following lines: (1) to investigate why ReliefS still works in cases where data is not purely numeric and explore different methods of handling nominal features; (2) to automatically determine the cost-effective percentage of instances for selective sampling and investigate its performance on large data sets (e.g., Web data); (3) to investigate other means of exploiting data characteristics for selective sampling; and (4) to apply selective sampling to the vast body of feature selection and other data preprocessing algorithms [38].

## Acknowledgments

We thank Bret Ehlert, Feifang Hu, Manoranjan Dash, Hongjun Lu, and Lance Parsons for their contributions to this work. We are grateful to the anonymous reviewers who have provided many helpful and constructive suggestions on an earlier version of this paper. An earlier short version of this work was published in the proceedings of the 19th International Conference on Machine learning, 2002. This work is in part based on the project supported by National Science Foundation under Grant No. IIS-0127815 for H. Liu, and on Grant-in-Aid for Scientific Research on Priority Areas (B), No. 759: Active Mining Project by Ministry of Education, Culture, Sports, Science and Technology of Japan for H. Motoda.

## References

- [1] R. Agrawal, T. Imielinski, and A. Swami. Database mining: A performance perspective. *IEEE Trans. on Knowledge and Data Engineering*, 5(6):914–925, December 1993.
- [2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. Int. Conf. Very Large Data Bases*, pages 487–499, Santiago, Chile, September 1994.
- [3] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.
- [4] S. D. Bay. The UCI KDD archive, 1999. <http://kdd.ics.uci.edu>.
- [5] C. L. Blake and C. J. Merz. UCI Repository of machine learning databases, 1998. <http://www.ics.uci.edu/~mlearn/MLRepository.html>.
- [6] A.L. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97:245–271, 1997.
- [7] P.S. Bradley, U. Fayyad, and C. Reina. Scaling clustering algorithms to large databases. In *Proceedings of the Fourth International Conference on Knowledge Discovery & Data Mining*, pages 9 – 15. AAAI PRESS, California, 1998.
- [8] C.J.C. Burges. A tutorial on support vector machines. *Data Mining and Knowledge Discovery*, 2, 1998.
- [9] W.G. Cochran. *Sampling Techniques*. John Wiley & Sons, 1977.
- [10] D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*, 15:201 – 221, 1994.
- [11] S. Das. Filters, wrappers and a boosting-based hybrid for feature selection. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 74–81, 2001.
- [12] M. Dash and H. Liu. Feature selection for classification. *Intelligent Data Analysis: An International Journal*, 1(3):131–156, 1997.



- [13] M. Dash and H. Liu. Feature selection for clustering. In *Proceedings of the Fourth Pacific Asia Conference on Knowledge Discovery and Data Mining, (PAKDD-2000)*. Kyoto, Japan, pages 110–121. Springer-Verlag, 2000.
- [14] M. Dash, H. Liu, and H. Motoda. Consistency based feature selection. In *Proceedings of the Fourth Pacific Asia Conference on Knowledge Discovery and Data Mining, (PAKDD-2000)*. Kyoto, Japan, pages 98–109. Springer-Verlag, 2000.
- [15] M. Dash, H. Liu, and J. Yao. Dimensionality reduction of unsupervised data. In *Proceedings of the Ninth IEEE International Conference on Tools with AI (ICTAI'97), November, 1997*, pages 532–539, Newport Beach, California, 1997. IEEE Computer Society.
- [16] T.G. Dietterich. Machine learning research: Four current directions. *AI Magazine*, pages 97–136, Winter 1997.
- [17] J. G. Dy and C. E. Brodley. Feature subset selection and order identification for unsupervised learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 247–254, 2000.
- [18] J. G. Dy and C. E. Brodley. Visualization and interactive feature selection for unsupervised data. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 360–364, 2000.
- [19] Y. Freund, H. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28:133–168, 1997.
- [20] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Software*, 3:209–226, 1977.
- [21] V. Gaede and O. Günther. Multidimensional access methods. *ACM Computing Surveys*, 30(2):170–231, 1998.
- [22] B. Gu, F. Hu, and H. Liu. *Sampling: Knowing Whole from Its Part*, pages 21–38. In Liu and Motoda [38], 2001.
- [23] M.A. Hall. *Correlation Based Feature Selection for Machine Learning*. PhD thesis, University of Waikato, Dept. of Computer Science, 1999.

- [24] M.A. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-00)*. Morgan Kaufmann Publishers, 2000.
- [25] Se June Hong. Use of contextual information for feature ranking and discretization. *IEEE Transactions on Knowledge and Data Engineering*, 9(5):718–730, 1997.
- [26] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In C. Nedellec and C. Rouveirol, editors, *Proceedings of 10th European Conference on Machine Learning*, pages 137 – 142, Chemnitz, Germany, 1998. Springer.
- [27] Y. Kim, W. Street, and F. Menczer. Feature selection for unsupervised learning via evolutionary search. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 365–369, 2000.
- [28] K. Kira and L.A. Rendell. The feature selection problem: Traditional methods and a new algorithm. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 129–134. Menlo Park: AAAI Press/The MIT Press, 1992.
- [29] K. Kira and L.A. Rendell. A practical approach to feature selection. In Sleeman and P. Edwards, editors, *Proceedings of the Ninth International Conference on Machine Learning (ICML-92)*, pages 249–256. Morgan Kaufmann, 1992.
- [30] J. Kivinen and H. Mannila. The power of sampling in knowledge discovery. In *SIGMOD/PODS' 94*, pages 77 – 85. ACM, 1994.
- [31] R. Kohavi and G.H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.
- [32] I. Kononenko. Estimating attributes : Analysis and extension of RELIEF. In F. Bergadano and L. De Raedt, editors, *Proceedings of the European Conference on Machine Learning, April 6-8*, pages 171–182, Catania, Italy, 1994. Berlin: Springer-Verlag.

- [33] I. Kononenko, E. Simec, and M. Robnik-Sikonja. Overcoming the myopia of inductive learning algorithms with RELIEFF. *Applied Intelligence*, 7:39–55, 1997.
- [34] P. Langley. Selection of relevant features in machine learning. In *Proceedings of the AAAI Fall Symposium on Relevance*, pages 140–144. AAAI Press, 1994.
- [35] E. Leopold and Kindermann J. Text categorization with support vector machines. how to represent texts in input space? *Machine Learning*, 46:423–444, 2002.
- [36] D. Lewis and W. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the Seventeenth Annual ACM-SIGR Conference on Research and Development in Information Retrieval*, pages 3 – 12, 1994.
- [37] H. Liu and H. Motoda. *Feature Selection for Knowledge Discovery and Data Mining*. Boston: Kluwer Academic Publishers, 1998.
- [38] H. Liu and H. Motoda, editors. *Instance Selection and Construction for Data Mining*. Boston: Kluwer Academic Publishers, 2001.
- [39] H. Liu, H. Motoda, and L. Yu. Feature selection with selective sampling. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 395 – 402, 2002.
- [40] T.M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
- [41] A. W. Moore. An introductory tutorial on kd-trees. Extract from Ph.D. Thesis Tech Report No. 209, Computer Laboratory, University of Cambridge, Robotics Institute, Carnegie Mellon University, 1991.
- [42] A. Y. Ng. On feature selection: learning with exponentially many irrelevant features as training examples. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 404–412, 1998.
- [43] K.S. Ng and H. Liu. Customer retention via data mining. *AI Review*, 14(6):569 – 590, 2000.
- [44] K. Nigam, A. K. Mccallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39:103–134, 2000.

- [45] M. Robnik-Sikonja and I. Kononenko. An adaptation of relief for attribute estimation in regression. In *Proceedings of Fourteenth International Conference on Machine Learning*, pages 296–304, 1997.
- [46] M. Robnik-Sikonja and I. Kononenko. Attribute dependencies, understandability and split selection in tree based models. In *Proceedings of Sixteenth International Conference on Machine Learning*, pages 344–353, 1999.
- [47] M. Robnik-Sikonja and I. Kononenko. Comprehensible interpretation of relief’s estimates. In *Proceedings of Eighteenth International Conference on Machine Learning*, pages 433–440, 2001.
- [48] M. Robnik-Sikonja and I. Kononenko. Theoretical and empirical analysis of Relief and ReliefF. *Machine Learning*, 53:23–69, 2003.
- [49] N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001.
- [50] G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 839–846, 2000.
- [51] M.R. Sikonja. Speeding up Relief algorithms with k-d trees. In *Proceedings of the Electrotechnical and Computer Science Conference ERK’98*, 1998.
- [52] N.A. Syed, H. Liu, and K.K. Sung. A study of support vectors on model independent example selection. In S. Chaudhuri and D. Madigan, editors, *Proceedings of ACM SIGKDD, International Conference on Knowledge Discovery and Data Mining*, pages 272 – 276, New York, NY, 1999. ACM.
- [53] L. Talavera. Feature selection as a preprocessing step for hierarchical clustering. In *Proceedings of International Conference on Machine Learning (ICML’99)*, pages 389–397, 1999.
- [54] C.A. Thompson, M.E. Califf, and R.J. Mooney. Active learning for natural language parsing and information extraction. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 406–414. Morgan Kaufmann, 1999.

- [55] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Machine Learning Research*, 2:45–66, 2001.
- [56] I.H. Witten and E. Frank. *Data Mining - Practical Machine Learning Tools and Techniques with JAVA Implementations*. Morgan Kaufmann Publishers, 2000.
- [57] E. Xing, M. Jordan, and R. Karp. Feature selection for high-dimensional genomic microarray data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001.
- [58] L. Yu and H. Liu. Feature selection for high-dimensional data: a fast correlation-based filter solution. In *Proceedings of the twentieth International Conference on Machine Learning*, pages 856–863, 2003.