Feature Selection with Selective Sampling

Huan Liu HLIU@ASU.EDU
Department of Computer Science & Engineering, Arizona State University, Tempe, AZ 85287-5406, USA

Hiroshi Motoda MOTODA@SANKEN.OSAKA-U.AC.JP
Institute of Scientific & Industrial Research, Osaka University, Ibaraki, Osaka 567-0047, Japan

Lei Yu LEIYU@ASU.EDU
Department of Computer Science & Engineering, Arizona State University, Tempe, AZ 85287-5406, USA

Abstract

Feature selection, as a preprocessing step to machine learning, has been shown very effective in reducing dimensionality, removing irrelevant data, increasing learning accuracy, and improving comprehensibility. In this paper, we consider the problem of active feature selection in a filter model setting. We describe a formalism of active feature selection called *selective sampling*, demonstrate it by applying it to a widely used feature selection algorithm *Relief*, and show how it realizes active feature selection and reduces the required number of training data for *Relief* to achieve time savings without performance deterioration. We design objective evaluation measures, conduct extensive experiments using bench-mark data sets, and observe consistent and significant improvement.

1. Introduction

Inductive learning is one of the major approaches to automatic extraction of useful patterns (or knowledge) from massive data. Data become increasingly larger in both columns (i.e., number of features) and rows (i.e., number of instances) in many applications such as genome projects, text mining, customer relationship management, and market basket analysis. This trend poses a severe challenge to inductive learning systems in terms of efficiency and effectiveness. Feature selection has been shown its impressive performance gains in attacking large dimensionality with many irrelevant features (Langley, 1994; Liu & Motoda, 1998; Dy & Brodley, 2000; Dash & Liu, 2000). In particular, feature selection can remove irrelevant features (and in many cases, render the impossible learning possible), increase efficiency in the learning task, improve learning performance like predictive accuracy, and enhance comprehensibility of learned results (Talavera, 1999). Although there exist numerous feature selection algorithms (Blum & Langley, 1997; Dash & Liu, 1997; Hall, 1999), 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 about a huge number of instances. Sampling is a common approach to this problem. Sampling is both random and blind. In this work, we explore the possibility if we can take advantage of the recent development in active learning (Roy & McCallum, 2001) to improve the performance of feature selection without increasing the number of sampled data points. In the following, we first review the models of feature selection and explain why a filter model is good for our consideration - handling a huge number of instances, then introduce active feature selection.

Feature selection algorithms broadly fall into the filter model or the wrapper model (Kohavi & John, 1997; Liu & Setiono, 1996). The filter model relies on general characteristics of the training data to select some features without involving any learning algorithm. The wrapper model requires one predetermined learning algorithm in feature selection and uses the performance of the learning algorithm to evaluate and determine which features are selected. As for each new subset of features, the wrapper model needs to learn a hypothesis (or a classifier), it tends to give superior performance as it finds features better suited to the predetermined learning algorithm, but it also tends to be more computationally expensive (Langley, 1994). When the number of instances becomes very large, the filter model is usually a choice due to its computational efficiency and neutral bias toward any learning algorithm. In the context of this work, we focus on the filter model. A representative example of the filter model is *Relief* - a widely used, very efficient feature selection algorithm (Kira & Rendell, 1992; Kononenko, 1994; Witten & Frank, 2000). Its details will be given in Section 2.1.

Active feature selection is inspired by the workings of active learning (Roy & McCallum, 2001). It is different from traditional supervised learning in which the learner has the freedom to select which data instances are added to its training set. The essence of active learning lies in its control over the choice of instances used in the training process. **Active feature selection** shares this essential character with active learning that it can influence what instances are used for feature selection by exploiting some characteristics of the data. Since a filter model is adopted in this work, we cannot employ any learning algorithm to actively choose instances. Therefore, the problem of active feature selection boils down to how we can actively choose useful instances for feature selection.

In section 2, we illustrate active feature selection using *Relief* with implementation details. In section 3, we design objective metrics for performance evaluation including time savings. Sectin 4 is an empirical study in which we evaluate the gains of active feature selection via extensive experiments and discuss the implications of the findings. Section 5 presents a conclusion with some work to be conducted in the near future.

2. Active Feature Selection for Relief

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 overcome the problem. Active feature selection avoids pure random sampling and is realized by selective sampling. The intuitive idea is to select only instances with higher probabilities to be informative in determining feature relevance.

As mentioned earlier, efficiency is critical for feature selection algorithms. To demonstrate the working of active feature selection, we use a well received, efficient algorithm *Relief* which can select statistically relevant features in linear time in the number of features and the number of instances. After introducing *Relief*, we illustrate how selective sampling can be approximated and implemented with a data structure (kd-trees (Friedman et al., 1977)), and present a new algorithm of active feature selection.

2.1 Relief

The key idea of *Relief* (given in Figure 1) is to estimate the quality of features according to how well their values distinguish between the instances that are near to each other. For this purpose, given a randomly selected instance X from a data set $\mathbf{S}$ with k attributes, *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 $\mathrm{W}[A_i]$ for all the features A_i depending on the difference $diff()$ on their values for X, M, and H. The process is repeated for m times, where m is a user-defined parameter (Kononenko, 1994; Kira & Rendell, 1992).

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

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

Figure 1. Original Relief algorithm.

The time complexity of *Relief* for a data set with N instances is $O(mkN)$. Clearly, efficiency is one of the major advantages of the *Relief* family over other algorithms. With m being a constant, the time complexity becomes $O(kN)$. However, since m is the number of instances for approximating probabilities, a larger m implies more reliable approximations. When N is very large, it often requires that $m << N$. The m instances are chosen randomly in *Relief*. Given a small constant m, we ask if by active feature selection, we can improve approximations to close to those using N instances.

2.2 Approximating selective sampling using kd-trees

The key to active feature selection is being able to choose m informative instances in the context of *Relief*. Without any prior knowledge about data, selective sampling is no different from random sampling for *Relief*. The idea of selective sampling stems from the fact that instances are not uniformly distributed and some instances are more representative than others (Aha et al., 1991). If one could find representative instances, only a small number of instances are needed.

The issue is how one can achieve that without accessing to a learning algorithm.

A *kd*-tree is an index structure often used for fast nearest neighbor search (Moore, 1991). It is a generalization of the simple binary tree, which uses k features instead of a single feature. Its interior nodes have an associated feature A_i and a value V that splits the data points into two parts: those with A_i-value less than V and those with A_i-value equal to or greater than V. The features at different levels of the tree are different, with levels rotating among the features of all dimensions. The root of the tree presents all the instances and the splitting is done recursively in each of the successor until the node contains no more than a predefined number of instances (called bucket size). The leaf nodes (buckets) represent mutually exclusive small subsets of instances which collectively form a partition of the whole data set. Which feature is chosen first to split can result in different *kd*-trees. For the purpose of selective sampling, we want to split data into different groups as early as possible. Hence, in our building a *kd*-tree, a feature is chosen first if it can maximize the data variance along that dimension. A median value of that feature is used to split the instances into two equal sized partitions. Such a *kd*-tree is a balanced tree and its leaf is called a bucket. The size of a bucket can be determined *a priori*. Since instances in each bucket are relatively close to each other, in selective sampling, one instance is randomly picked from each bucket. The time complexity of building such an optimized *kd*-tree is $O(kN\log N)$ where N is the total number of instances. The *kd*-tree can be built once and for all if necessary or dynamically when required. It is shown in (Sikonja, 1998) that *kd*-trees can speed up Relief algorithms. The focus of this work is to show the use of *kd*-trees in selective sampling.

2.3 Relief with selective sampling

Recall that the value of m is critical in approximating probabilities in *Relief*. Our first attempt is to selectively sample m instances according to the *kd*-tree. There are two ways of controlling the *kd*-tree splitting process if one does not want the bucket size to be 1. One is to use a given m, and the other is to use a predetermined bucket size t. One can easily establish that $t = N/m$. Therefore, if m is known, the splitting process stops when we reach the level where the bucket contains N/m or less instances. On the other hand, given a t, we have $m = N/t$. For instance, if we choose t to be 1 (i.e., each bucket contains only one instance), all the instances in the data set will be selected for *Relief*. If we increase t to 2, then all the buckets used for sampling will have no more than two instances, and the number of selected instances m will be $0.5N \le m \le N$. For a bucket size 3, m will be $0.33N \le m \le 0.5N$. For a bucket size $t = N$, the root node is not split at all, only one instance will be selected. From each bucket whose size is greater than 1, one instance is randomly selected.

In this work, we use bucket size t to control the number of selected instances m which is the number of buckets. *Relief* with selective sampling is referred as ReliefS detailed in Figure 2. One important point in `buildKDTree(t)` is that each feature should be normalized before the variance calculation in order to choose the feature with largest variance to split.

Given t - bucket size,

```
set all weights W[A_i] := 0.0;
buildKDTree(t);
m := number of buckets;
for j := 1 to m do begin
   randomly select an instance X ∈ Bucket[j];
   find nearest hit H and nearest miss M;
   for i := 1 to k do begin
      W[A_i] := W[A_i] − diff(A_i, X, H)/m
                   +diff(A_i, X, M)/m;
   end;
end;
```

Figure 2. Algorithm ReliefS with selective sampling.

To recap, *Relief* basically relies on the search of a predefined number of nearest neighbors (Kononenko, 1994), and the *kd*-tree data structure is designed for fast nearest neighbor search. The *kd*-tree divides the sample space into strata so that m instances can be selected from these strata. In the following, we will conduct an empirical study to exam if ReliefS can achieve what it is designed for. We use *ReliefF* (Kononenko, 1994; Witten & Frank, 2000) which extends *Relief* in many ways: it searches for several nearest neighbors to be robust to noise, and handles multiple classes.

3. Issues of Performance Evaluation

Let us first discuss the criteria for the performance metrics. Any performance measure should satisfy the following for *ReliefF*:

R1. Its value reaches the best when $m = N$.

R2. Its value improves as m increases.

R3. It is a function of the features of the data.

It is obvious that the best performance we can achieve is the features ranked by *ReliefF* with $m = N$. This ranked list of features is named $S_{F,N}$. The performance of ReliefS w.r.t. *ReliefF* can be measured in two aspects: (1) given various sizes of m and the subsets $S_{F,m}$ and $S_{S,m}$ selected by *ReliefF* and ReliefS respectively, we compare which subset ($S_{F,m}$ or $S_{S,m}$) is more similar to $S_{F,N}$ - the best subset by *ReliefF*; and (2) we compare whether features of the subsets $S_{F,m}$ and $S_{S,m}$ are in the same order of the features in $S_{F,N}$. Aspect 1 of performance measure is designed for feature subset selection; aspect 2 of performance measure is for feature ranking and basically it concerns the two lists of ReliefS and *ReliefF* with $m < N$ in comparison with *ReliefF* of $m = N$. We present next four different measures and check if they satisfy the above 3 requirements. In section 3.2, we discuss how to measure time savings due to active feature selection.

3.1 Performance measures

The optimal ranking of the features can be obtained by running *ReliefF* with all the instances ($m = N$) according to the weights $W[A_i]$. The optimal set (list) of features is $S_{F,N}$. Then a target subset of features T is an optimal subset of features which contains the top n weighted features in $S_{F,N}$. For a data set, an optimal subset of features is the top n features whose weights $\geq \gamma$, where γ is a threshold equal to $W[i]$ (the i-th largest weight in $S_{F,N}$ and the gap defined by $W[i]$ and $W[i+1]$ is sufficiently large (e.g., greater than the average gap among $k-1$ gaps).

Let S_F and S_S be the subsets of n features 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 $\mathcal{P}(T, R)$ where R can be either $S_{F,m}$ or $S_{S,m}$ with varying size m. We examine below what are the sensible candidates for $\mathcal{P}()$.

Precision

Precision is computed as the number of features in T that are also in R, normalized by dividing the number of features in R:

$$\frac{|\{x : x \in R \wedge x \in T\}|}{|R|}.$$

The value of Precision ranges from 0 to 1, where the value of 1 is achieved for perfectly matched sets.

Weighted Precision

Precision treats all features in T equally without considering the ordering of the features. To overcome this drawback, we develop the weighted precision measure. First, we assign each feature $A_i (1 \leq i \leq k)$ in the optimal sequence $S_{F,N}$ another weight $w_{A_i} = k - i + 1$(where k is the total number of features in the data set, and i is the rank of the feature A_i in $S_{F,N}$). Then the weighted precision is computed as:

$$\frac{\sum_{\forall A_j \in R} w_{A_j}}{\sum_{\forall A_i \in T} w_{A_i}}.$$

The weighted precision measure ranges from $(n+1)/(2k-n+1)$ to 1, where the value becomes the upper bound (1) when Precision is 1, and the lower bound is reached when the n features in R are the n features with the least ranking values from the optimal sequence $S_{F,N}$. Using weighted precision, we are able to evaluate the closeness between R and T by looking at both the number of matched features of the two subsets and the orders of the two subsets.

Distance Measure

Another way of considering the orders of the features in the two subsets is named Distance Measure (DM) which is the sum of distances of the same features in R and T. The distance of a feature between two sets is the difference of their positions in the ranking. Let T' be T in reverse order and S' be the set of k features whose top n features form R (n is determined by γ) using all k features. The maximum possible ranking distance between two sets T and T' that share the same features is:

$$D_{max} = \sum_{\forall A_i \in T} |position(A_i \in T) - position(A_i \in T')|.$$

Because subset R may not contain all the features in T, DM is given using S' as follows:

$$\frac{\sum_{\forall A_i \in T} |position(A_i \in T) - position(A_i \in S')|}{D_{max}}$$

D_{max} is used to normalize Distance measure. For sets T and R with Precision = 1, DM ranges from 0 to 1, where DM of 0 is achieved for the two sets having identical ranking, and DM of 1 is achieved for the two sets that are in absolutely reverse order.

Raw Distance

A straightforward performance measure is to directly calculate the sum of the difference of weights for each of the same feature in the optimal sequence $S_{F,N}$ and the sequence R. We name it Raw Distance (RD):

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

Metrics	Precision	Weighted Precision	Distance	Raw Distance
Complexity	$O(n^2)$	$O(nk)$	$O(nk)$	$O(k^2)$
Upper bound	1	1	1	None[1]
Lower bound	0	$(n+1)/(2k-n+1)$	0	0
Sensitivity	None	Low	High	High
γ setting	Yes	Yes	Yes	No

Table 1. Summary of Performance Measures.

where $W_S[A_i]$ and $W_R[A_i]$ are associated with $S_{F,N}$ and R, respectively. RD includes all the k features in the two sets. Thus, this measure avoids choosing a threshold for γ. When it is used for comparing the results of *ReliefF* and ReliefS, it serves the purpose well, although it cannot be used for measuring the performance of subset selection as it uses all features.

Table 1 provides a summary of these four metrics. Sensitivity is about how sensitive the measure is toward the order of features in a subset. γ setting is required in *Relief* for feature subset selection. Only Raw Distance does not require this threshold setting. Since the Distance measure is more sensitive than Weighted Precision, we choose Distance along with Precision and Raw Distance as performance measures in our experiments. Precision is a good and simple measure for feature subset selection where the order of features is not a concern.

3.2 Measuring time savings

A natural question is whether the use of selective sampling in feature selection would result in any time saving. This is because the initial building of *kd*-trees incurs certain costs. This question is best answered by providing time measures. The comparison can be conducted as follows: one can choose a performance measure from the above; for a given performance, ReliefS and *ReliefF* require different numbers of instances (m) to achieve the performance, thus we can measure their running time. Let T_{kd}, T_S, and T_F be time for *kd*-tree building, time for running ReliefS, and time for running *ReliefF* with a given performance, respectively. We will report T_{kd}, T_S, and T_F and compare $T_{kd}+T_S$ with T_F, the difference is either the saving or the loss.

4. Empirical Study

The objective of this section is to evaluate if active feature selection can, in the context of *ReliefF*, do better in selecting m instances. Since m is the number of instances used to approximating probabilities (seen in Figure 1), a larger m implies more reliable approximations. In (Kononenko, 1994), m is set to N to circumvent the issue of optimal m when many extensions of *ReliefF* are evaluated. In this work, however, we cannot assume that it is always possible to let $m = N$ as this would make the time complexity of *ReliefF* become $O(kN^2)$. In section 4.1, we describe the experimental setup and procedures of using real world data sets. In section 4.2, we present results and discussion of the experiments.

4.1 Experiment Setup

Title	N	Num	Nom	#C
Iris	150	4	0	3
Glass	214	9	0	7
Balance	625	4	0	3
Pima-Indian	768	8	0	2
Vehicle	846	18	0	4
Segment	2310	19	0	7
Waveform	5000	40	0	3
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 2. Summary of Benchmark Data Sets: N - number of instances, Num - numeric features, Nom - nominal features, #C - number of classes.

The experiments are conducted using Weka's implementation of *ReliefF* and ReliefS is also implemented in the Weka environment (Witten & Frank, 2000). All together 16 data sets are selected from the UCI Irvine machine learning data repository (Blake & Merz, 1998) with all nominal classes and varying data sizes. A summary of data sets is presented in Table 2. Three groups

	Precision		Distance		Raw Distance	
	ReliefF	ReliefS	ReliefF	ReliefS	ReliefF	ReliefS
Iris	1.0	1.0	0.050	0.006	0.057	0.016
Glass	0.980	0.992	0.158	0.097	0.078	0.046
Balance	0.861	0.931	0.433	0.265	0.040	0.016
Pima-Indian	0.902	0.917	0.277	0.214	0.020	0.014
Vehicle	0.997	1.0	0.205	0.105	0.055	0.026
Segment	1.0	1.0	0.082	0.030	0.060	0.016
Waveform	1.0	1.0	0.079	0.060	0.048	0.034
W/L/T	**0/4/3**		**0/7/0**		**0/7/0**	

Table 3. Average values for *ReliefF* and ReliefS of three different measures for continuous data.

of data are chosen. Group 1 contains only continuous data, Group 2 only nominal data, Group 3 mixed data. Each experiment is conducted as follows:

For each data set,

1. Run *ReliefF* using all the instances, and obtain the ranked list of features according to their weights, i.e., $S_{F,N}$. The parameter for k-nearest neighbor search is set to 5 (neighbors). This parameter remains the same for all the experiments.

2. Run *ReliefF* with increasing percentage of instances corresponding to bucket sizes from 7 to 1. At each percentage value, run *ReliefF* 30 times with different seeds and calculate measures Precision, Distance, and Raw Distance for each iteration to obtain average results for these performance measures to eliminate any idiosyncrasy in a single run. A curve is plotted for comparison purpose.

3. Run ReliefS with increasing percentage of instances. The percentage value is automatically decided by the decreasing bucket size from 7 to 1 for each data set as in Step 2. For each bucket size, run ReliefS 30 times and calculate Precision, Distance and Raw Distance each time and obtain average results as in Step 2 for *ReliefF*. A curve is plotted for comparison purpose.

4.2 Results and Discussions

Two sets of results are presented in Table 3 and Figure 3. Three performance measures of average results **for each data set** are reported in Table 3. They are averaged over results of different bucket sizes from 7 to 2 as *ReliefF* and *ReliefS* are the same when bucket size is 1. (W/L/T) in Table 3 summarizes Win/Loss/Tie in comparing *ReliefF* with ReliefS. For every data set, ReliefS is as good as or better than *ReliefF*. This suggests that instances selected using *kd*-tree are more effective than randomly chosen ones. Now let us look at the trends of the three performance measures when the number of instances increases for both *ReliefF* and ReliefS. Figure 3 shows the average results of all 7 data sets with different bucket sizes from 7 to 1. As the number of instances increases, all three measures improve and reach their best values when all instances are used in Relief. Figure 3 complements Table 3 in displaying results summarized along different aspects: numbers of instances and different data sets.

Figure 3 and Table 3 indicate that active feature selection has achieved improved results for continuous data. This is largely due to the use of *kd*-trees to partition the data. The *kd*-tree can be built once for all for each size of the data. After the *kd*-tree is built, we only need a maximum of $O(\log N)$ to reach a certain bucket of size t. Therefore, the subsequent run time of ReliefS is really the same as that of *ReliefF* after the *kd*-tree is built. To actually compare the running time of ReliefS with that of *ReliefF*, we follow section 3.2 and consdier the case of the smallest bucket size 2 (i.e., m_F is around 50% of the whole data for *ReliefF*) and use this performance (raw distance) to find corresponding m_S. Table 4 records the running times T_{kd}, T_S and T_F as well as m_S and m_F. We can observe that: (1) the time savings are consistent with the time complexity analysis of *Relief* - $O(kmN)$; (2) the larger the data set, the more savings in time; and (3) the ratio of T_{kd}/T_S decreases when data size increases.

Another interesting point to notice is that *ReliefF* works on both continuous, nominal and mixed data. Since variance is calculated in the *kd*-tree building, we naturally question if ReliefS can be directly extended to nominal and mixed data (Groups 2 and 3 in Table 2) by simply treating them as continuous. The results are reported in Table 5. ReliefS works as well as or better than *ReliefF* except for 3 cases (some particular bucket sizes for data sets PrimaryTumor, Zoo, Colic). The detailed re-

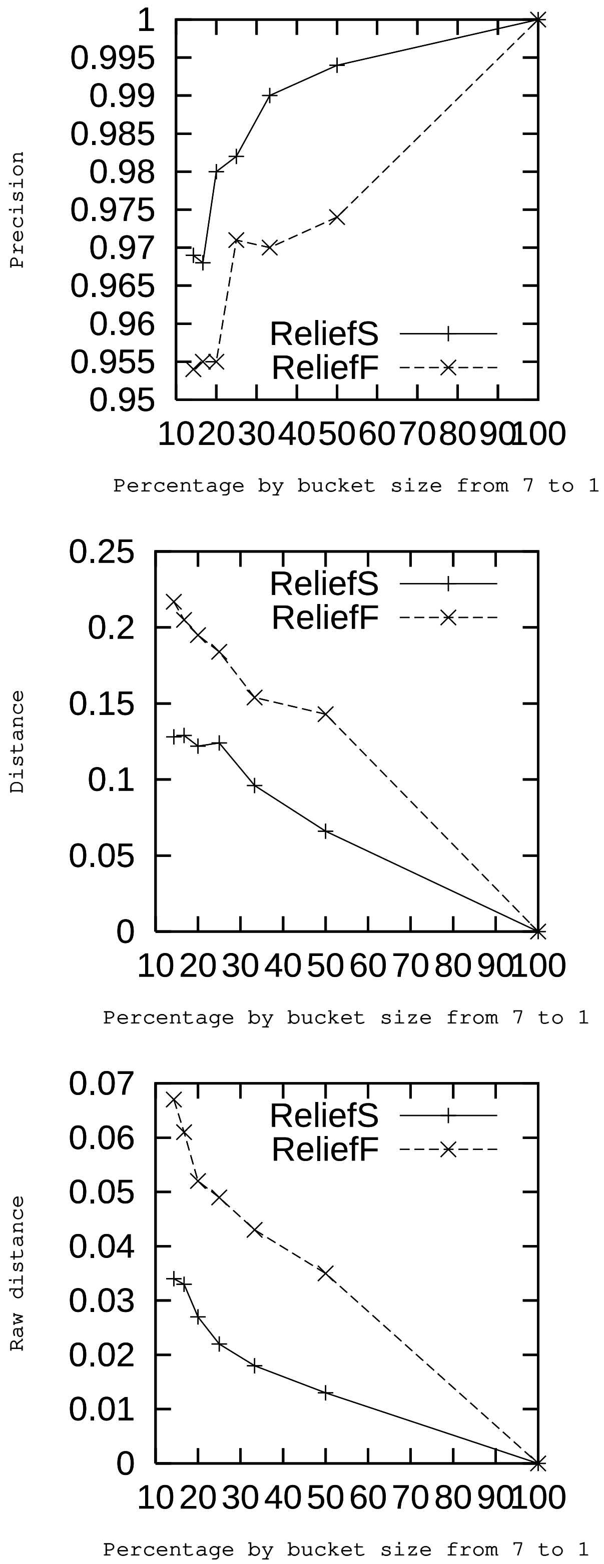

Figure 3. Average results of 3 performance measures on continuous data.

	ReliefS			*ReliefF*	
	T_{kd} (ms)	T_S (ms)	m_S	T_F (ms)	m_F
Iris	20	14	18	60	87
Glass	61	140	62	259	126
Balance	136	223	88	743	313
Pima	246	484	77	2684	476
Vehicle	530	2874	195	6790	499
Segment	1520	10870	277	53300	1317
Waveform	7170	359840	1900	529860	2950

Table 4. Time savings by ReliefS w.r.t. *ReliefF*. m_S and m_F are instances used by ReliefS and *ReliefF* to achieve the same performance in raw distance.

sults for individual data sets in postscript can be found at (http://www.public.asu.edu/~huanliu/icml-figs.ps). Comparing the results in Tables 3 and 5, we observe that both *ReliefF* and ReliefS work generally better on continuous data than on nominal or mixed data. For the Precision measure, the larger the better; for the two Distance measures, the smaller the better.

5. Conclusions and Further Work

Here we present a case of active feature selection. A formalism of active feature selection is formed as selective sampling. We chose an efficient feature selection algorithm *Relief* 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 representative instances are randomly selected from the partitions (one from each bucket). Extensive experiments are conducted to evaluate this novel formalism of active feature selection.

Although the experimental study demonstrates the effectiveness of the very first version of active feature selection, we plan future work along the following lines: (1) to further reduce the required instances without affecting the achieved performance by separating feature selection and feature ranking; (2) to investigate why ReliefS still works in cases where data are not purely continuous; (3) to automatically determine γ in feature subset selection as *Relief* is basically a feature ranking algorithm; and (4) to apply active feature selection to the vast body of feature selection and other algorithms (Liu & Motoda, 2001).

Acknowledgments

We gratefully thank Bret Ehlert, Feifang Hu, and Manoranjan Dash for their contributions to this work and anonymous reviewers and Area Chair for their constructive comments. This work is in part based on the project sup-

	Precision		Distance		Raw Distance	
	ReliefF	ReliefS	ReliefF	ReliefS	ReliefF	ReliefS
Breast-cancer	0.768	0.828	0.741	0.589	0.222	0.158
Primary-tumor	0.956	0.966	0.262	0.192	0.306	0.228
KRKPA7	0.959	0.984	0.117	0.066	0.172	0.095
Mushroom	1.0	1.0	0.685	0.290	0.153	0.055
Zoo	0.988	0.994	0.139	0.106	0.672	0.469
Autos	0.921	0.935	0.361	0.205	0.408	0.278
Colic	0.840	0.866	0.383	0.308	0.411	0.310
Vowel	1.0	1.0	0.077	0.020	0.047	0.019
Hypothyroid	0.965	0.974	0.123	0.078	0.074	0.045
W/L/T	**0/7/2**		**0/9/0**		**0/9/0**	

Table 5. Average values for *ReliefF* and ReliefS of three different measures for nominal and mixed data.

ported 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

Aha, D. W., Kibler, D., & Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning*, *6*, 37–66.

Blake, C., & Merz, C. (1998). UCI repository of machine learning databases. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

Blum, A., & Langley, P. (1997). Selection of relevant features and examples in machine learning. *Artificial Intelligence*, *97*, 245–271.

Dash, M., & Liu, H. (1997). Feature selection methods for classifications. *Intelligent Data Analysis: An International Journal*, *1*.

Dash, M., & Liu, H. (2000). Feature selection for clustering. *Proceedings of Fourth Pacific Asia Conference on Knowledge Discovery and Data Mining, (PAKDD-2000). Kyoto, Japan*. Springer-Verlag.

Dy, J. G., & Brodley, C. E. (2000). Feature subset selection and order identification for unsupervised learning. *Proceedings of the Seventeenth International Conference on Machine Learning* (pp. 247–254).

Friedman, J. H., Bentley, J. L., & Finkel, R. A. (1977). An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Software*, *3*, 209–226.

Hall, M. (1999). *Correlation based feature selection for machine learning*. Doctoral dissertation, University of Waikato, Dept. of Computer Science.

Kira, K., & Rendell, L. (1992). The feature selection problem: Traditional methods and a new algorithm. *Proceedings of the Tenth National Conference on Artificial Intelligence* (pp. 129–134). Menlo Park: AAAI Press/The MIT Press.

Kohavi, R., & John, G. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, *97*, 273–324.

Kononenko, I. (1994). Estimating attributes : Analysis and extension of RELIEF. *Proceedings of the European Conference on Machine Learning, April 6-8* (pp. 171–182). Catania, Italy: Berlin: Springer-Verlag.

Langley, P. (1994). Selection of relevant features in machine learning. *Proceedings of the AAAI Fall Symposium on Relevance*. AAAI Press.

Liu, H., & Motoda, H. (1998). *Feature selection for knowledge discovery data mining*. Boston: Kluwer Academic Publishers.

Liu, H., & Motoda, H. (Eds.). (2001). *Instance selection and construction for data mining*. Boston: Kluwer Academic Publishers.

Liu, H., & Setiono, R. (1996). A probabilistic approach to feature selection - a filter solution. *Proceedings of International Conference on Machine Learning (ICML-96), July 3-6, 1996* (pp. 319–327). Bari, Italy: San Francisco: Morgan Kaufmann Publishers, CA.

Moore, A. W. (1991). *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.

Roy, N., & McCallum, A. (2001). Toward optimal active learning through sampling estimation of error reduction. *Proceedings of the Eighteenth International Conference On Machine Learning*.

Sikonja, M. (1998). Speeding up Relief algorithms with k-d trees. *Proceedings of the Electrotechnical and Computer Science Conference ERK'98*.

Talavera, L. (1999). Feature selection as a preprocessing step for hierarchical clustering. *Proceedings of Internationl Conference on Machine Learning (ICML'99)*.

Witten, I., & Frank, E. (2000). *Data mining - pracitcal machine learning tools and techniques with JAVA implementations*. Morgan Kaufmann Publishers.