Active Feature Selection Using Classes

Huan Liu[1], Lei Yu[1], Manoranjan Dash[2], and Hiroshi Motoda[3]

[1] Department of Computer Science & Engineering
Arizona State University, Tempe, AZ 85287-5406
{hliu,leiyu}@asu.edu
[2] Department of Elec. & Computer Engineering
Northwestern University, Evanston, IL 60201-3118
manoranj@ece.northwestern.edu
[3] Institute of Scientific & Industrial Research
Osaka University, Ibaraki, Osaka 567-0047, Japan
motoda@sanken.osaka-u.ac.jp

Abstract. Feature selection is frequently used in data pre-processing for data mining. When the training data set is too large, sampling is commonly used to overcome the difficulty. This work investigates the applicability of active sampling in feature selection in a filter model setting. Our objective is to partition data by taking advantage of class information so as to achieve the same or better performance for feature selection with fewer but more relevant instances than random sampling. Two versions of active feature selection that employ class information are proposed and empirically evaluated. In comparison with random sampling, we conduct extensive experiments with benchmark data sets, and analyze reasons why class-based active feature selection works in the way it does. The results will help us deal with large data sets and provide ideas to scale up other feature selection algorithms.

1 Introduction

Feature selection is a frequently used technique in data pre-processing for data mining. It is the process of choosing a subset of original features by removing irrelevant and/or redundant ones. Feature selection has shown its significant impact in dealing with large dimensionality with many irrelevant features [1, 2] in data mining. By extracting as much information as possible from a given data set while keeping the smallest number of features, feature selection can remove irrelevant features, reduce potential hypothesis space, increase efficiency of the mining task, improve predictive accuracy, and enhance comprehensibility of mining results [3, 4]. Different feature selection methods broadly fall into the *filter model* [5] and the *wrapper model* [6]. 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 and uses the performance of the learning algorithm to evaluate and determine which features should be selected. The wrapper model tends to give superior performance as it finds features better suited to the predetermined learning algorithm, but it also tends to be computationally more expensive [7]. When the training data becomes very large, the filter model is usually a good choice due to its computational efficiency and neutral bias toward any learning algorithm. Many feature selection algorithms [8, 9] are being developed to answer challenging research issues: from handling a huge number of instances, large dimensionality, to dealing with data without class labels.

This work is concerned about a huge number of instances with class labels. Traditional feature selection methods perform dimensionality reduction using whatever training data is given to them. When the number of instances becomes very large, sampling is a common approach to overcome the difficulty [10, 11]. However, random sampling is blind. It selects instances at random without considering the characteristics of the training data. In this work, we explore the possibility of *active feature selection* that can influence which instances are used for feature selection by exploiting some characteristics of the data. Our objective is to actively select instances with higher probabilities to be informative in determining feature relevance so as to improve the performance of feature selection without increasing the number of sampled instances. Active sampling used in active feature selection chooses instances in two steps: first, it partitions the data according to some *homogeneity* criterion; and second, it randomly selects instances from these partitions. Therefore, the problem of active feature selection boils down to how we can partition the data to actively choose useful instances for feature selection.

2 Feature Selection and Data Partitioning

In this work, we attempt to apply active sampling to feature selection which exploits data characteristics by sampling from subpopulations[1]. Each subpopulation is formed according to a homogeneity criterion. Since this work is dealing with a large number of instances in feature selection, when choosing a feature selection method to demonstrate the concept of active sampling, *efficiency* is a critical factor. We adopt a well received, efficient filter algorithm *Relief* [12, 13] which can select statistically relevant features in linear time in the number of features and the number of instances. We first describe *Relief*, and then examine two ways of partitioning data into subpopulations using class information.

2.1 Relief for feature selection

The key idea of *Relief* (shown in Fig. 1) is to estimate the quality of features according to how well their values distinguish between the instances of the same and different classes that are near each other. *Relief* chooses features with n largest weights as relevant ones. For this purpose, given a randomly selected instance X from a data set S with k features, *Relief* searches the data set for

[1] We follow the practice in data mining that a given data set is treated as a population although it is only a sample of the true population, and subsets are subpopulations.

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 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 [12, 13]. Normalization with m in calculation of $W[A_i]$ guarantees that all weights are in the interval of [-1,1].

Given m - 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. $W[A_i]= W[A_i] - diff(A_i, X, H)/m$
 $+diff(A_i, X, M)/m$;
7. end;
8. end;

Fig. 1. Original Relief algorithm.

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 $m \ll N$. The m instances are chosen randomly in *Relief*. Given a small constant m, we ask if by active sampling, we can improve approximations to close to those using N instances.

2.2 Data partitioning based on class information

Intuitively, since *Relief* searches for nearest hits and nearest misses, class information is important and can be used to form subpopulations for active sampling. We investigate two ways of using class information below, and indicate why this scheme of active feature selection should work in helping reduce data size without performance deterioration.

Stratified sampling. In a typical stratified sampling [10], the population of N instances is first divided into L subpopulations of $N_1, N_2, ..., N_L$ instances, respectively. These subpopulations are non-overlapping, and together they comprise the whole population, i.e., $N_1 + N_2 + ... + N_L = N$. If a simple random sampling is taken in each stratum, the whole procedure is called *stratified random sampling*. One of the reasons that stratification is a common technique is that stratification may produce a gain in precision in the estimates of characteristics of the whole population. It may be possible to divide a heterogeneous population into subpopulations, each of which is internally homogeneous. If each stratum is homogeneous, a precise estimate of any stratum statistics can be obtained from a small sample in that stratum. These estimates can then be combined into a precise estimate for the whole population.

In dealing with a data set with class information, a straightforward way to form strata is to divide the data according to their class labels. If there are j classes $(c_1, c_2, ..., c_j)$, we can stratify the data into j strata of sizes $N_1, N_2, ..., N_j$. That is, the number of strata is determined by the number of classes. Each stratum contains instances with the same class. There are only two strata if $j = 2$. Time complexity of stratifying the data into j classes is $O(jN)$.

Entropy-based partitioning. Having only j classes, if one wishes to create more than j strata, different approaches should be explored. The key to stratification is to form homogeneous subpopulations. In order to divide the data into more than j subpopulations, finer strata need to be formed. We can group instances that are similar to each other into subpopulations of pure classes. This can be achieved by (1) dividing the data using feature values, (2) then measuring each subpopulation's *entropy*, and (3) continuing the first two steps until each subpopulation is pure or it runs out of features to divide subpopulations. This idea of applying entropy to measure purity of a partition is frequently used in classification tasks [14]:

$$entropy(p_1, p_2, ..., p_j) = -\sum_{i=1}^{j} p_i \log p_i \ ,$$

where p_i is a fraction estimating the prior probability of class c_i. One can now form subpopulations (or partitions) based on feature values[2]. After using q values of feature A_i to divide the data into q partitions, the expected entropy is the sum of the weighted entropy values of the partitions:

$$entropy_{A_i} = \sum_{o=1}^{q} w_o * entropy_o(p_1, p_2, ..., p_j), \qquad \sum_{o=1}^{q} w_o = 1 \ ,$$

where w_o is the percentage of the data that fall into partition o. It can be shown that for a pure partition (or all data points in the partition belong to one class), its entropy value $entropy_o()$ is 0. Hence, achieving pure partitions amounts to minimizing $entropy_{A_i}$. This partitioning process can continue until all partitions are pure or no further partitioning can be done. The formed partitions are in theory better than simple stratification described earlier because instances in the same partitions share the same feature values determined by partitioning. In other words, entropy-based partitioning uses feature values to partition data taking into account of class information. So, instances in a partition are close to each other besides having the same class value in an ideal case. It is also

[2] For continuous values, we find an optimal point that binarizes the values and minimizes the resulting entropy.

reasonable to anticipate that this entropy-based partitioning will result in more partitions than simple stratification for non-trivial data sets. Time complexity of entropy-based partitioning is $O(kN \log N)$ which is more expensive than that of stratified sampling ($O(jN)$).

With the above two methods for data partitioning, we are now able to partition a given data set into different partitions and then randomly sample from these partitions for *Relief* in choosing m instances as in Fig. 1. Since the number of instances in each stratum is different, to select a total of m instances, the number of instances sampled from a stratum (say, ith one) is proportional to the size (N_i) of the stratum, i.e., its percentage is $p_i\% = N_i/N$. Given m, the number of instances (m_i) sampled from each stratum is determined by $m * p_i\%$ for the ith stratum, and then random sampling is performed for each stratum to find m_i instances. The reason why stratified sampling works should apply to both methods and we expect gains of applying active sampling for feature selection. We present the details below.

3 Class-based Active Feature Selection

The two partitioning methods described in Section 2 constitute two versions of active sampling for *Relief* in Fig. 2: (1) *ReliefC* - *strata* are first formed based on class values, then m instances are randomly sampled from the strata, and the rest remains the same as in *Relief*; and (2) *ReliefE* - data is first partitioned using entropy minimization, then m instances are randomly sampled from the partitions, and the rest remains the same as in *Relief*. *ReliefC* in Line 3a of Fig. 2 uses $p\%$ instances randomly sampled from the strata of different classes, and *ReliefE* in Line 3b selects $p\%$ instances randomly sampled from the partitions determined by entropy minimization when splitting data along feature values, where $p\% \approx m/N$. Each partition has a distinct signature which is a combination of different feature values. The length of a signature is in the order of $\log N$.

In the following, we will conduct an empirical study to verify our hypothesis that active sampling should allow us to achieve feature selection with fewer instances (i.e., smaller m). If *ReliefC* and *ReliefE* can achieve what they are designed for, we want to establish which one is more effective. As a reference for comparison, we use *ReliefF* [13, 15] which extends *Relief* in many ways: it searches for several nearest neighbors to be robust to noise, and handles multiple classes.

4 Empirical Study

We wish to compare *ReliefC* and *ReliefE* with *ReliefF* and evaluate their gains in selecting m instances. Since m is the number of instances used to approximating probabilities, a larger m implies more reliable approximations. In [13], m is set to N to circumvent the issue of optimal m when many extensions of *Relief* 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 *Relief* become $O(kN^2)$.

Given $p\%$ - percentage of N data for *ReliefC* and *ReliefE*, and k - number of features,

```
1.  set all weights W[A_i] = 0.0;
2.  do one of the following:
    a.stratifying data using classes; // active sampling for ReliefC
    b.entropy-based partitioning; // active sampling for ReliefE
3.  corresponding to 2a and 2b,
    a.m = Σ (sample p% data from each stratum);
    b.m = Σ (sample p% data from each partition);
4.  for j = 1 to m do begin
5.    pick instance X_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_j, H)/m
                  +diff(A_i, X_j, M)/m;
9.    end;
10. end;
```

Fig. 2. Algorithms *ReliefC* and *ReliefE* with active sampling.

It is obvious that an optimal ranking of features can be obtained according to the weights $W[A_i]$ by running *ReliefF* with $m = N$. This optimal ranked list (or set) of features is named $S_{F,N}$. Let Y be one of $\{C, E\}$ and *ReliefY* denote either *ReliefC* or *ReliefE*. Given various sizes of m and the subsets $S_{F,m}$ and $S_{Y,m}$ selected by *ReliefF* and *ReliefY* respectively, the performance of *ReliefY* w.r.t. *ReliefF* can be measured in two aspects: (1) compare which subset ($S_{F,m}$ or $S_{Y,m}$) is more similar to $S_{F,N}$; and (2) compare whether features of the subsets $S_{F,m}$ and $S_{Y,m}$ are in the same order of the features in $S_{F,N}$. Aspect (1) is designed for feature subset selection to see if two subsets contain the same features; aspect (2) is for feature ranking and is more stringent than the first aspect. It compares the two ordered lists produced by *ReliefY* and by *ReliefF* of $m < N$ with reference to *ReliefF* of $m = N$. We discuss issues of performance measures below.

4.1 Measuring goodness of selected features

A goodness measure of selected features should satisfy (1) its value improves as m increases; (2) its value reaches the best when $m = N$; and (3) it is a function of the features of the data. Given an optimal ranked list $S_{F,N}$, a target set T is defined as the optimal subset of features which contains the top n weighted features in $S_{F,N}$. For a data set without knowledge of the number of relevant features, T is chosen as 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). To compare the performance of both *ReliefF* and *ReliefY* with different sizes of m, we can define a performance measure $\mathcal{P}(S_{F,N}, R)$ where

R can be either $S_{F,m}$ or $S_{Y,m}$ with varying size m. We examine below three measures for $\mathcal{P}()$.

Precision. Precision is computed as the number of features in T that are also in R_n (the subset of top n features in R), normalized by dividing the number of features in T:

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

Precision ranges from 0 to 1, where the value of 1 is achieved when subsets T and R_n are equal.

Distance Measure. Precision treats all features in T equally without considering the ordering of features. One way of considering the ordering of 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 $S'_{F,N}$ be $S_{F,N}$ in reverse order. The maximum possible ranking distance between the two sets $S_{F,N}$ and $S'_{F,N}$ that share the same features is:

$$D_{max} = \sum_{\forall A_i \in S_{F,N}} |position(A_i \in S_{F,N}) - position(A_i \in S'_{F,N})| \ , \ and$$

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

Since the subset R_n may not contain all the features in T, we use the full set R in the definition of DM. D_{max} is used to normalize DM so that it ranges from 0 to 1, where the value of 0 is achieved if the two sets T and R_n have identical ranking, otherwise, DM is larger than 0.

Raw Distance. A straightforward performance measure is to directly calculate the sum of the differences of weights for the same features in the optimal ranking list $S_{F,N}$ and R. We name it Raw Distance (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 $S_{F,N}$ and R, respectively. RD considers 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* with those of *ReliefE* and *ReliefC*, it serves the purpose well, although it cannot be used for measuring the performance of subset selection as it uses all features.

4.2 Data and experiments

The experiments are conducted using Weka's implementation of *ReliefF* [15]. *ReliefC* and *ReliefE* are also implemented in Weka. All together 12 data sets from the UC Irvine machine learning data repository [16] and the UCI KDD

Table 1. Summary of bench-mark data sets.

Title	# Total Instances	# Total Features	# Total Classes
WDBC	569	30 plus class	2
Balance	625	4 plus class	3
Pima-Indian	768	8 plus class	2
Vehicle	846	18 plus class	4
German	1000	24 plus class	2
Segment	2310	19 plus class	7
Abalone	4177	8 plus class	3
Satimage	4435	36 plus class	6
Waveform	5000	40 plus class	3
Page-Blocks	5473	10 plus class	5
CoIL2000	5822	85 plus class	2
Shuttle	14500	8 plus class	7

Archive [17] are used in experiments. All have numeric features with varied numbers of instances (from 569 to 14500), number of features (from 4 to 85), and number of classes (from 2 to 7). The data sets are summarized in Table 1.

We use increasing percentages of data with three versions of *Relief* and have *ReliefF* with $m = N$ as their reference point. 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 in *ReliefF* is set to 5 (neighbors). This parameter remains the same for all the experiments.
2. Specify five increasing percentage values P_i where $1 \leq i \leq 5$.
3. Run *ReliefE*, *ReliefC*, and *ReliefF* with each P_i determined in Step 2. For each P_i, run each algorithm 30 times and calculate Precision, Distance, and Raw Distance each time, and obtain their average values after 30 runs. Curves are plotted with average results.

4.3 Results and discussions

Intuitively, if active sampling works and the data is divided sensibly, we should observe that the finer the data is divided, the more gain there should be in performance improvement. For each data set, we obtain three curves for the three versions of *Relief* for each performance measure. Fig. 3 demonstrates two illustrative sets of average results for Precision, Distance, and Raw Distance. Recall that for Precision, 1 is the best possible value, while for Distance and Raw Distance, 0 is the best possible value. As shown in Fig. 3, for the Segment Data, we notice that all three versions of *Relief* perform equally well in Precision, but differently in Distance and Raw Distance. Precision is 1 indicates that all features selected are the same as if we use the whole N instances for selection. When Distance and Raw Distance have values greater than 0, it indicates that

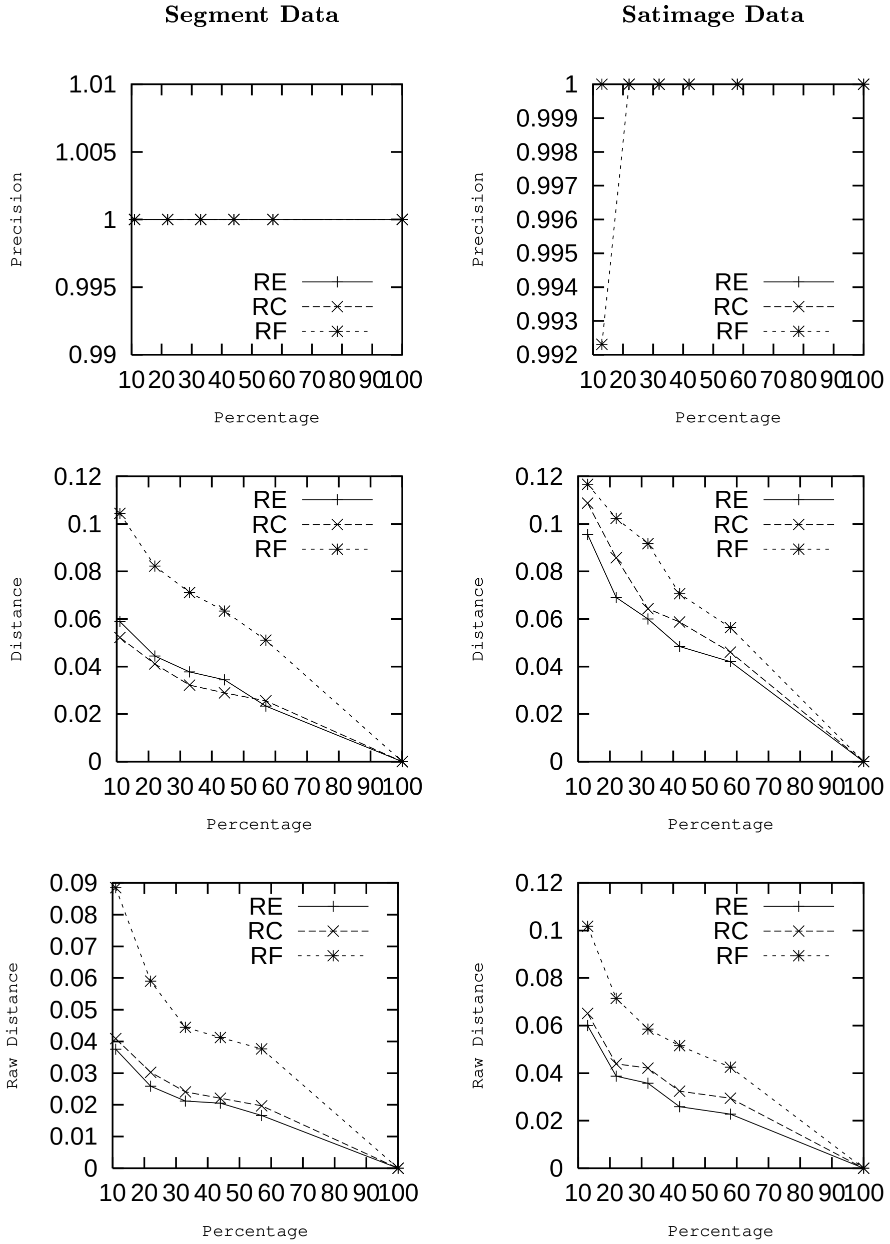

Fig. 3. Two illustrative sets of performance results on Segment and Satimage data sets.

the selected results are not exactly the same as that of using N instances. For the two sets of results (Segment and Satimage), we note one interesting difference between the two that *ReliefE* is, in general, worse than *ReliefC* for Segment in Distance Measure. It can be observed in all three versions of *Relief* that the more instances used, the better the performance of feature selection. A similar trend can also be observed for the Satimage data. It is clear that the two versions of active sampling generate different effects in feature selection.

Table 2. Precision, Distance, Raw Distance results: applying *ReliefE* (RE), *ReliefC* (RC) and ReliefF (RF) to feature selection on bench-mark data sets. We underline those figures that do not obey the general trend $\mathcal{P}(RE) \geq \mathcal{P}(RC) \geq \mathcal{P}(RF)$ and boldface those that are worse than $\mathcal{P}(RF)$.

	Precision			Distance			Raw Distance		
	RE	RC	RF	RE	RC	RF	RE	RC	RF
WDBC	0.992	**0.990**	0.991	$\underline{0.137}$	0.136	0.139	0.105	0.111	0.111
Balance	0.891	0.884	0.864	0.317	0.398	0.468	0.030	0.032	0.037
Pima-Indian	0.919	0.914	0.906	**0.249**	**0.249**	0.248	0.019	**0.020**	0.019
Vehicle	0.999	0.999	0.996	0.150	0.183	0.206	0.041	0.048	0.052
German	0.907	0.907	0.898	0.345	0.352	0.360	0.146	0.149	0.154
Segment	1.0	1.0	1.0	$\underline{0.040}$	0.036	0.074	0.025	0.027	0.054
Abalone	0.956	0.953	0.947	0.251	**0.277**	0.257	0.002	0.003	0.003
Satimage	1.0	1.0	0.998	0.064	0.073	0.088	0.039	0.042	0.065
Waveform	1.0	1.0	1.0	0.070	0.072	0.080	0.043	0.045	0.047
Page-Blocks	1.0	1.0	1.0	**0.214**	**0.220**	0.202	0.006	0.006	0.006
CoIL2000	1.0	1.0	1.0	0.054	0.056	0.060	0.102	0.106	0.110
Shuttle	1.0	1.0	1.0	0.0	0.0	0.0	0.002	0.002	0.003

Table 2 presents a summary of performance measures $\mathcal{P}_i$ averaged over five percentage values P_i and $1 \leq i \leq 5$, i.e.,

$$val_{Avg} = (\sum_{i=1}^{5} \mathcal{P}_i)/5 \ .$$

It is different from the results demonstrated in Fig. 3 that shows the progressive trends. Table 2 only provides one number (val_{Avg}) for each version of *Relief* (RE, RC, RF) and for each performance measure. In general, we observe that $\mathcal{P}(RE) \geq \mathcal{P}(RC) \geq \mathcal{P}(RF)$ where $\geq$ means "better than or as good as". In Precision, *ReliefC* has one case that is worse than *ReliefF*. In Distance, *ReliefE* has two cases that are worse than *ReliefC*, and two cases that are worse than *ReliefF*; *ReliefC* has 3 cases that are worse than *ReliefF*. In Raw Distance, *ReliefC* has one case that is worse than *ReliefF*. It is clear that for the 12 data sets, *ReliefE* is better than or as good as *ReliefC* with two exceptions in all three measures; and *ReliefC* is better than or as good as *ReliefF* with four exceptions in all three measures. The unexpected are (1) *ReliefE* does not show significant superiority over *ReliefC*; and (2) in a few cases, *ReliefE* performs worse than *ReliefC* as shown in Table 2 and Fig. 3, although *ReliefE* uses both feature values and class information to partition data. *ReliefE* and *ReliefC* can usually gain more for data sets with more classes.

The two versions of active sampling incur different overheads. As discussed earlier, between the two, the extra cost incurred by *ReliefC* is smaller - its time complexity is $O(jN)$. Time complexity of entropy-based partitioning is $O(kN \log N)$ which is more expensive than $O(jN)$ and usually $k \gg j$, where k is number of features. However, the additional costs for both versions incur only once. Because for 8 out of 12 cases *ReliefC* is better than *ReliefF*, with its low overhead, *ReliefC* can be chosen over *ReliefF* for feature selection. In *ReliefF*, it takes $O(N)$ to find the nearest neighbors; using signatures, it costs $O(\log N)$ for *ReliefE* to do the same. Therefore, the one-time loss of time in obtaining signatures can normally be compensated by the savings of at least m times of searching for neighbors. In addition, as an example seen in Fig. 3 (Segment Data), *ReliefE* and *ReliefC* using 10% of the data can achieve a performance similar to *ReliefF* using 50% of the data in terms of Distance and Raw Distance.

The three performance measures are defined with different purposes. For feature selection, in effect, Precision is sufficient as we do not care about the ordering of selected features. Many cells have value 1 in Table 2 for Precision measure. This means that for these data sets, the feature selection algorithms work very well - the usual smallest percentage ($P_1 = 10\%$) can accomplish the task: in order to have the average value to be 1, each $\mathcal{P}_i$ in $(\sum_{i=1}^{5} \mathcal{P}_i)/5$ should be 1. Similarly, we can infer that when the average Precision value is very close to 1, it indicates that active sampling can usually work with a smaller percentage of data. Since *Relief* is a ranking algorithm, when it selects features, it provides additional ordering information about selected features. When the value of Precision is 1, it does not mean that the orders of the two feature subsets are the same. We can further employ Distance (with a threshold γ for selecting features) and Raw Distance to examine whether their orders are also the same.

5 Concluding Remarks

In order to maintain the performance while reducing the number of required instances used for feature selection, active feature selection is proposed, implemented, and experimentally evaluated using a widely used algorithm *Relief*. Active sampling exploits data characteristics to first partition data, and then randomly sample data from the partitions. Two versions of active sampling for feature selection are investigated: (a) *ReliefC* - stratification using class labels and (b) *ReliefE* - entropy-based partitioning. Version (a) uses only class information, and version (b) splits data according to feature values while minimizing each split's entropy. The empirical study suggests that (1) active sampling can be realized by sampling from partitions, and the theory of stratified sampling in *Statistics* suggests some reasons why it works; (2) active sampling helps improve feature selection performance - the same performance can be achieved with fewer instances; and (3) between the two versions, in general, *ReliefE* performs better than *ReliefC* in all three measures (Precision, Distance, and Raw Distance).

6 Acknowledgments

We gratefully thank Bret Ehlert and Feifang Hu for their contributions to this work. This work is in part based on the project supported by National Science Foundation under Grant No. IIS-0127815 for H. Liu.

References

1. R. Kohavi and G.H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.
2. H. Liu and H. Motoda. *Feature Selection for Knowledge Discovery & Data Mining*. Boston: Kluwer Academic Publishers, 1998.
3. M. Dash and H. Liu. Feature selection methods for classifications. *Intelligent Data Analysis: An International Journal*, 1(3), 1997.
4. L. Talavera. Feature selection as a preprocessing step for hierarchical clustering. In *Proceedings of Internationl Conference on Machine Learning (ICML'99)*, 1999.
5. U.M. Fayyad and K.B. Irani. The attribute selection problem in decision tree generation. In *AAAI-92, Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 104–110. AAAI Press/The MIT Press, 1992.
6. G.H. John, R. Kohavi, and K. Pfleger. Irrelevant feature and the subset selection problem. In W.W. Cohen and Hirsh H., editors, *Machine Learning: Proceedings of the Eleventh International Conference*, pages 121–129, New Brunswick, N.J., 1994. Rutgers University.
7. P. Langley. Selection of relevant features in machine learning. In *Proceedings of the AAAI Fall Symposium on Relevance*. AAAI Press, 1994.
8. P. S. Bradley and O. L. Mangasarian. Feature selection via concave minimization and support vector machines. In *Proceedings of Fifteenth International Conference on Machine Learning*, pages 82–90, 1998.
9. M.A. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In *Proceedings of Seventeenth International Conference on Machine Learning (ICML-00)*. Morgan Kaufmann Publishers, 2000.
10. W.G. Cochran. *Sampling Techniques*. John Wiley & Sons, 1977.
11. B. Gu, F. Hu, and H. Liu. *Sampling: Knowing Whole from Its Part*, pages 21 – 38. Boston: Kluwer Academic Publishers, 2001.
12. 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.
13. 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.
14. J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
15. I.H. Witten and E. Frank. *Data Mining - Practical Machine Learning Tools and Techniques with JAVA Implementations*. Morgan Kaufmann Publishers, 2000.
16. C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/~mlearn/MLRepository.html.
17. S. D. Bay. The UCI KDD archive, 1999. http://kdd.ics.uci.edu.