Active Sampling: An Effective Approach to Feature Selection

Huan Liu* Hongjun Lu† Lei Yu‡

Abstract

Feature selection is frequently used in data preprocessing for data mining. It decreases number of features, removes irrelevant or noisy data, and increases mining performance such as predictive accuracy and comprehensibility. This work investigates active sampling in feature selection in a filter model setting. Three versions of active sampling are proposed and empirically evaluated: two employ class information and the other utilizes feature variance. They are applied to a widely used, efficient feature selection algorithm *Relief*. In comparison with random sampling, we conduct extensive experiments with benchmark data sets.

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 [4, 10]. 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, increase efficiency of the learning task, improve learning performance like predictive accuracy, and enhance comprehensibility of learned results [6, 15]. Many new feature selection algorithms are being developed to answer challenging research issues: from handling a huge number of instances, large dimensionality (e.g., thousands of features), to dealing with data without class labels. This work tackles feature selection with a huge number of instances. Sampling is a common approach to this problem. It is both random and blind. In this work, we explore the possibility if we can improve the performance of feature selection without increasing the number of sampled data points.

We adopt the *filter* model of feature selection [7] that relies on general characteristics of the training data to select some features without involving any learning algorithm. The other model is the *wrapper* model that requires one predetermined learning algorithm in feature selection and uses the performance of the learning algorithm to evaluate and determine which features should be 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. 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 mining/learning algorithm. Active sampling can influence what instances are used for feature selection by exploiting some characteristics of the data. While random sampling selects instances at random from a given data set, active sampling 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, active sampling boils down to how we can partition the data to actively choose useful instances for feature selection.

2 Active Sampling via Data Partitioning

In this work, we attempt to apply active sampling which exploits data characteristics by sampling from subpopulations. Each subpopulation is formed according to a homogeneity criterion. Data used in data mining can be generally categorized into two types: *with* or *without* class labels. In general, the former are used for classification and the latter used for clustering and association mining. We are concerned about data with class labels. Below, we examine ways of partitioning data into subpopulations using various information.

2.1 Data partitioning based on class information Intuitively, class information is important and can be used to form subpopulations for active sampling. We investigate two ways of using class information below.

Stratified sampling. In a typical stratified sampling [5], 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 of the population, i.e., $N_1 + N_2 + ... + N_L = N$. If a simple random sample is taken in each stratum (N_i), the whole procedure

*CSE, Arizona State University. hliu@asu.edu
†CS, HKUST. luhj@cs.ust.edu
‡CSE, Arizona State University. leiyu@asu.edu

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. To further divide the subpopulations determined by j classes, we essentially want to form finer strata in which instances are similar to each other in addition to class labels. This can be achieved by measuring each subpopulation's *entropy* which is used frequently in classification tasks [13]:

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

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. 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:

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

where w_o is the fraction 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 taking into account of class information. So, the instances in a partition are close to each other besides having the same class value. It is also reasonable to anticipate that this entropy-based partitioning will result in more partitions than simple stratification for non-trivial data sets. Given k, the number of features in a data set, time complexity of entropy-based partitioning is $O(kN \log N)$ which is more expensive than that of stratified sampling $(O(jN))$.

2.2 Data partitioning based on feature variance

The above two methods rely on class information to partition the data. The idea that active sampling may work stems from the fact that instances are usually not uniformly distributed and some instances are more representative than others [1]. If one could find such representative instances, only a small number of instances will be needed for the same task.

As a departure from the two partitioning methods introduced earlier that attempt to put "similar" instances together, we can *separate instances based on their dissimilarity*. One data structure that implements this idea is *kd*-trees [12]. It is a generalization of the simple binary tree, which uses k features instead of a single feature. 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-size partitions. Theoretically, 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 active sampling, one instance is randomly picked from each bucket. The time complexity of building such an optimized *kd*-tree is $O(kN \log N)$. The *kd*-tree can be built once and for all if necessary or dynamically when required [11].

With the above three 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 feature selection. The reason why stratified sampling works (given in section 2.1) should apply to all the three methods and we expect gains of applying active sampling for feature selection.

3 Active Sampling for Feature Selection

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 difficulty. Active sampling improves random sampling and is realized by partitioning the data first. The intuitive idea is to select only instances with higher probabilities to be informative in determining feature relevance.

When choosing a feature selection method to demonstrate the concept of active sampling, *efficiency* is a critical factor. We adopt a well received, efficient algorithm *Relief* [8, 9] which can select statistically relevant features in linear time in the number of features and the number of instances. Below we illustrate how active sampling can be applied to create new algorithms of active feature selection.

3.1 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 of the same and different classes that are near 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 [8, 9]. 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 $\mathrm{W}[A_i] = 0.0$;
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. $\mathrm{W}[A_i] = \mathrm{W}[A_i] - diff(A_i, X, H)/m + diff(A_i, X, M)/m$;
7. end;
8. end;

Figure 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. In our experiments, we use *ReliefF* [9, 17] which extends *Relief* in many ways: it searches for several nearest neighbors to be robust to noise, and handles multiple classes. 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 *ReliefF*. Given a small m, we ask if by active sampling, we can improve approximations to close to those using N instances.

3.2 Relief with active sampling The three partitioning methods described in Section 2 constitute three versions of active sampling for *Relief*:

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 *ReliefF*;
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 *ReliefF*; and
3. **ReliefS** - a kd-tree is first built, then m instances are randomly sampled from the buckets, and the rest remains the same as in *ReliefF*.

Given $p\%$ - percentage of N data for ReliefC and ReliefE, t - bucket size for ReliefS, and k - number of features,

1. set all weights $\mathrm{W}[A_i] = 0.0$;
2. do one of the following:
   a.stratified sampling; // for ReliefC
   b.entropy-based partitioning; // for ReliefE
   c.buildKDTree(t); // for ReliefS
3. corresponding to 2a, 2b, and 2c,
   a.$m = \sum$ (sample $p\%$ data from each stratum);
   b.$m = \sum$ (sample $p\%$ data from each partition);
   c.$m = \sum$ (1 instance sampled from each of m buckets of size t);
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. $\mathrm{W}[A_i] = \mathrm{W}[A_i] - diff(A_i, X_j, H)/m + diff(A_i, X_j, M)/m$;
9. end;
10. end;

Figure 2: Algorithms ReliefC, ReliefE, ReliefS with active sampling.

The three versions of *Relief* with active sampling are summarized in Figure 2. ReliefC in Line 3a of Figure 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 consisting of a combination of different feature values. The length of a signature is in the order of $\log N$. This property can be used to speed up the finding of nearest hits. It is clear that the re-

sulting number of partitions in ReliefE is usually larger than the number of strata in ReliefC. ReliefS involves kd-tree building (`buildKDTree`(t)). The kd-tree divides the sample space into buckets so that m instances can be selected from these buckets.

4 Evaluation

We compare ReliefC, ReliefE, and ReliefS with *ReliefF* and evaluate their gains 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. To compare the performance of both *ReliefF* and ReliefY with different sizes of m, let F be the original *ReliefF*, Y one of $\{C, E, S\}$, and we can define a performance measure $\mathcal{P}(T, R)$ where R can be either $S_{F,m}$ or $S_{Y,m}$ with varying size m, and T is the target set by *ReliefF* with $m = N$. We use three measures for $\mathcal{P}()$ suggested in [11]. **Precision**: It is computed as the number of features in T that are also in R, normalized by dividing the number of features in R. **Distance Measure**: The distance of a feature between two sets is the difference of their positions in the ranking. **Raw Distance**: It calculats the sum of the difference of weights for each of the same feature in the optimal sequence $S_{F,N}$ and the sequence S'.

4.1 Data and experimental procedures All together 12 data sets from the UC Irvine machine learning data repository [3] and the UCI KDD Archive [2] are used in experiments. All have numeric features with varied number of instances (from 569 to 145000), number of features (from 4 to 85), and number of classes (from 2 to 7).

The experiments are conducted using Weka's implementation of *ReliefF* [17]. ReliefC, ReliefE and ReliefS are our versions of active sampling for feature selection and also implemented in the Weka environment. We use increasing percentages of data with these four versions of *Relief* and have *ReliefF* with $m = N$ as their performance 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 is set to 5 (neighbors). This parameter remains the same for all the experiments.

2. Run ReliefS with different bucket sizes t_i corresponding to five percentage values P_i where $1 \leq i \leq 5$. For example, $t_5 = 2$ approximately corresponds to $P_5 \approx 50\%$ (P is 100% with $t = 1$) in experiments. At each P_i, ReliefS is repeated 30 times to calculate average performance values for Precision, Distance, and Raw Distance.

3. Run ReliefE, ReliefC, and *ReliefF* with each P_i determined in Step 2 for comparison purpose. 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.

4.2 Results and discussions Experimental results on benchmark data sets are reported in Table 1. We can still observe the general trend $\mathcal{P}(RS) \geq \mathcal{P}(RE) \geq \mathcal{P}(RC) \geq \mathcal{P}(RF)$ from Table 1. We observe the following: (1) the more instances are used in all four versions of *Relief*, the better the performance of feature selection; (2) More gains are observed when samples are small (close to 10%); (3) ReliefS can significantly improve the performance of feature selection with actively selected data points in most cases; and (4) ReliefE does not show significant improvement over the performance of ReliefC. We also observe steady superiority of ReliefE over ReliefC, and of ReliefC over *ReliefF*. This is because class-based partition method works better for most benchmark data sets with more than two classes. We name this observation as an issue of class-sensitivity in different versions of active learning. ReliefE and ReliefC can usually gain more for data sets with more classes. This is consistent with our intuition as both versions of active sampling rely on class information to stratify or partition the data. But ReliefS is insensitive to the number of classes as building a kd-tree does not require any class information.

5 Concluding Remarks

Inspired by active learning, active sampling for feature selection is proposed, implemented, and experimentally evaluated using a widely used feature selection algorithm *Relief*. Active sampling exploits the data characteristics to first partition data, and then randomly sample data from the partitions. Three versions of active sampling for feature selection are investigated: (a) ReliefC - stratification using class labels, (b) ReliefE - entropy-based partition, and (c) ReliefS - partition based on the kd-tree. Version (a) only uses class information, version (b) splits data according to feature values while minimizing each split's entropy, and version (c) divides data using dissimilarity - feature variance. The empirical study suggests that (1) active sampling can be realized by sampling from partitions, and the theory of stratified sampling in Statistics offers explanations why it works; (2) active sampling helps improve feature selection performance - the same performance can be achieved with much fewer instances; (3) among the three versions, ReliefS performs best in all three

	Precision				Distance				Raw Distance			
	RS	RE	RC	RF	RS	RE	RC	RF	RS	RE	RC	RF
WDBC	0.994	0.992	**0.990**	0.991	0.103	0.137	0.136	0.139	0.068	0.105	0.111	0.111
Balance	0.940	0.891	0.884	0.864	0.247	0.317	0.398	0.468	0.018	0.030	0.032	0.037
Pima-Indian	0.930	0.919	0.914	0.906	0.209	**0.249**	**0.249**	0.248	0.016	0.019	**0.020**	0.019
Vehicle	1.0	0.998	0.999	0.996	0.105	0.150	0.183	0.206	0.026	0.041	0.048	0.052
German	0.920	0.906	0.907	0.898	0.309	0.345	0.352	0.360	0.125	0.146	0.149	0.154
Segment	1.0	1.0	1.0	1.0	0.029	0.040	0.036	0.074	0.020	0.025	0.027	0.054
Abalone	0.971	0.956	0.953	0.947	0.176	0.251	**0.277**	0.257	0.001	0.002	0.003	0.003
Satimage	1.0	1.0	1.0	0.998	0.047	0.064	0.073	0.088	0.022	0.039	0.042	0.065
Waveform	1.0	1.0	1.0	1.0	0.056	0.070	0.072	0.080	0.036	0.043	0.045	0.047
Page-Blocks	1.0	1.0	1.0	1.0	0.043	**0.214**	**0.220**	0.202	0.003	0.006	0.006	0.006
CoIL2000	1.0	1.0	1.0	1.0	0.041	0.054	0.056	0.060	0.074	0.102	0.106	0.110
Shuttle	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.001	0.002	0.002	0.003
Loss	**0**	**2**	**1**	**N/A**	**0**	**4**	**3**	**N/A**	**0**	**0**	**1**	**N/A**

Table 1: Average values of Precision, Distance, Raw Distance results: applying ReliefS (RS), ReliefE (RE), ReliefC (RC) and ReliefF (RF) to feature selection on benchmark data sets.

measures (Precision, Distance, and Raw Distance).

6 Acknowledgments

We gratefully thank Bret Ehlert, Manoranjan Dash, Feifang Hu, and Hiroshi Motoda 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] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[2] S. D. Bay. The UCI KDD archive, 1999. http://kdd.ics.uci.edu.

[3] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/~mlearn/MLRepository.html.

[4] A.L. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97:245–271, 1997.

[5] W.G. Cochran. *Sampling Techniques.* John Wiley & Sons, 1977.

[6] M. Dash and H. Liu. Feature selection methods for classifications. *Intelligent Data Analysis: An International Journal*, 1(3), 1997.

[7] G.H. John, R. Kohavi, and K. Pfleger. Irrelevant feature and the subset selection problem. In W.W. asnd Hirsh H. Cohen, editor, *Machine Learning: Proceedings of the Eleventh International Conference*, pages 121–129, New Brunswick, N.J., 1994. Rutgers University.

[8] 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.

[9] 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.

[10] H. Liu and H. Motoda. *Feature Selection for Knowledge Discovery & Data Mining.* Boston: Kluwer Academic Publishers, 1998.

[11] 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.

[12] 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.

[13] J.R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

[14] 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.

[15] L. Talavera. Feature selection as a preprocessing step for hierarchical clustering. In *Proceedings of Internationl Conference on Machine Learning*, 1999.

[16] D.A. Talbert and D. Fisher. An empirical analysis of techniques for constructing and searching k-dimensional trees. In *Proceedings of the Sixth ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 26–33, Boston, MA, 2000. ACM Press.

[17] I.H. Witten and E. Frank. *Data Mining - Pracitcal Machine Learning Tools and Techniques with JAVA Implementations.* Morgan Kaufmann Publishers, 2000.