Instance Selection in Data Mining

Sameer Pradhan, Xindong Wu

Abstract— **Dealing with very large databases is one of the defining challenges in data mining research and development. No matter how powerful computers are or will be in the future, data mining research and development must consider how to manage ever-growing data that can be too large (e.g., terabytes of data) to be processed at one time. Instance selection is about approaches that select or search for a portion of a large database for use in data mining instead of using the whole database. One of the major approaches for instance selection is sampling in which a sample is selected for training and analysis. Other major approaches include windowing, data reduction, and selection of representative instances. This paper provides a comprehensive review on existing techniques for these approaches.**

Keywords— **Data Mining, Instance Selection, Sampling, Windowing**

I. Introduction

With the advances in data storage technology, the extensive use of database management systems and data warehousing technology, the sheer magnitude of business data has been increasing dramatically. An example is the 101 terabyte transaction database at Wal-Mart, which increases at a rate of 20 million transactions per day [53]. This is more than the rate at which data can be feasibly analyzed using today's computing power, therefore data selection has become an essential topic in data mining research and development.

Data selection comprises two-dimensional slashing of the original database, if you imagine the data consisting of horizontal rows of *instances* and vertical columns of *attributes* or *features*. This paper is different from the literature survey by Blum *et al.* [4], and we concentrate on approaches for speeding-up the knowledge acquisition process through the selection of instances or examples, commonly known as *instance selection*. We will *not* cover feature selection, which has received wide attention in the design of various algorithms and their data preprocessing.

The rest of this paper is organized as follows. Section II reviews various statistical sampling techniques for instance selection. Some of these techniques take into consideration the learning algorithm in use and the data characteristics, whereas others do not consider these factors at all. Section III looks at windowing techniques that can be used to reduce the computation time by avoiding re-discovery of rules. Section IV talks about selective elimination of instances so as to maximize generalization performance and, at the same time, make the mining process quicker. Section V reviews the possibility of using one learning algorithm to select instances for another algorithm.

Sameer Pradhan and Xindong Wu are with the Mathematics and Computer Science Department at the Colorado School of Mines, Golden, Colorado.

II. Sampling Techniques

There are two main reasons why it is sometimes necessary to sample the data to be mined: (i) The learning algorithm is computationally intensive, and/or (ii) There are too many instances to learn from. Data in the second category can contain either labeled or unlabeled instances. If it contains unlabeled instances, and if the labeling process is too expensive, e.g., speech recognition data, then the order of selection also plays an important role, as labeling selected instances incrementally, increases the domain knowledge, which can aid further selection.

In some databases, there is very little, or no prior information available, and hence the possibility of an intelligent search through the space of hypotheses is almost ruled out. It is in this case that uniform random sampling has proved to be the most frequently chosen option [42]. Such a sampling process, which is independent of both, the algorithm to be used for data mining, and the type of data that is to be mined is known as *static sampling* [25]. In static sampling, samples are chosen on the basis of statistical tests of hypotheses. The Probably Close Enough criterion [25] can be used to calculate a sample size that achieves a desirable trade-off between the accuracy of deductions, and the computational complexity. The sample size (n), using this criterion, satisfies the following relation.

$$P(acc(N) - acc(n) < \epsilon) \geq \delta \qquad (1)$$

where, $acc(N)$ is the obtainable accuracy after using the whole database, of size N; $acc(n)$ is the obtainable accuracy after using a sample of size n; ϵ is the tolerable difference in accuracies; and δ is the threshold probability of the classification process. A number of samples are extracted[1] from the database till the obtainable accuracy falls within a desirable range.

Sometimes, prior knowledge about the data is available to the data miner, e.g., information about relative class distributions, degree of redundancy, noise levels, etc., which can help decide which sampling process and which learning algorithm can give optimum results. Such sampling techniques are known as *dynamic sampling* [25]. Finding the best algorithm for a particular database, has been one of the important research issues in the field of machine learning. An expert system wrapper around several learning algorithms – MLToolBox [15], is one example of such efforts.

Different metrics have been devised to quantify redundancy in a database. One of them, the *Mølller's redundancy estimate* [31], uses conditional population entropy. However, it is unclear how this measure is useful – since a database that has exactly twice the instances as another will have the same redundancy. Another approach suggests

[1] The method of extraction is not central to the definition.

the classification of a database as *n-saturated*, when there is no subset n whose removal will cause a new theory to be learned. This concept seems to have a very high computational complexity at the same time it does not take into account the fact that different areas of a database can have different degrees of redundancy.

Most algorithms take into consideration some specific concerns and/or strategies while performing sampling. In *uncertainty sampling* [27], instances are selected, *not* on the basis of misclassification, but on the basis of an expert's confidence in the learned theory. Those instances whose class is the least certain, are carried over. *Decision theoretic subsampling* [32] deals with the calculation of a sample size used to select attributes that would help generate the best decision tree. *Density biased sampling* [35] is designed for applications where it is known that some of the classes are represented by very small numbers of instances, and might be ignored as noise. To correctly classify these minor representatives, it extracts more number of instances of these classes. *Peepholing* [7], is a sophisticated technique of subsampling that eliminates unpromising attributes and thresholds from consideration. But, as mentioned earlier, we will not delve on the attribute selection processes.

The following subsections discuss the above mentioned static and dynamic sampling techniques in more detail.

A. Density Biased Sampling

Density biased sampling [35] starts with clustering an unlabeled database, and selects more instances from the small, less dense clusters. This process has some resemblance to *stratified sampling*, in the case of labeled data, where the number of instances selected from a database, is proportional to the number of class representatives. However, density biased sampling extracts more instances from under-represented classes, and weighs them in order to preserve the underlying population distribution. The selected sample has the following characteristics:

- It is density preserving.
- It is biased by group size.
- Within a group/cluster, instances are selected randomly.

Density biased sampling is similar to *probability proportional to size sampling* [34], which is a multi-stage sampling process, where the sample is a weighted random sample without replacement and the probability of inclusion of each element is proportional to the size of the element. A comparison with an iterative hierarchical clustering algorithm called BIRCH[2] [62], *uniform random sampling*[3] (These are the two most frequently used techniques for clustering large databases), *inverse biased sampling* (IBS) [35] and *inverse root biased sampling*[4] (IRBS) [35] techniques, using synthetic data comprising three distinct cluster groups: (i) Evenly sized clusters, (ii) Clusters following the Zipf's distribution [63], and (iii) A group of one big and several small clusters[5], reveals that in the average case, IBS and IRBS are much better than uniform random sampling. As the cluster size becomes more and more skewed, the difference in performance increases. Also, IBS and IRBS outperform random sampling in group two, when the data follow Zipf's distribution and in group three, which comprises a group of one big and several small clusters. However, uniform random sampling is preferred when all the clusters are approximately the same size, although the accuracy margin is not too large.

B. Decision Theoretic Subsampling

When the number of training instances is very large, all of them cannot be feasibly used for information gain calculations. *Decision theoretic subsampling* [32], helps a decision tree construction algorithm (or any similar induction algorithm) select an optimum sample size that helps it make better decisions on which attribute to safely retain during induction. There is a trade-off between the selection of attributes with the most information, and the time required for the decision tree construction, which is measured by a quantity called *loss tolerance*. The conditional probability of the expected amount of information required to classify a random instance, given information about the value of a particular attribute, is distributed approximately normal. Therefore, we can find a breakeven sample size that produces a variance corresponding to the estimated loss tolerance, and use that sample to select an attribute, so as to generate a good decision tree. This involves the calculation of an expected required sample size (ERSS), above which, the decision tree quality is tolerable; and then extracts a sample of that size. However, a one-shot approach is not sufficient, and a sampling *strategy* has to be devised (as the ERSS might increase with the number of samples taken) to reach the required loss tolerance. The size of successive samples can be calculated using the formula

$$S = \sqrt{\frac{(ERSS - 1)c}{i}} \tag{2}$$

where, c is the overhead time required to process a sample, and i is the additional time required to process each instance of that sample. After a few thousand instances, majority estimates of the required sample size are fairly accurate.

C. Progressive Sampling

Progressive sampling is a static sampling strategy in which a database is sampled in stages, and a cumulative set of instances is used to infer knowledge. The way in which this can be accomplished is through the addition of a constant number of instances, or an increasing or decreasing number of instances[6]. The learning curves typically exhibit

[2]Balanced Iterative Reducing and Clustering using Hierarchies.

[3]The uniform random sampling was performed using Vitter's reservoir sampling algorithm [52].

[4]IBS and IRBS are two variations of density biased sampling.

[5]The simple k-means algorithm was used to perform clustering.

[6]Intuitively, the cumulation of decreasing number of instances is excluded in research till date. A combination of geometrically increasing and arithmetic sampling techniques might be beneficial, owing to the fact that it will minimize the trade-off faced by geometric sampling that the the next sample size can be sometimes a large percentage of the samples processed earlier, and in case the ideal sample size

a steep slope at the beginning, and becomes more gentle and gradually smoothens out with the increasing number of instances considered. The sample size (n_{min}) at which the learning curve starts to flatten, is identified as the point of convergence, above which no significant accuracy gain is obtained. Arithmetic sampling seems to be inefficient in case n_{min} is quite big, but at the same time, if the learning algorithm is incremental, such as ID5R [51], then the size of each progressive sample does not significantly decrease the efficiency. However, very few induction algorithms are incremental.

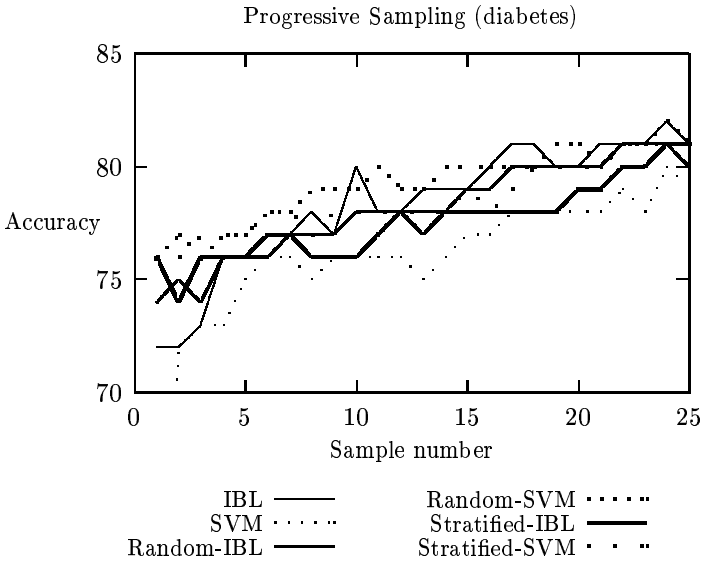

Fig. 1. Progressive Sampling on Diabetes database.

Unfortunately, n_{min} cannot be determined theoretically, as there are too many factors that affect it, which vary from one database to another, and even from one permutation of the same database to another. Most important factors that have been identified to have significant impact on the learning curve are *redundancy*, *noise-level* in the data, and the *technique used for sampling* [20]. Provost *et al.* used uniform random sampling to extract progressive samples, however, better techniques can be exploited to extract the progressive samples so as to attain faster convergence. We used Instance Based Learning algorithm (IBL) [1] and Support Vector Machines (SVM) [24], [6], [43] separately, to progressively select instances and then used random and stratified sampling to extract the same number of instances, to compare their learning rates, but the learning rates using IBL and SVM selected instances were not significantly different than that of random sampling or stratified sampling. This might be attributable to certain other reasons which we shall discuss in Section V.

Figure 1 shows the learning curve for the diabetes database from the machine learning repository at the University of California, Irvine [30]. Note that not the whole learning curve, but only the portion at which the curve starts flattening is shown. This appears ragged because of scaling. Also, each sample number indicates a collection of instances and not just one instance.

Provost *et al.* studied the effect of geometric sampling for this purpose and found it to be asymptotically optimum, as the time complexity to determine the convergence point, which is polynomial in time and space, is no worse than that of inducting the whole population [38].

D. Dynamic Programming Sampling

Progressive sampling guarantees asymptotic optimality, but does not say anything about absolute optimality. To achieve this, the problem of sampling has been formulated as a dynamic programming problem, with an objective of finding the sampling schedule with minimum cost of convergence. The algorithm so obtained is termed as *dynamic programming sampling* [38] and has a time complexity of $O(N^3)$, where N is the total number of examples. If $f(n)$ is the cost of creating a model of n instances and $\Phi(n)$ is the probability that the convergence requires more than n instances, and k samples are extracted, then the cost of convergence of a schedule S can be given by

$$C = \sum_{i=1}^{k} \Phi(n_{i-1}) f(n_i) \tag{3}$$

It can be seen that $\Phi(0) = 1$, that is, the probability of requiring more than 0 instances for convergence equals one. If there is any prior information about the likelihood of convergence, then that can be used to calculate the value of $\Phi(n)$, or else a uniform prior can be assumed. We can take advantage of the fact that the optimum schedule is composed of optimal sub-schedules and formulate the dynamic programming problem that achieves minimum cost of convergence. If $m[i,j]$ is the cost of the optimal schedule for $i \leq j$ instances, for a database that contains N instances, then the following recurrence can be used to compute $m[0,N]$

$$m[i,j] = min \begin{cases} \Phi(i) f(j) \\ min_{i<k<j} m[i,k] + m[k,j] \end{cases}$$

It is seen that the optimal schedule is highly dependent on $f(n)$ [38]. In general, the higher the complexity of $f(n)$, the more frequent the schedules need to be created. Another point worth noting is that although the sample sizes seem to be geometrically increasing, the multiplicative factor is not constant, in fact, it tends to decrease near the end, proving the intuition expressed earlier. The only problem with dynamic programming sampling seems to be the time complexity, but the assuring fact is that it is cubic in terms of the samples required for the model formulation, and not the whole database. So, a realistic schedule can be obtained at the sacrifice of some predictive accuracy. Also, the fact, that in most of the cases $n_{min} \ll N$ increases the feasibility of obtaining optimum schedules.

It is shown [38] that progressive geometric sampling is more efficient as long as the point of convergence is less than

is halfway into that sample, then a lot of processing time will be wasted. However, currently there is no way of determining where to stop geometric sampling and where to start arithmetic sampling. A possible good strategy is to start halfway through the whole database, or when the next sample exceeds some percentage of the total samples processed.

half the sample size, and as the knowledge of n_{min}, becomes increasingly precise, geometric sampling outperforms dynamic programming sampling. Dynamic programming has also been applied to feature selection by Mangasarian [28], and there seems to be a strong connection between its use to select relevant instances.

E. Uncertainty Sampling

There is another way to reduce the instances, and at the same time, increase predictive accuracy of the classifier, by making the process of supervised learning more efficient in terms of the expert's time and expertise usage, using *uncertainty sampling* [27] to present him/her with instances from the database, whose labels are uncertain, even after using the knowledge contained in any previously labeled data, and then using his/her input to classify some other instances whose labels were uncertain before. Lewis *et al.* [27] examine a heterogeneous approach in which a classifier of one type selects instances for training a classifier of another type. This technique yields classifiers faster and with lower error rates than the ones generated using random sample ten times larger. It is worth noting, however, that there seems to be a strong connection between the classifying algorithm used to sample and the one used to classify, as another recent study [45], [46] shows that random sampling proves to be a better choice in case the two algorithms are of conflicting nature, although there is slight difference in both the approaches. More on this can be found in Section V.

Theoretical analysis and practical experience have proved [27] that better classifiers can be learned from a small number of instances, if the learning algorithm is allowed to create artificial instances or membership queries that are presented to the expert for labeling. However, this method suffers from a disadvantage that many of these queries can be non-sensical, therefore, the best way around this is to create a stream of unlabeled instances and ask the expert to label them. Once an instance is labeled, the algorithm checks whether it can help label some others – so far unlabeled instances, and labels them. Then, it presents the user with another instance from this set, whose class is still sufficiently uncertain. This process continues till all the instances are labeled with a certain level of confidence. This process has a two-folded nature; it uses the expert's time intelligently, and at the same time, speeds up the classification process. These approaches can be considered to be a combination of *sequential* and *stratified* sampling. Single classifier approaches to uncertainty sampling have been criticized by some researchers [60] on the grounds that one classifier is not representative of the set of all the classifiers consistent with the data: *the version space* [27]. The degree to which this is a problem is not yet established. Uncertainty sampling is comparable with windowing, which we will discuss in Section III; the main difference being that in windowing all the class labels are known and only the misclassified instances are added to the new window.

Unfortunately, uncertainty sampling has a strong connection with the type of classifier used to select the instances. Despite containing a disproportionately large number of instances from the low frequency classes, it still yields an accurate classifier. The C4.5 [40] algorithm was modified by Catlett [27] to accept the relative costs of two types of errors: false negatives and false positives. This number is called the loss ratio (LR). A loss ratio of 1 indicates that both the errors have equal cost (the default assumption in C4.5). Exactly what figure should be used is a question for further research and motivates a sensitivity analysis of the effect of this parameter on the accuracy of the generated classifier. The class values are determined, not by majority votes, but by comparison with a probability threshold $\frac{LR}{LR+1}$, or its reciprocal, whichever is appropriate. Pruning minimizes the expected loss instead of estimated errors.

Uncertainty sampling technique has been used, mostly, for text categorization. Texts often reside in large databases supporting boolean queries, a restricted version of propositional logic. Since decision rules can be easily converted to this form they turn out to be a better choice. Another reason for accepting them is that they are also easier to understand to humans. It was found [27] that the decision rules that C4.5 produced by using uncertainty sampling of around 1000 instances, were significantly more accurate than the ones obtained by applying C4.5 to random sample four times larger.

F. Selective Sampling

In cases where there is an abundance of unlabeled data and the cost of labeling is very high, *selective sampling* [13], [44] plays an important role. In this case an algorithm chooses instances which it considers informative. Initially, almost all the instances are informative, but as the process continues, some of the available instances are more informative than the others, and choosing these instances can accelerate the learning process. Seung *et al.* [44], [17] have suggested a *query by committee* algorithm that helps achieve this. This process is iterative; in each iteration, it selects a sample randomly from the population of unlabeled instances, and then, picks up n hypotheses[7] randomly from the version space, and labels the selected instance. In case all the hypotheses label the instance identically, then that instance is not considered informative enough, and the process continues. However, if the hypotheses label the instance differently, then an expert is asked to provide the correct label for that instance, and it is added to the set of selected instances. This process continues till the algorithm is confident, to a certain extent, of the accuracy of predictions based on the selected instances, i.e., when a sufficiently large number of samples are rejected.

Freund *et al.* [17] prove that in perceptron-type learning algorithms, the prediction error decreases exponentially fast in the number of queries, and show that such an exponential decrease is guaranteed for a general class of learning problems, where an accuracy ϵ is obtained after extracting $O(\frac{1}{\epsilon})$ samples and calling the expert $O(log\frac{1}{\epsilon})$ times in case of *deterministic* and *noiseless* concepts. Cohn *et al.* [13] have also developed a *selective sampling* strategy, which is a

[7] This is the size of the committee

part of the concept of *active learning*, where the classifier is trained with instances that are progressively selected from the *region of uncertainty*, i.e., part of the concept space which the classifier cannot classify using the knowledge at hand, with a certain degree of confidence. This region of uncertainty is difficult to determine, even in case of simple concepts, but Cohn *et al.* [13] have developed an approximate method to – either overdetermine it, or underdetermine it, so that most, if not all of the instances that are examined are non-redundant in the learning process. These, again, are examples of dynamic sampling techniques.

G. Other Methods

Depending on the type of databases that are being mined, sampling strategies need to be changed. In case the positive instances are not sufficient enough, the classification may not be very good. Randomly sampling over the population is one way in which the desired sample can be obtained, as the random sample naturally contains more percentage of the common categories. Another way is to use *uncertainty sampling*, but since this method is driven by the failure of a classifier, it tends to be prejudiced towards the rare categories.

Another sampling strategy is *proportion enforced sampling* [60], where the database is split using a parameter k such that if k is 2, then one instance in the pool will be included in each of the two subsets. Another method, called *completeness-driven sampling* [60], ensures that all the instances are represented in the sample. In this case, the learning curve has a steep slope in the beginning, which then gradually plateaus off, resulting in increased accuracy, but at a vast computational expense. The principle of *Pareto analysis*[8] states that 80% knowledge can be acquired from just 20% of the data. This also suggests that significant knowledge can probably be retained by eliminating the majority of instances, which ones, however is the question.

Random sampling of larger databases seems to provide good fidelity knowledge for smaller sample sizes, whereas smaller databases need larger samples to generate the same level of fidelity. It is observed that, in case of smaller databases, there is little speedup as the sample tends to identify false large item sets [61]. Whereas, in case of larger databases, the speedup is quite significant (sometimes as high as four times [61]), which can again be attributed to the increased probability of significant redundancy in large databases (more on this can be found in Section III), and so, random sampling is preferable for larger databases. When the *binomial distribution* was used to select samples from a database randomly and the sample size was determined using *Chernoff bounds* [61], which is a theoretical method for calculating the sample size so that the probability of success equals the *support*[9] required – in case of association rule discovery, it was found that the sample sizes were very conservative and in many cases exceeded the size of the total database, since the formula is independent of size of the database under consideration. So, for all practical purposes, sample sizes less than advised by the *Chernoff bounds* equation, give sufficiently *confident*[9] association rules.

It was observed [33], using C4.5, with its three pruning techniques, viz., *error-based pruning* (EBP – the C4.5 default), *reduced error pruning* (REP) and *minimum description length pruning* (MDL); and using two rule learning algorithms, viz., C4.5 RULES and RIPPER [12], that, as the size of the training data increases, the model size keeps on increasing linearly, (in the cases under consideration) even after a point at which the classification accuracy ceases to show a significant increase. This might a result of the learning algorithms being able to discern increasingly weak structures, or they are susceptible to fitting noise and the problem is exacerbated with large databases. When the class labels are randomized, the decision tree learning algorithms come up with a number of rules, although the data is devoid of any useful structure. However, the rule learning algorithms – especially RIPPER work as expected, generating almost zero rules. In case of decision tree learning algorithms, this seems to be explained by the phenomenon of *bias propagation* [33]. The reason for rule learning algorithms is an open question. Methods like Tabu search [22] are also being exploited to aid the selection process [3]. Brin *et al.* have developed a *dynamic data mining* [5] algorithm to explore large rule spaces by sampling. Chaudhari *et al.* [9] have formulated bounds on the amount of data required to generate a satisfactory histogram in relation to database query and association analysis. Ting *et al.* [49] have exploited *batch learning* to speed-up the learning process. Tsumoto *et al.* [50] have developed methods based on recursive iteration of resampling methods and multiple statistical testing.

III. Windowing

Windowing selects a window of instances from the set of training instances, to learn a concept, and then, depending on the correctness of the concept, new instances are added to the window and the process is repeated till all the training instances are correctly classified [40]. This process can be considered to be a variation of sampling, since it uses only a sample of the training set to formulate rules, which are used to test the remaining database in order to eliminate the instances that are correctly classified, from further consideration. This makes the learning process much quicker by avoiding rediscovery of rules.

Windowing was first proposed as a procedure for efficient use of the ID3 [56] decision tree construction algorithm. Very poor predictive accuracies were obtained when windowing was applied to noisy data. To fight this limitation, Quinlan implemented a new version of windowing in 1993. The new version differs in the following respects: (i) It tries to select the instances so as to make the class distribution more uniform (This is supposed to accrue gains in predictive accuracy in domains with skewed distributions), (ii)

[8] *Pareto analysis*, named after the 19^{th} century economist Vilfredo Pareto, is based on the principle that most effects come from few causes, i.e., 80% of the effects come from 20% of the possible causes.

[9] *Support* and *confidence* are terms associated with the association rule discovery process, and provide a quantitative view of the rules generated. For more information, refer to [10, pg. 869].

It includes half the misclassified instances in the next window in order to guarantee faster convergence in presence of noise, and (iii) It can stop before all the instances are correctly classified, in case it finds that there is no significant gain in predictive accuracy. However, it was found that windowing was not a very good strategy [56] and the interest in it died for sometime till 1994, when it was revived owing to a couple of papers [26], [29] that emasized the use of sampling in data mining. It was also observed that windowing worked better with rule learning algorithm like DOS[10] [18], [20], than with decision tree learning algorithm like C4.5. Windowing as applied to DOS was much faster than plain DOS. Thus, windowing seems to work better with rule learning algorithms as compared to decision tree learning algorithms. This might be explained by the fact that in decision tree algorithms, the instances are selected randomly and tend to maintain the population distribution whereas in case of windowing, the process is started by selecting random instances to fill the window, but further down the line, when the instances in the window are modified, all the instances that were misclassified by the theory learned from the previous windowing process are added to the new window, thus creating a skew distribution of instances in the window. Thus, for each individual rule in the target theory, windowing tries to identify the minimum number of instances from which the rule can be learned. Separate and conquer rule based algorithms try to maximize the number of positive instances covered, and at the same time, try to minimize the negative instances that pass the test. Whereas in case of C4.5, the information gain (or average entropy), is subtracted from the information value, which is same for all the tests. More evidence of this sensitivity also comes from the fact that making the class distribution in the initial window more uniform, produces better results.

Fürnkranz found that, contrary to the conclusions by Catlett [7][11], windowing can be very useful in cases of *noise-free* data. He proposed a new algorithm that increased the efficiency of windowing by preventing *relearning* of the rules in previous windows, and also discussed an improved noise handling strategy [19], [20], [2]. The subject of noise handling in windowing is quite complex and cannot be resolved using just some noise filtering algorithm. The modified algorithm removes instances classified by good rules from further consideration. However this does not guarantee that the good rule will classify all the instances correctly, and thus the apparently consistent rules are tested in the next iteration. This way the window size remains small and so does the learning time. This improved version has a disadvantage that it tends to find overly-specialized rules, owing to the low instance size. To overcome this disadvantage, a procedure is implemented at the end of the algorithm, to arrange all the rules in an ascending order of the number of instances correctly classified, and then remove all the rules whose instances are covered by the more general rules (or, the rules that come later in the indexing). This improved version, called WINDOS-95, does show considerable improvement in terms of the number of instances processed in a window.

A noise-tolerant windowing algorithm to efficiently handle noisy data with the help of the noise-tolerant learning algorithm – I-RIP, called I-WIN [19], [20], not only carries over the misclassified instances to the new window, but also removes all the instances that are correctly classified by the good rules from the earlier window. I-WIN outperforms I-RIP at lower noise levels, but reverse is the case when the noise levels increase [19]. We can say that I-WIN outperforms I-RIP as the size of the training data increases or as the redundancy in the database increases.

Another characteristic of the database that favors windowing is the presence of redundancy[12]. The more the number of redundant instances in the training set, the more useful the windowing process will be, whereas in case of training sets that do not have a single redundant instance, all of them have to go through the window one time or the other thereby nullifying the basic advantage of windowing (as is the case of the Quinlan's KRKN chess database [19, pg. 141]).

An improved version of windowing, called *integrative windowing* [20] tries to exploit the fact that regions of the databases that are already covered by good rules, need not be considered in subsequent iterations. The algorithm is so named because of its technique of successively integrating the learned rules into the final theory. Contrary to conventional windowing, this algorithm does not merely add the incorrectly classified instances, but also removes instances from the window if they are covered by consistent rules. These experiments showed some interesting characteristics [20]. In case of a noise-free database (the Mushroom database) the effect of windowing was very prominent. Whereas, as mentioned earlier, in case of a non-redundant database (the KRKN database) the predictive accuracy was 100% *only* when all the instances were used for training purpose and that redundancy is of crucial importance for the success of windowing. Mϕller's redundancy estimate [20], however, fails to explain the success of windowing as related to redundancy.

The effect of noise in the windowing algorithm is that the window ends up having all the misclassified instances (noise itself) in it and this tends to hinder the pace of knowledge discovery as well as its quality. In order to adopt to noisy domains, the algorithm has to be modified in certain respects as follows.

- **Consistency-Check** When is a rule learned from the window good enough? The consistency check is resolved with the introduction of an estimate which has to be significantly above the default accuracy.
- **Completeness-Check** When should we stop adding rules to the theory? To handle the completeness check, the size of a window is doubled when no new rules are found using the instances from it.
- **Resampling** Which instances should be added to the window? Resampling adds a fresh set of instances to

[10] Dull Old Separate and conquer rule learning algorithms.

[11] Note that it is not actually contrary. Wirth [56] *et al.* do mention that windowing is favorable in case of noise-free data.

[12] Redundancy not only favors windowing, but also many other algorithms, e.g., *progressive sampling.*

the window when it is observed that the noise level in the window is substantial. The instances that are added comprise ones that are covered with insignificant rules and all the uncovered positive instances. The modification in the algorithm that aids windowing in case of noisy databases is that the rules that are learned in the current window are tested on the entire database for consistency. If the rules pass this test then they are understood to be consistent rules. Also, a rule that is found to be significant will be added to the list of rules and all the instances that are covered by this rule are removed from the database altogether.

IV. Instance Reduction Techniques

The *nearest neighbor algorithm* [14] and its derivatives – *condensed nearest neighboring* [23], *selective nearest neighboring* [41] and *reduced nearest neighboring* [21], are some research attempts at reducing the size of the training data, while maintaining the generalization performance. These are often quite successful at learning a concept from a training set and provide a good generalization on subsequent input vectors. However, they often retain the entire training set in the memory, resulting in large memory requirements, slow execution speed and sensitivity to noise. Distance functions in these neighbouring schemes also have a significant impact on their ability to generalize correctly. Several variations have been proposed by Wilson *et al.* [54]. Some algorithms, for e.g., RISE [16], even seek to reduce the storage requirements and speed up classification by modifying the instances themselves instead of just deciding on which ones to keep. Techniques such as *kd-trees* [47] and *projection* [36] can reduce clustering time, but still require the same memory. Reducing the size of the training set can reduce the storage requirements and time required for the processing. Several other issues can be considered in the move towards instance reduction.

- **Instance Representation** One choice of designing a training set reduction to decide whether to retain a subset of instances or whether to modify the instances using a different representation.
- **Direction of Search** Can be incremental, decremental or in a batch mode. As the names suggest, this involves the consideration of all, none, or a group of instances in the beginning, and then, depending on their importance in classification, they are removed or added to the next set of instances that would be considered.
- **Intuition of which instances to keep** Another factor that distinguishes nearest neighbor algorithms is whether they seek to retain the border, central or other points.
- **Choice of the distance function** Usually the Euclidean distance function is used, but others such as the Heterogeneous Euclidean-Overlap Metric (HEOM), Value Difference Metric (VDM), the Heterogeneous Value Difference Metric (HVDM), Interpolated Value Difference Metric (IVDM), and Windowed Value Difference Metric (WVDM) that are described in detail by Wilson *et al.* [54], can also be used.

Wilson *et al.* propose an approach for down-sizing the training data in case of exemplar-based learning algorithms, while trying to maintain or increase the generalization accuracy. The *decremental reduction optimization procedures* (DROP1-5) and *decremental encoding length* (DEL) procedure are summarized as follows.

- **DROP1: The basic algorithm** – Remove an instance, if at least as many of its associates (i.e., neighbors) in the whole set would be classified correctly without it. This algorithm tends to remove the noisy instances and the instances at the center of clusters, since their associates are quite likely to be classified correctly without them. In other words, this algorithm tends to retain non-noisy border points.
- **DROP2: Ordering the removal** There is a potential problem with DROP1, in case of noisy instances, if many of its neighbors are removed first. The removal of these noisy instances causes misclassification of their associates. DROP2 solves this problem by considering the effect of the removal of an instance on the whole of the training set instead of considering only those that are remaining at that time. In other words, an instance is removed from the training set only if at least as many of its associates (including those that have already been removed) are classified correctly without it.
- **DROP3: Filtering noise** DROP2 sorts the training set in order to remove the central points before the border points. One problem with this method is that the noisy instances are also border points and cause the order of removal to be drastically changed. A noisy point in the center of a cluster causes many internal points to be considered as border points, which will remain even after the noisy instance is removed. To overcome this, DROP3 uses a noise filtering pass, in which all the instances misclassified by their k nearest neighbors are removed, before sorting the instances.
- **DROP4: More careful noise filtering** This is very similar to DROP3, the only difference being that it does a more careful noise filtering. It removes the instance misclassified by its k-nearest neighbors only if its removal does not affect the classification of other instances.
- **DROP5: Smoothening the decision boundary** DROP5 modifies DROP2 in such a way that it removes instances starting with the instances that are nearest to their nearest enemies. Thus it repeatedly executes the removal pass performed by DROP2 till no further improvement is possible.
- **DEL or Decremental Length Encoding** It is very similar to DROP3, the only difference being that it uses the encoding length heuristic to decide in each case whether an instance can be removed. It uses decremental reduction and removes an instance if, over and above its misclassification by its k-nearest neighbors, it does not increase the encoding length cost.

Experiments have shown [55] that DROP1 has higher storage requirements as well as lower generalization accuracy, probably owing to the reason that it does not use

information provided by the pruned instances to determine whether further instances should be pruned. DROP2-DROP5 show generalization accuracy within 1% of all the full k-nearest neighbor classifiers ($k = 3$), at the same time reducing the storage down to 14%, which was comparatively a very good reduction. DROP3 has the highest generalization accuracy and at the same time the lowest storage requirements, among all the other DROP algorithms.

V. Heterogeneous Instance Selection

Recently, experiments have been done to study the scaling-up of inductive algorithms, like C4.5 [40] and HCV [58], while minimizing accuracy trade-offs associated with the decreased sample size, thus making the mining process feasible. Till date, different techniques viz. *windowing*, *random sampling*, *stratified sampling*, *peepholing* and *information-theoretic peepholing*, have been explored to this effect. Syed *et al.* [46]used SVM (support vector machines – a pattern recognition algorithm) and IBL (an instance based learning algorithm), to sample the databases. SVM technique was identified as a possible sampling strategy since, till date a significant amount of research has indicated the following characteristics [45], [43]:

- They concisely represent the whole population.[13]
- Different kernel functions can simulate different classifier architectures (for e.g., polynomial, Gaussian RBF and Neural Network classifiers).

SVM and IBL were used to select instances from the original whole database, and then, the same number of instances were selected using random and stratified sampling. The representativeness of these samples were used as training sets for classifiers C4.5 and HCV. However, the instances selected by SVM and IBL, failed to provide better accuracies with C4.5 and HCV, as compared to that generated by instances selected using random or stratified sampling. This might be attributed to the fact that both these algorithms, especially SVM, work with numeric attributes to generate the support vectors, which are data points that lie near the class boundaries, and therefore do not provide enough information for the decision tree induction algorithm C4.5 and rule induction algorithm HCV, to discriminate between the classes correctly.

It was seen in our experiments that random sampling can sometimes deliver catastrophic predictive accuracies, therefore, though it is the simplest method to extract samples, it cannot be completely relied upon.

Some researchers believe that a single classifier cannot be used to sample data for another classifier (refer Section II-E), so it can be tested whether a unique collection of the instances selected by these algorithms, can lead to a better training set. Also, the success of uncertainty sampling in Section II-E, might be attributed to the fact that it involves text data, and the fact that the size of the database, and the number of features involved are huge.

[13]Although it has been shown by Schölkopf [43], it has not been proven for heterogeneous classification process (similar to heterogeneous uncertainty sampling, refer to Section II-E) that we will be considering in this section), which may or may not benefit from the support vector set, which is consists, primarily, of boundary points.

Syed *et al.* [45] present an incremental algorithm using SVMs and propose a criterion for evaluating the goodness of this algorithm, to empirically determine whether the proposed technique performs well according to the given criterion. When a comparison was made between the following two alternative methods of creating the support vectors – one in which a set of support vectors from a previously learned database were used to classify the new information that comes in, by discarding the 'redundant' instances at successive incremental step, and the other, in which the whole data are saved and the SVM classification algorithm is used to classify them. It was found that the predictive accuracy of the first method showed the same trend as for the second one, thus bringing out empirical evidence that the support vectors could remember previously seen data concisely. To check the validity of the above observation, or to see whether the support vectors represented a non-redundant set of instances, another two experiments were carried out. In the first experiment, the whole data was used to train the SVM to generate support vectors. These support vectors were then used to classify the test data. In the other experiment, a different 10% of the support vectors were considered as the test data and the remaining were used to form the training set, thus checking whether the original support vectors represented a non-redundant set. Experimental results showed that there was considerable discrepancy between the predictive accuracies of the classification when the SVM was trained using the entire data and when it was trained using 90% of the support vectors, thus showing that the support vectors represented a non-redundant set of vectors, that concisely represented the data.

VI. Conclusions

Most research efforts on dealing with large databases in data mining have concentrated on scaling up inductive algorithms [37]. These efforts include the design of fast induction algorithms, data partitioning, and using relational representations. The instance selection approaches reviewed in this paper complement these efforts reviewed in [37]. We have not covered data partitioning in this paper, because it is not directly related to instance selection that selects or searches for a portion of a large database for use in data mining instead of using the whole database. Data partitioning (as in incremental batch learning [11] and multi-layer induction [59]) divides a large database into subsets and each subset is used in data mining either sequentially or in parallel.

Instance selection is different from bagging and boosting which generate multiple versions of a predictor (a decision tree or a set of rules) by running an existing learning algorithm many times on a set of re-sampled data. Bagging and boosting aim to improve the predictive accuracy by using multiple versions of a predictor, and a single instance can be selected again and again in the generation of the different versions of the predictor. Instance selection is also different from stacked generalization [57], [48] and hierarchical meta-learning [8], because in stacked generalization and hierarchical meta-learning, a high-level mining model

is used to combine lower-level models with the original data to achieve higher predictive accuracy.

References

[1] Aha, D., Kibler, D., and Albert, M., "Instance-Based Learning Algorithms", *Machine Learning*, Volume 6, 1991, 37-66.

[2] Angluin, D., and Laird, P., "Learning From Noisy Examples", *Machine Learning*, Volume 2, Number 4, 1987, 343-370.

[3] Bennet, K., and Blue, J., "An Extreme Point Tabu Search Method for Data Mining", *Rensselaer Polytechnic Institute Math Report: Number 228*, 1996.

[4] Blum, A., and Langley, P., "Selection of Relevant Features and Examples in Machine Learning", *Artificial Intelligence*, 1997, 245-271.

[5] Brin, S., and Page, L., "Dynamic Data Mining: Exploring Large Rule Spaces by Sampling", *Stanford University, Paper: Number 261.*, 1996.

[6] Burges, C., "A Tutorial on Support Vector Machines for Pattern Recognition", *Data Mining and Knowledge Discovery*, Volume 2, Number 2, 1998, 121-167.

[7] Catlett, J., "MegaInduction", *PhD Thesis – Basser Department of Computer Science, University of Sydney*, 1991.

[8] Chan, P., "An Extensible Meta-Learning Approach for Scalable and Accurate Inductive Learning", *PhD Dissertation, Dept of Computer Science, Columbia University*, 1996.

[9] Chaudhuri, S., Motwani, R., and Narasayya, V., "Random Sampling for Histogram Construction: How Much is Enough ?", *Proceedings of the ACM SIGMOD Conference*, 1998, 436-447.

[10] Chen, M., Han J., and Yu, P., "Data Mining: An Overview from a Database Perspective", *IEEE Transactions on Knowledge and Data Engineering*, Volume 8, Number 6, 1996, 866-883.

[11] Clearwater, S.H., Cheng, T.P., Hirsh, H., and Buchanan, B.G., "Incremental Batch Learning", *Proceedings of the Sixth International Workshop on Machine Learning*, 1989, 366–370.

[12] Cohen, W., "Fast Effective Rule Induction", *Proceedings of the Twelfth International Conference on Machine Learning*, 1995, 115-123.

[13] Cohn, D., Atlas, L., and Ladner, R., "Improving Generalization with Active Learning", *Machine Learning*, Volume 15, Number 2, 1994, 201-221.

[14] Cover, T., and Hart, P., "Nearest Neighbour Pattern Classification", *IEEE Transactions on Information Theory*, Volume IT-13, Number 1, 1967, 21-27.

[15] Craw, S., Sleeman, D., Graner, N., Rissakis, M., and Sharma, S., "Consultant: Providing Advice for the Machine Learning Toolbox", *Proceedings of Expert Systems 92: 12th Annual Technical Conference of the British Computer Society Specialist Group on Expert Systems*, 1992, 5-23.

[16] Domingos, P., "Rule Induction and Instance-Based Learning: a Unified Approach", *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI)*, 1995, 1226-1232.

[17] Freund, Y., Seung, S., Shamir, E., and Tishby, N., "Selective Sampling Using the Query by Committee Algorithm", *Machine Learning*, Volume 28, 1997, 133-168.

[18] Fürnkranz, J., "More Efficient Windowing", *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, 1997, 509-515.

[19] Fürnkranz, J., "Noise-Tolerant Windowing", *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, 1997, 852-857.

[20] Fürnkranz, J., "Integrative Windowing", *Journal of Artificial Intelligence Research*, Volume 8, 1998, 129-164.

[21] Gates, G., "The Reduced Nearest Neighbour Rule", *IEEE Transactions on Information Theory*, Volume IT-18, Number 3, May 1972, 431-433.

[22] Glover, F., "Tabu Search Methods in Artificial Intelligence and Operations Research", *ORSA Artificial Intelligence*, Volume 1, Number 2, 1987.

[23] Hart, C., "The Condensed Nearest Neighbour Rule", *IEEE Transactions on Information Theory*, Volume 14, 1968, 515-516.

[24] Joachims, T., "Making Large-Scale SVM Learning Practical", *Advances in Kernel Methods – Support Vector Learning*, Schölkopf, B., Burges, C. , and Smola, A. (*eds.*), MIT Press, 1998.

[25] John, G., and Langley, P., "Static vs. Dynamic Sampling for Data Mining", *Proceedings of the Second International Conference of Knowledge Discovery and Data Mining*, 1996, 367-370.

[26] Kivinen, J., and Mannila, H., "The Power of Sampling in Knowledge Discovery", *Proceedings of the Thirteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 1994, 77-85.

[27] Lewis, D., and Catlett, J., "Heterogeneous Uncertainty Sampling for Supervised Learning", *Proceedings of the Eleventh International Conference on Machine Learning*, 1994, 148-156.

[28] Mangasarian, O., "Mathematical Programming in Data Mining", *University of Wisconsin, Mathematical Programming Technical Report: 96-05*, 1996.

[29] Mannila, H., "Methods and Problems in Data Mining", *Proceedings of the International Conference on Database Theory*, 1997, 41-55.

[30] Mercz, C., and Murphy, P., *UCI ML Repository*, Department of Information and Computer Science, `http://www.ics.uci.edu/ mlearn/MLRepository.html`.

[31] Mϕller, M., "Supervised Learning on Large Redundant Training Sets", *International Journal of Neural Systems*, Volume 4, Number 1, 1993, 15-25.

[32] Musick, R., Catlett, J., and Russel, S., "Decision Theoretic Subsampling for Induction on Large Databases", *Proceedings of The Tenth International Conference on Machine Learning*, 1993, 212-219.

[33] Oates, T., and Jensen, D., "Large Datasets Lead to Overly Complex Models: an Explanation and a Solution", *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, 1998, 294-298.

[34] Olken, F., and Rotem, D., "Random Sampling from Database Files: A Survey", *Proceedings of the Fifteenth International Conference (SSDBM) on Statistical and Scientific Database Management*, 1990, 92-111.

[35] Palmer, C., and Faloutsos, C., "Density Biased Sampling: An Improved Method for Data Mining and Clustering", *Carnegie Mellon University, Technical Report: CMU-CS-99-113*, May 26, 1999.

[36] Papadimitriou, C., and Bentley, J., "A Worst-Case Analysis of Nearest Neighbour Searching by Projection", *Lecture Notes in Computer Science: Automata Languages and Programming*, 1980, 470-482.

[37] Provost, F.J, Kolluri, V., "A Survey of Methods for Scaling Up Inductive Algorithms", *Data Mining and Knowledge Discovery*, Volume 3, Number 2, 1999, 131-169.

[38] Provost, F., Jensen, D., and Oates, T., "Efficient Progressive Sampling", *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, 1999, 23-32.

[39] Quinlan, J., "Induction of Decision Trees", *Machine Learning*, Volume 1, Number 1, 1986, 81-106.

[40] Quinlan, J., "C4.5 : Programs for Machine Learning", Morgan Kaufmann Publishers, 1993.

[41] Ritter, G., Woodruff, H., Lowry, S., and Isenhour, T., "An Algorithm for a Selective Nearest Neighbour Decision Rule", *IEEE Transactions on Information Theory*, Volume 21, Number 6, November 1975, 666-669.

[42] SAS Institute's "Enterprise Miner", `http://www.sas.com/software/components/miner.html`

[43] Schölkopf, B., Burges, C., and Vapnik, V., "Extracting Support Data for a Given Task", *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, 1995, 252-257.

[44] Seung, H., Opper, M., and Sompolinsky, H., "Query by Committee", *Proceedings of the Fifth Workshop on Computational Learning Theory*, 1992, 287-294.

[45] Syed, N., Liu, H., and Sung, K., "Incremental Learning with Support Vector Machines", *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-99)*, 1999.

[46] Syed, N., Liu, H., and Sung, K., "A Study of Support Vectors on Model Independent Example Selection",*Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-99)*, 1999, 272-276.

[47] Sproull, R., "Refinements to Nearest-Neighbour Searching in k-Dimensional Trees", *Algorithmica*, Volume 6, 1991, 579-589.

[48] Ting, K. and Witten, I.H., "Stacked Generalization: When Does It Work?", *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-97)*, 1997, 866–871.

[49] Ting, K., Low, B., and Witten I., "Learning from Batched Data: Model Combination Versus Data Combination", *Knowledge and Information Systems*, Volume 1, 1999, 83-106.

[50] Tsumoto, S., and Tanaka, H., "Automated Selection of Rule Induction Methods Based on Recursive Iteration of Resampling

Methods and Multiple Statistical Testing", *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, 1995, 312-317.

[51] Utgoff, P., "Incremental Induction of Decision Trees", *Machine Learning*, Volume 4, 1989, 161-186.

[52] Vitter, J., "Random Sampling with a Reservoir", *ACM Transactions on Mathematical Software*, Volume 11, 37-57.

[53] Wal-Mart, `http://www.wal-mart.com/newsroom/8_17_expand.html`

[54] Wilson, D., and Martinez, T., "Improved Heterogeneous Distance Functions", *Journal of Artificial Intelligence Research*, Volume 6, 1997, 1-34.

[55] Wilson, D., and Martinez, T., "Reduction Techniques for Examplar-Based Learning Algorithms", To appear in *Machine Learning*, Volume 38, Number 3, 2000.

[56] Wirth, J., and Catlett, J., "Costs and Benefits of Windowing in ID3", *Proceedings of the Fifth International Conference on Machine Learning*, 1988, 87-99.

[57] D.H. Wolpert, "Stacked Generalization", *Neural Networks*, Volume 5, 1992, 241–259.

[58] Wu, X., "Rule Induction with Extension Matrices", *Journal of the American Society for Information Science*, Volume 49, 1998, 435-454.

[59] Wu, X. and Lo, W., "Multi-Layer Incremental Induction", *Proceedings of the 5th Pacific Rim International Conference on Artificial Intelligence*, 1998, 24–32.

[60] Yang, Y., "Sampling Strategies and Learning Efficiency in Text Categorization", *AAAI Spring Symposium on Machine Learning in Information Access*, 1996, 88-95.

[61] Zaki, M., Parthasarathy, S., Li, W., and Olighara, M., "Evaluation of Sampling for Data Mining of Association Rules", *Proceedings of the Seventh International Workshop on Research Issues in Data Engineering*, 1997, 42-50.

[62] Zhang, T., Ramakrishnan, R., and Livny, M., "BIRCH: An Efficient Data Clustering Method for Very Large Databases", *Proceedings of the ACM SIGMOD Conference*, 1996, 103-114.

[63] Zipf, G., "Human Behaviour and Principle of Least Effort: An Introduction to Human Ecology", *Addison Wesley*, 1949.