Data Mining and Knowledge Discovery, 6, 115–130, 2002

On Issues of Instance Selection

HUAN LIU hliu@asu.edu
Department of Computer Science and Engineering, Arizona State University, Tempe, Arizona, USA

HIROSHI MOTODA motoda@sanken.osaka-u.ac.jp
Institute of Scientific and Industrial Research, Osaka University, Osaka, Japan

1. Background and motivation

The digital technologies and computer advances with the booming internet uses have led to massive data collection (corporate data, data warehouses, webs, just to name a few) and information (or misinformation) explosion. Szalay and Gray described this phenomenon as "drowning in data" (Szalay and Gray, 1999). They reported that each year the detectors at the CERN particle collider in Switzerland record 1 petabyte of data; and researchers in areas of science from astronomy to the human genome are facing the same problems and choking on information. A very natural question is "now that we have gathered so much data, what do we do with it?" Raw data is rarely of direct use and manual analysis simply cannot keep pace with the fast growth of data. Data mining and knowledge discovery (KDD), as a new emerging field comprising disciplines such as databases, statistics, machine learning, comes to the rescue. KDD attempts to turn raw data into nuggets and create special edges in this ever competitive world for science discovery and business intelligence.

The KDD process is defined in Fayyad et al. (1996) as *the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data*. Data Mining processes include data selection, preprocessing, data mining, interpretation and evaluation. The first two processes (data selection and preprocessing) play a pivotal role in successful data mining. Facing the mounting challenges of enormous amounts of data, much of the current research concerns itself with scaling up data mining algorithms (Provost and Kolluri, 1999). Researchers have also worked on scaling down the data (Liu and Motoda, 1998b; Reinartz, 1999)—an alternative to the algorithm scaling-up. The major issue of scaling down data is to select the relevant data and then present it to a data mining algorithm. This line of work is in parallel with the work on algorithm scaling-up and the combination of the two is a two edged sword in mining nuggets from massive data sets.

Selection pressures are present everywhere. They can be found in scientific enterprises, organizations, business environments, human being or even bacteria evolution. Making a decision is also about selecting among choices. Selection seems a necessity in the world surrounding us. It stems from the sheer fact of limited resources. No exception for data mining. Many factors give rise to data selection. First, data is not purely collected for data mining or for one particular application. Second, there are missing data, redundant data, and errors during collecting and recording. Third, data can be too overwhelming to

handle. Instance selection is one avenue to the empire of data selection. Data is stored in a *flat file* and described by terms called *attributes* or *features*. Each line in the file consists of attribute-values and forms an *instance*, also named as a *record, tuple*, or a *data point* in a multi-dimensional space defined by the attributes. Data reduction can be achieved in many ways. By selecting features (Blum and Langley, 1997; Liu and Motoda, 1998a), we reduce the number of columns in a data set; by discretizing feature-values (Fayyad and Irani, 1993; Bloedorn and Michalski, 1998; Hussain et al., 1999), we reduce the number of possible values of discretized features; and by selecting instances, we reduce the number of rows in a data set (Blum and Langley, 1997). It is the last on which we focus in this collection. Instance selection is to choose a subset of data to achieve the original purpose of a data mining application as if the whole data is used. Many variants of instance selection exist such as squashed data (DuMouchel et al., 1999), critical points (Aha et al., 1991), boarder cases (Burges, 1998), prototype construction (or instance averaging) (Chang, 1974), in addition to many forms of sampling.

The ideal outcome of instance selection is a model independent, minimum sample of data that can accomplish tasks with little or no performance deterioration, i.e., for a given data mining algorithm M, its performance $\mathcal{P}$ on a sample s of selected instances and on the whole data w is roughly

$$\mathcal{P}(M_s) \doteq \mathcal{P}(M_w).$$

By model independence, we mean that for any two data mining algorithms M_i and M_j, let $\Delta\mathcal{P}$ be the performance difference in using data s with respect to using data w,

$$\Delta\mathcal{P}(M_i) \doteq \Delta\mathcal{P}(M_j).$$

Performance issues of instance selection will be elaborated in Section 3.

Although selection pressures always exist, the surge of data mining applications and wide availability of data mining algorithms/products give rise to the pressing need for data reduction. Among many, one well studied topic is feature selection, extraction and construction (Liu and Motoda, 1998a). Instance selection, as another topic for data reduction, is recently getting more and more attention from researchers and practitioners. There are many reasons for this new trend: first, instance selection concerns some aspects of data reduction that feature selection cannot blanket; second, it is possible to attempt it now with advanced statistics and accumulated experience; and third, doing so can result in many advantages in data mining applications.

The issue of instance selection comes to the spot light because of the vast amounts of data and increasing needs of preparing data for data mining applications. More often than not, we have to perform instance selection in order to obtain meaningful results. Instance selection has the following prominent functions:

Enabling Instance selection renders the impossible possible. As we know, every data mining algorithm is somehow limited by its capability in handling data in terms of sizes, types, formats. When a data set is too huge, it may not be possible to run a data mining algorithm or the data mining task cannot be effectively carried out without data reduction.

Instance selection reduces data and enables a data mining algorithm to function and work effectively with huge data.

Focusing The data includes almost everything in a domain (recall that data is not solely collected for data mining), but one application is normally only about one aspect of the domain. It is natural and sensible to focus on the relevant part of the data for the application so that search is more focused and the mining is more efficient.

Cleaning The GIGO (garbage-in-garbage-out) principle applies to almost all, if not all, data mining algorithms. It is therefore paramount to clean data, if possible, before mining. By selecting relevant instances, we can usually remove irrelevant ones as well as noise and/or redundant data. The high quality data will lead to high quality results and reduced costs for data mining.

The above three main functions of instance selection may intertwine. For example, cleaning can sometimes be a by-product of the first two. Focusing can also serve a function of enabling under certain circumstances.

From its data aspect, instance selection takes the whole data w as input and churns out a sample s of the data as output. Although benefits reaped from this exercise are many, instance selection does require resources (machine, labor, and time). Therefore, we need to consider the gain vs. the loss due to instance selection. This problem is tightly associated with another two issues: (1) sample size and (2) mining quality. Usually, the more (relevant) data, the better the mining quality. In the context of data reduction, we are required to reduce data but maintain the mining quality. Hence, in selecting instances, we often face an issue of trade-off between the sample size and the mining quality. In this sense, instance selection is an optimization problem that attempts to maintain the mining quality while minimizing the sample size. The issue of measuring the mining quality is associated with the sample quality issue in which we aim to achieve a true representative sample with the minimum size.

Clearly, instance selection is not a simple matter. This special issue is to address the difficult issues related to instance selection in KDD, to call for the sharing of research results and accomplishments, and to join forces to advance this field with practical impact. In the next section, we will briefly review major lines of research and development in instance selection. The first common reaction to instance selection is *random sampling*, but we shall see that there are works other than random sampling. In the end of Section 2, we attempt to gain insights from the existing methods in our search of the next generation of instance selection methods. In Section 3, we are concerned about the evaluation issues of instance selection such as what and how to measure instance selection. Section 4 is about the related work on data reduction or performance improvement. Section 5 is a brief introduction of the four articles selected for this special issue with emphasis on their distinctive contributions. In the last section, we summarize this article and project further research, development, and application for instance selection.

2. Major lines of research and development

A spontaneous response to the challenge of instance selection is, without fail, some form of sampling. Although it is an important part of instance selection, there are other approaches

that do not rely on sampling, but resort to search or take advantage of data mining algorithms. As we can select the best feature to build a decision tree (stump), for example, we can also select the most representative data points for a decision tree induction algorithm to efficiently divide the data space. In the following, we start with sampling methods, and proceed to other instance selection methods associated with data mining tasks such as classification and clustering.

2.1. Sampling

Sampling is a procedure that draws a sample S_i by a random process in which each S_i receives its appropriate probability π_i of being selected. In practice we seldom draw a probability sample by writing down the S_i and π_i because it is intolerably laborious with a large data set, where sampling may produce billions of possible samples. The draw is most commonly made by specifying probabilities of inclusion for the individual instances and drawing instances, one by one or in groups until the sample of desired size is constructed (Cochran, 1977). It is not without a reason that people tend to first think of sampling as a tool for instance selection. Sampling methods are useful tools (Kivinen and Mannila, 1994), have existed for quite many years and are available in many system packages.

Sampling does not consciously search for relevant instances. One cannot help asking "how are the three functions (enabling, focusing and cleaning) of instance selection accomplished in sampling?" What does the wonder is the random mechanism underlying every sampling method. Enabling and cleaning are possible as the sample is usually smaller than the original data and noise and irrelevant instances in the sample will become accordingly less if sampling is performed appropriately. Although it does not take into account the task at hand, some forms of sampling can, to a limited extent, help focusing. We present some commonly used sampling methods below.

Simple random sampling It is a method of selecting n instances out of the N such that every one of the $\binom{N}{n}$ distinct samples has an equal chance of being drawn. In practice a simple random sample is drawn instance by instance. Since an instance that has been drawn is removed from the data set for all subsequent draws, this method is also called random sampling without replacement. Random sampling with replacement is entirely feasible: at any draw, all N instances of the data set are given an equal chance of being drawn, no matter how often they have already been drawn. This is the most frequently used and seen sampling method.

Stratified random sampling The data set of N instances is first divided into subsets of $N_1, N_2, \ldots, N_l$ instances, respectively. These subsets are nonoverlapping, and together they comprise the whole of the data set (i.e., $\sum_i^l N_i = N$). The subsets are called strata. When the strata have been determined, a sample is drawn from each stratum, the drawings being made independently in different strata. The sample sizes within the strata are denoted by $n_1, n_2, \ldots, n_l$, respectively. If a simple random sample is taken in each stratum, the whole procedure is described as stratified random sampling. It is often used in some applications that we wish to divide a heterogeneous data set into subsets, each of which is internally homogeneous. Its close cousins are systematic sampling and cluster sampling.

Adaptive sampling In many sampling applications, one may feel inclined to make decision on further sampling based on what has been observed so far. Adaptive sampling refers to a sampling procedure for selecting instances to be included in the sample that depends on results obtained from the sample. The primary purpose of adaptive sampling is to take advantage of data characteristics in order to obtain more precise estimates. It can be considered as sampling and mining are performed side by side to take advantage of the result of preliminary mining for more effective sampling and vise versa. Its variants can be found in sequential sampling and progressive sampling.
Applying sampling methods The simplest way of applying sampling is to go through the data once and obtain a sample, then work on the sample only. However, better results can often be achieved following the common pattern below:

– Obtain some approximate mining result with a sufficiently large sample s,
– Refine the result with the whole data w.

The issue of sample size is nontrivial and will be discussed separately later. The above pattern is an ideal case as we only need to scan the whole data set twice. However, in some cases, a few samples may be needed in the first step so that sampling is conducted a few times. In some cases, the above pattern (both steps) may be repeated a few times. If a mining algorithm is data hungry and memory intensive, it is obvious that one should maximize the use of the data obtained in Step 1 and minimize the use of the whole data; the ideal case would be going through Step 2 only once.
Determining a sample size This issue can be dealt with theoretically and empirically. There are sampling theories and learning theories that tell us how large the sample should be for what kind of results we expect. In general, the more precise the result we want, the larger the sample size. Examples can be found in Mitch (1997) in which PAC learning theory (Valiant, 1984), the VC dimension (Vapnik, 1995), and Chernoff Bounds are mentioned for determining sample sizes. Learning curves are commonly used in empirical studies of a required sample size (Provost et al., 1999). As the theoretical results are usually worst-case and learning curves are average-case, they are not necessarily consistent with each other. Some more examples can be found in (Piatetsky-Shapiro and Connell, 1984; Harris-Jones and Haines, 1997).

Moreover, in data mining applications, we are often constrained by the computer's memory capacity in processing data with a limited size. Besides the efficiency issue of the mining algorithm (e.g., a decision tree induction algorithm vs. a neural network one), we often end up using as much data as the computer can take. In other words, theoretical bounds help little in this situation. With increased memory capacity, we will be better and better guided by theoretical bounds in determining sample size.

Sample size is also related to mining quality as the average-case learning curves usually suggest. However, samples of the same size could vary in terms of their qualities. In particular, some samples are more representative or resembling the original data than others. Hence, there is a need for measuring sample quality; we then wish to establish the positive correlation between sample quality and mining quality.

2.2. *Methods associated with classification*

Let us now examine the alternatives to instance selection that rely on *search*. Methods in this category generally count on data mining applications. One of the key data mining applications is classification. It is to discover patterns from data with two purposes: (1) classification—grouping the data points in terms of common properties, and (2) prediction—predicting the class of an unseen instance. The data for this type of application is usually labeled with so-called class values. Instance selection in the context of classification has been attempted by researchers according to the classifiers being built. Instance selection here clearly serves the purposes of enabling, focusing, and cleaning. Besides the explicitness shown in ways of selecting instances, sample size can be automatically determined in the process. We include below four types of selected instances.

Critical points They are the points that matter the most to a classifier. The issue was originated from the learning method of Nearest Neighbor (NN) (Cover and Hart, 1967). Instead of generalizing the data, NN does nothing in terms of learning;[1] only when it is required to classify a new instance does NN search the data to find the nearest neighbor in the data for the new instance and use the nearest neighbor's class value to predict it. During the latter phase, NN could be very slow if the data is large and be extremely sensitive to noise. Therefore, many suggestions have been made to keep only the critical points so that noisy data points are removed as well as the data set is reduced. Here we use Instance-Based Learning (IBL) (Aha et al., 1991) as an example of NN to illustrate the concept of critical points. IB1 is a modified version of NN with normalized feature values. IB2 is similar to IB1 but saves only misclassified instances (critical points) so that the storage requirements for IB2 can be significantly smaller than IB1. The idea is that instances near the concept boundaries are possible to wrongly classified. IB2 is prone to noise as noisy instances are almost always misclassified. An improve version of IB2 is IB3 that can better handle noise.

Boundary points They are the instances that lie on borders between classes. Support vector machines (SVM) provide a principled way of finding these points through minimizing structural risk (refer to a tutorial in Burges, 1998). Using a non-linear function ϕ to map data points to a high-dimensional feature space, a non-linearly separable data set becomes linearly separable. Data points on the boundaries, which maximize the margin band, are the support vectors. In Scholkopf et al. (1995), they apply the SVM algorithm to reduce data by only keeping the support vectors as they contain all the information a given classifier needs for constructing the decision function. Support vectors are instances in the original data sets. Boundary points and critical points are different in the ways how they are found.

Prototypes They are representatives of groups of instances via averaging (Chang, 1974). A prototype that represents the typicality of a class is used in characterizing a class, instead of describing the differences between classes. Therefore they are different from critical points or boundary points.

Tree based sampling Decision trees (Breiman et al., 1984; Quinlan, 1993) are a commonly used classification tool in data mining and machine learning. An attribute is chosen to split the data forming branches when we grow a tree from its root. It is a recursive process and stops when some stopping criterion is satisfied (for example, a fixed number of layers has been reached or all instances are classified). In order to build a compact yet accurate tree,

some attribute selection criteria have been used; among many, Gini (Breiman et al., 1984) and Information Gain (Quinlan, 1993) are widely used. Instance selection can be done via the decision tree built. In Breiman and Friedman (1984), they propose *delegate sampling*. The basic idea is to construct a decision tree such that instances at the leaves of the tree are approximately uniformly distributed. Delegate sampling then samples instances from the leaves in inverse proportion to the density at the leaf and assigns weights to the sampled points that are proportional to the leaf density.

2.3. Methods associated with clustering

When data is unlabled, methods associated with classification algorithms cannot be directly applied to instance selection. The widespread use of computers results in huge amounts of data stored without labels (web pages, transaction data, newspaper articles, email messages) (Baeza-Yates and Ribeiro-Neto, 1999). Clustering is one approach to finding regularities from unlabled data. Being constrained by limited resources, we cannot just simply dump all the data to a computer (no matter how powerful the computer is) and let it crunch. Besides, in many applications, the experience from practice suggests that not all instances in a data set are of equal importance to the description of clustering. In the following, we discuss three types of selected instances in clustering.

Prototypes They are pseudo data points generated from the formed clusters. The basic idea is that after the clusters are found in a space of high dimension, one may just keep the prototypes of the clusters and discard the rest. The k-means clustering algorithm is a good example of this sort. Given a data set and a constant k, the k-means clustering algorithm is to partition the data into k subsets such that instances in each subset are similar under some measure. The k means are iteratively updated until a stopping criterion is satisfied. The prototypes in this case are the k means.

Prototypes & sufficient statistics In Bradley et al. (1998), they extend the k-means algorithm to perform clustering in one scan of the data. By keeping some points that defy compression plus some sufficient statistics, they demonstrate a scalable k-means algorithm. From the viewpoint of instance selection, it is a method of representing a cluster using both defiant points and pseudo points that can be reconstructed from sufficient statistics, instead of keeping only the k means.

Data description in a hierarchy When the clustering produces a hierarchy (concepts of different details (Fisher, 1987) or taxonomy (Everitt, 1974)), it is obvious that the prototype approach to instance selection will not work as there are many ways of choosing (defining) appropriate clusters. In other words, it is an issue about which layer in the hierarchy is the most representative description (or basic concepts that are neither too general nor too specific). This is because at the leaf level, each instance forms a cluster, and at the root level, all instances are under one cluster. In COBWEB (Fisher, 1987), for example, when basic concepts are determined, pseudo data points are the descriptions of basic concepts in probability distributions over the space of possible attribute values.

Squashed data They are some pseudo data points generated from the original data. In this aspect, they are similar to prototypes as both may or may not be in the original data set.

Squashed data are different from prototypes in that each pseudo data point has a weight and the sum of the weights is equal to the number of instances in the original data set. Presently two ways of obtaining squashed data are (1) model free (DuMouchel et al., 1999) and (2) model dependent (likelihood based (Madigan et al., 2002)).

In DuMouchel et al. (1999), the approach to squashing is model-free and relies on moment-matching to ensure that the original data and the squashed data are sufficiently similar. In other words, the squashed data set is attempted to replicate the moments of the original data. In Madigan et al. (2002), the approach to squashing is to declare a statistical model in advance for data squashing. Their approach partitions the data using a likelihood-based clustering and then selects pseudo data points in order to mimic the posterior distribution. They show that the data squashed this way can also be used in models other than that used for the squashing.

2.4. Instance labeling

In real world applications, classification models are extremely useful analytic techniques. Although large amounts of data are potentially available, the majority of data are not labeled as we mentioned earlier in Section 2.3. Manually labeling the data is a labor intensive and costly process. Researchers investigate whether experts are asked to only label a small portion of the data that means the most to the task if it is too expensive and time consuming to label all data. This is equivalent to a problem of selecting which unlabeled data for labeling. Usually an expert can be engaged to label a small portion of the selected data at various stages. So we wish to select as little data as possible at each stage, and use an adaptive algorithm to guess what else should be selected for labeling in the next stage. Clustering techniques are often used to group unlabeled data in the beginning. Instance labeling is closely associated with adaptive sampling (discussed earlier in Section 2.1), clustering, active learning (McCallum and Nigam, 1998; Cohn et al., 1994), and boosting (Freund, 1994, 1995). The last two topics will be discussed in the related work (Section 4).

2.5. Towards the next step of instance selection

As shown above, instance selection has been studied and employed in various tasks (sampling, classification, clustering, and instance labeling). The life would become much simpler if we could have a universal model of instance selection. The most important of all is that we could have a component of instance selection available as a library function for applications (and it is transparent to the user). The tasks shown above are, however, very unique in themselves as each task has different information, requirements, and underlying principles. It is clear that a universal model of instance selection is out of the question. However, is it possible for us to generalize what we have discussed so far, explore their properties to invent new and better instance selection methods? In Syed et al. (1999b), for example, they investigated if support vector machines could be used as a general model for instance selection in classification. The reason to choose SVMs is the nice features exhibited by SVMs and the statistical learning theory under SVMs—it aims to minimize both empirical risk

and structural risk. The learning algorithms included in the limited scope of investigation are C4.5 (Quinlan, 1993), SVMs (Joachims, 1998), Multilayered Perceptron, IBL (instance based learning) (Aha et al., 1991). Since the instances selected by SVMs were empirically shown to be worse than the sample selected by random sampling for some other learning algorithms, the conclusion is that SVMs cannot serve as a universal model for instance selection with respect to classification. An instant derivation of this finding is that different groups of learning algorithms need different instance selectors in order to suit their learning/search bias well (Brodley, 1995). This negative results should not deter our efforts in searching for a specific universal model of instance selection. It is hoped that this collection of works by various researchers can lead to more concerted study and development of new methods for instance selection. This is because even modest achievements along this line will harvest great profits in data mining applications: this would allow the selected instances to be useful for a group of mining algorithms in solving real world problems.

3. Evaluation issues of instance selection

Naturally, other things being equal, an instance selection algorithm would be the best if it achieves the most reduction. However, we are not content with sheer data reduction. The goal of instance selection is multi-fold.Therefore, evaluation of instance selection is also related to tasks for which instance selection is performed. Some examples are (1) *Sampling*—performance is of sufficient statistics and can depend on the data distribution. For example, if it is of normal distribution, means and variances are the two major measures, higher moments are also used to check if sufficient statistics are preserved (DuMouchel et al., 1999). (2) *Classification*—performance is more about predictive accuracy as well as aspects such as comprehensibility and simplicity, when applicable. (3) *Clustering*—performance is naturally about clusters: inter- and intra-cluster similarity, shapes of clusters, number of clusters, etc.

In practice, researchers often conduct experimental studies to evaluate performance. One can divide evaluation methods into two categories: *direct* and *indirect* measures. Direct measures work only with data and indirect measures involve at least a data mining algorithm. The basic idea for direct measures is to keep as much resemblance as possible between the selected data and the original data so that the data selected can be used by any data mining algorithms. Some examples of direct measures are entropy (Cover and Thomas, 1991), moments (Smith, 1998), and histograms (Chaudhuri et al., 1998).

Ironically, it would be more direct to see the effect of instance selection using indirect measures. For example, a classifier can be used to check whether instance selection results in better, equal, or worse predictive accuracy. The more sophisticated version of measuring predictive accuracy is to engage multiple runs of cross validation (Weiss and Kulikowski, 1991; Weiss and Indurkhya, 1998). In many cases, labeled data sets are used to measure the quality of clusters. The class labels are not used during clustering. After clustering, for each class, we can define precision and recall to evaluate the clusters: how many data points in a cluster are correctly assigned over the total data points in the cluster—the percentage of the two is *precision*, and how many correctly assigned data points are in a cluster with respect to the total data points that should belong to the cluster—the percentage of the two

is *recall*. One can observe the effect of instance selection by comparing the two sets of precision and recall before and after instance selection. In the case that class labels are not known, one can examine how different the standard clustering measures (inter- and intra-cluster similarity, number of clusters, etc.) are before and after instance selection. In short, conventional evaluation methods in sampling, classification, and clustering can be used in assessing the performance of instance selection.

Since instance selection is usually a preprocessing step between data and mining, one should also be aware of two practical and important points in instance selection: *platform compatibility* and *database protection*. The former suggests that the subset of selected instances should be compatible with the mining algorithm used in application. For example, if a data mining algorithm cannot take weighted instances, then it does not make sense to have squashed data as selected instances. The latter requires that under any circumstances, the original data should be kept intact. It is also wise to compare any new instance selection algorithm with a basic sampling method (say random sampling) because the latter is simple and fast, and maintains the compatibility between the original data and the selected instances.

The bottom line performance measures are *time complexity* and *space complexity*. The former is about the time required for instance selection to be performed. The latter is about the space needed during instance selection. Time spent on instance selection should, in general, be no more than what a data mining algorithm requires. The space requirement for instance selection should not be as demanding as data mining. Instance selection stems from the need that we have to reduce the data as it is just too much to handle in some applications. Nevertheless, we can sometimes relax the requirements on time and space a little because instance selection is normally performed once in a while. In other words, even if it requires similar time or space complexities as a data mining algorithm does, it may be still acceptable.

4. Related work

Instance selection is just one of many data reduction techniques. We briefly review some related techniques that can be used either directly or indirectly to reduce data or to identify distinguish instances or attributes. Considering data as a flat file as is in instance selection, we can transpose an instance selection problem to a problem of attribute selection. An updated collection on feature selection, extraction and construction (Liu and Motoda, 1998a) provides a comprehensive range of techniques in this regard. The basic link of this work to instance selection is that we can reduce duplicates of instances by removing irrelevant attributes. Take an example of feature selection (Blum and Langley, 1997). Many evaluation measures have been designed for feature selection. In most experimental cases, the number of features can be reduced drastically without performance deterioration, or sometimes with performance improvement. This type of work, however, is not directly designed for instance selection, but can indirectly reduce instances by removing duplicates.[2]

Another related piece of work is feature discretization (see a recently completed survey of several discretization algorithms in Hussain et al. (1999)). It is a technique to discretize continuous attributes into intervals (or feature quantization). The original motivation is that some mining or learning algorithms can only deal with discrete values, so continuous

values have to be discretized first. Some discretization algorithms can play a role of feature selection as those features with one value after discretization can be removed. Therefore discretization of feature values into intervals can also indirectly remove some instances.

Boosting (Schapire, 1990; Freund, 1995) is, in a general sense, of instance selection though there is no data reduction involved. It is a general method which attempts to boost the accuracy of any given learning algorithm. One of the main ideas of the AdaBoost algorithm (Freund and Schapire, 1997) is to maintain a distribution or set of weights over the instances. It repeatedly updates the weights after generating weak hypotheses. The wrongly classified instances in many iterations will have larger weights, otherwise the instances will have smaller weights. At the termination of the algorithm, the difficult instances for the chosen learner will have higher weights.

Active learning (Cohn et al., 1994, 1996) also concerns instance selection. It assumes some control over the subset of the input space and is an approach to improving learning performance by identifying a specific set of data during learning instead of employing all data available. The key issue is to choose instances most needed for the learning at a particular learning time. One way is to measure uncertainty of the estimated predictions—uncertainty sampling (Lewis and Gale, 1994; Lewis and Catlett, 1994). The degree of uncertainty influences the amount of data as well as which subset of data. Query-by-committee is another way of choosing instances for learning (Seung et al., 1992). The disagreement on a certain query among the committee will cause more data related to the query to be selected.

Incremental learning, in a classical definition (Utogoff, 1989; Langley, 1996), uses the data when it is available maintaining a "best-so-far" concept. The subsequently available data is used to revise and improve the concept. However, not every classifier is capable of learning incrementally. Incremental learning can be extended to a relaxed version—keeping the core data instead of maintaining a concept. Instance selection plays a role in selecting the core data to alleviate the storage problem. In Syed et al. (1999a), they examine this type of incremental learning for support vector machines and show that support vectors can be used as the core data for learning in the next phase.

In the work of on-line analytic processing (OLAP), there is one task that is related to instance construction, i.e., aggregated or summarized data in a data warehouse (Devlin, 1997). It's really about focusing and efficient online analytic processing. Aggregation strategies rely on the fact that most common queries will analyze either a subset or an aggregation of the detailed data. Aggregation consists of grouping the data according to some criterion and totaling, averaging, or applying some other statistical methods to the resultant set of data. The definition of a specific aggregation is driven by the business need, rather than by the perceived information need. The appropriate aggregation will substantially reduce the processing time required to run a query, at the cost of preprocessing and storing the intermediate results.

5. Articles in this special issue

The four articles in this special issue challenge the problem of instance selection from different angles and complement each other. We can find three specific techniques (Madigan et al., 2002; Domingo et al., 2002; Brighton and Mellish, 2002) vs. a general framework

(Reinartz, 2002); an on-line (incremental) approach (Domingo et al., 2002) vs. off-line (batch) ones (Brighton and Mellish, 2002; Madigan et al., 2002); task-generic treatment (Domingo et al., 2002; Madigan et al., 2002; Reinartz, 2002) vs. task-specific one (Brighton and Mellish, 2002); selection (Domingo et al., 2002) vs. removal (Brighton and Mellish, 2002); and filtering (Domingo et al., 2002; Brighton and Mellish, 2002) vs. construction (Madigan et al., 2002). Domingo et al. (2002) addresses the problem of deciding an appropriate sampling size. They propose an adaptive on-line sampling method that solves this problem. Sampling size is not fixed a priori but adaptively decided based on the samples so far seen. This adaptiveness allows in many cases to solve a problem with a much smaller number of instances than that required for the worst case. Their problem is to select a nearly optimal hypothesis from a finite known hypothesis space by using a fraction of data that are obtained by on-line sequential sampling without going through the whole data. Optimality is measured by a utility function that can be calculated by the sampled data, thus it is not limited to a specific task. However, since it assumes a hypothesis space which is not very large, it is meant to be used as a tool to solve some sub-tasks that fit in the framework. They show that there are many potential sub-tasks that this method can be used effectively.

Brighton and Mellish (2002) focuses on the instance selection problem of nearest neighbor (NN) classifiers. Thus, the data mining task is classification. They argue that there is no free lunch in that a single instance selection method can work for all data sets, and emphasize that the structure of class definition (e.g., whether the class distribution is homogeneous or heterogeneous in the feature space) is a decisive factor. Their method works in batch mode as many NN clasifiers do, and targets at homogeneous class distribution. They introduce, for an instance, a notion of "reachable set" (instances that are neighbors to this instance) and "coverage set" (instances that have this instance as one of their neighbors), borrowing an idea first developed in case based reasoning. Their algorithm removes instances which have a reachable-set size greater than the coverage-set size, retaining only the critical data points. Thus, they view "selection as removal". They also conduct an extensive study to compare their approach with other state-of-art techniques.

Madigan et al. (2002), on the other hand, proposes a method of creating (squashing) data points that preserve some statistical measure. As discussed in Section 2.3, they use a statistical model to squash the data and choose a log-likelihood profile as the measure about how a probability density function of variables behaves around the optimal statistical parameters. Thus it is not restricted to a specific task. Their algorithm clusters the original data according to this measure and create data points, one for each cluster in such a way that its contribution to the likelihood matches the total contribution of the data points in the cluster. They apply their method to logistic regression for various data sizes, and show that their method consistently outperforms the random sampling of the same sample size (in some case by a factor of 10^5). What is interesting is that the same squashed data points obtained by using a logistic regression model can still work well when used for neural networks training for classification. This means that the proposed method can achieve excellent squashing performance even when the target statistical analysis departs from the model used to squash the data.

Reinartz (2002) views the problem of instance selection as a special case of a more general problem—*focusing* and tries to provide a unifying framework for various approaches of

instance selection. The focusing task has three elements: focusing input, focusing output, and focusing criterion. The focusing criterion must take into account the focusing context, i.e., data mining goal, data characteristics, and data mining algorithm. Instantiating this to instance selection task can generate most of the known approaches. Sampling, clustering, and prototyping are the three basic generic components of the unified framework, and each of them can be instantiated to individual algorithms. The article demonstrates how several known algorithms can be described by this framework. The most important merit of this unified view is that it allows for systematic comparisons of various approaches along different aspects of interest. In the article, an initial attempt (focusing advice) is shown for analytical studies to find out which instantiation of instance selection is best suited in given contexts.

6. Conclusion and future work

With the constraints imposed by computer memory and mining algorithms, we experience selection pressures more than ever. The central point of instance selection is *approximation*. Our task is to achieve as good mining results as possible by approximating the whole data with the selected instances and hope to do better in data mining with instance selection (that is, *less is more*) as it is possible to remove noisy and irrelevant data in the process.

There are many ways of achieving approximation of the original data via instance selection. Therefore, it would be nice if there were a single general purpose instance selection method that is guaranteed of good performance in any situation. Unfortunately, the best we have in line with this hope is random sampling, though it obviously ignores the nature of data mining tasks, mining goals, and data characteristics. We have presented an initial attempt to categorize the methods of instance selection in terms of sampling, classification, clustering, and instance labeling. We addressed fundamental issues of instance selection, outlined representative sampling methods and task-dependent instance selection methods, and discussed evaluation issues associated with instance selection. As each method has its strengths and weaknesses, identifying what to gain and what to lose is crucial in designing an instance selection method that matches the user's need. One of the immediate tasks we are facing is how to develop some theory that can guide the use of instance selection and help a user apply it in data mining applications.

Much work still remains to be done. Instance selection deals with scaling down data. When we understand better instance selection, it is natural to investigate if this work can be combined with other lines of research in overcoming the problem of huge amounts of data, such as algorithm scaling-up, feature selection and construction. It is a big challenge to integrate these different techniques in achieving the common goal—effective and efficient data mining.

Acknowledgments

The idea of this special issue on instance selection came up in late 1998 when we discussed in different occasions with Usama Fayyad, Pat Langley, Heikki Mannila, and Michael Pazzani. The pointers on the topic they provided were instrumental in defining the scope of instance

selection and getting this work started. We would also extend our enormous gratitude to many authors who made this special issue possible via their quality submissions.

Notes

1. Because it does nothing until a prediction is requested, the NN type of learning is also termed as lazy learning (Aha, 1997).
2. This could lead to the change of class distribution of the data.

References

Aha, D. (Ed.). 1997. Lazy Learning. Dordrecht: Kluwer Academic Publishers.

Aha, D.W., Kibler, D., and Albert, M.K. 1991. Instance-based learning algorithms. Machine Learning 6:37–66.

Baeza-Yates, R. and Ribeiro-Neto, B. 1999. Morden Information Retrieval. New York: Addison Wesley and ACM Press.

Bloedorn, E. and Michalski, R. 1998. Data-Driven Constructive Induction: A Methodology and Its Applications. In Feature Extraction, Construction and Selection: A Data Mining Perspective. Boston: Kluwer Academic Publishers, pp. 51–68.

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

Bradley, P., Fayyad, U., and Reina, C. 1998. Scaling clustering algorithms to large databases. In Proceedings of the Fourth International Conference on Knowledge Discovery & Data Mining, pp. 9–15.

Breiman, L. and Friedman, J. 1984. Tool for large data set analysis. In Statistical Signal Processing, E. Wegman and J. Smith (Eds.). New York: M. Dekker, pp. 191–197.

Breiman, L., Friedman, J., Olshen, R., and Stone, C. 1984. Classification and Regression Trees. Wadsworth & Brooks/Cole Advanced Books & Software, Monterey: CA.

Brighton, H. and Mellish, C. 2002. Advances in instance selection for instance-based learning. Data Mining and Knowledge Disovery, An International Journal, 6(2):153–172.

Brodley, C.E. 1995. Recursive automatic bias selection for classifier construction. Machine Learning, 20(1/2): 63–94.

Burges, C. 1998. A tutorial on support vector machines. Journal of Data Mining and Knowledge Discovery, 2:121–167.

Chang, C. 1974. Finding prototypes for nearest neighbor classifiers. IEEE Transactions on Computers, C-23.

Chaudhuri, S., Motwani, R., and Narasayya, V. 1998. Random sampling for histogram construction: How much is enough? In Proceedings of ACM SIGMOD, International Conference on Management of Data, L. Haas and A. Tiwary (Eds.). New York: ACM, pp. 436–447.

Cochran, W. 1977. Sampling Techniques. New York: John Wiley & Sons.

Cohn, D., Atlas, L., and Ladner, R. 1994. Improving generalization with active learning. Machine Learning, 15:201–221.

Cohn, D., Ghahramani, Z., and Jordan, M. 1996. Active learning with statistical models. Journal of Artificial Intelligence Research, 4:129–145.

Cover, T. and Hart, P. 1967. Nearest neighbor pattern classification. IEEE Transactions on Information Theory, IT-13:21–27.

Cover, T.M. and Thomas, J.A. 1991. Elements of Information Theory. New York: Wiley.

Devlin, B. 1997. Data Warehouse from Architecture to Implementations. Reading, MA: Addison Wesley Longman, Inc.

Domingo, C., Gavaldà, R., and Watanabe, O. 2002. Adaptive sampling methods for scaling up knowledge discovery algorithms. Data Mining and Knowledge Disovery, An International Journal, 6(2):131–152.

DuMouchel, W., Volinsky, C., Johnson, T., Cortes, C., and Pregibon, D. 1999. Squashing flat files flatter. In Proceedings of the 5th ACM Conference on Knowledege Discovery and Data Mining.

Everitt, B. 1974. Cluster Analysis. London: Heinemann.

Fayyad, U. and Irani, K. 1993. Multi-interval discretization of continuous-valued attributes for classification learning. In Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence, pp. 1022–1027.

Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. 1996. From data mining to knowledge discovery: An overview. In Advances in Knowledge Discovery and Data Mining, U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy (Eds.). Menlo Park, CA: AAAI Press/The MIT Press, pp. 495–515.

Fisher, D. 1987. Knowledge acquisition via incremental conceptual clustering. Machine Learning, 2:139–172.

Freund, Y. 1994. Sifting informative examples from a random source. In Advances in Neural Information Processing Systems, pp. 85–89.

Freund, Y. 1995. Boosting a weak learning algorithm by majority algorithm. Information and Computation, 121(2):256–285.

Freund, Y. and Schapire, R. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer Systems and Science, 55(1):119–139.

Harris-Jones, C. and Haines, T.L. 1997. Sample size and misclassification: Is more always better? Working Paper AMSCAT-WP-97-118, AMS Center for Advanced Technologies.

Hussain, F., Liu, H., Tan, C., and Dash, M. 1999. Discretization: An enabling technique. Technical Report: TRC6/99, School of Computing, National University of Singapore.

Joachims, T. 1998. Text categorization with support vector machines: Learning with many relevant features. In Proceedings of 10th European Conference on Machine Learning, C. Nedellec and C. Rouveirol (Eds.). Chemnitz, Germany, pp. 137–142.

Kivinen, J. and Mannila, H. 1994. The power of sampling in knowledge discovery. In SIGMOD/PODS' 94, pp. 77–85.

Langley, P. 1996. Elements of Machine Learning. San Mateo, CA: Morgan Kaufmann.

Lewis, D. and Catlett, J. 1994. Heterogeneous uncertainty sampling for supervised learning. In Proceedings of the Eleventh Conference on Machine Learning, pp. 148–156.

Lewis, D. and Gale, W. 1994. A sequential algorithm for training text classifiers. In Proceedings of the Seventeenth Annual ACM-SIGR Conference on Research and Development in Information Retrieval, pp. 3–12.

Liu, H. and Motoda, H. (Eds.). 1998a. Feature Extraction, Construction and Selection: A Data Mining Perspective. Boston: Kluwer Academic Publishers.

Liu, H. and Motoda, H. 1998b. Feature Selection for Knowledge Discovery Data Mining. Boston: Kluwer Academic Publishers.

Madigan, D., Raghavan, N., DuMouchel, W., Nason, M., Posse, C., and Ridgeway, G. 2002. Liklihood-based data squashing: A modeling approach to instance construction. Data Mining and Knowledge Discovery, An International Journal, 6(2):173–190.

McCallum, A. and Nigam, K. 1998. Employing EM in pool-based active learning for text classification. In Proceedings of the Fifteenth International Conference on Machine Learning, pp. 350–358.

Mitchell, T. 1997 Machine Learning. New York: McGraw-Hill.

Piatetsky-Shapiro, G. and Connell, C. 1984. Accurate estimate of the number of tuples satisfying a condition. In ACM SIGMOD Conference, pp. 256–276.

Provost, F., Jensen, D., and Oates, T. 1999. Efficient progressive sampling. In Proceedings of the 5th ACM Conference on Knowledege Discovery and Data Mining.

Provost, F. and Kolluri, V. 1999. A survey of methods for scaling up inductive algorithms. Journal of Data Mining and Knowledge Discovery, 3:131–169.

Quinlan, J. 1993. C4.5: Programs for Machine Learning. San Mateo, CA: Morgan Kaufmann.

Reinartz, T. 1999. Focusing Solutions for Data Mining. New York: Springer. LNAI 1623.

Reinartz, T. 2002. A unifying view on instance selection. Data Mining and Knowledge Disovery, An International Journal, 6(2):191–210.

Schapire, R. 1990. The strength of weak learnability. Machine Learning, 5(2):197–227.

Scholkopf, B., Burges, C., and Vapnik, V. 1995. Extracting support data for a given task. In Proceedings of the First International Conference on Knowledge Discvoery and Data Mining, U. Fayyad and R. Uthurusamy (Eds.). pp. 252–257.

Seung, H., Opper, M., and Sompolinsky, H. 1992. Query by committee. In Proceedings of the Fifth Annual Workshop on Computational Learning Theory, Pittsburgh, PA, pp. 287–294.

Smith, P. 1998. Into Statistics. Singapore: Springer-Verlag.
Syed, N., Liu, H., and Sung, K. 1999a. Handling concept drifts in incremental learning with support vector machines. In Proceedings of ACM SIGKDD, International Conference on Knowledge Discovery and Data Mining, New York, S. Chaudhuri and D. Madigan (Eds.). pp. 317–321.
Syed, N., Liu, H., and Sung, K. 1999b. A study of support vectors on model independent example selection. In Proceedings of ACM SIGKDD, International Conference on Knowledge Discovery and Data Mining, New York, S. Chaudhuri and D. Madigan (Eds.). pp. 272–276.
Szalay, A. and Gray, J. 1999. Drowning in data. Scientific American www.sciam.com/explorations/1999/.
Utogoff, P. 1989. Incremental induction of decision trees. Machine Learning, 4:161–186.
Valiant, L. 1984. A theory of the learnable. Communications of the Association for Computing Machinery, 27:1134–1142.
Vapnik, V. 1995. The Nature of Statistical Learning Theory. New York: Springer-Verlag.
Weiss, S. and Indurkhya, N. 1998. Predictive Data Mining. San Francisco, California: Morgan Kaufmann.
Weiss, S. and Kulikowski, C. 1991. Computer Systems That Learn. San Mateo, California: Morgan Kaufmann.

Huan Liu is Associate Professor in Computer Science and Engineering at Arizona State University. He holds a B.E. in Electrical Engineering and Computer Science from Shanghai JiaoTong University, M.S. and Ph.D. in Computer Science at University of Southern California. He held positions at Telecom Australia Research Laboratories (Telstra), and at School of Computing, National University of Singapore. His major research interests include data mining, machine learning, data reduction, metadata, and real-world applications. Dr. Liu publishes extensively in these areas. More information can be found at http://www.public.asu.edu/∼huanliu.

Hiroshi Motoda is a professor in the division of Intelligent Systems Science at the Institute of Scientific and Industrial Research of Osaka University since 1996. Before joining the university, he had been with Hitachi since 1967, participated in research on core management, control and design of nuclear power reactors, expert systems for plant diagnosis at the Energy Research Laboratory, and on machine learning, knowledge acquisition and diagrammatic reasoning at the Advance Research Laboratory. In addition to these, his current research interests include scientific knowledge discovery and data mining. He received his Bs, Ms and PhD degrees in nuclear engineering from University of Tokyo. He was on the board of trustee of the Japan Society of Software Science and Technology (JSSST), the Japanese Society for Artificial Intelligence (JSAI) and the Japanese Cognitive Science Society (JCSS), the chair of SIG-KBS and SIF-FAI of JSAI, and on the editorial board of JSAI, JCSS and Knowledge Acquisition and IEEE Expert. He is now on the editorial board of Artificial Intelligence in Engineering, International Journal of Human-Computer Studies, Knowledge and Information Systems: An International Journal and Intelligent Data Analysis: An International Journal. He is a member of AAAI, IEEE Computer Society, JSAI, JSSST, IPSJ and JCSS.