
Data Mining and Knowledge Discovery, 6, 191–210, 2002

A Unifying View on Instance Selection

THOMAS REINARTZ thomas.reinartz@daimlerchrysler.com
DaimlerChrysler AG, Research and Technology, FT3/KL, P.O. Box 2360, 89013 Ulm, Germany

Editors: Mannila, Liu, Motoda

Received September 18, 1999; Revised January 15, 2001

Abstract. In this paper, we consider instance selection as an important focusing task in the data preparation phase of knowledge discovery and data mining. Focusing generally covers all issues related to data reduction. First of all, we define a broader perspective on focusing tasks, choose instance selection as one particular focusing task, and outline the specification of concrete evaluation criteria to measure success of instance selection approaches. Thereafter, we present a unifying framework that covers existing approaches towards solutions for instance selection as instantiations. We describe specific examples of instantiations of this framework and discuss their strengths and weaknesses. Then, we outline an enhanced framework for instance selection, generic sampling, and summarize example evaluation results for several different instantiations of its implementation. Finally, we conclude with open issues and research challenges for instance selection as well as focusing in general.

Keywords: data reduction, focusing, sampling, instance selection

1. Introduction

In recent years, knowledge discovery in databases (KDD) and data mining became more and more important as the number and size of existing data sources grow at phenomenal rates, and all industry segments from financial sectors to telecommunications rely on analysis of data to compete.

By now, KDD is defined as a complex, iterative, and interactive process which requires more than loading data into an intelligent algorithm and waiting for automatically achieved results to deploy them (e.g., Fayyad et al., 1996). Instead, KDD contains several phases, each consisting of various tasks and activities. The CRoss-Industry Standard Process for Data Mining (CRISP-DM) suggests to separate the KDD process into six different phases: Business understanding, data understanding, data preparation, modeling, evaluation, and deployment. CRISP-DM also describes tasks, activities and results of single steps within each phase in detail (Chapman et al., 1999).

As more experience exists in the field from applications in the past, it is clear that data preparation is one of the most important and time consuming phases in KDD. Preparation tasks such as data selection, data cleaning, data construction, data integration, and data formatting often determine the success of data mining engagements.

In this paper, we emphasize the importance of data selection since the size of today's databases often exceeds the size of data which current data mining algorithms handle

properly. Hence, we argue to use data reduction to shrink the data before data mining, then apply data mining algorithms to the reduced data set, and still achieve sufficient results if the selection strategy is appropriate for the current situation.

From our perspective, data reduction covers a broad spectrum of different tasks varying in the data that is reduced, in the context in which data reduction is performed, and in the specific way of evaluation. As a broader term for all issues arising in the context of data reduction, we use the term *focusing* throughout this paper (Matheus et al., 1993). As we will show, instance selection is one particular focusing task, still varying in different manners.

This paper is organized as follows. In the next section, we define a broader perspective on focusing tasks and describe instance selection as a subtask in focusing. Thereafter, we outline different evaluation strategies to measure success of instance selection approaches and define an example criterion in more detail. Then, we specify a unifying framework which covers existing instance selection methods as instantiations. We review state of the art approaches for instance selection as instantiations of this framework and discuss their strengths and weaknesses. Based on these considerations, we outline an enhanced framework, generic sampling, and summarize example evaluation results for several different instantiations of its implementation. Finally, we conclude with open issues and remaining research challenges in the area of instance selection and focusing in general.

2. Focusing tasks

The first issue in dealing with focusing is to define focusing tasks in detail. The definition of focusing tasks includes the specification of evaluation criteria to measure success of approaches solving these tasks in the context of data mining. The more we are able to characterize a task, its description, its context, and evaluation criteria, the easier it is to develop or select appropriate methods to solve the task. Hence, we propose a framework for the definition of focusing tasks and respective evaluation criteria.

2.1. Notation

In the following, we assume data in form of a (database) *table*. A table is a pair of a set of *attributes* and a set of *tuples*. Each *attribute* is characterized by its *name*, its specific *type*, and its *domain* of values. Each *tuple* contains a sequence of attribute values (see Definition 1).

Definition 1 (Attribute, tuple, and table). We define an attribute a_j by a unique *name*, an attribute *type*, and a domain $\mathrm{dom}(a_j) = \{a_{j1}, a_{j2}, \ldots, a_{jk}, \ldots, a_{jN_j}\}$ or $\mathrm{dom}(a_j) \subseteq \mathbb{R}$. N_j denotes the number of (occurring) values in attribute domain $\mathrm{dom}(a_j)$.

We specify a *tuple* t_i as a sequence of values $t_i = (t_{i1}, t_{i2}, \ldots, t_{ij}, \ldots, t_{iN})$. Each value t_{ij} is an element of attribute domain $\mathrm{dom}(a_j)$, i.e., $\forall t_{ij}, 1 \leq j \leq N : t_{ij} \in \mathrm{dom}(a_j)$.

Assume a set of attributes $A = \{a_1, a_2, \ldots, a_j, \ldots, a_N\}$, and a (multi-)set of tuples $T = \{t_1, t_2, \ldots, t_i, \ldots, t_M\}$. We define a *table* as a pair (A, T).

We denote the *power set*, i.e., the set of all subsets, of A and T as $A^{\subseteq}$ and $T^{\subseteq}$, respectively.

The terminology for descriptions of data sets is not unique. Whereas the database community uses *tuple* to refer to a row in a database table, the same concept in statistics is an *observation, item* or *data point*, and in machine learning it is an *example, case*, or *instance*. Similarly, an *attribute* is also called a *variable* or *feature*. We prefer to use tuple and attribute here, although we exceptionally use other terms as synonyms.

2.2. *Focusing specification*

Usually, we define tasks by specifications of input, output, and the relation between them. This idea leads to Definition 2 which adopts this notion to focusing.

Definition 2 (Focusing specification). Assume a table (A, T). A *focusing input* f_{in} is either a *set of tuples* ($f_{in} \subseteq T$), a *set of attributes* ($f_{in} \subseteq A$), or a *set of values* ($f_{in} \subseteq \mathrm{dom}(a_j)$, $a_j \in A$). A *focusing output* f_{out} is either a simple *subset* ($f_{out} \subseteq f_{in}$), or a *constrained subset* ($f_{out} \subset f_{in}$, and $P(f_{in}, f_{out})$), or a *constructed set* of artificial entities ($f_{out} \not\subset f_{in}$, and $P(f_{in}, f_{out})$). P is a predicate or a function that relates f_{in} to f_{out}. A *focusing criterion* $f(f_{in}, f_{out})$ is either *relevance* or *representativeness*.

A *focusing specification* is a tuple $(f_{in}, f_{out}, f(f_{in}, f_{out}))$ with a focusing input f_{in}, a focusing output f_{out}, and a focusing criterion $f(f_{in}, f_{out})$.

2.2.1. Focusing input. The original *focusing input* usually corresponds to a table. Each table is composed of a set of attributes and a set of tuples, and each attribute has its own domain of values. These three components result in three different values for the input of focusing specifications.

The focusing input is either a *set of attributes*, a *set of tuples*, or a *set of values* of a specific attribute domain. Hence, the focusing input specifies which component of a table is manipulated. Each type of input results in different tasks which need different solutions.

2.2.2. Focusing output. The overall *focusing output* still represents all components of a table. However, the output differs depending on the input and type of operations performed during focusing. The input is the first restriction of the output. If the input is a set of tuples, the output is also a set of tuples. Similarly, if the input is a set of attributes or a set of values, the output is a set of attributes or a set of values, respectively.

We further distinguish between the following three types of outputs. The basic type is a *simple subset* of the input. In case of simple subsets, we do not define any additional constraints on the output except that it is a subset of the input. The second type is a *constrained subset*. Whereas the first type allows any arbitrary subset of the input, the second type restricts the output to specific subsets that meet desired constraints. Hence, it is a specialization of the first type. Finally, the output is a *constructed set*, if we allow the output to include artificial representatives of the input. This type of output no longer corresponds to subsets of the input but constructions of new entities.

Note, in case of constrained subsets and constructed sets, we assume predicates P or functions P. A predicate P defines the set of constraints which the output meets. A function P specifies the construction principles how to generate the output from the input.

2.2.3. Focusing criterion. The *focusing criterion* determines the relation between input and output. At an abstract level, we distinguish between two different criteria, *relevance* and *representativeness*. In general, relevance means that focusing restricts the available information in the input to a subset which is still appropriate to achieve the data mining goal. In contrast, representativeness ensures that the output still represents the entire information in the input.

Specific focusing criteria are again either predicates or functions. In case of predicates, focusing criteria evaluate to true, if the relation between input and output meets the criterion. In order to get more fine-grained criteria, we also consider functions which return positive real values. These functions enable more detailed comparisons of different pairs of inputs and outputs.

At this level, more precise definitions of relevance and representativeness are not possible, since focusing criteria also have to take into account the focusing context. After the discussion of this context, we outline specific evaluation criteria that implement notions of relevance and representativeness.

2.3. Focusing context

Besides the core components of focusing tasks described previously, additional aspects affect focusing tasks and their solutions since focusing depends on phases before and after data preparation in the knowledge discovery and data mining process, and the results of other phases influence focusing. Here, we concentrate the *focusing context* on the most important aspects: Data mining goal, data characteristics, and data mining algorithm.

2.3.1. Data mining goal. In general, the *data mining goal* describes the primary purpose of an engagement. Data description and summarization, segmentation, concept description, deviation detection, dependency analysis, classification, and prediction are typical data mining goals (Chapman et al., 1999). Each of these goals has its own characteristics and hence its own influence on focusing tasks.

For example, assume we consider a segmentation goal in credit scoring to identify similar profiles of good customers, in order to retain existing customers or to approach new customers which share one of these specific profiles. Then, only good customers are of interest rather than good and bad customers, and the definition of relevance in this context requires value *good* for attribute *credit* of a customer.

2.3.2. Data characteristics. Similarly, *data characteristics* influence the definition of focusing tasks, too. Data characteristics summarize information on the shape of data. Typical characteristics range from simple statistics up to information on the data quality.

For example, assume we consider tables with high redundancy in terms of duplicate tuples, and the data mining goal is to identify all unique entries, in order to analyze variations among customer profiles. Then, focusing outputs are already representative, if they contain at least one tuple for each subset of redundant tuples.

2.3.3. Data mining algorithm. Finally, the *data mining algorithm*, which we want to apply to achieve the data mining goal, is relevant for definitions of specific focusing tasks. Even for a fixed data mining goal and fixed data characteristics there exist numerous alternative algorithms, and the specific choice of approach is important for defining the focusing task.

For example, if we compare top down induction of decision trees and nearest neighbor classifiers, we observe that relative frequencies of attribute values are important for the induction of decision trees, whereas basic nearest neighbor classifiers do not care about frequencies but need at least a single tuple from each subset of sufficiently similar tuples which belong to the same class in order to gain maximum accuracy.

2.4. Instance selection as a focusing task

In summary, this section defined prerequisites for formal specifications of focusing tasks. Each considered aspect of specification and context influences the definition of focusing tasks as well as the development or selection of appropriate solutions. In conclusion, *focusing tasks* consist of focusing specification and focusing context. Evaluation criteria implement focusing criteria and enable comparisons of solutions which aim at optimizing the result of evaluation.

Instance selection is a particular focusing task where the input is a set of tuples and the output is a (simple or constrained) subset of the input. From now on, we will restrict the focus of this paper on this particular instance selection task, and start with an outline of specific evaluation strategies for instance selection.

3. Evaluation criteria for instance selection

The definition of focusing criteria in Section 2.2 is too general for evaluation of specific instance selection results. In this section, we argue to consider the focusing context in order to define specific evaluation criteria to measure success of instance selection.

3.1. Filter and wrapper evaluation

First of all, we distinguish between *filter* and *wrapper* approaches (John et al., 1994). Filter evaluation only considers data reduction but does not take into account any data mining activities. For example, measures of statistical representativeness such as representing approximately the same modus in a sample as in the original input are typical filter evaluation measures.

Contrary, wrapper approaches explicitly emphasize the data mining aspect and evaluate results by using the specific data mining algorithm to trigger instance selection. For example, selecting an initial subset, running the data mining algorithm on this initial subset, evaluating the respective data mining results, and then incrementally enlarge the initial subset until the data mining results become sufficiently well is a typical wrapper method for evaluation.

In this paper, we suggest four different components for each of filter and wrapper evaluation. For filter evaluation, we propose to consider typical statistical characteristics to measure the appropriateness of instance selection. For example, we usually expect a sample to represent the same or similar characteristics for *modus* and *mean, variance, distribution*, and *joint distribution* within attribute domains of interest.

For wrapper evaluation, we suggest aspects such as *execution time, storage requirements, accuracy*, and *complexity* (of data mining results). If we assume classification as the data mining goal, for example, accuracy is often described in terms of the number of correct classifications given a classifier and a separate test set. If we use top down induction of decision trees to build the classifier, the number of nodes in the resulting decision tree represents the complexity of the result, for instance. For other goals and algorithms, we have to specify these notions accordingly.

3.2. Isolated and comparative evaluation

The second major differentiation in evaluation distinguishes between *isolated* and *comparative* approaches. An evaluation strategy is isolated if it only takes into account a single instance selection approach and its results. In contrast, comparative approaches compare different solutions and their results to each other. If an approach does not yield optimal results but all competitive approaches are worse, then, this approach is still good in terms of comparative evaluation if no other approach does better.

3.3. Separated and combined evaluation

For each single aspect, we can define a single *separated* evaluation criterion. However, in some cases it makes more sense to consider more than only a single aspect for evaluation. Therefore, we can also define *combined* criteria that either cumulate different aspects or that also combine isolated and comparative evaluations. Moreover, we can also add relevance weights to different components of evaluation and stress the most important factor given a specific data mining context depending on user requirements.

3.4. An example evaluation criterion

An example for an isolated separated wrapper evaluation criterion takes into account execution time of instance selection and data mining. If we only evaluate solutions according to execution time, we expect that for appropriate solutions the sum of execution time of instance selection and data mining on the output is smaller than execution time of data mining on the entire input. Consequently, we define the evaluation criterion which takes into account execution time as a relation between these two execution time values. Definition 3 specifies the respective evaluation criterion $E_T^{\leftrightarrow}$.

Definition 3 (Evaluation criterion $E_T^{\leftrightarrow}$). Assume a table (A, T), a focusing input $f_{in} \subseteq T$, an instance selection approach $F : T^{\subseteq} \longrightarrow T^{\subseteq}$, a focusing output $f_{out} = F(f_{in}) \subseteq f_{in}$,

if F is deterministic, or a set of focusing outputs F_{out}, $\forall f_{out} \in F_{out} : f_{out} = F(f_{in}) \subseteq f_{in}$, if F is non-deterministic, a data mining algorithm D, a function t that measures execution time of F and D, an additional value $\sigma(t) \geq 0$ that depicts the sum of t and its standard deviation in case of non-deterministic instance selection and their averaged application, and $0 \leq \eta_\sigma \leq 0.5$. We define the evaluation criterion $E_T^{\leftrightarrow}$ as a function

$$E_T^{\leftrightarrow} : F \times D \times T^{\subseteq} \times T^{\subseteq} \longrightarrow]0; \infty[,$$

$$E_T^{\leftrightarrow}(F, D, f_{in}, f_{out}) := (1 - \eta_\sigma) \cdot \left(\frac{t(F, f_{in}) + t(D, f_{out})}{t(D, f_{in})} \right) + \eta_\sigma \cdot \left(\frac{\sigma(t(F, f_{in})) + \sigma(t(D, f_{out}))}{t(D, f_{in})} \right).$$

The first part of $E_T^{\leftrightarrow}$ returns values smaller than 1, if the sum of execution time of instance selection and data mining on the output is smaller than execution time of data mining on the input. It reveals values larger than 1, if the inverse is true. If we compare instance selection approaches according to this evaluation criterion, more appropriate solutions with respect to execution time yield smaller values of $E_T^{\leftrightarrow}$.

At this point, we emphasize the difference between deterministic and non-deterministic instance selection approaches. *Deterministic* methods always generate the same output given the same input, whereas *non-deterministic* solutions possibly create different outputs although the input is the same. Systematic sampling for a fixed input order and a pre-specified start position is an example for a deterministic approach, whereas random sampling is a typical non-deterministic solution.

It is important to know whether the instance selection approach is deterministic or not, since evaluation of non-deterministic approaches has to consider the potential of randomly generating the worst possible result in a single try or conversely the best output that is possible. Thus, a single application of a non-deterministic approach is not sufficient for an appropriate instance selection and its evaluation. We consider this aspect in Definition 3 for non-deterministic solutions, by referring to average applications. We run non-deterministic approaches several times and average their results. For simplicity, we do not explicitly compute average values in Definition 3 but rather assume that t already covers the result of computations of average values for all outputs in F_{out}.

The second term in $E_T^{\leftrightarrow}$ additionally punishes non-deterministic approaches since their applications probably yield different outputs with different qualities. Thus, we take the average quality of multiple experiments as the evaluation result and punish the worst possible case as an additional amount of $E_T^{\leftrightarrow}$ by adding the standard deviation of t across multiple experiments. Note, in Definition 3 function σ covers both the average value plus the standard deviation.

For deterministic approaches, both parts of $E_T^{\leftrightarrow}$ have the same value since the standard deviation of deterministic approaches is zero, and hence the sum of the average value (i.e., a single value for deterministic approaches) and the standard deviation is the average value itself.

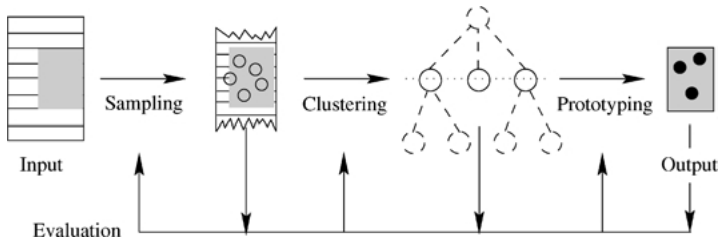

Figure 1. A unifying framework for instance selection.

4. A unifying framework for instance selection

Up to now, we specified instance selection as a focusing task and outlined evaluation strategies to measure success of instance selection approaches. In this section, we present a unifying framework which covers individual state of the art approaches related to instance selection.

The unifying framework consists of three steps (see figure 1):

1. Sampling
2. Clustering
3. Prototyping

The general assumption of the framework is either to utilize (statistical) sampling techniques, or to search for sets of sufficiently similar tuples and to replace each set by a single or a few prototypes. Instantiations of this framework only differ in their way to apply sampling, clustering, and prototyping.

For instance, an example instantiation of this framework works as follows. First, an application of a statistical sampling technique draws an initial sample. In the next step, a clustering technique groups the initial sample into subsets of similar tuples. For each of these subsets, the prototyping step selects or constructs a smaller set of representative prototypes. The set of prototypes then constitutes the final output of instance selection.

At any intermediate point, instance selection approaches possibly apply an evaluation step in order to judge whether the output already meets the pre-defined evaluation criterion. In this framework, evaluation criteria are as simple as checking the sample size, or as complex as applying data mining algorithms and analyzing their results.

The order of steps in this framework is not strict. Instantiations of this framework are able to apply the basic steps in any order, or skip some steps completely. Similarly, instantiations sometimes also skip one of the basic steps, perform other steps first, and then return to the previously ignored step. And finally, some approaches combine more than a single step into a compound step.

4.1. Sampling

The first step of the unifying framework uses (statistical) *sampling* techniques (Cochran, 1977; Scheaffer et al., 1996). Originally, sampling techniques supply surveys in human

populations. Starting with an initial question which requires measurements of variables in the population, sampling leads to approximate answers by drawing samples from the population, measuring variables of interest in these samples, and then concluding from results on samples to the entire population. Typically, variables of interest are numeric values such as averages, sums, or proportions.

As a general advantage of sampling, we recognize that most sampling techniques are efficient in terms of execution time, and from KDD perspectives costs for drawing samples from tables and calculating statistical values are low. The theory of sampling is also well understood, although it only applies to estimations of a few numeric statistical values, and a similar theory for the appropriateness of sampling in KDD does not exist. We also notice that statistical sampling techniques generally do not take into account the focusing context and represent non-deterministic solutions. Hence, there are always fractions of uncertainty in practice, and single experiments using sampling techniques are not sufficient for KDD purposes in general.

Due to space limitations, we can not describe specific sampling approaches in detail but typical instantiations of the unifying framework apply *simple random sampling, systematic sampling*, or *stratified sampling*.

4.2. Clustering

The second step of the unifying framework uses *clustering* techniques to identify subsets of similar tuples within the input. In statistics as well as in machine learning exist many efforts to identify and to describe groups of similar objects (see Hartigan (1975) and Genari (1989), for example). The key idea to utilize clustering techniques for instance selection purposes suggests to select only a few representatives for each cluster of similar objects. This set of representatives then forms the output of instance selection.

Again, detailed descriptions of clustering methods are beyond the scope of this paper. All approaches mainly differ in their way to build clusters (coverage, separation, and structure) and whether they only create clusters without descriptions or whether they also provide characterizations of each cluster.

4.3. Prototyping

The third step in the unifying framework is *prototyping*. In general, prototypes are more condensed descriptions of sets of tuples. Prototyping assumes that a single tuple is able to represent information of an entire subset of tuples, and we distinguish between the following types of approaches:

– *Prototype selection* results in a subset of the original set of tuples and is often based on notions of *prototypicality*. For each tuple in the original subset of tuples, prototypicality measures the degree of representativeness of this tuple among all tuples in the subset (Zhang, 1992). Then, we select those tuples which maximize prototypicality as the representative subset of prototypes (see Skalak (1993), Skalak (1994), and Smyth and Keane (1995) for further examples).

– *Prototype construction* generates prototypes which do not necessarily refer to existing tuples. Therefore, this type of prototyping uses a specific function to explicitly build new tuples that represent information of an entire subset of tuples. For example, if all attributes are quantitative, we average all occurring values for each attribute separately, and the sequence of average values constitutes the center of all tuples as the prototypical representation (see Linde et al. (1980), Barreis (1989), and Sen and Knight (1995) for further examples).

In terms of the framework for the definition of focusing tasks, prototype selection as well as prototype construction define specific predicates or functions that refer to the respective predicate or function in the definition of focusing outputs (see Definition 2). Prototype selection criteria correspond to predicates of constrained subsets, whereas prototype construction requires definitions of functions for constructed sets.

5. Example instantiations of the unifying framework

In this section, we discuss several example instantiations of the unifying framework for instance selection. We focus our attention on particularly promising approaches that are both efficient in computation and appropriate for data mining. For each approach, we emphasize the most prominent step of the framework which the method mainly applies.

5.1. Dynamic sampling using the PCE criterion

Dynamic sampling using the *Probably Close Enough (PCE) criterion* follows up windowing (Quinlan, 1993; John and Langley, 1996). In general, *dynamic sampling* covers all solutions which use wrapper evaluation and iteratively enhance the current output until the evaluation becomes true. In contrast, *static sampling* generates an output only once and uses this output for data mining.

The PCE criterion is a specific evaluation criterion used in dynamic sampling. The idea in dynamic sampling using the PCE criterion is to select an output that is probably good enough to yield the same quality of data mining results as if we use the entire input.

The PCE criterion allows instance selection to stop as soon as the probability that the difference between accuracies of classifiers generated on the input and the current output is higher than a first threshold no longer exceeds a second threshold. Similar to windowing, dynamic sampling using the PCE criterion iteratively enlarges the current output by random selection until the PCE criterion holds.

5.1.1. Instance selection using the PCE criterion. In terms of the framework for the definition of focusing tasks, the PCE criterion refers to focusing on constrained subsets. The PCE criterion is true, if the probability that the difference between accuracies of classifiers generated on the input and the current output exceeds Δ_1 is not higher than Δ_2 (See Definition 4).

Definition 4 (PCE criterion). Assume a table (A, T), a focusing input $f_{in} \subseteq T$, a focusing output $f_{out} \subseteq f_{in}$, a data mining algorithm D, a probability function p on classification accuracies a, and $0 \leq \Delta_1,\ \Delta_2 \leq 1$. Predicate $P(f_{in},\ f_{out})$ in dynamic sampling using the PCE criterion is

$$P(f_{in},\ f_{out}) :\Leftrightarrow p(a(D(f_{in})) - a(D(f_{out})) > \Delta_1) \leq \Delta_2.$$

5.1.2. Advantages and disadvantages. In terms of the unifying framework, dynamic sampling using the PCE criterion utilizes simple random sampling and wrapper evaluation. Evaluation applies data mining algorithms to the current output, estimates accuracies of resulting classifiers on an additional sample, and incrementally enlarges the output, if the quality of the current output is probably not good enough. Note, if data mining algorithms are not able to incrementally extend classifiers using additional tuples, dynamic sampling implies iterative data mining from scratch for each enlarged output.

As an instance selection approach, dynamic sampling using the PCE criterion is not efficient. Moreover, experimental results reported by John and Langley (1996) show only small improvements in terms of classification accuracies in comparison to static sampling while generating classifiers on larger samples than static sampling.

Recently, Provost et al. (1999) proposed the *progressive sampling scheme* which extends the ideas of the PCE criterion towards a more generic algorithm which suggests a different strategy to incrementally extend the current output by geometric sampling.

5.2. Leader sampling

Leader sampling (LEASAM) is a derivative of leader clustering which adopts the clustering algorithm for instance selection without explicitly constructing the clusters. Leader clustering constructs exhaustive, non-overlapping, non-hierarchical clusterings of the focusing input (Hartigan, 1975). Therefore, we assume a similarity measure *Sim* to compute similarities between pairs of tuples, and a similarity threshold δ. The algorithm generates a partition of all tuples into clusters and selects a leader tuple for each cluster, such that each tuple in a cluster is within a distance of $1 - \delta$ from the leader tuple.

Leader sampling uses the same approach to perform instance selection. It only needs one pass through the input and assigns each tuple to the cluster of the first leader tuple which is sufficiently similar (in terms of δ) to this tuple. It selects new leader tuples for tuples which are not close enough to any existing leader tuple. As the final output of instance selection, LEASAM returns all leader tuples as representative prototypes.

5.2.1. Prototype selection in leader sampling. In terms of the framework for the definition of focusing tasks, leader sampling refers to constrained subsets as the output. Definition 5 states the prototype selection criterion in LEASAM. This criterion evaluates to true, if for all tuples in the input, at least one tuple in the output exists which exceeds similarity threshold δ in comparison to this tuple, and for all pairs of distinct tuples in the output, similarities between these tuples are smaller than δ.

Definition 5 (Prototype selection in LEASAM). Assume a (database) table (A, T), a focusing input $f_{in} \subseteq T$, a focusing output $f_{out} \subseteq f_{in}$, a similarity measure Sim, and $0 < \delta \leq 1$. Predicate $P(f_{in}, f_{out})$ in LEASAM is

$$\begin{aligned} P(f_{in}, f_{out}) \quad & :\Leftrightarrow \\ & \forall t_i \in f_{in}\ \exists t_{i'} \in f_{out} : Sim(t_i, t_{i'}) \geq \delta \\ \wedge \quad & \forall t_i, t_{i'} \in f_{out} : t_i \neq t_{i'} \Rightarrow Sim(t_i, t_{i'}) < \delta. \end{aligned}$$

5.2.2. Advantages and disadvantages. In terms of the unifying framework for instance selection, leader sampling is again a compound approach of clustering and prototyping. Leader sampling builds clusters (though not storing them) and leader tuples at the same time. Leader tuples correspond to prototypes, and each leader represents a single cluster. Thus, the set of all leaders defines a set of prototypes that represents the entire input. Note, predicate P in definition 5 describes the constraints that hold for the output of LEASAM but P is not used during instance selection as an evaluation criterion. Hence, leader sampling does not apply any explicit evaluation criterion as dynamic sampling using the PCE criterion does by estimating and comparing classification accuracies for evaluation.

The positive feature of leader sampling is that it only requires one sequential pass through the data. It only needs to retain leader tuples, not the clusters. On the other hand, several negative properties follow from immediate assignments of tuples to clusters as soon as the first leader exceeds similarity threshold δ.

The first negative property is that clusterings are not invariant with respect to reordering of tuples. For example, the first tuple is always the leader tuple of the first cluster. The second negative property is that the first clusters are always likely to contain more tuples than later clusters, since they get first chance at each tuple as it is allocated. Thirdly, representativeness of leaders mainly depends on similarity threshold δ. For small values of δ, leader clustering selects only a few leaders, whereas for large values of δ, the set of leaders is likely to contain many tuples. It is difficult to set δ to the most appropriate value in terms of representativeness of resulting leaders in advance.

As an extension to leader sampling that overcomes these drawbacks, we refer to *advanced leader sampling* (Reinartz, 1997). This novel approach provides two additional preparation steps, sorting and stratification (see Section 6.1). Furthermore, we also point to *similarity-driven sampling* which supports automatic mechanisms for estimation and adaptation of similarity threshold δ during execution (Reinartz, 1998).

5.3. PL

The second example related to prototyping is PL (Datta and Kibler, 1995). This approach learns prototypical concept descriptions for sets of tuples. Although the final output of PL contains prototypical concept descriptions for each class, PL uses a top-down splitting mechanism to partition tuples for each class separately, and then learns prototypes for each partition. The disjunction of all prototypes for each class forms the prototypical concept description of this class.

Table 1. Algorithm PL(f_{in}, *partition, prototype*).

```
begin
    f_out := ∅;
    k := 1;
    while k ≤ N_N do
        S := partition({t_i ∈ f_in | t_iN ≡ a_Nk});
        l := 1;
        while l ≤ | S | do
            f_out := f_out ∪ prototype(s_l);
            l := l + 1;
        enddo
        k := k + 1;
    enddo
    return(f_out);
end;
```

Table 1 shows an implementation of PL. After initialization of the output, PL generates prototypes for each class a_{Nk} separately. First, PL partitions all tuples in the input which belong to the same class into subsets $S := \{s_1, \ldots, s_l, \ldots, s_L\}$. Then, PL constructs a single prototype for each of the resulting clusters. At the end, PL returns the set of all prototypes as the output.

Similar to top down induction of decision trees, the *partition* step recursively separates sets of tuples into subsets according to attribute values. For qualitative attributes, the partition contains subsets of tuples which all have the same value within each subset. For quantitative attributes, the partition consists of subsets of tuples which all have either values below a selected threshold or values above this threshold within each subset.

5.3.1. Prototype construction in PL. In terms of the framework for the definition of focusing tasks, PL corresponds to focusing on constructed sets. Definition 6 describes prototype construction in PL. For each predictive attribute, PL computes a single cumulative value of all tuples within each subset of the partition. For qualitative attributes, this cumulated value is the modus $m_j(s_l)$, i.e., the value that most often occurs in the subset. For quantitative attributes, the cumulated value is the mean $\mu_j(s_l)$ of all values.

Definition 6 (Prototype construction in PL). Assume a table (A, T), a focusing input $f_{in} \subseteq T$, a clustering $S = \{s_1, \ldots, s_l, \ldots, s_L\}$ of f_{in}, and a focusing output $f_{out} \subseteq f_{in}$. Function $P(f_{in}, f_{out})$ in PL is

$$P(f_{in}, f_{out}) :\Leftrightarrow \forall t_i \in f_{out}\ \exists s_l \in S : t_i \equiv prototype(s_l) \quad \text{with}$$

$$prototype(s_l) := (t_{i^*1}, \ldots, t_{i^*j}, \ldots, t_{i^*N}), \quad \text{and}$$

$$t_{i^*j} := \begin{cases} m_j(s_l), & \text{if } a_j \text{ qualitative} \\ \mu_j(s_l), & \text{else} \end{cases}$$

Note, Definition 6 simplifies prototype construction in PL, since it does not include strength values as implemented in PL.

5.3.2. Advantages and disadvantages. In terms of the unifying framework, PL again performs clustering in combination with prototyping. The partition step in PL constructs exhaustive, non-overlapping, hierarchical clusters, and the prototyping step constructs a single prototype for each cluster. Except the inherent evaluation within the partition step, PL does not apply any evaluation criterion.

PL is similar to edited nearest neighbor approaches. However, its advantage is the straightforward strategy to identify subsets of sufficiently similar tuples which are then represented by single prototypes. Besides the necessity of having class labels in the input data, the main disadvantage of PL is the complexity of the partition step which is mainly due to expensive computations for quality assessments of partitions. Moreover, PL's output does not necessarily refer to existing tuples in the original data set and hence is not really an instance selection approach but an instance construction method.

As an alternative to PL, we mention the recent development by DuMouchel et al. (1999) for squashing flat files flatter.

6. Evaluation

In this section, we briefly present a condensed description of an experimental evaluation of different instance selection approaches. For this purpose, we utilize the *generic sampling* approach which results from providing arbitrary combinations of the unifying framework with enhancements in advanced leader sampling and similarity-driven sampling (Reinartz, 1999).

6.1. Generic sampling

Algorithm GENSAM which implements the generic sampling approach provides two additional preparation steps, sorting and stratification, in comparison to the unifying framework for instance selection plus an intelligent sampling step.

6.1.1. Sorting. *Sorting* uses the following order relation between tuples which considers attribute values in order of attribute relevance starting with the most important attribute.[1] If value t_{iN} for the most important attribute a_N in tuple t_i is larger than value $t_{i'N}$ in tuple $t_{i'}$, then sorting places t_i after $t_{i'}$ in the sorted table. If both tuples have the same value for a_N, sorting recursively proceeds with the next important attribute in the same way. Relative positions of completely identical tuples remain constant.

For local comparisons between attribute values, we distinguish between quantitative and qualitative attributes. For quantitative attributes, we use the regular numeric order, whereas for qualitative attributes, we employ the lexicographic order.

6.1.2. Stratification. *Stratification* constructs a strata tree similar to PL described in Section 5.3. It separates tuples into smaller subsets according to attribute values. Again, stratification considers attribute values in order of attribute relevance and also distinguishes between quantitative and qualitative attributes. For quantitative attributes, a stratum contains tuples with values in the same interval generated by discretization (e.g., Dougherty

et al., 1995), whereas for qualitative attributes, a stratum includes tuples with the same value. Stratification treats missing or unknown values as special cases and builds extra strata for all tuples with special values for the same attribute.

The stratification procedure continues with the next important attribute at the next lower level in the hierarchy of strata as long as a pre-defined maximum number of tuples within a single stratum is exceeded or no more attributes for stratification are available.

6.1.3. Intelligent sampling. For the intelligent sampling step, generic sampling is able to supply any instantiation of the unifying framework for instance selection. In its implementation in GENSAM, generic sampling provides random sampling, systematic sampling, leader sampling, and similarity-driven sampling.

All in all, generic sampling provides an enhanced framework which allows applications of many different instance selection approaches by combining the three steps, sorting, stratification, and sampling, in any way.

6.2. Experimental setting

For experimental studies, we selected the data mining goal classification, several data sets, representing varying data characteristics, from the UCI repository, and two different data mining algorithms for classification, C4.5 (Quinlan, 1993) and instance-based learning (Aha et al., 1991; Kohavi et al., 1996).

As instance selection approaches, we chose several different instantiations of generic sampling. In this paper, we concentrate on simple random sampling (R), simple random sampling with stratification (RS), systematic sampling (S), systematic sampling with sorting (SS), leader sampling (L), and leader sampling with sorting and stratification (LS). For all approaches, we used various parameter settings, and selected the best results for each approach for evaluation.

For each data set, for each data mining algorithm, and for each instance selection approach, experimentation started with randomly splitting the data set into train and test data (about 80% and 20% of the data, respectively). Then, we applied each instance selection approach to the training set, run each data mining algorithm on the resulting output, and finally tested the generated classifiers on the test set to estimate predictive accuracy. Note, for non-deterministic instance selection approaches we conducted multiple runs and averaged results.

For evaluation and comparison of different instance selection approaches, we considered three different combined evaluation criteria which rank the results according to more than only a single aspect in filter and wrapper evaluation as well as in isolated and comparative modes. Below, we present the results for combined filter evaluation.

6.3. Focusing advice

The overall goal of experimental evaluation is to generate focusing advice, i.e., to provide guidance in selecting the best suited instance selection approach given the context. Therefore, we defined *focusing scores* in the following way.

Table 2. Focusing advice for filter evaluation.

Context	R	RS	S	SS	L	LS
Few tuples	0	+1	−1	+4	+5	0
Many tuples	−1	+1	+2	+5	+4	0
Few attributes	−1	+1	+2	+5	+4	−2
Many attributes	−3	−3	−1	0	+5	+6
No qual. attributes	−1	+1	+1	+5	+4	0
No quant. attributes	−1	+2	+3	+5	+2	0
More qual. attributes	−1	+1	+1	+5	+4	0
More quant. attributes	−1	+1	+1	+5	+4	0
Many classes	+3	+3	+3	+5	−1	−4
Average	−0.6	+1.1	+1.4	+4.6	+3.4	−0.1

First, we considered situations where a specific instance selection approach behaves particularly good or particularly bad in terms of the evaluation criterion separately. We added a *plus* for good success and assigned a *minus* for poor performance. Thereafter, we analyzed comparisons between different approaches. For each pair of comparisons, we related results of the first approach to success of the second. The superior solution again got a plus, whereas the inferior solution deserved a minus. At the end, we summed up all plus and minus scores which resulted in final scores for each instance selection approach. Note, in individual cases, scores abstract from specific experimental results but keep most important relations among focusing solutions.

If we apply focusing solutions in KDD, we use the resulting score tables to decide which instance selection approach is preferable. Rows in each table allow comparisons among different approaches for specific data characteristics, whereas columns relate concrete approaches across varying data characteristics. Negative scores mean that this approach is generally less appropriate in this context, whereas positive scores recommend applications of this approach. The lower the score is, the worse is the solution, and the higher the value is, the better is the solution.

For example, Table 2 shows focusing advice in terms of focusing scores for six instance selection approaches given different aspects of the focusing context according to combined filter evaluation. Here, we describe the context by the number of tuples, the number of attributes, the type of attributes, and the number of class labels.

In summary, Table 2 shows that on average systematic sampling with sorting (SS) is superior to other approaches, followed by leader sampling (L) without any preparation in advance. For full discussions of the experimental evaluation and more focusing advice, we refer to Reinartz (1999).

7. Conclusions

In summary, this paper addressed different issues in the area of instance selection. First of all, we generally defined focusing tasks and showed that instance selection is one particular

focusing task in the data preparation phase of the knowledge discovery and data mining process. We outlined evaluation strategies for instance selection and discussed an example criterion in detail.

Second, we presented a unifying framework for instance selection that covers existing attempts towards solutions. We discussed several examples, demonstrated how they fit into the framework, and briefly analyzed their strengths and weaknesses. For each approach, we also pointed to recent developments as enhancements.

Finally, we also outlined an enhanced framework, generic sampling, and summarized example results of experimental studies with several instantiations of this approach in the context of classification tasks with varying data characteristics as focusing advice.

Although size and complexity of existing data and information sources grow at phenomenal rates and the importance of data reduction techniques became clear, it is interesting to see that there is relatively few research and technology in this area that directly contributes to solutions for these challenges. It is unclear whether the complexity of this problem is too high or whether the data mining community feels that statistical sampling approaches are sufficient. From our perspective, there is much more research necessary to overcome limitations of existing data mining algorithms that can not handle large scale databases properly. Data reduction is one particular important aspect to help in those situations where we are not able to scale up algorithms.

Below, we outline further research based on these considerations and raise more research challenges for the future.

7.1. More research for focusing solutions

The overall work partly presented here also includes initial attempts for analytical studies to find out which instantiation of instance selection is best suited in given contexts as well as more detailed experimental results comparing different approaches in different domains. The analytical studies and experimental results imply first attempts to define heuristics for the selection of the most appropriate method for instance selection.

Initially, we started with theoretical attempts to prove the appropriateness of specific instance selection approaches in concrete data mining contexts. We selected random sampling, classification, and nearest neighbor classifiers as a starting point, and ended up in an average case analysis for the expected classification accuracy given specific sample sizes and nearest neighbor classification in even more constrained situations.

The average case analysis worked out well and comparisons between predicted accuracies and experimental results showed that this analysis is able to correctly foresee true results. Hence, this analysis either helps to predict which accuracy to expect for a given sample size, or to determine the sample size given an expected accuracy.

However, the assumptions on the data mining context which made this analysis feasible are too restricted and not applicable in general. Consequently, we went on the experimental way. We selected several alternative instance selection approaches, more than only a single data mining algorithm, and data sets with varying data characteristics. We run systematic experiments, compared the results using different evaluation strategies, and inferred heuristics for the relation between the characteristics of the focusing context and the appropriateness of specific approaches.

These heuristics, exemplified above, are now able to help in relatively many application scenarios to select the best suited focusing solution given the focusing context. However, these heuristics remain the result of an experimental study, and are consequently by no means theoretically proven.

7.2. Open issues and research challenges

In essence, we claim that the specification of focusing tasks defines the scope of potential research in the area of data reduction. The definition of various evaluation criteria allows systematic comparisons of approaches along different aspects of interest. The unifying framework of existing solutions helps the community to compare their developments in a methodological way and to discuss their strengths and weaknesses. The generic sampling approach outlined opens the perspective for future research to allow for flexible solutions that can be specialized according to specific contexts. Finally, the analytical studies and experimental results provide an initial attempt to help practitioners to find the best suited solution for instance selection.

The area of instance selection (or focusing in general) still needs more effort. There is no consistent and data mining oriented way for systematically evaluating the appropriateness of existing mechanisms or novel solutions. Current methods are still not able to really scale up data mining approaches to huge databases in terms of Giga-, Terra-, and Petabyte. Moreover (and possibly more important), there is also no theoretically proven guidance in selecting the right instance selection approach given a specific context.

Hence, we argue to work on the following issues for future research in the area of instance selection (and focusing in general):

- define standardized evaluation criteria to enable systematic comparison of existing and novel approaches;
- scale up intelligent focusing approaches by combining technologies from machine learning research and the database community;
- develop more intelligent focusing solutions that provide data reduction techniques beyond pure statistical sampling and make use of the specific characteristics of concrete contexts in data mining;
- extend attempts of analytical studies and experimental results to understand the relation between different instance selection techniques and to come up with reliable heuristics and guidelines for the selection of best suited focusing solutions given a specific data mining environment.

The work presented and outlined here is certainly a first step towards these directions but still does not provide an overall answer to all questions.

Note

1. Attribute relevance is either defined by domain experts or computed automatically using existing state of the art approaches (e.g., Wettschereck et al., 1995).

References

Aha, D.W. and Bankert, R.L. 1996. A comparative evaluation of sequential feature selection algorithms. In Learning from Data: AI and Statistics V, D. Fisher and H.-J. Lenz (Eds.). New York, NY: Springer, pp. 199–206.

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

Barreis, E.R. 1989. Exemplar-Based Knowledge Acquisition. Boston, MA: Academic Press.

Chapman, P., Clinton, J., Khabaza, T., Reinartz, T., and Wirth, R. 1999. The CRISP-DM Process Model, www.crisp-dm.org/pub-paper.pdf.

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

Datta, P. and Kibler, D. 1995. Learning prototypical concept descriptions. In Proceedings of the 12th International Conference on Machine Learning, July, 9–12, Tahoe City, CA, A. Prieditis and S. Russell (Eds.). San Mateo, CA: Morgan Kaufmann, pp. 158–166.

Dougherty, J., Kohavi, R., and Sahami, M. 1995. Supervised and unsupervised discretization of continuous features. In Proceedings of the 12th International Conference on Machine Learning, July, 9–12, Tahoe City, CA, A. Prieditis and S. Russell (Eds.). Menlo Park, CA: Morgan Kaufmann, pp. 194–202.

DuMouchel, W., Volinsky, C., Johnson, T., Cortes, C., and Pregibon, D. 1999. Squashing flat files flatter. In Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August, 15–18, San Diego, California, USA, S. Chaudhuri and D. Madigan (Eds.). New York: ACM, pp. 6–15.

Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. 1996. Knowledge discovery and data mining: Towards a unifying framework. In Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, August, 2–4, Portland, OR, E. Simoudis, J. Han, and U. Fayyad (Eds.). Menlo Park, CA: AAAI Press, pp. 82–88.

Genari, J.H. 1989. A survey of clustering methods. University of California, Irvine, CA, Technical Report 89–38.

Hartigan, J.A. 1975. Clustering Algorithms. New York: John Wiley and Sons.

John, G.H., Kohavi, R., and Pfleger, K. 1994. Irrelevant features and the subset selection problem. In Proceedings of the 11th International Conference on Machine Learning, July, 10–13, Rutgers University, New Brunswick, NJ, W.W. Cohen and H. Hirsh (Eds.). San Mateo, CA: Morgan Kaufmann, pp. 121–129.

John, G.H. and Langley, P. 1996. Static versus dynamic sampling for data mining. In Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, August, 2–4, Portland, OR, E. Simoudis, J. Han, and U. Fayyad (Eds.). Menlo Park, CA: AAAI Press, pp. 367–370.

Kohavi, R., Sommerfield, D., and Dougherty, J. 1996. Data Mining Using MLC++: A Machine Learning Library in C++, http://robotics.stanford.edu/~ronnyk.

Linde, Y., Buzo, A., and Gray, R. 1980. An algorithm for vector quantizer design. IEEE Transactions on Communications, 28:85–95.

Matheus, C.J., Chan, P.K., and Piatetsky-Shapiro, G. 1993. Systems for knowledge discovery in databases. IEEE Transactions on Knowledge and Data Engineering, 5(6):903–913.

Provost, F., Jensen, D., and Oates, T. 1999. Efficient pogressive sampling. In Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August, 15–18, San Diego, CA, USA, S. Chaudhuri and D. Madigan (Eds.). New York: ACM, pp. 23–32.

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

Reinartz, T. 1997. Advanced leader sampling for data mining algorithms. In Proceedings of the ISI '97 Satellite Conference on Industrial Statistics: Aims and Computational Aspects, August, 16–17, Athens, Greece, C.P. Kitsos (Ed.). Athens, Greece: University of Economics and Business, Department of Statistics, pp. 137–139.

Reinartz, T. 1998. Similarity-driven sampling for data mining. In Principles of Data Mining and Knowledge Discovery: Second European Symposium, PKDD '98, September, 23–26, Nantes, France, J.M. Zytkow and M. Quafafou (Eds.). Heidelberg: Springer, pp. 423–431.

Reinartz, T. 1999. Focusing Solutions for Data Mining: Analytical Studies and Experimental Results in Real-World Domains, LNAI 1623. Heidelberg: Springer.

Scheaffer, R.L., Mendenhall, W., and Ott, R.L. 1996. Elementary Survey Sampling, 5th ed. New York: Duxbury Press.

Sen, S. and Knight, L. 1995. A genetic prototype learner. In Proceedings of the 14th International Joint Conference on Artificial Intelligence, August, 20–25, Montreal, Quebec, Canada, C.S. Mellish (Ed.). San Mateo, CA: Morgan Kaufmann, Vol. I, pp. 725–731.

Skalak, D.B. 1993. Using a genetic algorithm to learn prototypes for case retrieval and classification. In Proceedings of the AAAI '93 Case-Based Reasoning Workshop, July, 11–15, Washington, DC, D. Leake (Ed.). Menlo Park, CA: American Association for Artificial Intelligence, pp. 64–69, Technical Report WS-93-01.

Skalak, D.B. 1994. Prototype and feature selection by sampling and random mutation hill climbing algorithms. In Proceedings of the 11th International Conference on Machine Learning, July, 10–13, Rutgers University, New Brunswick, N.J., W.W. Cohen and H. Hirsh (Eds.). San Mateo, CA: Morgan Kaufmann, pp. 293–301.

Smyth, B. and Keane, M.T. 1995. Remembering to forget. In Proceedings of the 14th International Joint Conference on Artificial Intelligence, August, 20–25, Montreal, Quebec, Canada, C.S. Mellish (Ed.). San Mateo, CA: Morgan Kaufmann, Vol. I, pp. 377–382.

Wettschereck, D., Aha, D., and Mohri, T. 1995. A review and comparative evaluation of feature weighting methods for lazy learning algorithms. Naval Research Laboratory, Navy Center for Applied Research in Artificial Intelligence, Washington, D.C., Technical Report AIC-95-012.

Zhang, J. 1992. Selecting typical instances in instance-based learning. In Proceedings of the 9th International Conference on Machine Learning, July, 1–3, Aberdeen, Scotland, D. Sleeman and P. Edwards (Eds.). San Mateo, CA: Morgan Kaufmann, pp. 470–479.

Thomas Reinartz is currently a senior researcher in the data mining group at DaimlerChrysler Research and Technology in Ulm, Germany. He received his Ph.D. in Computer Science from the University of Kaiserslautern. His research interests include knowledge acquisition, machine learning, data mining, and case-based reasoning.