D2I

Integrazione, Warehousing e Mining di sorgenti eterogenee

Programma di ricerca (cofinanziato dal MURST, esercizio 2000)

Analysis and comparison of methods and algorithms for data mining

Tiziana Catarci, Paolo Ciaccia, Giovambattista Ianni, Stefano Lodi, Luigi Palopoli, Marco Patella, Claudio Sartori

D3.R1 **30 aprile 2001**

Sommario

In this report we review and compare data mining methods and algorithms. After a short introduction on the general concepts of data mining we focus on four specific topics, metaquerying, data clustering, similarity queries and visualization, and go deeper, analyzing the various approaches and proposals known in the literature and, where applicable, in the market. In particular, in each of the four parts an effort is made to find out a common model to compare results and applicability.

Tema	Tema 3: Data Mining
Codice	D3.R1
Data	30 aprile 2001
Tipo di prodotto	Rapporto tecnico
Numero di pagine	52
Unità responsabile	RM
Unità coinvolte	BO, CS, RM
Autore da contattare	Claudio Sartori Dipartimento di Elettronica, Informatica e Sistemistica Università degli Studi di Bologna Viale Risorgimento, 2, 40136 Bologna, Italia csartori@deis.unibo.it

Analysis and comparison of methods and algorithms for data mining

Tiziana Catarci, Paolo Ciaccia, Giovambattista Ianni, Stefano Lodi,
Luigi Palopoli, Marco Patella, Claudio Sartori

30 aprile 2001

Abstract

In this report we review and compare data mining methods and algorithms. After a short introduction on the general concepts of data mining we focus on four specific topics, metaquerying, data clustering, similarity queries and visualization, and go deeper, analyzing the various approaches and proposals known in the literature and, where applicable, in the market. In particular, in each of the four parts an effort is made to find out a common model to compare results and applicability.

Contents

1 Introduction

In recent years, advances in hardware and systems for automated data acquisition, such as geographic satellites, for the recording of business transaction such as bar code technology, and mass storage devices have led an increasing number of organizations to accumulate huge amounts of business, scientific and demographic data. Such large repositories of data are usually kept off-line because owning organizations are unable to turn them into usable information. In fact, although it is widely accepted that large data repositories may contain evidence of knowledge which could be highly valuable in a business or scientific environment, the task of exploring and analyzing large bodies of data remains at present a very difficult one.

On one hand, it is apparent that only a minimal fraction of a large data body can be explored or analyzed by traditional methods. In fact, such methods are based on either a direct interaction with the retrieved data, or simple summaries and reports, or statistical and machine learning algorithms which however are not designed to cope with very large data sets, and are thus generally unusable.

On the other, the exploration and analysis of data is an important task in the environments where database systems are traditionally deployed, but current database systems do not provide the functionalities to help analysts in performing it. Notably, the technology to provide such functionality has not been an important area of database research which, for some decades, has satisfied the demand for a technology of distributed information systems with great emphasis on efficient and reliable storage and retrieval of very large amounts of data, capable of concurrency and access control.

For the above reasons, there is general consensus among researchers involved in the disciplines related to data and information management and analysis, including machine learning, expert systems, database systems, statistics, and data visualization, that there is a need for a new generation of tools for automated data mining and knowledge discovery. The attempt to satisfy this need has resulted in the emerging field of research referred to as *Knowledge Discovery in Databases* (KDD).

A concise definition of KDD can be found in [35]:

> Knowledge discovery in databases is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data.

The definition above can be briefly explained as follows. The objects which we search for are patterns, i.e., expressions in some language stating properties in a subset of the data which are "simpler" than the data. Such patterns must be valid, in that they must hold to a certain extent in new data, novel, that is previously unknown, potentially useful to the organization owning the data as a factor in a decision making process, and ultimately understandable by humans in order to explain the structure of the data and interpret the results. Notice that KDD is defined as a process since it is generally an activity performed by iterating a sequence of stages (data preparation, search for patterns, evaluations of results, refinement).

KDD is distinct from *data mining*, which is defined in [35] as follows:

> Data mining is a step in the KDD process consisting of particular data mining algorithms that, under some acceptable computational efficiency limitations, produces a particular enumeration of patterns.

In the last few years, several data mining algorithms and related techniques have been proposed. It is beyond the scope of this introduction to review all of them. Here we will succinctly describe some of the most popular ones.

Classification. It is defined as the learning from the data of a function to map, or classify, data points into classes. Such a function can be used straightforwardly in deciding which action

to take if the classes are naturally associated to different actions, e.g. deciding whether or not to grant a loan to a customer.

Association Rules. Statements of conditional distribution between two events of the form $\{A_1 \wedge \ldots \wedge A_{n-1}\}$ and $\{A_1 \wedge \ldots \wedge A_n\}$, where $A_1, \ldots, A_n$ are taken from a set of boolean properties which are meaningful in the database, and represent propositions that are true in the event. For instance, $\frac{|\{Bread \wedge Butter \wedge Milk\}|}{|\{Bread \wedge Butter\}|} = .9$, meaning "90% of the transactions which contain bread and butter also contain milk", is an association rule in a transaction database [2]. A typical application of association rules is in the improvement of selling strategies of large retailing companies.

Metaquerying. Metaqueries [65] are generic descriptions of classes of patterns from a relational database. For instance, a metaquery $T \leftarrow L_1, \ldots, L_m$, where T and L_i are literal schemes $Q(Y_1, \ldots, Y_n)$, can be instantiated, or answered, by substituting the predicate variables with predicate (i.e. relation) names from the database. Notice that, unlike association rules, metaqueries link information from several tables. Therefore a typical application of metaquerying is in discovering conditional patterns in complex relational databases.

Clustering. Clustering, or *cluster analysis*, is a method for data exploration, originally proposed by statisticians, whose aim is to group data according to some notion of similarity [46]. In the optimal case, this leads to a maximal similarity between members of the same group and to minimal similarity between members of different groups. Clustering is useful when knowledge about the existence of homogeneous sub-populations in the data helps in deciding which specific actions to take for each sub-population, e.g. in psychiatry, medicine, social services, and market research [32].

Approximate Similarity Queries. Similarity queries between two objects are frequently used in data exploration and mining: Clustering is often based on a notion of similarity between objects, and thus requires similarity queries to be executed; the exploration of multimedia databases can be performed iteratively by query execution and refinement. In all applications (such as the above) where multiple expensive similarity queries have to be executed, the user may be willing to accept a time/quality trade-off, i.e. an error with respect to the exact case is traded for an often dramatic improvement in execution time.

Data Visualization. Various data visualization modalities, together with other techniques specifically oriented to the discovery of correlation or rules, are often used in data mining systems. These visualization modalities are often used to support other techniques, essentially to visualize database results. Furthermore, the availability of different visualizations allows the user to discover new properties, their correlation and find any unexpected values. The user may apply other data mining techniques for further analysis of such "unusual", or rather interesting data. For this reason, data visualization can be seen as a data mining method by itself.

In the rest of the report, we review in greater depth the last four data mining techniques above: Metaquerying, clustering, similarity search, and data visualization. In particular, in each of the four parts we made an effort to find out a common model to compare results and applicability.

Part I

Metaquerying (*Giovanbattista Ianni and Luigi Palopoli*)

2 Introduction

Metaquerying [65] (also called metapattern) is a promising approach for datamining in relational and deductive databases. Metaqueries serve as a generic description of a class of patterns a user is willing to discover. Unlike many other mining tools, patterns discovered using metaqueries can link information from several tables in databases. These patterns are all relational, while most machine-learning systems can only learn propositional patterns and/or work on a single relation. Metaqueries can be specified by human experts or alternatively, they can be automatically generated from the database schema.

Intuitively, a metaquery has the form

$$T \leftarrow L_1, ..., L_m \tag{1}$$

where T and L_i are literal schemes $Q(Y_1, ..., Y_n)$ and Q is either an ordinary predicate name or a predicate variable. In this latter case, $Q(Y_1, ..., Y_n)$ can be instantiated to an atom with predicate symbol denoting a relation in the database. The instantiation must be done in a way that is consistent with variable names. An answer to a metaquery is an (ordinary) rule obtained by consistently substituting second order predicates with relation names.

Shen *et al.* [65] are, to best of our knowledge, the first who have presented a framework that uses metaqueries to integrate inductive learning methods with deductive database technology. For example (taken from [65]), let P,Q, and R be predicate variables and **DB** be a database, then the metaquery

$$R(X, Z) \leftarrow P(X, Y), Q(Y, Z).$$

specifies that the patterns to be discovered are relationships of the form

$$r(X, Z) \leftarrow p(X, Y), q(Y, Z).$$

where p, q, and r are relations from **DB**. For instance, for an appropriate database **DB**, one possible result of this metaquery is the rule:

$$speaks(X, Z) \leftarrow citizen(X, Y),\ language(Y, Z). \tag{2}$$

A rule which serves as an answer to a metaquery is usually accompanied by two indices, that indicate its "plausibility degree". In [12], for example, each rule in the answer is supplied with *support* and *confidence*. The *support* indicates how frequently the body of the rule is satisfied, and the *confidence* measures what fraction of the tuples that satisfy the body, also satisfy the head. A confidence of 0.93, for example, for the rule (2) means that out of all pairs (X, Z) that satisfy the body of the rule, 93% also satisfy the head. An admissibility threshold for the support and confidence is usually provided by the user.

Similar to the case of association rules, the notions of *support* and *confidence* and other indexes have two major purposes:

1. to avoid presenting negligible information to the user.
2. to cut off the search space by early detection of low support and confidence.

Support and confidence have been also defined for other datamining techniques such as association rules [3].

Metaquerying was implemented in several datamining systems [65, 26, 27] and was arguably very useful in knowledge discovery [73, 65, 57, 37]. Theoretical studies about this technique can be found in [12] and [7].

In the following we will provide a formal definition of the concept of metaquery, and of the most common plausibility indices. Then, we will overview on systems which featured some version of metaquerying. Next, we will show some of the computational issues related to this technique, and we will outline some fundamental difference between metaquerying and association rules.

3 What is a metaquery and how to evaluate it

Let U be a countable domain of constants. A database DB is $(D, R_1, ..., R_n)$ where $D \subseteq U$ is finite, and each R_i is a relation of fixed arity $a(R_i)$ such that $R_i \subseteq D^{a(R_i)}$.

unless As stated above, a metaquery **MQ** is a second-order template describing a pattern to be discovered [65]. Such a template has the form

$$T \leftarrow L_1, ..., L_m \tag{3}$$

where T and L_i are literal schemes. Each literal scheme T or L_i has the form $Q(Y_1, ..., Y_n)$ where Q is either a *predicate (second order) variable* or a relation symbol, and each Y_j $(1 \leq j \leq n)$ is an ordinary (first order) variable. If Q is a predicate variable, then $Q(Y_1, ..., Y_n)$ is called a *relation pattern* of arity n, otherwise it is called an *atom* of arity n. The right-hand-side $L_1, ..., L_m$ is called the *body* of the metaquery, while T is called the *head* of the metaquery. A metaquery is called *pure* if each of its relation patterns with the same predicate variable has the same arity.

Intuitively, given a database instance **DB**, answering a metaquery **MQ** on **DB** amounts to finding all substitutions σ of relation patterns appearing in **MQ** by atoms having as predicate names relations in **DB**, such that the Horn rule $\sigma(\mathbf{MQ})$ (obtained by applying σ to **MQ**) encodes a dependency between the atoms in its head and body. The Horn rule is supposed to hold in **DB** with a certain degree of plausibility. The plausibility is defined in terms of *indexes* which we will formally define shortly.

Let **MQ** be a metaquery and **DB** a database. Let $pv(\mathbf{MQ})$, $ls(\mathbf{MQ})$, and $rep(\mathbf{MQ})$ denote the set of predicate variables, the set of literal schemes, and the set of relation patterns occurring in **MQ**, respectively (note that $rep(\mathbf{MQ}) \subseteq ls(\mathbf{MQ})$. Moreover, let $rel(\mathbf{DB})$ denote the set of relation names of **DB** and $ato(\mathbf{DB})$ denote the set of all the atoms of the form $p(T_1, \ldots, T_k)$ where $p \in rel(\mathbf{DB})$, k is the arity of p, and each T_i is an ordinary variable .

Semantics is defined via types of *metaquery instantiations*: an instantiation type specifies how relation patterns can be instantiated, turning a metaquery to an ordinary Horn rule over the given database. Next, we define three different types of metaquery instantiations, that we call *type-0, type-1* and *type-2*, respectively. In the literature, metaquerying semantics has not been always precisely defined, even if a kind of type-2 semantics is usually assumed (see, e.g.,[58]).

Definition 3.1 Let **MQ** be a metaquery and **DB** a database. An *instantiation* (on **MQ** and **DB**) is a mapping $\sigma : rep(\mathbf{MQ}) \rightarrow ato(\mathbf{DB})$, whose restriction $\sigma' : pv(\mathbf{MQ}) \rightarrow rel(\mathbf{DB})$ is functional.

The condition above says that predicate names of **MQ** are *consistently* substituted with relation names from **DB**.

Definition 3.2 Let **MQ** be a pure metaquery. An instantiation σ is *type-0* if for any relation pattern L and atom A, $\sigma(L) = A$ implies that L and A have the same list of arguments.

That is, under type-0 semantics, each predicate variable is always matched to a relation with the same arity and ordinary variables are left untouched. As an example, consider the database $\mathbf{DB}_1$ shown in Figure 1 and the metaquery

$$R(X,Z) \leftarrow P(X,Y), Q(Y,Z) \tag{4}$$

A possible type-0 instantiation for **MQ** is

$$\sigma = \{\langle R(X,Z), \mathrm{UsPT}(X,Z)\rangle, \langle P(X,Y), \mathrm{UsCa}(X,Y)\rangle, \langle Q(Y,Z), \mathrm{CaTe}(Y,Z)\rangle\}$$

which yields the following Horn rule when applied to **MQ**:

$$\mathrm{UsPT}(X,Z) \leftarrow \mathrm{UsCa}(X,Y), \mathrm{CaTe}(Y,Z)$$

Definition 3.3 Let **MQ** be a pure metaquery. An instantiation σ is *type-1* if for any relation pattern L and atom A, $\sigma(L) = A$ implies that the arguments of A are obtained from arguments of L by permutation. *consistently mapped by* σ

With type-1 instantiations, variable ordering within relation patterns "does not matter". As an example, under this semantics, from metaquery (4) and $\mathbf{DB}_1$, among others, both the following Horn rules can be obtained:

$$\mathrm{UsPT}(X,Z) \leftarrow \mathrm{UsCa}(X,Y), \mathrm{CaTe}(Y,Z)$$
$$\mathrm{UsPT}(X,Z) \leftarrow \mathrm{UsCa}(Y,X), \mathrm{CaTe}(Y,Z)$$

The third type of instantiation takes a step further by allowing a relation pattern of arity k to be matched with an atom of arity k', with $k' \geq k$, padding "remaining" arguments to free variables:

Definition 3.4 Let **MQ** be a metaquery. An instantiation σ is *type-2* if for any relation pattern L and atom A, $\sigma(L) = A$ implies the following:

- the arity k' of A is greater-than or equal-to the arity of L;
- k of the arguments of A coincide with the k arguments of L, possibly occurring in different positions;
- the remaining $k'-k$ arguments of A are variables not occurring elsewhere in the instantiated rule.

With type-2 instantiations we can express interesting patterns ignoring how many extra attributes a physical relation may have. Should the relation *UsPT* be defined with an additional attribute, as in Figure 2, the metaquery (4) can be instantiated, using a type-2 instantiation, to

$$\mathrm{UsPT}(X,Z,T) \leftarrow \mathrm{UsCa}(Y,X), \mathrm{CaTe}(Y,Z)$$

User	Carrier
John K.	Omnitel
John K.	Tim
Anastasia A.	Omnitel

Carrier	Technology
Tim	ETACS
Tim	GSM 900
Tim	GSM 1800
Omnitel	GSM 900
Omnitel	GSM 1800
Wind	GSM 1800

User	Phone Type
John K.	GSM 900
John K.	GSM 1800
Anastasia A.	GSM 900

Figure 1: The relations UsCa, CaTe, and UsPT of $\mathbf{DB}_1$

User	Phone Type	Model
John K.	GSM 900	Nokia 6150
John K.	GSM 1800	Nokia 6150
Anastasia A.	GSM 900	Bosch 607

Figure 2: The new relation $UsPT$

Note that a type-0 instantiation is a type-1 instantiation where the chosen permutation of relations's attributes is the identity, whereas a type-1 instantiation is a type-2 instantiation, where the arity of the atoms matches the arity of the relation patterns they are substituted for.

Note, moreover, that type-2 instantiations may apply to any metaquery, while type-0 and type-1 instantiations require pure metaqueries.

4 Plausibility Indexes

In datamining applications, one is generally interested in discovering plausible patterns of data that represent significant knowledge in the analyzed data sets. In other words, it is not typically required for a discovered pattern to absolutely characterize the entire given data set, but a significant subset thereof. This idea is embedded in the usage of *plausibility indexes*. In the literature, several plausibility indices definitions are found, such as *support*, *confidence*, *base* and *strength* [58, 12, 3]. In fact, support is similar to base and confidence is similar to strength [12]. In the analysis that follows, we shall use support, confidence and another useful index that we call *cover*. We first provide formal definitions of the indices that we use.

For any set F, let $|F|$ denote its size. We assume the reader is familiar with relational algebra and Datalog (see [83]). Unless otherwise specified, a predicate name p will denote its corresponding underlying relation as well, and an atom $p(\mathbf{X})$ will denote the corresponding database relation, where the list of arguments $\mathbf{X}$ is used, as in Datalog, to positionally refer to p's columns. For a set of atoms R, $att(R)$ is the set of all the variables of all the atoms in R, and $\mathbf{J}(R)$ is the natural join of all the atoms in R.

Definition 4.1 A plausibility index, or index in short, is a function which maps a database instance **DB** and a Horn rule $h(\mathbf{X}) \leftarrow b_1(\mathbf{X}_1), ..., b_n(\mathbf{X}_n)$ (defined over **DB**) to a rational number representing a value within $[0, 1]$.

Definition 4.2 Let R and S be two sets of atoms. Then the fraction of R in S, denoted $R \uparrow S$, is

$$\frac{|\pi_{att(R)}(\mathbf{J}(R) \bowtie \mathbf{J}(S))|}{|\mathbf{J}(R)|}.$$

In particular, whenever $|\pi_{att(R)}(\mathbf{J}(R) \bowtie \mathbf{J}(S))| = 0$, $R \uparrow S$ is equal to 0.

Definition 4.3 Let r be a Horn rule and let **DB** be a database. Let $h(r)$ and $b(r)$ denote the sets of atoms occurring in the head and in the body of r, respectively. Then

- the *confidence* of r is $cnf(r) = b(r) \uparrow h(r)$,
- the *cover* of r is $cvr(r) = h(r) \uparrow b(r)$,
- the *support* of r is $sup(r) = \max_{a \in b(r)}(\{a\} \uparrow b(r))$;

In the sequel, the set of plausibility indexes $\{cnf, cvr, sup\}$ will be denoted **I**.

Intuitively, $sup(r)$ measures how much the body (or part of it) of an instantiation contains satisfying tuples. When an instantiation scores an high support, a pattern search algorithm may conclude that it is worth to further consider such an instantiation, because there is at least one relation with a high percentage of its tuples satisfying the instantiated body. The concept of base [58] is similar. A technical discussion about differences between these two indices is carried out in [12].

When an instantiation scores a high confidence, we can conclude that a high percentage of the assignments which satisfy the body also satisfy the head relation. Hence, confidence implies how much valid the rule is over the given database. Given a rule r, the indices $cnf(r)$ and $sup(r)$ are equivalent to *confidence* and *support* defined in [12]. The purpose of strength [58] is similar to confidence.

Conversely, cover tells which is the percentage of implied tuples belonging to the head relation. The latter index is found useful in those application where is necessary to decide if it is worth to store the head relation or to compute it with a reasonably matching view.

5 State of the art evaluation systems

5.1 Efficient metaquery answering

The Fleximine environment [28] provides a complete KDD system, designed as a testbed for datamining research. Its open-ended design allows the integration of many existing datamining techniques such as association rule induction and metaquerying. A detailed description on how metaquerying is implemented in Fleximine can be found in [12]. Algorithms employed in Fleximine in order to answer metaqueries are similar to classical Constraint Satisfaction techniques. Fleximine employs a very simple CSP algorithm (forward checking with Backjumping [24]), and the indices used are support and confidence. The basic instantiation algorithm is shown in Figure 3.

At the end of the stage (if it succeeds) each relation pattern $R(X_1, ..., X_n)$ that appear in the metaquery is instantiated. That is, R is bound to some relation name r and each variable is bound to an attribute ("field") of the relation. We assume that a procedure $att(r, X)$ can return the attribute in r to which the variable X is bound.

Next, a filtration stage carries out the following steps: filtering out rules with low support, and filtering out rules with low confidence. Confidence is computed only for rules with sufficient support. Fleximine focused so far on algorithms for computing support. Three alternatives are proposed:

1. Join approach: the straightforward way: computing the equijoin of the body of the rule, then computing $S_i = \frac{|J_{r_i}|}{|r_i|}$ for each instantiated relation r_i in the body, and then taking the maximum.

2. Histogram approach: Using histograms for estimating support. Computing the support using the Join approach only for rules with high estimated support.

3. Histogram + memory approach: same as the histogram approach, except that intermediate results are stored in memory, and reused whenever the same computation is requested.

The straightforward way to calculate the support is computing the support S_i of each relation by performing a natural Join, and then taking $\text{Max}\{S_i | 1 \leq i \leq m\}$. This is done by the Join approach mentioned above. Since the Join is an expensive operation, some low-cost procedures are introduced in order to guarantee an early detection of low support. The other two approaches compute an upper bound on the support and then compute the exact support only for rules

1. *Instantiate(SR)*
2. Input: SR: a set $R_0, R_1, ..., R_n$ of relation patterns, partially instantiated, each with a set of constraints $C_0, ..., C_n$. Initially the constraints sets are all empty.
3. Output: Relations $r0, r1...rn$ that match the relation patterns in SR with the variables bound to attributes.
   (a) If SR is completely instantiated then return SR.
   (b) Pick the next uninstantiated relation variable R_i from SR, {here we should use CSP *variable* ordering techniques to choose}
   (c) Pick up the next possible instantiation r to R_i (r should meet the constraints in C_i and should be of the same arity as R_i). {here we should use CSP *value* ordering techniques to choose}
   (d) For each relation pattern R_j not yet instantiated do:
      i. if R_j has a common field variable X with R_i, add to the C_j the constraint that X must be bound to an attribute with type T_X, where T_X is the type of the attribute that X is bound to in r.
   (e) Call Instantiate recursively.

Figure 3: **Algorithm for the instantiation stage**

1. **compute-support**($r_1, ..., r_m$, $MinSupport$)
2. Input: Set of relations $r_1, ..., r_m$, where each of the attributes is bound to a variable. A support threshold $MinSupport$.
3. Output: if the rule whose body is $r_1, ..., r_m$ has support equal or larger than $MinSupport$, return the support, otherwise return -1.
4. $RelSetCopy = RelSet = \{r_1, ..., r_m\}$; $LowSupp = true$;
5. While ($RelSet \neq \emptyset$) **and** $LowSupp$ do
   (a) Let $r \in RelSet$;
   (b) $s = S_i$-upbound$(r, RelSetCopy)$;
   (c) i. if $s \geq MinSupport$ then $LowSupp = false$
      ii. else $RelSet = RelSet - \{r\}$;
6. (a) If $LowSupp$ then return -1
   (b) else return Join-support$(r_1, ..., r_m)$

Figure 4: **computing support for a rule body**

1. S_i-**upbound-brave**(r_i,R)
2. input: A relation r_i and a set of relations R.
3. output: An upper bound on S_i for a rule whose body is $R \cup \{r_i\}$.
4. $s = 1.0$
5. If there is $r' \in R$ such that r' and r_i have a common variable X then s =upbound$(r_i, r', att(r_i, X), att(r', X))$;
6. return s;

Figure 5: **computing S_i bravely**

1. S_i-**upbound-cautious**(r_i,R)
2. input: A relation r_i and a set of relations R.
3. output: An upper bound on S_i for a rule whose body is $R \cup \{r_i\}$.
4. $s = 1.0$
5. (a) for each relation r' in R such that r_i and r' have variables in common
   (b) do
      i. for each common variable X of r_i and r'
      ii. do
      s' =upbound$(r_i, r', att(r_i, X), att(r', X))$;
      if $s' < s$ then s=s';
6. return s;

Figure 6: **computing S_i cautiously**

with high enough upper bound of support. The idea is summarized in Algorithm compute-support in Figure 4. Note that once one relation with high S_i is found, the procedure Join-support$(r_1, ..., r_m)$ is called. This procedure simply computes the exact support using Join.

The procedure S_i-upbound called by the algorithm compute-support returns an upper bound for the value S_i for a single relation r_i in the body of the rule. This can be done by one of the two procedures: S_i-upbound-brave or S_i-upbound-cautious, shown in Figures 5 and 6, respectively. The basic idea is that an upper bound can be achieved by taking the join of a relation r_i with any other relation with which r_i has variables in common. Procedure S_i-upbound-brave does this by picking one arbitrary relation with which r_i has a common variable, and procedure S_i-upbound-cautious does this by considering *all* relations with which r_i has variables in common, and taking the minimum. Procedure S_i-upbound-cautious works harder than procedure S_i-upbound-brave but it achieves a tighter upper bound and hence can save more Join computations.

5.2 Data type handling

When we deal with real life databases, some issue arise towards data types involved in a metaquery instantiation. Of course, several instantiations may not make sense since they could be link attributes with different data types from different tables. An approach intended to early exclude from evaluation those instantiated metaqueries which do not meet data types constraint, can be found in [58]. This paper introduces the concept of *overlap*. Two attribute from different tables cannot be joined if they do not share a appropriately large set of values. Let C_x and C_y be two attributes and V_x and V_y be their value sets, respectively. The *overlap* is

1. **upbound-histo**(r_1, r_2, att_1, att_2)
2. input: Two relations r_1 and r_2, att_1 an attribute of r_1, att_2 an attribute of r_2. att_1 and att_2 are of the same type.
3. output: An upper bound on the support where only r_1 and r_2 are considered.
4. Let U be the set of all distinct values in att_1 of r_1 and att_2 of r_2.
5. Let h_1 be a histogram with domain U of the values in att_1 of r_1. (If there is no such histogram, build it).
6. Let h_2 be a histogram with domain U of the values in att_2 of r_2. (If there is no such histogram, build it).
7. return Histo($r_1, r_2, h_1, h_2, |U|$);

Figure 7: Computing support from histograms

$$\text{Overlap}(C_x, C_y) = \max\left(\frac{|V_x \cap V_y|}{|V_x|}, \frac{|V_x \cap V_y|}{|V_y|}\right)$$

An overlap table may be employed in order to eliminate unnecessary connections (e.g. height vs. temperature) and/or to drive automatic pattern generation.

5.3 Unsupervised generation of metapattern

Since the task of generating suitable metaqueries could be a heavy burden for the end user, some completely automatic metaquery generation system has been introduced [65, 58]. The proposed approach consists in starting with a set of most general metapatterns, and incrementally generate the interesting ones as the process of discovery continues. The search process is not complete, but guided in (hopefully) fruitful directions by plausibility indixes. [58] proposes a specific approach for what is called the *transitivity* pattern class. The automatic pattern generation is done by considering all the patterns belonging to the considered class, and by literal pattern insertion. The underlying idea is that adding a literal pattern to a metaquery can improve the strength (or the confidence) of instantiated rules. As an example the metaquery

$$R_1(Y_3, Y_1) \leftarrow P_1(Y_1, Y_2), Q_1(Y_2, Y_3), S_1(Y_2, Z) \tag{5}$$

may have more chance to generate rules scoring an high value of confidence with respect to the metaquery

$$R_1(Y_3, Y_1) \leftarrow P_1(Y_1, Y_2), Q_1(Y_2, Y_3)$$

from which 5 is derived from (although lower values for support are expected).

6 Computational Issues

We present here several results regarding the complexity of metaquerying. We assume the reader is familiar with basic notions regarding complexity classes and, in particular, the definition of the polynomial hierarchy [79]. Moreover, we recall that AC^0 is the class of decision problems solved

by uniform families of circuits of polynomial size and constant depth [10, 70], and LOGCFL coincides with the class of decision problems logspace-reducible to a context free language [70]

As for the complexity of queries, metaquery complexity can be defined according to two complexity measures, namely, *combined complexity* and *data complexity* [84]. Let T denote an instantiation type out of $\{0,1,2\}$ and I denote an index, i.e. a function that given a Horn rule and a database instance returns a rational value $\in [0,1]$. Let $\mathbf{DB}$ denote a database instance, $\mathbf{MQ}$ a metaquery, and k a rational threshold value s.t. $0 \leq k < 1$. Then, the *data* complexity of the metaquery problem $\langle \mathbf{DB}, \mathbf{MQ}, I, k \rangle_T$ is the complexity of deciding if there exists a type-T instantiation σ for $\mathbf{MQ}$ such that $I(\sigma(\mathbf{MQ})) > k$, when $|\mathbf{DB}|$ varies and k and $\mathbf{MQ}$ are fixed. Furthermore, the *combined* complexity of the metaquery problem $\langle \mathbf{DB}, \mathbf{MQ}, I, k \rangle_T$ is the complexity of deciding if there exists a type-T instantiation σ for $\mathbf{MQ}$ such that, when $|\mathbf{DB}|$ and $|\mathbf{MQ}|$ varies and k is fixed, $I(\sigma(\mathbf{MQ})) > k$.

In the literature it is usually assumed that, in answering a metaquery $\mathbf{MQ}$, one looks for rules that satisfy $\mathbf{MQ}$ and have a certain level of support and confidence [12, 58]. Here we preferred to split the metaquery problem as to refer to one index at a time. The rationale is the following. First, this allows us to single out more precisely complexity sources. Second, technical presentation is simpler this way. Third, complexity measures for problems involving more than one index can be obtained fairly easily from metaquerying problems having only one index. The forthcoming results are taken from [7]. We begin by analyzing the cases when the threshold value k is set to 0.

Theorem 6.1 *Let $I \in \mathbf{I}$. The combined complexity of $\langle \mathbf{DB}, \mathbf{MQ}, I, 0 \rangle_T$ is NP-complete, for any instantiation type $T \in \{0,1,2\}$.*

Next, we consider the combined complexity of metaquerying when the fixed threshold k is s.t. $0 \leq k < 1$.

Theorem 6.2 *Let $I \in \mathbf{I}$. The combined complexity of $\langle \mathbf{DB}, \mathbf{MQ}, I, k \rangle_T$, with $0 \leq k < 1$, and instantiation type $T \in \{0,1,2\}$ is NP-complete if $I = cvr$ or $I = sup$ and it is in PSPACE if $I = cnf$.*

Thus, as far as the combined complexity measure is concerned, metaquerying is intractable. Next we discuss some tractable subcases, dealing with the concept of *acyclic* metaquery.

Definition 6.3 Let $\mathbf{MQ}$ be a metaquery. The set of (both predicate and ordinary) variables of $\mathbf{MQ}$ is denoted $var(\mathbf{MQ})$. Define the *hypergraph* $H(\mathbf{MQ}) = \langle V, E \rangle$ associated to $\mathbf{MQ}$ as follows. $V = var(\mathbf{MQ})$ and E contains an edge e_i for each literal scheme $r_i(\mathbf{X}_i)$ in $\mathbf{MQ}$, where e_i is the set of variables occurring in $r_i(\mathbf{X}_i)$. We say that $\mathbf{MQ}$ is *acyclic* if $H(\mathbf{MQ})$ is acyclic.

For example, the metaquery

$$\mathbf{MQ}_1 = P(X,Y) \leftarrow P(Y,Z), Q(Z,W)$$

is acyclic, whereas the slight different metaquery

$$\mathbf{MQ}_2 = P(X,Y) \leftarrow Q(Y,Z), P(Z,W)$$

is cyclic.

Theorem 6.4 *Let $\mathbf{MQ}$ be an acyclic metaquery and $I \in \mathbf{I}$. The combined complexity of $\langle \mathbf{DB}, \mathbf{MQ}, I, 0 \rangle_0$ is LOGCFL-complete under logspace reductions.*

However, acyclicity is not sufficient to guarantee tractability in general, as shown next for instantiation types other than type-0.

Theorem 6.5 *Let $I \in \mathbf{I}$ and $T \in \{1,2\}$ and let $\mathbf{MQ}$ be an acyclic metaquery, then $\langle \mathbf{DB}, \mathbf{MQ}, I, 0 \rangle_T$ is NP-hard.*

Similarly to most query languages, the data complexity is much lower than the combined complexity. In particular, in some interesting case, it lies very low in the complexity hierarchy, as proven next.

Theorem 6.6 *Under the data complexity measure (fixed metaquery and threshold value, variable database instance), $\langle \mathbf{DB}, \mathbf{MQ}, I, 0 \rangle_T$ is in AC^0, for any $I \in \mathbf{I}$ and for any $T \in \{0,1,2\}$.*

In the general case, the data complexity of metaquerying is within P.

Theorem 6.7 *The data complexity of the metaquerying problem $\langle \mathbf{DB}, \mathbf{MQ}, I, k, T \rangle$, $0 \leq k < 1$ is in P, for $I \in \mathbf{I}$ and $T \in \{0,1,2\}$.*

7 Metaquerying and Association Rules

Roughly speaking, an association rule is a rule $X \rightarrow Y$, where X and Y are set of items [2]. The interest value of a rule is tested on a transaction database (i.e. a set of sets of items), and, like metaqueries, interestingness is measured using the concepts of support and confidence, which are suitably defined (some other indexes of usefulness of an association rule are proposed in [14] and [56]). Contrary to metaquerying techniques, association rules are largely studied in literature for what it concerns efficient evaluation (e.g. [4, 18]), and maintenance (e.g. [19, 17]). There exists many variants of that technique, such as sequential pattern [5], and generalized association rules [78]. A preliminary result about computational properties of association rules was given in [66].

Although both techniques may artfully simulate the other, the kind of patterns which can be learned using association rules is more specific, and requires data stored on a single table, whereas metaquerying takes advantage of relational schemes. Metaquerying can be directly applied to native relations and/or on existing views as well; association rule learning systems often need preprocessing steps in order to denormalize data on a single table, instead. The denormalization process introduces problems for dealing with redundancy, and in building the required transaction table. Such troubles are usually solved by human choice. Suppose you want to mine association rules on the following set of relations:

Student(StudID, ..., Sex, ...), Course(CourseID, ..., Hours, ...),

Student_Course(CourseID, StudID)

As an example, taking the join of these three tables would give wrong values for what it concerns Sex percentages (e.g. there are students following many courses): thus the process of building a transactions table from this preprocessed view should be careful. A good way to do this would be denormalizing Student w.r.t. Course, i.e. building a table of transactions where each student corresponds to a transaction and each course is a new (boolean) field telling if a given student follows a given course. Conversely, one might be interested in Courses, and then the transaction table should be built in a symmetric way (considering students as new fields of Courses). In general, the process of mining association rules involves decisional processes like this: thus, although association rules represent a very useful data mining technique, its direct applicability is sometimes difficult.

Part II
Data clustering (*Stefano Lodi and Claudio Sartori*)

8 Introduction

Cluster analysis is a method for data exploration, originally proposed by statisticians, whose aim is to group data according to some notion of similarity [46]. In the optimal case, this leads to a maximal similarity between members of the same group and to minimal similarity between members of different groups.

More recently, cluster analysis has been proposed as a *data mining* activity [35], named *data clustering*. In this environment the problem has further constraints, requiring efficiency for data sets which are much larger than the ones usually described in statistical or pattern recognition literature. For such problem instances, algorithms with subquadratic complexity are the only candidates to practical use. If the data set exceeds main memory size, then the additional constraint of I/O minimization has also to be satisfied.

Although the emphasis is placed on efficiency, it is not the case that methods proposed for data clustering trade effectiveness for efficiency. Many research efforts have contributed original algorithms whose accuracy is comparable to known statistical methods. The problem of clustering nonquantitative data has also received considerable attention. Generally, theories for nonquantitative data imply weaker properties and are therefore less amenable to efficient processing.

9 A Comparison Model

To establish a comparison model for data clustering algorithms, we separate design choices, theoretical complexity, experimental efficiency, and effectiveness in recognizing clustered data.

9.1 Design Choices

The approaches proposed in the literature will be classified according to the following properties.

1. Mathematical structure of the data space.
2. Effective construction of clusters.
3. Degree of dynamic processing (none, partial, full).
4. Cardinality reduction (such as random sampling).

9.1.1 Structure of the Data Space

Many data clustering algorithms are only enabled to process data which belong to specified mathematical structures, because designers can take advantage of mathematical properties to achieve better efficiency or accuracy.

We can thus classify the approaches into three categories according to the required structure. Since mathematical structures form a hierarchy, we will classify an effort in the most general category that describes it.

1. *Quantitative Data:* In this group, which includes two proposals, we collect all approaches to the problem of clustering tuples of numerical data, which make use of, at least, the vector sum operation.

2. *Categorical Data:* Two efforts deal with categorical data, i.e., sets of data tuples such that no operation on values or tuples, and no predicate on values or tuples other that equality can be considered meaningful. It is also assumed that the number of allowed values is small. (At most in the order of thousands.)

3. *Metric Data:* The four proposals in this class require only the definition of a *distance function*, that is, a real function of object pairs $d\colon S \times S \to \mathbb{R}$ satisfying the four *metric axioms*:

$$d(x,x) = 0 \tag{6}$$
$$d(x,y) \geq 0 \qquad \text{(nonnegativity)} \tag{7}$$
$$d(x,y) = d(y,x) \qquad \text{(symmetry)} \tag{8}$$
$$d(x,y) + d(y,z) \geq d(x,z) \qquad \text{(triangle inequality)} \tag{9}$$

   Since quantitative data sets are usually equipped with a Minkowski metric, e.g. the euclidean distance, the class of metric approaches subsumes class 1.

9.1.2 Cluster Construction (Labeling)

In many applications, the output of a clustering algorithm must include an explicit grouping of objects into clusters. Such grouping is usually represented as a *labeling function* on pairs of obiects, which takes equal values if and only if both objects belong to the same cluster.

All approaches in this survey provide such functionality, although some approaches are flexible, in that labeling is an optional phase to be run at the end of normal processing. If labeling is not performed, the output of the algorithm consists of some kind of numerical or graphical summary of the data which gives insight about the clustering structure of the data.

9.1.3 Dynamic Processing

A *dynamic algorithm* must perform a *pre–processing stage*, followed by a number of *update stages* [64]. The pre–processing stage reads an initial instance of the problem and computes the initial state which will be used in the iterations and, possibly, an initial solution. An update stage reads an update of the current instance and generates a new state and a new solution, starting the current state.

Traditionally, clustering algorithms are static: no part of the result of a run of an algorithm on a data set $\mathcal{D}$ can be used by subsequent runs on updates of $\mathcal{D}$. Only recently, it has been pointed out that dynamic clustering algorithms are useful in the following scenarios.

- *Data warehousing:* A data warehouse materializes views from a set of base (operational) tables and performs on them some analytical task. Usually a data warehouse is rebuilt during the periods of inactivity of the operational systems [67], but problem size can easily lead to situations where the inactivity time is not sufficient for recomputing clusters from scratch.

- *On line clustering:* In some applications, the data set is extremely large and the labeling of objects is not required. In these cases, the user may want to resort to a compromise between accuracy of clustering and running time, making use of an on-line algorithm to be halted when the (approximate) result is considered satisfactory.

We may classify the approaches as follows.

1. *Static:* No dynamic processing can be performed. Most approaches (five) fall into this category.

2. *Partially dynamic:* This category includes two strictly related approaches. In both, the whole data base is not needed in advance: cluster statistics are recomputed incrementally as objects are read from the data base.

3. *Fully dynamic:* The only approach in this class supports both insertions and deletions of objects.

9.1.4 Cardinality Reduction

In knowledge discovery, one simple way to cope with the ever increasing problem size is to reduce the cardinality of the data set to analyze, and then apply the algorithm to the reduced data set. Of course, the problem is then to bring theoretical or experimental evidence that the results can be extended to the entire data set accurately enough.

Among the work surveyed here, the only technique which has been employed is *random sampling*. In one approach, known external memory algorithms are utilized to extract a sample from the data set and then the proposed method is applied to the sample.

9.2 Accuracy

Authors usually validate che accuracy of clustering algorithms experimentally: Both the proposed algorithm and algorithms known from the literature are run on a few synthetic and real data sets, and the results are compared, mostly in an informal way. Very often, the data sets used for the comparison are two-dimensional data sets containing "natural" clusters, i.e., clusters which are perceived as such by a human observer. This is possible only in case the algorithms to be compared are applicable to quantitative or metric data. For categorical data, either the application example determines the perceived natural clusters, or the clustered structure of the data set satisfies some rigourous formal definition of cluster assumed by the authors.

Among the clusters frequently used for comparisons, one finds:

1. clusters with widely different densities,

2. arbitrary shape clusters,

3. clusters with variable density,

4. clusters connected by thin lines of points,

5. clusters with overlapping projections,

6. clusters with hierarchical structure (clusters inside clusters).

The result of the comparison usually shows empirical subjective evidence that either the new proposed algorithm produces one cluster for every natural cluster in the data set, whereas algorithm in the literature aggregate or split clusters, or produce clusters having a larger overlap with the natural clusters.

It must be noted that a formal comparative analysis of accuracy is usually impossible since only a few works among the ones surveyed here define clusters formally.

9.3 Complexity

Due to the variety of cluster definitions and the frequent use of heuristics, theoretical complexity analysis alone is usually not sufficient to measure the efficiency of data clustering algorithms. Therefore, most works include both theoretical complexity and experimental evaluation of the running time.

In some cases, comparative experiments have been conducted. However, source or compiled code for many algorithms is not available. Therefore, some comparisons were performed using reimplementations.

10 Survey

In this section, we will briefly review some approaches in the literature. The methods will be grouped according to the basic principle or technique used to cluster the data, namely:

1. *Density estimation*,
2. *Descriptive Statistics*.

10.1 Approaches based on Density Estimation

The intuition behind density based clustering is simple: Clusters are regions of the space that are densely populated by data points. One may therefore map every data set to a function whose value represents density at a data or space point and use it to direct the grouping of points into clusters of the data set.

We will see that all the approaches surveyed here propose density measures which are simplified (in fact, not normalized) versions of density estimates known from the statistical literature. Thus, we recall a few of those estimates following [74]. For simplicity, we define the estimates assuming the real line as measurable space.

Let $\mathcal{S} = \mathbb{R}$ be the measurable data space and $\mathcal{D} \subseteq \mathcal{S}$ a data set of size N. Let h be a small real number, called *window width*. The *naive estimator* si defined by

$$\hat{f}(x) = \frac{|\mathcal{D} \cap (x-h, x+h)|}{2Nh}\,. \tag{10}$$

Let $K : \mathcal{S} \to \mathbb{R}$ be a function, called *kernel function*, such that

$$\int_{-\infty}^{\infty} K(x)dx = 1. \tag{11}$$

Usually K will be a symmetric probability density function. The *kernel estimator* with kernel K is defined by

$$\hat{f}(x) = \frac{1}{Nh}\sum_{i=1}^{N} K(\frac{x - x_i}{h}). \tag{12}$$

By *rank* $r(y, x, \mathcal{D})$ of y w.r.t. x in $\mathcal{D}$ we mean the number $k \in \mathbb{N}$ such that exactly $k-1$ points of $\mathcal{D}$ are closer to x than y. If y has rank k w.r.t. x, we also say y is the kth nearest neighbour of x. The distance of the kth nearest neighbour of x from x in $\mathcal{D}$ is defined as $d_k(x, \mathcal{D}) = d(x, y)$, where $r(y, x, \mathcal{D}) = k$. The *kth nearest neighbour density estimator* is defined by

$$\hat{f}(x) = \frac{k}{2N d_k(x, \mathcal{D})}. \tag{13}$$

10.1.1 DBSCAN

The primitive notion of the DBSCAN approach [30] is the definition of *core point*. Intuitively, a core point is a point whose density is estimated to be higher than a given threshold.

Let $\epsilon \in (0, \infty)$, $MinPts \in \mathbb{N}$. A point $x \in \mathcal{D}$ is a core point w.r.t. ϵ, $MinPts$ if and only if it is the center of an ϵ-neighbourhood containing at least $MinPts$ other data points. A binary

relation of *direct density reachability* is established between every point in the ϵ-neighbourhood of a core point and the core point itself. Then, the transitive closure of direct density reachability, *density reachability*, or $>_{\mathcal{D}}$, is defined. Finally, the binary relation *density connectivity* is defined: $x, y \in \mathcal{D}$ are density connected iff there exists a point $z \in \mathcal{D}$ such that $x >_{\mathcal{D}} z$ and $y >_{\mathcal{D}} z$. Finally, a cluster is defined as a nonempty subset such that:

1. C is closed under density connectivity,
2. any two points in C are density connected.

By definition, all points not belonging to any clusters are *noise points*. An algorithmically convenient characterization of a cluster is given by the authors essentially as follows. If c is a core point, the *set generated by* c is defined as $S_c = \{x \in \mathcal{D} : x \text{ density reachable from } c\}$. It can be proved that S_c is a cluster and that S_c equals the set generated by any core point in it. Therefore, an expansion procedure may construct a cluster by selecting an initial core point as seed and recursively adding reachable points until no new point can be added.

The implemented expansion procedure is passed a core point and a cluster identifier, and keeps in memory a queue of seeds. Iteratively, a point x is dequeued, and, if x is a core point, a range query `range`(x, ϵ) is executed. All points in the result are enqueued, with their cluster identifier changed to the passed parameter. When the queue is empty, the procedure returns. The main algorithm scans the data set and calls the expansion procedure once for every core point that is not yet classified (at the time of the call).

The algorithm executes at most N range queries. Since DBSCAN has been proposed mainly for spatial data clustering, an access method supporting range queries is usually available. In this case, the average case I/O complexity will typically be $O(|\mathcal{D}| \log |\mathcal{D}|)$. (e.g. if the method is the R*-tree and the range query region is "small".)

It is apparent that DBSCAN can be applied to metric data sets as well, provided a metric access method is used instead of a spatial one.

Since the two parameters ϵ and *MinPts* must be known a–priori, a heuristic procedure based on nearest neighbours to derive ϵ known *MinPts* is provided. However, some authors have observed that this remains a difficult task.

It is easy to see that that the DBSCAN approach is based on the naive density estimator.

10.1.2 IncrementalDBSCAN

IncrementalDBSCAN [29] is a fully dynamic version of DBSCAN. The goal of this effort is to propose a method which is equivalent to DBSCAN and outperforms it for a wide range of update sizes.

The approach can be intuitively justified as follows. Inserting or deleting a data point changes the reachability relation, and consequently the clusters induced by the relation. Since reachability is defined in terms of the core point property, which is a local property, it is likely that updates can be handled without accessing the entire data set, thereby improving over the static algorithm.

Formally, the set of points (objects) which may change their cluster membership is modeled by the following definition: If x is the inserted or deleted point, the set $\mathit{Affected}_{\mathcal{D}}(x)$ of *affected objects* of x is defined as the union of the ϵ-neighbourhood of x and the set of objects density reachable in $\mathcal{D} \cup \{x\}$ from the ϵ-neighbourhood of x. A consequence of the definition is that objects not in $\mathit{Affected}_{\mathcal{D}}(x)$ induce the same generated set before and after the update (see section 10.1.1 for the definition of generated set). Therefore, the new clustering could be obtained by running DBSCAN on the set $\mathit{Affected}_{\mathcal{D}}(x)$.

It is sufficient, however, to take a much smaller set of points as "seed" for the expansion procedure. In fact, if c is a core object in the updated data set, then the cluster generated by c in the updated data set is contained in $\mathit{Affected}_{\mathcal{D}}(x)$ if and only if c is density reachable by an

object in the ϵ-neighbourhood of x by a chain of length 2. This fact ensures that, for insertion, only core objects directly reachable from *new* core objects are part of the seed, whereas, for deletion, only core objects directly reachable from *former* core objects are part of the seed.

The authors derive the following theoretical formula for the speed-up factor of IncrementalDBSCAN vs. DBSCAN:

$$SF = \frac{N + f_{ins}m - f_{del}m}{m(f_{ins}r_{ins} + f_{del}r_{del})}, \tag{14}$$

where f_{ins}, r_{ins}, f_{del}, r_{del} are the percentage and number of range queries for insertions and deletions, respectively, and m is the size of the update. The evaluation of the algorithm on both a large synthethic data set and a real world data set shows that the average number of range queries is in the order of few units. In practice, the speed-up is in the order of tens even for large numbers of daily updates.

10.1.3 DENCLUE

The DENCLUE [48] method is an attempt to satisfy two important requirements of clustering in multimedia data bases: Noise immunity and robustness at high dimensionality.

The key idea is to construct an estimate of the overall density, find its modes (called *density attractors*), and group points around the modes following the gradient of the estimate. *Noise points* are defined as being exactly the points grouped with modes at which the estimate is less than a threshold.

The density estimate is defined as

$$f^{\mathcal{D}}(x) = \sum_{i=1}^{N} f^{x_i}(x), \tag{15}$$

where $f^y(x)$ is called an *influence function*, and models the influence of data point y on the density at x. In this general setting, no restriction is placed on the form of the dependency of $f^y(x)$ from y. If $f^y(x)$ is just a function $f(x)$ shifted to y, then the density function $f^{\mathcal{D}}(x)$ can be interpreted as a multivariate application of the kernel estimate.

As for kernel estimates, typical examples of good influence functions are the *square wave influence function*:

$$f^y(x) = \begin{cases} 1, & \text{if } d(x,y) \leq \sigma, \\ 0, & \text{otherwise,} \end{cases} \tag{16}$$

and the *gaussian influence function*

$$f^y(x) = \exp(-\frac{d(x,y)^2}{2\sigma^2}) \tag{17}$$

where σ is a parameter serving as window width (see section 10.1).

Two different types of cluster are defined in DENCLUE, *center-defined* and *arbitrary-shape*. A center-defined cluster is a set of data points attracted by the same density attractor, at which density is not less than a threshold ξ. An arbitrary-shape cluster for a set of attractors X is a set of data points, each attracted by some attractor in X at which density is not less than a threshold ξ, with the condition that any two of attractors in X can be connected by a path all points of which are at density not less than ξ.

The authors prove that the probability that density attractors do not change when the number of noise points (i.e., uniformly distributed in the relevant portion of the data space) goes to infinity has limit 1. This result shows that the approach has good noise immunity.

By choosing an appropriate influence function, the DENCLUE approach can be shown to generalize other approaches. For instance, it is apparent that DBSCAN can be simulated by choosing the square wave influence function. Notice however that formally DBSCAN clusters differ slightly from arbitrary-shape clusters. In fact, points whose density is less than ξ are never included in a DBSCAN cluster, whereas they are included in an arbitrary shape cluster if their attractor's density is not less than ξ.

The implemented algorithm partitions the data space into hypercubes of side 2σ, assigning a unique one-dimensional key to each. Empty cubes are discarded and the remaining keys are stored into an access structure, such as a B^+-tree. For any two cubes such that the means of their data points are less than 4σ apart, a connection between them is created to speed up the traversal of the data space during the clustering phase. The density function is approximated by extending summation, for computing density at x, only over points which are in cubes connected to the cube of x, having their means at most 4σ apart from x. Clustering is than performed by following density gradient in small finite steps, halting when density decreases.

The worst case complexity of the approach is $O(|\mathcal{D}| \log |\mathcal{D}|)$. However, the author claim that the average experimental complexity is much lower, about $O(\log |\mathcal{D}|)$.

In spite of the approximations adopted and a worst case complexity not improving over other methods, experimental validation with a real world high dimensional data set of molecular biology with large amounts of noise shows that the implemented algorithm considerably outperforms DBSCAN by a large margin.

10.1.4 OPTICS

In many data sets, the clustering structure is *hierarchical*. Thus, methods which fix a single density threshold tend to hide nested clusters which can separated from one another only by densities below the threshold. Moreover, the clustering will depend heavily on the threshold.

The goal of the OPTICS [8] method is to construct a robust, easy to interpret graphical description of the data, without requiring the specification of critical parameters. The actual labeling of the data can be performed optionally. The key idea is to try to describe the hierarchical clustering structure as a 2-dimensional plot (*pointindex*, *value*). Roughly, point indices and values are assigned in such a way that, for each cluster, there exist a distinct maximal interval of point indices such that no internal point is a local maximum greater than any of the border values. Hierarchical clusters are shown as nested intervals satisfying the above property.

The formal definition of clusters constructed by the OPTICS method is quite involved. We will therefore describe only the method. Let $d_k(x)$ be the distance of x from its kth neighbour. Two parameters $MinPts \in \mathbb{N}$ and $\epsilon \in (0, \infty)$ are assumed. The *core distance* of x is defined as $d_{MinPts}(x)$, for every x such that $d_{MinPts}(x) \leq \epsilon$, and undefined otherwise. As DBSCAN, the OPTICS algorithm is based on a set expansion procedure: At every step, a point is deleted from the set and a range query $\texttt{range}(x, \epsilon)$ is executed only if its core distance is defined. All points in the query result are added to the set. The procedure ends when the queue is empty. In contrast to DBSCAN, the set is maintained as a priority queue, where every point has a dynamically changing real priority, called *reachability distance*. For point $x \in \texttt{range}(y, \epsilon)$, the reachability distance of x w.r.t y is defined only if the core dist the maximum between the core distance of y and $d(x, y)$. When a point x is inserted as part of the result of a range query $\texttt{range}(y, \epsilon)$, its priority is set to the reachability distance w.r.t. y. If later, on deletion of another object z with smaller priority than x, x is encountered again in the query result for z, the minimum between the current priority and its reachability distance w.r.t. z is kept for x.

This ingenious mechanism ensures that the ordering of the points follows a steep path up the estimated density curve to a mode, and goes to a different mode only after trying to gather as many points as possible from the cluster around the current mode. The resulting plot shows a steep downward slope at the begininning of each cluster, when points with a small reachability

distance are added. The plot tends to level in the cluster center, and finally the plotted value grow, possibly more slowly than at the beginning of the cluster (depending on the sharpness of the cluster borders).

The definition of core distance implies that OPTICS is related to the kth nearest neighbour density estimator.

10.2 Approaches based on Descriptive Statistics

10.2.1 BIRCH

The important issues of noise immunity and resource limitations were first addressed by the BIRCH method [90]. Given the amount of memory available, BIRCH tries to compute the most accurate clustering, while minimizing the number of I/Os. The complete method is quite elaborate; we describe in the following the core of the approach, which essentially combines a dynamic index structure with easily maintainable, synthetic descriptions of sets of points.

Any finite vector subspace can be sinthetically described by a triple of elementary statistics, called *clustering feature* (CF): Cardinality, vector sum, and sum of vector squares. Such statistics are incrementally updatable in $O(1)$ time whenever a new point is added to the set, and sufficient for computing in $O(1)$ time the centroid of the set and various distances between the set and a point, and scatter measures (such as set radius or diameter).

BIRCH maintains incrementally a collection of clusters. Let $\mathcal{C}$ be the collection computed over a data set $\mathcal{D}$ and let T be a fixed threshold (a parameter of the algorithm). When a new point x is inserted into $\mathcal{D}$, if for no cluster $C \in \mathcal{C}$, the diameter of $C \cup \{x\}$ is less than T, then a new cluster $\{x\}$ is added to $\mathcal{C}$. Otherwise x is added to the cluster that minimizes the distance of x from C over all $C \in \mathcal{C}$. Hence the invariant maintained by BIRCH is that every cluster diameter is less then the fixed threshold T.

To improve efficiency, BIRCH also maintains a multiway balanced search tree entirely contained in main memory, the *CF-tree*. Every internal node in the tree is a sequence of pairs (CF, p) representing the set of data points described by all its CFs. For each (CF, p), CF describes the set represented by the node pointed to by p. In the leaves, CFs represent clusters. The insertion of a new point is performed by following the path from the root that minimizes, at each traversed node, a predefined distance between the point and the set represented by the CF. After updating the CF, the internal nodes along the path to the leaf are updated. When leaf capacity is exceeded, a split mechanism is triggered. If the insertion exhausts the available memory, then the threshold T is set to $T' > T$ and a new tree for T' is built with the leafs of the current tree. It is proved that the new tree is always smaller, and rebuilding only requires h extra pages of memory, where h is the height of the current tree.

10.2.2 BUBBLE

The BUBBLE method [39] extends the approach of BIRCH to clustering in arbitrary metric spaces. The authors abstract out the structural features of BIRCH which do not depend on vector operations and define a generic framework, BIRCH*, which can be instantiated to generate concrete clustering algorithms.

The fundamental idea the BIRCH* framework shares with BIRCH is to avoid storing sets of points explicitly during the computation: Instead, any set of points $S \subseteq \mathcal{D}$ is represented implicitly by the value of a function $CF^*(S)$ called *generalized cluster feature*. To be advantageous, a clustering feature should be:

1. much smaller than the set it represents: $|CF^*(S)| \ll |S|$,
2. incrementally updatable, i.e., there exists an efficiently computable function g_{CF^*} such that $CF^*(S \cup \{x\}) = g_{CF^*}(x, CF^*(S))$,

3. sufficient to compute efficiently various scatter measures and distances between sets, so that for a measure m, there is a function g_m such that $g_m(CF^*(S)) = m(S)$, and similarly for a distance.

The algorithm maintains a multiway search tree of cluster features, iteratively reading a point x from the data set and selecting a cluster feature in a leaf to be updated. The chosen feature CF is such that the distance between x and CF is minimized over all cluster features, and $m(CF) \leq T$, where m a predefined scatter measure and T a threshold. If no feature can be found satisfying the threshold requirement, a new feature is initiated with point x. When inserting x, the leaf containing the required feature can be found quickly by navigating the tree: At every internal node the child minimizing the distance to x is chosen to continue. When leaf capacity is exceeded, a split mechanism is triggered. Memory constraints are enforced by increasing the threshold T and rebuilding a new smaller tree.

The features designed by the authors for metric data are based on the concept of clustroid, which is the metric counterpart of the centroid. A *clustroid* of an object set S is an object $x \in S$ that minimizes the sum of square distances to all other objects in S, also called the *rowsum* of x. The definition is justified by proving that the clustroid is always the object closest to the centroid of the image of S under multidimensional scaling. The scatter measure used to verify the threshold requirement upon insertion of x is simply the distance between x and the clustroid of the set.

The CF for the BIRCH algorithm clearly satisfies requirements 1, 2, 3 above. However, the clustroid fails to satisfy them, since to update the clustroid of S, all objects in S must be present in main memory. Therefore the authors propose a heuristic strategy (the details of which we omit due to lack of space) based on keeping only a small number of representative objects for each leaf cluster. At any internal node, the cluster feature of an entry is a set of objects, sampled from the set of all objects stored in the features of the entry's child.

In the BUBBLE-FM instantiation, the number of distance computations for navigating the tree is reduced by mapping sample objects at internal nodes to $\mathbb{R}^k$ by FastMap, an efficient implementation of incremental multidimensional scaling.

BUBBLE and BUBBLE-FM are evaluated on three data sets against cascading multidimensional scaling (FastMap) and BIRCH, and are shown to compute clusters having much less distortion (the sum of mean square errors over all clusters). Experimental time complexity both vs. clusters and data set size is almost linear, matching thus the performance of BIRCH. Finally, BUBBLE-FM achieves higher quality clustering on a data cleaning problem in the domain of bibliographic data bases with respect to previous known methods from information retrieval. A direct comparison with other metric and incremental approaches, such as DBSCAN and IncrementalDBSCAN, is however missing. Although details about the test data sets are given, the computing environment is unknown and thus it is impossible even to infer a comparison.

10.2.3 ROCK

The ROCK method [44] for clustering categorical data is motivated by the poor accuracy exhibited by traditional hierarchical clustering algorithms when applied to such data.

The issue is explained by an example. In the domain of market basket analysis, the objects to be clustered are transactions, i.e., subsets of the set of products on sale. Hierarchical methods iteratively merge the closest pair of clusters according to a given intercluster similarity, which is defined in terms of object similarity. Common definitions for the intercluster similarity are the maximum/minimum/average similarity over pairs of objects (not belonging to the same cluster).[1] A popular similarity measure between sets is the *Jaccard coefficient*, defined as the number of common objects divided by the number of objects in either set.

[1] The resulting hierarchical methods are known as single link, complete link, and group average, respectively.

The key empirical observation is that sets of transactions may be recognized as distinct "natural" clusters, because all transactions in each set consist only of items from a subset that is characteristic of the set, although the characteristic subsets overlap. Unfortunately, in such situation, the similarity between the sets may still be small under any intercluster similarity measure, because of the influence exerted by transactions sharing the few items in common to the clusters. Thus, a hierarchical algorithm may merge the clusters at an early stage.

The remedy proposed by the authors is to render object similarity more context sensitive, as opposed to traditional similarity definitions which are based only on the structural properties of the objects. To that purpose, they introduce the novel concept of link between objects.

Let $s\colon \mathcal{D} \to [0,1]$ be a similarity measure and $\theta \in [0,1]$ a threshold. Objects $x, y \in \mathcal{D}$ are said to be *neighbours* if and only if $s(x,y) \geq \theta$. The *link between objects* $x, y \in \mathcal{D}$ is the number of common neighbours to x and y. It is apparent that objects in different transaction clusters are much less likely to have high link value than high similarity. Intercluster similarity is then defined as the sum of links over all pairs of objects not in the same cluster, divided by its expectation.

Having defined suitable object and intercluster similarities, the algorithm follows the usual hierarchical scheme: starting with the clustering where each cluster is a singleton, it maintains a priority queue of cluster pairs, ordered by intercluster similarity, and merges at every step the clusters in the queue's maximum element. The algorithm halts when the expected number of clusters is reached.

Notice that links have to be computed for every pair of objects, resulting in superquadratic time and and space complexity in the worst case. Therefore, the authors propose to apply the algorithm to a sample taken from the data set by standard external memory techniques, with linear worst case I/O complexity. Labeling is performed by selecting a fraction of objects from each cluster and assigning an object to the cluster that maximizes the number of neighbours among the objects selected from the cluster, divided by its expectation. Hence, labeling requires a linear number of I/O operations and subquadratic CPU time. (The actual complexity will depend on the size of the selection as a function of the data set size.)

10.2.4 CACTUS

Whereas dissimilarity or distance functions are used very frequently for quantitative data, such functions are not naturally defined for categorical data. The CACTUS algorithm [38] discovers clusters in categorical data without assuming or defining any dissimilarity or distance function. Clusters are regions where the actual number of tuples is high when compared to its expectation, under the assumption that attributes are independent and values are equally likely (the so-called *attribute-independence assumption*).

Let the universe of attributes be $A_1, \ldots, A_n$ and $\Delta(A_1), \ldots, \Delta(A_n)$ their domains. Formally, an *interval region* R is the cartesian product of exactly one subset for each of the attribute domains: $R = R_1 \times \cdots \times R_n$, where $R_i \in \Delta(A_i)$, $i = 1, \ldots, n$. To compare the expected and actual number of tuples in a region, the authors define two notions of support. The *support* $\sigma_{\mathcal{D}}(a_i, a_j)$ *of an attribute value pair* $(a_i, a_j) \in \Delta(A_i) \times \Delta(A_j)$ is defined as the number of tuples having value a_i at A_i and a_j at A_j, i.e., $\sigma_{\mathcal{D}}(a_i, a_j) = |\{t \in \mathcal{D} : t.A_i = a_i \wedge t.A_j = a_j\}|$. The *support* $\sigma_{\mathcal{D}}(R)$ *of a region* R is defined as $|R \cap \mathcal{D}|$. Under the attribute-independence assumption, the expected supports are $\frac{|\mathcal{D}|}{|\Delta(A_i)||\Delta(A_j)|}$, and $|\mathcal{D}|\frac{|R_1|\cdots|R_n|}{|\Delta(A_1)|\cdots|\Delta(A_n)|}$, respectively.

Given $\alpha \in \mathbb{R}$, $\alpha > 1$, clusters are defined as maximal regions where all marginal bivariate distributions over $\{A_1, \ldots, A_n\}$ have higher support than expected by at least a factor α everywhere in the cluster, and the support of the region is higher than expected by at least a factor α.

The algorithm is divided into three phases: *summarization*, *clustering*, and *validation*. The summarization phase computes *inter-attribute* and *intra-attribute summaries*. The former are

the bivariate distributions on two attributes which equal the actual bivariate marginal distributions where the support is higher than expected by α, and equal zero elsewhere. The latter capture the similarity of attribute values from one attribute in the following way. If $a_1, a_2 \in \Delta(A_i)$, then their similarity w.r.t. A_j is the number of values $x \in \Delta(A_j)$, $i \neq j$, which have support higher than expected by a factor α with each of a_1, a_2.

In the clustering phase, first cluster projections on one attribute are generated for all attributes, then the projections are stepwise extended to the remaining attributes: First to clusters on two attributes, then the latter are extended to clusters on three attributes, and so on.

Cluster projections on an attribute A are computed from cluster projections on A of clusters on two attributes (2-clusters), which in turn can be computed from summaries. Unfortunately, the exact computation of all cluster projections of 2-clusters on one attribute is a difficult problem (NP-complete). Therefore, an approximate approach is devised, which allows to compute the clustering phase in a fraction of the time needed for summarization.

The validation phase filters out all candidate clusters not having the required support (third part of the definition above).

The authors show that the computation of inter-attribute summaries, which is the most expensive operation in CACTUS, can be done in one scan (at most a few scans) of the data base, for realistic values of the dimensionality, attribute cardinality and memory size. Interestingly, I/O costs are dominated by CPU costs and therefore the difference between one or more scans is negligible. Experimentally, CACTUS is shown to be more efficient than the STIRR algorithm[2] by a factor from 3 to 10.

Indeed, the most interesting feature of the CACTUS algorithm is its ability to cluster data in absence of a natural dissimilarity between values and tuples. Concerning efficiency, the plots drawn by the authors show that the time taken by CACTUS is linear w.r.t. to input size, clustering 5 million tuples on 10 attributes in about 3 minutes. When the number of attributes is increased, the running time rapidly grows (to the order of tens of minutes for 50 attributes).

11 Conclusions

The works reviewed in the present survey, by no means exhaustive, constitute a fairly large body of work on data clustering. Many important issues, such as scalability, applicability under severe limitations of computational resources, robustness in presence of large amounts noise and clusters with a variety of different structures, have been addressed by some proposal.

Up to now, little attention has been devoted to dynamic data, i.e., data which are frequently modified and, therefore, require frequent updates of the group composition. An example of this situation is found in *data warehousing*. A data warehouse usually materializes views from a set of base (operational) tables and performs on them some processing, such as clustering. Usually a data warehouse is rebuilt during the periods of inactivity of the operational systems [67], but problem size can easily lead to situations where the inactivity time is not sufficient for recomputing clusters from scratch. A more efficient solution could reuse old clustering information and consider separately new (or updated) data, obtaining new clusters from the old ones.

Another application scenario in the realm of data warehouses is *approximate on-line clustering*. In a growing number of application scenarios, the analysis of data warehouses is performed concurrently by many users over data sets whose size is in the order of hundreds Megabytes. As a consequence, severe limitations may be put on the computational resources available to every user. This situation is further exacerbated by data mining tasks, which are intrinsically iterative. Users of data mining services could benefit from approximate on-line computation in two ways. Firstly, exact answers are not always needed. For instance, the result of a clustering task might be to put forth evidence of a clustering structure, rather than to partition explicitly the data set

[2] Not surveyed here.

into clusters. Second, data mining queries returning unwanted results can be detected early and halted. This is especially useful when utilizing clustering algorithms, which require parameters with small stability ranges.

	Techn.	Data sp.str.	Lab.	Card.red.	Dyn.proc.
BIRCH	Stat.	quantitative	Y/N	N	partial
DBSCAN	Dens.est.	metric	Y	N	N
DENCLUE	Dens.est.	quantitative	Yg	Y	N
Incr.DBSCAN	Dens.est.	metric	Y	N	full
OPTICS	Dens.est.	metric	Y/N	N	N
BUBBLE	Stat.	metric	Y	N	partial
ROCK	Stat.	categorical	Y	Y	N
CACTUS	Stat.	categorical	Y	N	N

Table 1: Classification of clustering methods

Part III
Similarity queries (*Paolo Ciaccia and Marco Patella*)

12 Introduction

In this Section we review the problem of approximate similarity search, proposing a classification schema able to characterize existing approaches with respect to the kind of used approximation and on how the quality of the result is measured.

Similarity queries are a search paradigm which is profitably used when dealing with mining of data. In its essence, the problem is to find objects which are similar, up to a given degree, to a given query object. In this way, for example, we are able to cluster together similar objects by executing several similarity queries [31]. In order to assess the similarity between pair of objects, usually a notion of distance is used, being understood that low values of distance correspond to high degrees of similarity. More formally, we are faced with the following problem: Given a set of objects $\mathcal{O} \subset \mathcal{U}$ drawn from a generic metric space $\mathcal{M} = (\mathcal{U}, d)$, where $\mathcal{U}$ is a domain (the *feature space*) and $d : \mathcal{U} \times \mathcal{U} \to \Re_0^+$ is a non-negative and symmetric binary function that also satisfies the triangle inequality, retrieve the object(s) which are closest (i.e. that lead to the lowest values of d) to a user-specified query object $q \in \mathcal{U}$. Typical similarity queries include *range* queries (where all the objects in $\mathcal{O}$ whose distance to q does not exceed a user-specified threshold α are requested) and k-nearest neighbor (k-NN) queries (where the k objects in $\mathcal{O}$ which are closest to q are requested).

Several access structures have been proposed to speed up the resolution of (exact) similarity queries: They can be broadly classified (depending on their field of applicability) as multi-dimensional (or spatial) and metric access methods (the former only apply when the feature space is a vector space). Recent studies, however, pointed out the fact that using such access structures is sometimes not very efficient (e.g. when the feature space is a high-dimensional vector space [87]): In such cases, the most efficient way to exactly solve similarity queries is to sequentially scan the entire data-set, comparing each object against the query object q. Obviously, such solution is not viable for very large data-sets.

To speed-up the search it is common to offer to the user a quality/time trade-off: If the user is willing to save search time, he/she has to accept a degradation in the quality of the result, i.e. an error with respect to the exact case. Approximate similarity search, therefore, has the goal to reduce search times for similarity queries by introducing an error in the result. Since k-NN queries represent the most used type of similarity queries (because the user can control the query selectivity, i.e. the cardinality of the result set), in the following we will concentrate on this kind of queries.

12.1 Applicability Scenarios

The field of applicability of approximate similarity search is very wide. As an example, consider a multimedia database, where the user can query the system to search for images similar to a given one: Usually, such kind of search consists of several steps, each refining the query by changing, through relevance feedback techniques [11], both the query object and the distance function used to compare objects. In this light it is clear that the first steps of the search do not require that an exact result is found, but only that a result can be quickly obtained that is approximately similar to the exact result, in order to proceed with further steps as soon as possible.

As another example, consider modern Decision Support Systems (DSSs), where complex queries are posed to the underlying database system over Gigabytes (or even Terabytes) of data. Such queries are expected to be computationally very expensive even if the exploratory nature

of many DSS applications often do not require an exact answer [15]. In this scenario, users are frequently ready to accept an approximate result if the query solution time is reduced by some orders of magnitude.

Another common case when approximate queries arise is that of cluster-based similarity search: In this context, the data-set is first divided in clusters, i.e. sets of objects sharing common characteristics (e.g. which are similar to each other), by means of data mining techniques [54, 55]; then, at query time, the query object is compared against clusters' representatives and only those cluster that are closest to the query are accessed to retrieve objects similar to the query [85, 13, 59]. It is clear that such techniques cannot guarantee that the exact result is found.

13 A Classification Schema

Being understood that exact similarity search is sometimes difficult, several approaches have been recently proposed to solve the approximate problem. Virtually every approach proposes a new technique and new kind of metrics to evaluate the quality of the approximate result, yet all of them fail in relating the presented approach with other techniques presented in literature. The goal of this Section is to present a schema able to classify existing approaches by means of the following coordinates:

1. The type of data to which the approach can be applied.
2. The metrics introduced to quantify errors produced by approximation.
3. The guarantees in the quality of the result offered by the approach.
4. The degree of interaction with the user, i.e. the possibility the user has to *tune* the technique parameters to adapt to his/her actual needs.

The above coordinates have been chosen in order to evaluate the field of applicability of existing techniques for approximate similarity search. In fact, it is understood that if a technique A is applicable only to a subset of the data to which another technique B is applicable, then A is less general than B. On the other hand, it could be the case that A is more efficient or leads to lower errors: We are not interested in overall efficiency or accuracy of existing techniques here, but only on how they are achieved and how they can be measured.

13.1 Data Types

Since (exact) similarity queries require a notion of distance to be defined, an approximate technique is usually applied to a set of objects $\mathcal{O} \subset \mathcal{U}$ drawn from a generic metric space $\mathcal{M}$ (see Section 12). Examples of metric spaces include the D-dimensional vector space $\Re^D$ with the Euclidean distance L_2 or the set Σ^* of (finite length) strings obtained from an alphabet of symbols Σ with the edit distance d_{edit} (i.e. the minimum number of symbols that have to be inserted, deleted or substituted in order to transform a string into another).

Since, however, in most of the cases the data-set $\mathcal{O}$ is drawn from a vector space, several techniques exploit this fact by explicitly referring to coordinates of the space. Of course, this limits the applicability of such techniques, since they cannot be used on non-vectorial objects (e.g. strings with the edit distance). Moreover, some techniques are only applicable when the distance used to measure the (dis-)similarity between objects is an L_p metric[3] or, even more restrictive, the Euclidean distance L_2.

In this light, the following classification is given, in decreasing order of applicability:

[3] We recall that the definition of the L_p distance between two points x and y in a D-dimensional space is as follows: $L_p(x,y) = \left(\sum_{i=1}^{D} |x[i]-y[i]|^p\right)^{1/p}$, $1 \leq p < \infty$, $L_\infty(x,y) = \max_{i=1}^{D} |x[i]-y[i]|$.

MS (metric spaces) Methods in this class are applicable to the more general case of objects drawn from a generic metric space.

VS (vector spaces) In this class fall all those techniques that explicitly use objects' coordinates, thus are only applicable to vector spaces. However, these techniques do not make any assumption on the metric used to compare vectors (thus arbitrarily chosen distance functions can be used, e.g. quadratic form functions where the distance between vectors is defined by way of a positive definite symmetric matrix [72]).

$\mathbf{VS}_{L_p}$ (vector spaces, L_p distance) Techniques belonging to this class can only be applied when the considered objects are vectors in a D-dimensional space and the distance used to compare them is an L_p metric (thus no correlation between coordinates is allowed). Specific classes can be obtained by instantiating p (e.g. the class $\mathbf{VS}_{L_2}$ contains techniques that only applies to Euclidean spaces, i.e. when the distance used is the L_2 Euclidean metric). If p is not instantiated, then the technique is applicable to any vector space with an L_p metric, independently of the value of p.

As examples of the above classification method, we now describe three approximations techniques, assigning each of them to the proper class.

Example 1
In [40] the authors propose the use of Locality-Sensitive Hashing (LSH) to transform a D-dimensional vector x into a sequence of C bits (binary vector) $v(x)$. Since the L_1 distance between vectors can be approximated by the Hamming (edit) distance between the corresponding binary vectors, they propose a hashing technique to index only the binary vectors $v(x)$. Of course, both accuracy and efficiency of the technique highly depend on the number C of bits used for approximating vectors. Since the approximation for the Hamming distance only applies to the L_1 metric, this technique is of class $\mathbf{VS}_{L_1}$.

Example 2
In [86] approximate nearest neighbor search techniques based on the VA-file [87] are presented. Such structure, in its essence, is a sequential structure containing approximations of vectors using a fixed number b of bits. Exact k-NN search is performed by first executing a sequential scan of the structure using the query distance on vectors approximations, which yields a number $M > k$ of *candidate vectors*, and then applying a refinement step, where the distance is evaluated on real vectors and only the k "best" vectors are kept. Proposed techniques suggest either to reduce the number of considered approximations by reducing the query radius (VA-BND) or to avoid the refinement phase by returning only the "best" k candidate vectors, using the approximations (VA-LOW). Since no assumption is made on the distance to be used, both techniques fall in the **VS** class.

Example 3
In [42] the authors propose the P-Sphere tree, a 2-level index structure for approximate 1-NN search. In order to find the nearest neighbor for the query point, the leaf node which is closest, according to the used distance function, to the query point is accessed. The query is solved through a simple linear scan of objects contained in such node. In this case, no assumption is made on the query distance to be used (which, however, should be the same used to build the tree) and no coordinates are used, thus this technique is classified as **MS**.

13.2 Error Metrics

In this Section we review the most relevant error measures introduced to evaluate the accuracy of approximate techniques for similarity search. From the point of view of approximation, existing techniques can be classified as follows:

CS (changing space) To this class belong approximate methods that change the metric space, either by changing the distance used to compare objects or by modifying the feature space, then solve the exact problem on the so obtained approximate space. The goal is to obtain a "simpler" exact problem. Examples of such techniques are those that approximate vectors using a fixed number of bits, or dimensionality reduction techniques [16].

RC (reducing comparisons) Techniques in this class use the exact distance to compare objects but reduce the number of objects to be compared against the query in order to obtain a speedup with respect to the exact search. Examples of such techniques are those that prune from the search regions of the space according to the computation of bounds.

Of course, if we use a **CS** technique, the *ranking* of objects, i.e. the ordering of objects in the data-set with respect to the distance to the query object, is changing with respect with the exact case. On the other hand, using a **RC** technique, since the distance is not changed, the ranking of objects with respect to the exact case changes because some results are missed.

Therefore, **CS** methods commonly use comparisons in ranking of objects between approximate and exact results to measure the accuracy of their approximation (*ranking* measures), whereas **RC** techniques use measures of *precision/recall* (i.e. how many exact results are returned by the approximate query)[4] or measure the difference in distance between the exact and the approximate result (*distance* measures). Other measures include the percentage of correct results, i.e. the percentage of times in which the approximate result is equal to the exact result.

In the following we will describe some error measures presented in literature, classifying them by means of above categories.

Example 4
In [9] the authors propose the BBD-tree, which is a primary memory structure able to answer to approximate k-NN queries in a time that is logarithmic in the number of objects included in the data-set $\mathcal{O}$. To reduce the number of tree cells accessed, during the search the query radius is reduced by a factor of ϵ with respect to the radius used for exact search. Therefore, this method can be classified as **RC**. The measure proposed to rate the accuracy of the proposed algorithm is the *average relative error* which is defined as the ratio between the distance of the approximate NN and the exact nearest neighbor with respect to the query minus 1: This measure is also used in [20], where it is called *effective error* ϵ_{eff}. More formally, if we denote the query object as q, its exact NN as $nn(q)$, and the approximate NN as $\widetilde{nn}(q)$, it is

$$\epsilon_{eff} = \frac{d(q, \widetilde{nn}(q))}{d(q, nn(q))} - 1 \tag{18}$$

Clearly, ϵ_{eff} is a *distance* measure.

Example 5
In [89] three different algorithms are presented to solve approximate k-NN queries with M-tree [21]. All of them use heuristic conditions to prematurely stop the exact k-NN search on the tree. Thus, such methods fall in the **RC** class. To measure the accuracy of proposed heuristics on k-NN queries, the *relative distance error* $\overline{\epsilon}$ is proposed, which is defined as the average ϵ_{eff_i}, $i = 1, \ldots, k$:

$$\overline{\epsilon} = \frac{1}{k} \sum_{i=1}^{k} \left(\frac{d(q, \widetilde{nn}_i(q))}{d(q, nn_i(q))} - 1 \right) \tag{19}$$

where $nn_i(q)$ and $\widetilde{nn}_i(q)$ denote the i-th exact and approximate NN, respectively. It can be easily proven that $\overline{\epsilon} \leq \epsilon_{eff_k}$.

[4] It has to be noted that, since we only consider k-NN queries, the number of retrieved objects (the result of the approximate query) is equal to the number k of relevant objects (the result of the exact query), so that the two notions of precision and recall used in Information Retrieval [71] are coincident.

Example 6
The technique proposed in [36] combines clustering and dimensionality reduction to approximate k-NN search. During the search, only the clusters which are closest to the query are considered and, for all the points in such clusters, only a fraction of dimensions is used to assess the distance to the query. To improve accuracy, the user can increase both the number of visited clusters and the fraction of considered dimensions. This technique, therefore, combines characteristics of both classes **CS** (for using only some dimensions) and **RC** (for looking up only in some clusters).

Example 7
The VA-LOW technique discussed in Example 2 belongs to the **CS** class, since the approximate results are chosen by considering only the vector approximations. One of the metrics proposed to measure the result quality is the *normalized rank sum*, i.e. the inverse of the sum of ranks in the exact result of objects in the approximate result, computed as

$$nrs = \frac{k(k+1)}{2 \cdot \sum_{i=1}^{k} rank(\widetilde{nn}_i(q))} \tag{20}$$

where the function $rank(x)$ returns the ranking of object x in the exact result. Of course, this is a *ranking* measure.

13.3 Quality Guarantees

Having determined how approximate techniques measure the result quality, it is worth considering whether each method is able to bound its performance above a predetermined level. In other words, we are asking if an approximate technique can guarantee its error measure to be lower than a (user-specified) threshold. The classification we give is as follows:

NG (no guarantees) In this class fall those methods that only use heuristic conditions to approximate the search; thus such methods are not able to give any formal bound on the error introduced by the approximation.

DG (deterministic guarantees) Techniques in this class are able to deterministically bound from above the error introduced by approximation.

PG (probabilistic guarantees) Approximate methods following this approach are only able to give probabilistic guarantees on the quality of query result. This means that, depending on the query object, the accuracy of the result can fall below the specified threshold, but, when averaging results for several queries, the quality guarantees are met. To achieve this goal, information about distribution of data is needed. In this light, techniques belonging to this class can be further divided into two basic types according to how much it is known about objects' distribution [63].

- **PG_{par} (probabilistic guarantees, parametric)** Approaches in the parametric class assume that the used data-set follows a certain distribution; the only unknown information concerns a few parameters that need to be estimated (e.g. through sampling). Of course, when the considered objects do not follow the modeled distribution, quality guarantees cannot be met.
- **PG_{npar} (probabilistic guarantees, non-parametric)** In this case, little assumptions (or no assumption at all) are made on distribution of objects, so that such information has to be estimated through sampling and stored in a suitable way (e.g. through histograms).

Example 8
The third technique proposed in [89] (see also Example 5) stops the k-NN search whenever the improvement in the distance between the query and its k-th NN falls below a threshold κ. In this case, however, no guarantee can be given on the accuracy of the approximate result, defined, for example, through the relative distance error $\overline{\epsilon}$ defined by Equation 19. Thus, this method is in the **NG** class.

Example 9
The algorithm for approximate search proposed for BBD-trees in [9] (see also Example 4), and the first technique proposed in [89] both use a value ϵ to reduce the query radius during the search. In both cases, it is guaranteed that $\epsilon_{eff} \leq \epsilon$, thus both techniques belong to class **DG**.

Example 10
In [13] the DBIN structure is proposed as a 2-level index for solving the k-NN problem. The method assume that the data-set is composed of K clusters, and that distribution of objects within each cluster can be modeled by way of a Gaussian distribution, parameterized by a mean vector and a covariance matrix. At query time, the cluster that best fits the query object is found, and the NN is computed by considering objects in that cluster. Then, remaining clusters are accessed iff the probability that the NN has not been found yet is higher than a user-specified threshold. Such probability is computed by relying on the assumption of a Gaussian model, with parameters estimated at index construction time. Since the correct NN is found only with high probability and a Gaussian distribution is assumed (where mean and covariance have to be estimated), this method is in the $\mathbf{PG}_{par}$ class.

Example 11
The PAC technique proposed in [20] is a paradigm for approximate 1-NN search with metric access methods, where the effective error ϵ_{eff} measure (Equation 18) is allowed to exceed the user-specified accuracy threshold ϵ with a probability limited by the user-specified confidence δ. To guarantee this, the distance between the query objects and its NN is estimated from the distance distribution [22] of indexed objects. Since this latter information is not known at query time, such distribution is estimated (through sampling) and stored (in a histogram). By above considerations, this technique can be classified in the $\mathbf{PG}_{npar}$ class.

13.4 User Interaction

The last classification we propose relates to the possibility given to the user to specify, at query time, the parameters for the search (e.g. the maximum error allowed). Some techniques, in fact, are inherently static, in the sense that a structure is built by using a set of parameters to offer some guarantees: If the user wants to change, for example, the accuracy of the result, he/she has to modify the value of the parameters and to rebuild the structure from scratch. Other methods, on the other hand, exploit a single structure that is not bound to any parameter and can be used with different sets of parameters, according to user's needs.

SA (static approach) When using a technique in this class, the user cannot vary the set of parameters for query approximation, but is bound to those specified when building the approximate structure. Usually, to provide several quality of result profiles, different structures are built, using different sets of parameters, and the user is given the possibility to choose the structure that best fits his/her actual needs.

IA (interactive approach) Methods in this class are not bound to a specific set of parameters, but can be interactively used by varying such parameters at query time. Usually, interactive techniques are obtained as modifications of the exact similarity search method, that can be executed by requesting a maximum error of 0%.

Example 12
In the P-Sphere technique presented in [42] (see also Example 3), the size of leaf nodes, i.e. the number of objects in each data page, is estimated by taking into account a user-specified accuracy. Of course, if the accuracy parameter is changed, the P-Sphere tree has to be rebuilt from scratch. Therefore, this method is static and belongs to the **SA** class.

Example 13
In [47] the authors propose the generalized NN search as a new approach for high-dimensional NN search. The key idea here is to find a suitable projection to reduce the space dimensionality; then, the NN search is performed on the reduced space using the original distance function and projected points. Of course, the higher the value of the dimensionality D' of the reduced space, the better accuracy is obtained by this technique. Since the user can specify, at query time, the value of D', this method can be classified as **IA**.

14 Some Relevant Cases

In this Section we use the schema introduced in Section 13 to classify some of the approaches for approximate similarity search presented in recent years. For each method presented in the following Sections, the classification is expressed as a 4-tuple consisting of the following "coordinates": $(< \textit{data type} >, < \textit{error metric} >, < \textit{quality guarantee} >, < \textit{user interaction} >)$.

14.1 *Fastmap* [34]: $(\mathbf{MS}, \mathbf{CS}, \mathbf{NG}, \mathbf{SA})$

The *Fastmap* technique [34] has been proposed as a tool for mining and visualization of metric data-sets. In its essence, the *Fastmap* algorithm is able to map a set of objects drawn from a generic metric space to a D'-dimensional Euclidean space, where D' is a user-specified value, such that distances between objects are preserved as much as possible. Of course, this approach can also be used for approximate searching, since performing a similarity search in the target D'-dimensional space can be viewed as an approximate search in the original metric space. Since the method applies to general metric spaces, it belongs to the **MS** class; the transformation of the space is reflected in a transformation of the distance used to compare objects, thus this technique is in the **CS** class; in the paper, the authors give no guarantee on the error introduced for distance in the target space,[5] hence the quality guarantee class is **NG**; finally, as for user interaction, the mapping in the D'-dimensional space has to be made before any index structure is built on the transformed objects, thus *Fastmap* falls in the **SA** class.

14.2 DBIN [13]: $(\mathbf{VS}, \mathbf{RC}, \mathbf{PG}_{par}, \mathbf{IA})$

The DBIN (density based indexing) method was presented in [13] as an approach to solve approximate similarity queries in high-dimensional spaces. The base assumption is that the distribution of objects in the space can be modeled as a mixture of Gaussian distributions. Each point, therefore, can be associated to a cluster, parametrized with a mean vector and a covariance matrix, by using an expectation-maximization algorithm. When searching for the NN of a query point, the clusters obtained in the building phase are ranked according to the probability that the query point belongs to them; then, each cluster is accessed (and points in that cluster compared to the query) until the probability that the NN has not been found falls below an user-specified tolerance. Since no assumption is made on the distance used to compare vectors (even if analytical results are given only in the case of quadratic form distance functions), this method falls in class **VS**; the distance used to compare vectors is the exact distance, thus

[5] The error between exact distances and distances between transformed objects can be limited, for the relevant case of vector spaces, by exploiting the Johnson-Lindenstrauss lemma. However, for the general case of metric spaces, no general rule has been proposed so far.

the class of this technique is **RC**; as for quality guarantees, this technique assumes that indexed objects are distributed in clusters according to a Gaussian distribution, for which the mean and the covariance are estimated in the building phase, hence this method belongs to class $\mathbf{PG}_{par}$; since the user can specify the tolerance parameter, used when stopping the search, this method is of class **IA**. Finally, the metric used to measure quality of the result is the percentage of times the exact NN has been found (this is a *precision/recall* measure).

14.3 PAC [20]: $(\mathbf{MS}, \mathbf{RC}, \mathbf{PG}_{npar}, \mathbf{IA})$

PAC (probably approximately correct) nearest neighbor queries, introduced in [20], represent a probabilistic approach to approximate 1-NN search in metric spaces, where the error in the result can exceed a specified accuracy threshold ϵ with a probability that is limited by a confidence parameter δ. The PAC paradigm can be applied to *any* distance-based (either multi-dimensional or metric) index tree that is based on a recursive and conservative decomposition of the space (thus it is of **MS** class). The only information that is needed by the algorithm to prune index nodes from the search is the value of r_δ^q, the maximum value of distance from the query object q for which the probability that the exact NN of q has a distance lower than d is not greater than δ:

$$r_\delta^q = sup\{r | \Pr\{d(q, nn(q)) \leq r\} \leq \delta\} \tag{21}$$

In the paper, this value is estimated by using the distance distribution of indexed objects with respect to the query object (estimated through sampling and stored as an histogram); it is therefore clear that this approach is probabilistic and non parametric ($\mathbf{PG}_{npar}$ class). The distance used to query the distance-based index structure is the exact one, the approximation is introduced by reducing the number of object to be compared against the query object q by means of r_δ^q and of the ϵ parameter, and quality of the result is measured by the effective error ϵ_{eff} defined by Equation 18, thus the class of this approach is **RC**; finally, since the accuracy and the confidence parameters (ϵ and δ, respectively) can be specified at query time, this technique belongs to the **IA** class.

14.4 VA-BND [86]: $(\mathbf{VS}, \mathbf{RC}, \mathbf{PG}_{npar}, \mathbf{IA})$

In [86] two approximate query evaluation techniques are presented for the VA-File. The VA-File structure [87] approximates vectors using a fixed number of bits, and stores such approximations in a file. For exact k-NN search, the approximation file is sequentially scanned to exclude vectors that can not be in the result set through the computation of bounds on exact distances (such scan is very fast since the computation of bounds between approximations has to consider only a few bits); finally, exact vectors corresponding to approximations included in the result of the previous scan (the candidate vectors) are compared against the query point to compute the final result. Since the approximations of the VA-File are only applicable to vector spaces and any distance can be used to compare vectors (even if computation of bounds can be a difficult task if complex metrics are used), all approximate techniques developed for this structure fall in the **VS** class. In the paper, two different metrics are proposed to measure the accuracy of approximate techniques:

1. The *ratio of false dismissals* rfd, defined as

$$rfd = \frac{1}{k} \cdot \sum_{i=1}^{k} \begin{cases} 1 & \text{if } rank(\widetilde{nn_i}(q)) > k \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

   which counts the number of exact results not included in the approximate result (divided by k). This can be classified as a *precision/recall* measure and it is the same measure used in [13] (see Section 14.2), where it is called *discounted accuracy*.

2. The *normalized rank sum* nrs, defined by Equation 20, is computed as the inverse of the sum of rankings in the exact result of objects in the approximate result, normalized in order to obtain a value in the interval $[0, 1]$. This is a *ranking* measure and it is used to better distinguish the quality of result with respect to the above measure.

The first approach to reduce the complexity of similarity searching in the VA-File through approximation proposes to adapt the computation of distance between approximate vectors. The user is given the possibility to specify a value α to adapt computed bounds: Higher values of α correspond to higher errors in the result but the candidate set will consist in a lower number of vectors. Since the approximation is introduced in the computation of bounds and not on the exact distance, this technique can be classified as **RC**. The number of vectors missed can be computed as a function of the distance distribution between objects, thus this technique can give probabilistic guarantees on the ratio of false dismissals rfd as a function of the parameter α; therefore, the class for this method is $\mathbf{PG}_{npar}$. Finally, since the parameter α can be specified at query time, the VA-BND technique is in class **IA**.

14.5 VA-LOW [86]: (VS, CS, DG, SA)

The second approximate technique for the VA-file (also presented in [86]) proposes to completely omit the second refinement phase and to return, as the approximate result, the k vectors corresponding to the best approximations. Of course, in this case, the approximation in the result comes from using, in computing the distance, the approximate vectors instead of the exact vectors, thus this method can be classified as **CS**. The error in computation of the distance on approximate vectors can be controlled by means of the quantity of bits used for the approximations: The more bits are used, the better the approximation but the slower the first sequential phase. Since a bound on the error between the distance on approximate vectors and the exact distance can be easily computed, this technique falls in the **DG** class. As for the interaction with the user, it is clear that the only parameter used, i.e. the number of bits used for vectors' approximation, has to be specified before the actual VA-File is built, so that the class for this technique is **SA**.

15 Comments and Extensions

We believe that the proposed classification schema can be very fruitful for the analysis of approximate techniques for similarity search. By using such schema interesting relations and similarities between techniques can be found that may not be evident at a first sight. As an example, consider the PAC approach (reviewed in Section 14.3) and the VA-BND technique (Section 14.4): Both are classified as belonging to the **RC**, $\mathbf{PG}_{npar}$, and **IA** classes, the only difference being in the fact that the VA-File only applies to vector spaces. Indeed, at a closer look, these two method share several analogies:

- In both cases the approach requests for an additional confidence parameter (δ and α, respectively) representing the quality of the result the user is willing to obtain. The lower the value of the parameter, the lower the error and the higher the search costs.
- Both methods use information about the distance distribution in order to estimate the distance between the query object and its nearest neighbor.
- In both cases the distance distribution and the confidence parameter are jointly used to derive bounds to stop the search.
- Different error metrics are proposed for the two methods, but estimates on other metrics can be easily obtained (e.g. on rfd and nrs for PAC, and on ϵ_{eff} for VA-BND).

It is clear that, by using the proposed classification schema, we are able to immediately understand the field of applicability of a particular approximate technique. In this way, we can conceive whether, for example, a method is more general, i.e. it applies to a superset of scenarios, with respect to another, or how its quality measures relate to those proposed for other techniques. In search for the "best" approximate technique for a specific scenario at hand, in fact, different aspects are to be considered, in particular the generality/efficiency trade-off: A more general method is expected to have a lower efficiency (i.e. to lead to higher search costs) than a method that applies to a lower number of cases, since it is expected that the latter is able to exploit some domain-specific information that the former cannot to take into account; this applies immediately when considering methods that apply to metric spaces or only to vector spaces. The same considerations can be made when dealing with quality guarantees: Parametric approaches usually attain better performance with respect to non-parametric ones, yet they are only applicable to particular distributions of objects. On the other hand, deterministic techniques, in some senses, are the most general ones, since guarantees are met in all possible cases and not only in a probabilistic way. Finally, it is clear that interactive approaches are more general than static ones, since the user is given the possibility to choose at query time the desired quality of the result, which is inversely related to search costs needed to obtain the approximate result.

Limits of approximate techniques can also be discovered by means of the proposed classification schema. As an example, consider the PAC technique reviewed in Section 14.3: Since only the distance between the query object and its nearest neighbor is estimated, it is clear that such approach cannot deal with approximate k-NN queries when $k > 1$. In order to extend the PAC approach to generic k-NN queries, we need to estimate the distance between the query object and its k-th nearest neighbor, e.g. by using formulas from [22].

Finally, we would like to point out some issues that are currently open for research. In our view, the most pressing one is that of extending approximate similarity techniques to deal with the case of *complex* queries, where different similarity predicates are jointly evaluated to derive the similarity between the object and the query [33, 23]. In the simplest case, all the similarity predicates refer to a single *feature*, thus, for example, we could ask for the objects which are most similar to a query object q_1 *and* to a query object q_2: In this case, the overall similarity score for an object $O \in \mathcal{U}$ is obtained by computing the similarity between O and q_1, and that between O and q_2, and combining them through a scoring function [33]. In order to solve complex queries, some existing approaches suggest to independently solve the two sub-queries, i.e. to consider objects that are sufficiently close to q_1 *or* to q_2, and then to combine the so-obtained results [33, 45, 88], whereas other techniques are able to consider the complex query *as a whole* and to process it with classical distance-based access methods [23]. The former approach is usually much less efficient than the latter, since a lot of work is wasted during the first phase to consider objects that are not contained in the final result (for a comparison, see [23]). Besides specific problems, however, it is clear that this kind of similarity search is affected by the same problems that limit (simple) similarity search, thus one could conceive approximate techniques applying to complex similarity queries. By our knowledge, no previous work dealt with such issue, probably because of the preliminary state for approximate techniques for simple similarity search. We plan to investigate this issue in the future.

Part IV
Visualization (*Tiziana Catarci*)

16 Introduction

Various data visualization modalities, together with other techniques specifically oriented to the discovery of correlation or rules, are often used in data mining systems. These visualization modalities are often used to support other techniques, essentially to visualize database results. Furthermore, the availability of different visualizations allows the user to discover new properties, their correlation and find any unexpected values. The user may apply other data mining techniques for further analysis of such "unusual", or rather interesting data.

Thanks to the use of animation, some recent systems (such as MineSet [43] of Silicon Graphics) allow the user to "fly over" data to see multiple levels of detail. The advantage of using data visualization technique is that the user need not know the type of phenomena to observe in order to notice something unusual or interesting. For example, through statistic tests the analyst must formulate precise queries, as "do the data evaluate this condition?" or "are these values consistent with the distribution of Poisson?" No statistic test can answer questions such as "is there something unusual in this data set?" Also, the use of interactive visualization techniques allow the user to quickly and easily change the type of visualized information, just as the particular method used (that is moving from histograms to scatterplots or parallel coordinates).

However, one limit of existing systems is the absence of a global interaction environment. In other words, what is missing is a complete data-mining oriented visual metaphor, which supports both the phase of definition of the specific operations that are to be applied to the data and the analysis of the end results. The aspects of more interest to the existing systems are on the contrary relative to the application, more or less sophisticated, of specific visualization techniques of data traditionally proposed in literature [75, 82, 69], which essentially focus on the visualization of histograms and other diagrams typically used to represent quantitative data, tridimentional and otherwise, cluster of points, distribution of points, cluster relations, maps, again in two or three dimensions.

Visualization was initially used for data mining by the military, to analyze submarine acoustic data, and also in the petrochemical sector, to analyze the course of seismic prospects. However, the visual analysis of data may be applied in different applicative sectors. For example, in medical research it may be applied to identify factors that characterize patients with a particular disease, or in chemical analysis for the discovery of new compounds, or in the financial sector for the discovery of how data on warehouse goods are related.

In the following relevant characteristics of a set of notable systems are discussed. Such systems are divided into two categories: systems in which visualization is a basic technique, associated to a limited number of functionalities and other mining techniques, and more complex systems which provide the user with a complete set of algorithms and tools.

17 Data Visualization based Systems

Data characteristics that are very difficult to understand analyzing a certain number of lines and columns in the database may become obvious when visualized through drawings. The following systems belong to this category: AVS/Express [80], CrossGraphs [51], Data Desk [25], Datascope [61], DEVise [62], ADVIZOR/200 [52], Descartes [41], JWAVE [68], Spotfire [6], Open Visualization Data Explorer [50] and VisualMine [76]. All these systems use statistical data analysis techniques, 2D and 3D visualization and various diagrams which better highlight the data characteristics. Few systems support advanced functionalities such as multidimensional data visualization, typical of On-Line Analytical Processing (OLAP) systems - animation or the

use of special visual tools to query the database (filters of dynamic queries [1, 81]). The use of Web technology is more widespread whereas integration of different systems is limited. Unix and PC platforms are almost equally distributed among the various systems (see Table 2).

System property	AVS Expr	Cross-graphs	Data Desk	Data Scope	DEVise	ADVISOR 2000	Descartes	JWave	Spotfire	Vis DX	Visual Mine
Statistical Analysis	x	x	x	x	x	x	x	x	x	x	x
2D and 3D images	x	x	x	x	x	x	x	x	x	x	x
Graph Based	x	x	x	x	x	x	x	x	x	x	x
Animation	x	-	-	-	-	x	-	x	x	x	?
Dynamic query filters	-	-	-	x	-	-	x	-	x	-	-
Developed for internet/intranet	x	-	-	-	-	-	x	x	x	-	-
Presence of API	x	-	-	-	-	-	-	-	-	-	-
Access to data source	x	-	x	x	x	x	-	x	x	x	x
ODBC	x	-	-	x	-	x	-	-	x	-	x
Integration with other systems	-	-	-	-	-	-	-	-	x	x	x
Platforms											
- Unix	x	x	-	-	x	-	x	x	-	x	x
- PC	x	x	x	x	-	x	x	x	x	-	x
- Mac	-	-	x	-	-	-	x	x	-	-	-

Table 2: *Key Features of DM Systems based on Visualization*

AVS/Express

Description: AVS/Express allows users from various scientific, technical and trade sectors to use functionalities for visualization, data analysis and image processing. AVS includes traditional visualization tools as 2D designs, various types of diagrams, image processing and advanced tools that allow interactive rendering and 3D image visualization, through systems such as SGI Realty or CAVE. Simply click a certain point on the screen to view data during exploration.

The system allows import and export of data in TIFF, JPEG, GIF, SGI, SUN, BMP, PBM and NetCDF formats; it also allows output in AVI, MPEG and VRML, giving the possibility to publish the results of the analysis directly on the World Wide Web. Currently the user may choose between two types of software: AVS/Express Multipip e Edition and AVS/Express Visualization Edition. The former concentrates on the 3D aspect and is intended to help understand data by using new interaction methods such as virtual reality. The latter, on the other hand, includes a subset of Express visualization modules (data visualization and processing), apart from the possibility of merging existing modules in C/C++.
Platforms: Windows 98/NT, Digital, HP, IBM, Silicon Graphics, Sun.
Producer: Advanced Visual Systems

ADVIZOR/2000

Description: ADVIZOR/2000 was intended to help users understand their data and render the study much faster by using visualization techniques. It was introduced as a program strictly linked to Microsoft Excel. The key feature of ADVIZOR is its ability to simplify visualization and understanding of multidimensional relations. To obtain this, metaphors and visual perspectives are used to maximize the user's ability in studying his/her data.
Platforms: Windows 95/98/NT
Producer: Visual Insights

CrossGraphs

Description: CrossGraphs is a clinical data visualization program that uses statistical graphs and cross-charts to help discover relations between data and compare the paths of many data subsets. CrossGraph provides more than a dozen graphical components for better visualizing

data relations. It is also possible to develop new ones. In fact, by using *Customization Option*, CrossGraph may be used as a powerful graphical library exploited by other programs through OLE. In addition, it is possible to export the produced drawings in Word documents or PowerPoint presentations, or import data from Excel electronic sheets. CrossGraph may be used interactively as data visualization and exploration tool or in batch modality to produce high quality reports for off-line analysis.
Platforms: Windows 95/98/NT, HP, Sun.
Producer: PPD Informatics

Data Desk

Description: Data Desk is a fast and easy to use software that helps people to understand their data. It is equipped with interactive tools used for visualizing and analyzing data, based on the concepts and philosophy of "Exploratory Data Analysis". Data Desk employs many of the traditional statistical and graphical techniques to discover the structure of a data set. Special colors and symbols in the diagrams can easily highlight cluster and abnormal points; sliders may be used for dynamic data analysis; Graphics (2D and 3D) and the derived tables are linked such that points selected in a graphic design can be highlighted in all the others.
Platforms: Mac and Windows 98/NT.
Producer: Data Description

DataScope

Description: DataScope provides solutions in various sectors including banking, finance, insurance and telecommunication. Among the peculiar characteristics of DataScope is the ability to query a database without using an actual query language. In fact, no formula or command is required for a query, using the mouse is enough. Basically the system uses statistical diagrams for non-numerical data, whereas just one visualization method based on empirical distribution diagrams is used for numerical data. DataScope classifies database fields in three categories:

- Identifier Fields: This category helps to identify records. An identifier field is normally a name or another unique identifier (ID).
- Numerical Fields: These fields contain quantity such as population age, price, etc. DataScope is also able to process numerical fields where data is not available for all records.
- Discrete Fields: In this category, the fields have only little different values that are used for classification of records rather than quantitative analysis. Some examples are fields that contain type identifiers, yes/no answers, etc.

DataScope can use up to 16 different windows to visualize the database content. Each window represents one or two database fields; in this way various alternatives can be analyzed, using 16 different fields or pairs of fields. All windows are connected to each other so as to display all information concerning currently selected record. The visualization mode depends on the field category. Identifier fields are viewed in a list form. Numerical fields are presented according to their empirical distribution function. The value given by this function for a selected data is a number between 0 and 100. The percentage indicates the position of the selected data in respect of others and as such it is very easy to examine a value in relation to another. Lastly, discrete fields are visualized as traditional graphics i.e. pie chart or bar chart. The chart shows the category (or the class) to which the selected record belongs. Relational diagrams may be used to visualize information on the relationships between two numerical fields. These diagrams are useful when trend is evaluated and when searching for unusual characteristics. Different relational diagrams can be visualized.

DataScope allows access to every type of database thanks to the use of the ODBC connecting library.

There are two versions of the program:

- DataScope Professional, which uses two modular components:
  - Explorer - Visualization and Decision Support.
  - Predictor - Prediction and Data Mining.
- DataScope Power, which, apart from the two modular components, uses:
  - Scheduler - Automate the Data Mining Process
  - Datamap - Import from Relational Databases

Platforms: Windows 95/98/ME/2000/NT.
Producer: Cygron Research & Development, Ltd.

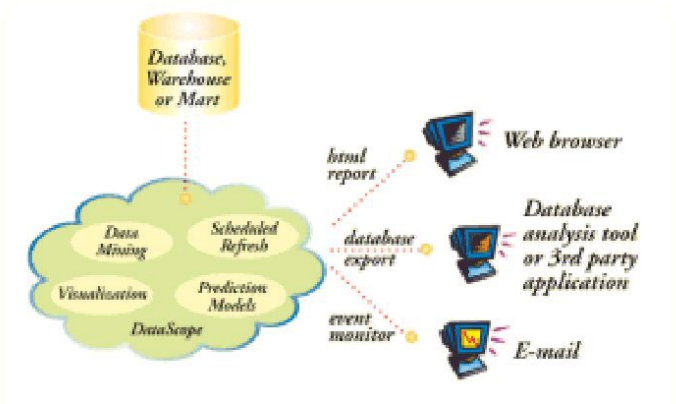

Figure 8: *Datascope Architecture*

DEVise

Description: DEVise (Data Exploration and Visualization) is a data exploration system that allows the user to develop and easily visualize huge databases (possibly containing, or connected indirectly to, multi-media objects) from different data sources. The importance of this system has been in the development of powerful, but intuitive, data visualization and primitive query constructs. The main characteristics of DEVise are therefore the visual query interface, whose constructs may be memorized and reproduced on other data sets; the effective management of huge databases and the ability to query records using visual constructs. The system allows the import of ASCII data only. Furthermore, most of the DEVise functions are obtainable from Internet using Java DEVise Java Screen.
Platforms: Unix system
Inventors: Miron Livny, Raghu Ramakrishnam and Kent Wenger.

Descartes (IRIS)

Description: Descartes is a research prototype that supports space data visual analysis for Geographic Information Space applications (GIS). Descartes automatically produces high quality thematic maps for statistical data selected by the user. The system applies a general knowledge on the best combination and representation of the data on the maps, using heuristic rules together with specific metadata of the application and an object-oriented knowledge representation

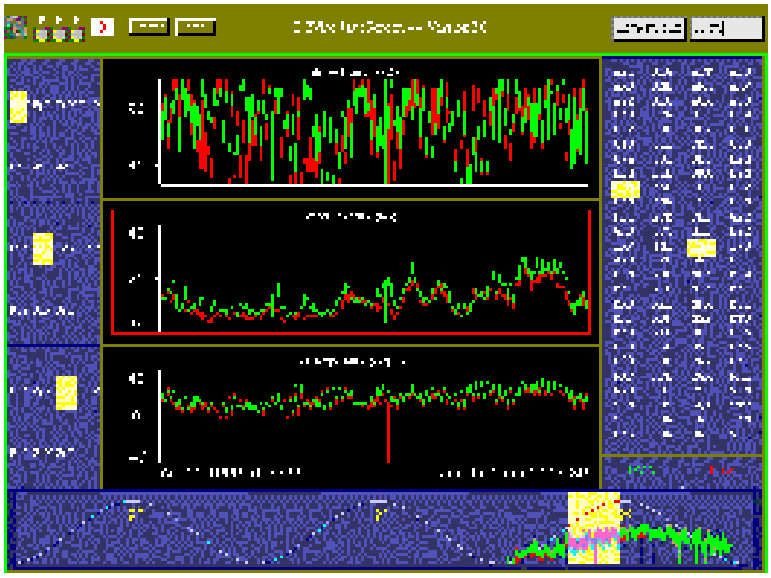

Figure 9: *DEVise JavaScreen*

language. Contrary to other GIS systems, Descartes completely automates the production of these maps, allowing the user to concentrate on the data analysis rather than the cartographic design. Also, because information and access are key elements of geographical information, Descartes has a module called writeHTML, which generates HTML pages directly from database records. This way, the user can simply click and make available his/her information on an intranet or Internet network. Descartes is in two forms: as a single program using Windows 95/98/NT or as an application based on Java applets.
Inventors: Nathalia V. and Gennady L. Andrienko

JWAVE

Description: JWAVE is a component of PVWAVE Web Development Environment, and visual data analysis software that integrates advanced (2D and 3D) graphics, statistical and numerical functions with very flexible access to data, assisted by a RAD (Rapid Application Development) environment. JWAVE has a multi-layer architecture: the JWAVE client, the JWAVE server and PVWAVE. The JWAVE client comprises a system of classes and Java applets for the client interface. On the other hand, the JWAVE server acts as a Unix or NT service that forwards the client requests to PVWAVE.
Platforms: Windows 95/98/NT, Digital, HP, Sun, Open VMS, NeXTStep, Convex, Data General.
Producer: Visual Numerics, Inc.

Spotfire

Description: Spotfire is a tool used for the analysis and visual exploration of huge databases, strictly connected to an advanced query language. Thanks to the dynamic techniques of interactive and animated visualization, Spotfire allows the user, who has no knowledge of statistics, to access thousands of database objects and notice certain trends, paths or anomalies among the data. Just click an object to find the desired details, including multi-media data. Colors, icons, advanced animation, 2D and 3D computer graphics can be used for this purpose. Tables generated with normal electronic pages, containing numeric and text data, can be visualized and explored in Spotfire with minimum effort. The application of Spotfire to real life situations is quite significant. For example, it is possible to interactively visualize telephone calls or

study environmental data of a certain region or analyze financial and commercial data, and so forth. Spotfire imports data from data warehouses, text files and electronic sheets (supported by ODBC), and creates a visual data mining environment for the user. At any time it is possible to obtain information concerning an object (also multi-media) visualized in the form of a dot on th screen, by clicking on it. Visualization is flexible and it is possible to capitalize on the colors and icons in the discovery of hidden paths, clusters or isolated data. Spotfire appears with a main window divided into various areas. The selected records are visualized on the left side of the screen. A query tool (sliderbar, alphaslider, ...) for each column of imported table is visualized on the upper right side of the screen. These tools are used to choose a subset of values in a column, so as to select a certain number of rows. Interacting with the tools, the visualization of the data updates automatically to reflect the changes. Lastly, the lower right side of the screen could be used to visualize details, upon request. By just selecting a certain point in the visualization window, the desired detail information on a database object (typically a record), would appear in a complete form.
Platforms: server: Windows NT; client: Windows 95/98/2000/NT
Producer: IVEE Development AB.

OpenVisualizationDataExplorer (DX)

Description: Open Visualization Data Explorer is a general-purpose software for visualization and analysis of mainly scientific data, especially 3D data originating from simulation. The architecture of the system is based on a client-server model; the client hosts the user's graphics interface, which uses Xwindows and Motif, whereas the server effects all necessary computation. DX provides an extensive library of modules, so the user composes her/his programs to render the image, choosing and connecting these modules by means of a visual programming editor. DX is similar to a GIS system, but it further offers the possibility to investigate huge quantity of data. DX supports large number of data sets and works simultaneously with many types of data.
Platforms: Sun, IBM, DEC, Data General.
Producer: IBM.

VisualMine

Description: VisualMine (Visual Data Mining Environment) is based on an advanced 3D-visualization technology. After loaded data from the database, the user employs the Mapping Window module to select which type of visual metaphor s/he intends to use. Data are then converted into graphics. Clusters, anomalies and relationships inside huge data sets are emphasized through the 3D representation. Moreover, it is also possible to represent geographic data, as maps, in various levels of detail. It is worth noting that VisualMine supports access to Oracle, Informix and Sybase databases through the SQL language, while through ODBC it allows one to access dBase, FoxPro, Access and Excel files. Furthermore, by means of the new API libraries, users of VisualMine can work jointly with other data-mining products. In particular, the last versions of the product support a full integration with Clementine (see next section). Lastly, the latest of the product is the introduction of VisualMine Wizard, which allows users to perform data mining activities in a semi-automatic manner.
Platforms: Workstation Unix and Windows 95 and beyond or NT
Producer: Artificial Intelligence Software.

18 Multistrategy Systems

This section refers to tools that adopt diverse data mining techniques, from neural network to data visualization, making provision for a more effective search of paths and rules within large

databases. These systems basically operate in the Unix environment, since they exploit the power of parallel architectures to analize the available data with several techniques (see Table 3).

System properties	Clementine	Decision Series	Intelligent Miner	Mine Set
Neural Networks	x	x	x	x
Decision Trees	-	-	x	x
Induction Rules	x	x	x	x
Discovery of Rules	-	x	x	x
Data Visualization	x	x	x	x
Use of Induction Properties of Algorithms	-	-	-	-
Export of Rules in Prolog, SQL or others	x	-	-	-
Developed for internet/intranet	-	-	-	-
API	x	-	x	-
Access to other sources				
- Data	x	x	x	x
- ODBC	-	-	-	-
Integration with other systems	x	x	x	x
Programs				
- Unix	x	x	x	x
- PC (Windows)	-	-	-	-
- Mac	-	-	-	-

Table 3: *Key Features of Multistrategy Systems*

As shown in Table 3 the following systems fall within this category: Clementine [60], Decision Series [77], Intelligent Miner [49], KnowledgeSTUDIO [53], and MineSet [43].

Clementine

Description: Clementine is based on a visual programming interface and it supports every aspect of data mining, from data access (Oracle, Ingres, electronic sheets, etc...), to manipulation utilizing diverse data mining techniques. Clementine is completely based on the CRISP-DM data mining model. CRISP-DM is a project financed by the European Commission, to define a standard approach to data mining projects. The objective of the project is to define and validate a general method independent of the type of business. This approach renders the implementation of large data mining applications faster, more efficient and less costly. The system analysis driver is based on machine learning algorithms as neural networks and induction of rules, the creation of graphs, cluster methods and regression. Moreover, with Clementine Solution Publisher it is possible to distribute data mining results, thanks also to the possibility of automatically generating Ansi C standard code. Clementine is an open system and can be entirely configured by the user; in fact it has a development library.
Platforms: Sun, HP, Silicon Graphics, DEC, VAX/VMS.

Producer: Integral Solutions Ltd.

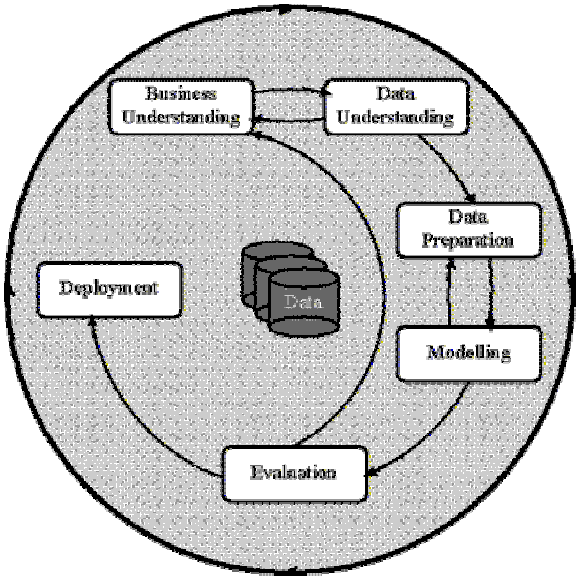

Figure 10: *CRISP Approach*

Decision Series

Description: Decision Series is a set of integrated tools for the discovery of rules. It uses neural nets (DecisionNet) in learning to identify paths from training examples, using backpropagation as analysis algorithm. To cluster information which are similar according to certain metrics. Decision Series uses DecisonCluster and DecisionKmeans modules. The clusters could be established a priori or determined by the system itself. Finally, within the system there is a module for the search of association rules and time-series (DecisionAR). Using the DecisionAccess interface, Decision Series may interface its environment with several relational DBMS (e.g., Informix, Oracle, Sybase, Teradata) and report tools (e.g., BusinessObjects).
Platforms: Unix systems
Producer: NeoVista Solutions, Inc.

Intelligent Miner

Description: the IBM Intelligent Miner groups diverse data mining techniques in one tool, which works in client-server modality. It is capable of creating classification models and prediction by using decision trees and neural networks. Furthermore, it uses rules discovery techniques, such as associations and series, and data visualization functionalities. It can also access various data sources, such as Oracle, Sybase and DB2 for OS/390. The results of the analysis are recorded in DB2 tables. The system is also provided with programming libraries.
Platforms: AIX, OS/390, OS/400, Solaris and Windows 2000/NT.
Producer: IBM

KnowledgeSTUDIO

Description: Thanks to an intuitive interface, KnowledgeSTUDIO simplifies and streamlines several data-mining problems. It can import data from many statistical tools, such as dBase, Excel, SPSS, SAS. Alternatively, the data could be extracted in a native format and analyzed by a dedicated server (KnowledgeSERVER), optimized for mining. KnowledgeSTUDIO offers diverse mining techniques: five different algorithms for decision trees, five for neural networks

and clustering. There is also the possibility to include customized algorithms through the KnowledgeSTUDIO Algorithm Development Kit (ADK). The system provides a simple way of dividing the data into a number of equivalent partitions known as "statified samples", which contain the types of records specified by the user, to train the neural networks or another predictive model. Using VisualBasic, PowerBuilder, Delphi, C++ or Java, other data mining methods can be included in the analysis. KnowledgeSTUDIO is capable of constructing decision trees automatically or interactively. Rules obtained in this manner can be expressed in pseudo-code, SQL or Java. Conversely, cluster-based analyses can be used to identify groups of interest. Lastly, neural networks can be simply created through a wizard; KnowledgeSTUDIO uses ad-hoc algorithms, instead of Backpropagation, by which it reduces the analysis time without compromising the predictive power. ANGOSS Software offers other mining products, particularly for e-business (KnowledgeWebMiner) and firms (KnowledgeSERVER for Siebel).
Platforms: Windows 95, 98, NT and 2000, Solaris (Server)
Producer: ANGOSS Software Corporation

MineSet

Description: MineSet is an environment inclusive of many interactive systems, which provide diverse functionalities, from data access and conversion to diverse data mining techniques. This allows the user better to better explore and comprehend the data. Currently, MineSet provides the support for the generation and analysis of association rules, the analysis of classification models and the determination of the most important columns in a database. Each data mining system is aided by a data mining visual system that allows data analysis according to different visual metaphors. The data mining systems that constitute MineSet are:

- *Association Rule Generator*, that allows the identification of rules, which describe relationships between entities in databases. The resultant rules serve the purpose of defining a predictive model based on analyzed data that can be used to comprehend tendencies and predictions in the future. Ride Visualizer visualizes obtained results and makes provision for more rules to be analyzed.

- *Decision Tree Inducer*, that generates decision trees directly from the data specified by the user. Tree Visualizer enables the navigation through the trees, representing them in 3-dimensional landscape.

- *Evidence Inducer*, that is useful when producing classifications containing incomplete data. The results are visualized through Evidence Visualizer.

- *Column Importance*, this is a utility for discovering the most important columns in the database, in terms of data classification. This is very useful when selecting what dimensions of data to use for data mining visual analysis.

Other data mining visual systems are Map Visualizer, which allows the analysis of tendencies based on space data relationships, such as time and age and Scatter Visualizer that observes the tendencies projected in the 3 dimensional space. Figure 11 shows the representation, by means of the Map Visualizer, of information relative to the sale of products for the entire the United States. The geographic information on the member states and the data on the latitudinal and longitudinal coordinates cannot be retrieved from the database. The user, during the construction of the map, is requested to provide such information.

The Tool Manager is used to access the MineSet data; it provides access to the most diffused relational DBMSs, and can also access non-relational data sources. It then associates different visual paradigms to the data to facilitate their exploration. One of the limits of MineSet is its indissoluble link with the Silicon Graphics environment.

Platforms: IRIX 6.2 and beyond
Producer: Silicon Graphics, Inc.

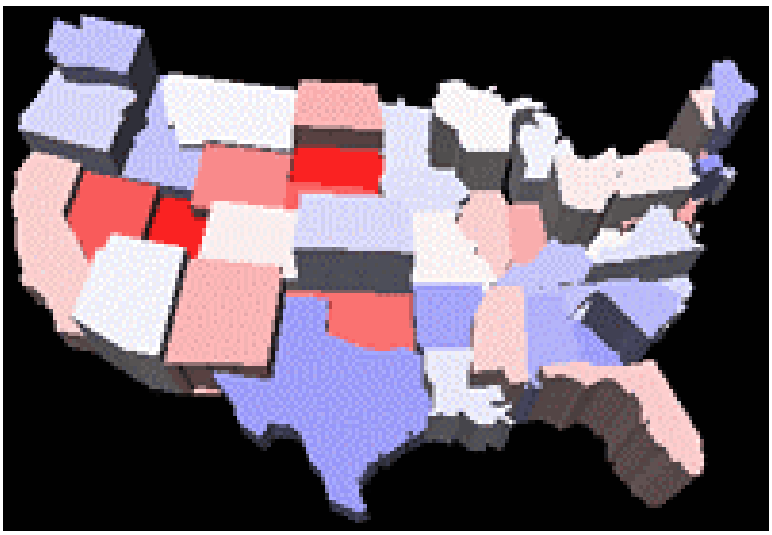

Figure 11: *MineSet Map Visualizer*

18.1 Open Problems

One of the major problems related to the use of data mining tools is the user's difficulty in understanding what to do. Generally, when the more or less expert user begins to analyze data, he/she has no precise idea of what to look for and how to proceed. In such cases the system should provide an appropriate visualization of the more relevant data, so that the user can be better oriented in taking the initial steps. On the contrary, most existing systems begin with the assumption that the user knows exactly what to look for and how to search (i.e. which techniques to use).

Another problem is the dynamic interaction with the system that must respond in a satisfactorily fast way to the requests made by the user during the data analysis. This aspect is connected to the use of appropriate data structures that enable prompt responses in respect of data updating.

There is yet another problem concerning providing data mining tools with a multi-paradigmatic environment, equipped with a user model, which allows the system to have a certain degree of self-adaptability to the different characteristics of the various users, from the amateur to the expert, from the manager to the analyst.

Lastly, because one data-mining technique is not enough to extract all the information from a data set, we look forward to the complete integration in one environment, even if composed of separate tools, of the current data mining techniques and other systems as DBMSs and electronic sheets. In this manner the output of a tool could serve as input of another and it would be possible to automatically select the most appropriate technique based on the circumstances. For example, the output of a rule discovery technique could be better highlighted by a specific data visualization technique.

References

[1] Ahlberg A. and Shneiderman B. Visual information seeking: Tight coupling of dynamic query filters with starfield displays. In *Proceedings of ACM Conference on Human Factors in Computing Systems (CHI)*, 1994.

[2] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993*, pages 207–216. ACM Press, 1993.

[3] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993*, pages 207–216. ACM Press, 1993.

[4] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 487–499. Morgan Kaufmann, 1994.

[5] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In Philip S. Yu and Arbee L. P. Chen, editors, *Proceedings of the Eleventh International Conference on Data Engineering, March 6-10, 1995, Taipei, Taiwan*, pages 3–14. IEEE Computer Society, 1995.

[6] C. Ahlberg. Spotfire. `http://www.ivee.com`.

[7] F. Angiulli, Rachel Ben-Eliyahu-Zohary, G. B. Ianni, and L. Palopoli. Computational properties of metaquerying problems. In *PODS-2000: Proceddings of the 19th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 237–244, San Diego, CA, 2000.

[8] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. OPTICS: Ordering points to identify the clustering structure. In Alex Delis, Christos Faloutsos, and Shahram Ghandeharizadeh, editors, *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadephia, Pennsylvania, USA*, pages 49–60. ACM Press, 1999.

[9] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching. *Journal of the ACM*, 45(6):891–923, November 1998.

[10] D. A. Mix Barrington, N. Immerman, and H. Straubing. On uniformity within NC_1. *Journal of Computer and System Sciences*, 41(3):274–306, 1990.

[11] Ilaria Bartolini, Paolo Ciaccia, and Florian Waas. Using the wavelet transform to learn from user feedback. In *Proceedings of the First DELOS Network of Excellence Workshop on Information Seeking, Searching and Querying in Digital Libraries*, Zurich, Switzerland, December 2000. Online publication `http://www.ercim.org/publication/ws-proceedings/DelNoe01/`.

[12] R. Ben-Eliyahu-Zohary and E. Gudes. Towards efficient metaquerying. *Proceedings of IJCAI 1999, Stockholm(Sweden)*, pages 800–805, 1999.

[13] Kristin P. Bennett, Usama M. Fayyad, and Dan Geiger. Density-based indexing for approximate nearest-neighbor queries. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 1999)*, pages 233–243, San Diego, CA, August 1999.

[14] Sergey Brin, Rajeev Motwani, Jeffrey D. Ullman, and Shalom Tsur. Dynamic itemset counting and implication rules for market basket data. In Joan Peckham, editor, *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA*, pages 255–264. ACM Press, 1997.

[15] Kaushik Chakrabarti, Minos N. Garofalakis, Rajeev Rastogi, and Kyuseok Shim. Approximate query processing using wavelets. In *Proceedings of 26th International Conference on Very Large Data Bases (VLDB 2000)*, pages 111–122, Cairo, Egypt, September 2000.

[16] Kaushik Chakrabarti and Sharad Mehrotra. Local dimensionality reduction: A new approach to indexing high dimensional spaces. In *Proceedings of 26th International Conference on Very Large Data Bases (VLDB 2000)*, pages 89–100, Cairo, Egypt, September 2000.

[17] David Wai-Lok Cheung and Jiawei Han. Maintenance of discovered knowledge: A strategy for updating association rules. *KDOOD/TDOOD*, pages 58–60, 1995.

[18] David Wai-Lok Cheung, Jiawei Han, Vincent Ng, Ada Wai-Chee Fu, and Yongjian Fu. A fast distributed algorithm for mining association rules. In *Proceedings of the Fourth International Conference on Parallel and Distributed Information Systems, December 18-20, 1996, Miami Beach, Florida, USA*, pages 31–42. IEEE Computer Society, 1996.

[19] David Wai-Lok Cheung, Sau Dan Lee, and Ben Kao. A general incremental technique for maintaining discovered association rules. In Rodney W. Topor and Katsumi Tanaka, editors, *Database Systems for Advanced Applications '97, Proceedings of the Fifth International Conference on Database Systems for Advanced Applications (DASFAA), Melbyurne, Australia, April 1-4, 1997*, volume 6 of *Advanced Database Research and Development Series*, pages 185–194. World Scientific, 1997.

[20] Paolo Ciaccia and Marco Patella. PAC nearest neighbor queries: Approximate and controlled search in high-dimensional and metric spaces. In *Proceedings of the 16th International Conference on Data Engineering (ICDE 2000)*, pages 244–255, San Diego, CA, March 2000.

[21] Paolo Ciaccia, Marco Patella, and Pavel Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB'97)*, pages 426–435, Athens, Greece, August 1997.

[22] Paolo Ciaccia, Marco Patella, and Pavel Zezula. A cost model for similarity queries in metric spaces. In *Proceedings of the 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'98)*, pages 59–68, Seattle, WA, June 1998.

[23] Paolo Ciaccia, Marco Patella, and Pavel Zezula. Processing complex similarity queries with distance-based access methods. In *Proceedings of the 6th International Conference on Extending Database Technology (EDBT'98)*, pages 9–23, Valencia, Spain, March 1998.

[24] Rina Dechter. Enhancement schemes for constraint processing: Backjumping, learning, and cutset decomposition. *Artificial Intelligence*, 41:273–312, 1990.

[25] Data Description. Data Desk. `http://www.datadesk.com/DataDesk/`.

[26] C. Domshlak, D. Gershkovich, E. Gudes, N. Liusternik, A. Meisels, T. Rosen, and S. E. Shimony. FlexiMine - a flexible platform for KDD research and application construction. a short version in kdd-98. *Technical Report FC-98-04, Ben-Gurion University*, 1998.

[27] C. Domshlak, D. Gershkovich, E. Gudes, N. Liusternik, A. Meisels, T. Rosen, and S. E. Shimony. FlexiMine-homepage. *Ben-Gurion University, Mathematics and Computer Science. URL: www.cs.bgu.ac.il/~ kdd*, 1998.

[28] C. Domshlak, T. Rosen, E. Shiller, and S. E. Shimony. Knowledge Discovery with FlexiMine. In *AIMTRW-98, AI Meets the Real World Conference*, Stamford, CT, September 1998.

[29] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Michael Wimmer, and Xiaowei Xu. Incremental clustering for mining in a data warehousing environment. In *Proc. 24th Int. Conf. Very Large Data Bases, VLDB*, pages 323–333, 24–27 August 1998.

[30] Martin Ester, Hans-Peter Kriegel, Jorg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In Evangelos Simoudis, Jia Wei Han, and Usama Fayyad, editors, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, page 226. AAAI Press, 1996.

[31] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 226–231, Portland, OR, 1996.

[32] Brian S. Everitt. *Cluster Analysis*. Edward Arnold, 3rd edition, 1993.

[33] Ronald Fagin. Combining fuzzy information from multiple systems. In *Proceedings of the 15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'96)*, pages 216–226, Montreal, Canada, June 1996.

[34] Christos Faloutsos and King-Ip Lin. FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pages 163–174, San Jose, CA, June 1995.

[35] Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. AAAI Press/MIT Press, March 1996.

[36] Hakan Ferhatosmanoglu, Ertem Tuncel, Divyakant Agrawal, and Amr El Abbadi. Approximate nearest neighbor searching in multimedia databases. In *Proceedings of the 17th International Conference on Data Engineering (ICDE 2001)*, pages 503–511, Heidelberg, Germany, April 2001.

[37] Y.J. Fu and J.W. Han. Meta-rule-guided mining of association rules in relational databases. *DOOD95 workshop on integration of knowledge discovery with deductive and object oriented databases, Singapore*, 1995.

[38] Venkatesh Ganti, Johannes Gehrke, and Raghu Ramakrishnan. CACTUS: Clustering categorical data using summaries. In Surajit Chaudhuri and David Madigan, editors, *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 73–83, N.Y., August 15–18 1999. ACM Press.

[39] Venkatesh Ganti, Raghu Ramakrishnan, Johannes Gehrke, Allison Powell, and James French. Clustering large datasets in arbitrary metric spaces. In *Proc. 15th IEEE Conf. Data Engineering, ICDE*, 23–26 March 1999.

[40] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *Proceedings of 25th International Conference on Very Large Data Bases (VLDB'99)*, pages 518–529, Edinburgh, Scotland, UK, September 1999.

[41] G.L.Andrienko. Descartes. `http://allanon.gmd.de/and/and.html`.

[42] Jonathan Goldstein and Raghu Ramakrishnan. Contrast plots and P-Sphere trees: Space vs. time in nearest neighbor searches. In *Proceedings of 26th International Conference on Very Large Data Bases (VLDB 2000)*, pages 429–440, Cairo, Egypt, September 2000.

[43] Silicon Graphics. MineSet. `http://www.sgi.com/software/mineset/`.

[44] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Rock: A robust clustering algorithm for categorical attributes. In *Proceedings of the 15th International Conference on Data Engineering, 23-26 March 1999, Sydney, Austrialia*, pages 512–521. IEEE Computer Society, 1999.

[45] Ulrich Güntzer, Wolf-Tilo Balke, and Werner Kießling. Optimizing multi-feature queries for image databases. In *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB 2000)*, pages 419–428, Cairo, Egypt, September 2000.

[46] John A. Hartigan. *Clustering algorithms*. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons., New York, 1975.

[47] Alexander Hinneburg, Charu C. Aggarwal, and Daniel A. Keim. What is the nearest neighbor in high dimensional spaces? In *Proceedings of 26th International Conference on Very Large Data Bases (VLDB 2000)*, pages 506–515, Cairo, Egypt, September 2000.

[48] Alexander Hinneburg and Daniel A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)*, pages 58–65, New York City, New York, USA, 1998. AAAI Press.

[49] IBM. Ibm intelligent miner. `http://www-4.ibm.com/software/data/iminer/fordata/`.

[50] IBM. Open visualization data explorer (DX). `http://www.research.ibm.com/dx/`.

[51] Belmont Research Inc. Cross Graphs. `http://www.belmont.com/cg.html`.

[52] Visual Insights. Advizor/2000. `http://www.vdi.com/`.

[53] Angoss Software International. Knowledgestudio. `http://www.angoss.com/ProdServ/AnalyticalTools/`.

[54] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1988.

[55] Anil K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, September 1999.

[56] Roberto J. Bayardo Jr., Rakesh Agrawal, and Dimitrios Gunopulos. Constraint-based rule mining in large, dense databases. In *Proceedings of the 15th International Conference on Data Engineering, 23-26 March 1999, Sydney, Austrialia*, pages 188–197. IEEE Computer Society, 1999.

[57] B. Kero, L. Russell, S. Tsur, and W. M. Shen. An overview of data mining technologies. *DOOD95 workshop on integration of knowledge discovery with deductive and object oriented databases, Singapore*, 1995.

[58] B. Leng and W.M. Shen. A metapattern-based automated discovery loop for integrated data mining - unsupervised learning of relational patterns. *IEEE Transactions on knowledge and data engineering*, 8(6):898–910, 1996.

[59] Chen Li, Edward Y. Chang, Hector Garcia-Molina, and Gio Wiederhold. Clustering for approximate similarity search in high-dimensional spaces. *IEEE Transactions on Knowledge and Data Engineering*. To appear.

[60] Integral Solutions Limited. Clementine. http://www.spss.com/.

[61] Cygron Research & Development Ltd. DataScope. http://www.cygron.com/.

[62] Livny M., Ramakrishnan R., and Wenger K. DEVise. http://www.cs.wisc.edu/ devise/.

[63] M. V. Mannino, P. Chu, and T. Sager. Statistical profile estimation in database systems. *ACM Computing Surveys*, 20(3):191–221, September 1988.

[64] Peter Bro Miltersen, Sairam Subramanian, Jeffrey Scott Vitter, and Roberto Tamassia. Complexity models for incremental computation. *Theoretical Computer Science*, 130(1):203–236, August 1994.

[65] B. G. Mitbander, K. Ong, W.M. Shen, and C. Zaniolo. *Metaqueries for data mining*, chapter 15, pages 375–398.

[66] Shinichi Morishita. On classification and regression. *Discovery Science*, pages 40–57, 1998.

[67] Inderpal Singh Mumick, Dallan Quass, and Barinderpal Singh Mumick. Maintenance of data cubes and summary tables in a warehouse. In Joan Peckham, editor, *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, pages 100–111, Tucson, Arizona, 13–15 June 1997. *SIGMOD Record* 26(2), June 1997.

[68] Visual Numerics. Jwave. http://www.vni.com/vnihome.html.

[69] Friedhoff R.M. and Peercy M.S. *Visual Computing*. Scientific American Library, 2000.

[70] W. L. Ruzzo. On uniform circuit complexity. *Journal of Computer and System Sciences*, 22(3):365–383, 1981.

[71] Gerard Salton. *Automatic Text Processing: The Transformational, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, Reading, MA, 1988.

[72] Thomas Seidl and Hans-Peter Kriegel. Efficient user-adaptable similarity search in large multimedia databases. In *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB'97)*, pages 506–515, Athens, Greece, August 1997.

[73] W. M. Shen. Discovering regularities from knowledge bases. *Intelligent Systems*, 7(7):623–636, 1992.

[74] Bernard W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall, 1986.

[75] Card S.K., Mackinlay J.D., and Shneiderman B. *Readings in Information Visualization*. Morgan Kaufmann, 1999.

[76] Artificial Intelligence Software. VisualMine. http://www.visualmine.com/.

[77] Neo Vista Solutions. Decision Series. http://www.accrue.com/products/decision.html.

[78] Ramakrishnan Srikant and Rakesh Agrawal. Mining generalized association rules. In Umeshwar Dayal, Peter M. D. Gray, and Shojiro Nishio, editors, *VLDB'95, Proceedings of 21th International Conference on Very Large Data Bases, September 11-15, 1995, Zurich, Switzerland*, pages 407–419. Morgan Kaufmann, 1995.

[79] L. J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1976.

[80] Advanced Visual Systems. Avs/express. `http://www.avs.com/software/soft-t/avsxps.html`.

[81] Catarci T. and Santucci G. The prototype of the dare system. In *Proceedings of the ACM SIGMOD Conf. on Management of Data*, 2001.

[82] E.R. Tufte. *Envisioning Information.* Graphics Press, 1990.

[83] Jeffrey D. Ullman. *Principle of database and knowledge-base systems*, volume I of *Principle of computer science series.* Computer Science Press, Inc., 1988.

[84] M. Y. Vardi. The complexity of relational query languages. *Proceedings of the Fourteenth ACM SIGACT Symposium on Theory of Computing (STOC-82)*, pages 137–146, 1982.

[85] Wei Wang, Jiong Yang, and Richard R. Muntz. STING: A statistical information grid approach to spatial data mining. In *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB'97)*, pages 186–195, Athens, Greece, August 1997.

[86] Roger Weber and Klemens Böhm. Trading quality for time with nearest-neighbor search. In *Proceedings of the 7th International Conference on Extending Database Technology (EDBT2000)*, pages 21–35, Konstanz, Germany, March 2000.

[87] Roger Weber, Hans-Jörg Schek, and Stephen Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of the 24th International Conference on Very Large Data Bases (VLDB'98)*, pages 194–205, New York, NY, August 1998.

[88] Leejay Wu, Christos Faloutsos, Katia P. Sycara, and Terry R. Payne. FALCON: Feedback adaptive loop for content-based retrieval. In *Proceedings of 26th International Conference on Very Large Data Bases (VLDB 2000)*, pages 297–306, Cairo, Egypt, September 2000.

[89] Pavel Zezula, Pasquale Savino, Giuseppe Amato, and Fausto Rabitti. Approximate similarity retrieval with M-trees. *The VLDB Journal*, 7(4):275–293, 1998.

[90] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: An efficient data clustering method for very large databases. In H. V. Jagadish and Inderpal Singh Mumick, editors, *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 103–114, Montreal, Quebec, Canada, 4–6 June 1996. *SIGMOD Record* 25(2), June 1996.