Data Surveying
Foundations of an Inductive Query Language

Arno Siebes
CWI
Database Research group
P.O.Box 94079
NL-1090 GB Amsterdam
The Netherlands
e-mail: arno@cwi.nl

Abstract

Data mining systems have to evolve from a set of specialised routines to more generally applicable *inductive query languages* to satisfy industry's need for strategic information. This paper introduces such an inductive query language called *Data Surveying*. Data Surveying is the discovery of "interesting subsets" of the database. Groups of customers whose behaviour deviates from average customer behaviour are examples of such interesting subsets. A user specifies what makes a subset interesting through a *survey task*. The wide applicability of this scheme is illustrated by a variety of examples.

To implement an inductive query language system, the "what" (the kind of strategic information sought) has to be made independent from the "how" (how this strategic information is discovered). In other words, the discovery algorithms have to be *task independent*. In this paper, operators on the search space are introduced to achieve this independence. The discovery algorithms are defined relative to these operators. To enforce efficient discovery, the notion of *polynomial convergence* is defined for these algorithms.

Domain knowledge plays an important role in the specification of both the survey task and the operators.

Keywords: Foundations of Data Mining, Inductive Query Languages, Domain Knowledge

Introduction

One of the main promises of data mining, (Piatetsky-Shapiro & Frawley 1991; Fayyad *et al.* 1995; Fayyad & Uthurusamy 1994; Holsheimer & Siebes 1994), for industry is the discovery of strategic information in their multi-gigabyte production databases. Data mining research at CWI is targeted at developing systems that will fulfill this promise.

Strategic information can be equated with *interesting subgroups* in a database. For example, for an insurance company knowledge of groups of clients with a (highly) deviating claim-behaviour is vital to stay competitive. Identifying groups of clients whose trend deviates over time from the average is also strategic. For retailers, Agrawals association rules (Agrawal, Imielinski, & Swami 1993; Agrawal *et al.* 1995) provide strategic information. This problem can be (re)formulated as the search for (large) groups of baskets that share a number of items.

The discovery of interesting subgroups is what we call *Data Surveing*. Data surveying is more like an "inductive query language" than a specific data mining problem, in fact, it encompasses and extends many techniques from data mining and statistics[1]. The advantage is that data surveying is widely applicable. The disadvantage is the required generality of the "discovery engine" and the potential loss of performance compared with tailored solutions caused by this generality.

The same considerations are, of course, true for conventional (deductive) query languages. In that case, the advantages of the generality outweigh by far the disadvantages of the loss of performance. One of the objectives of our research is to find out whether this will also hold for inductive query languages.

This paper introduces data surveying formally and illustrates its wide applicability. To implement such an inductive query language, the discovery of the results (the "how") has to be made independent of the actual discovery task (the "what"). This paper shows how this independence can be achieved by defining operators on the search space.

More in particular, in Section 2 the subgroups are formalised through *descriptions*, the notion of interestingness through *quality functions*, and data surveying through *survey tasks*. Subsequently it is shown that data surveying encompasses many techniques from statistics and data mining. This generality of data surveying comes at a price, viz., the survey tasks have to be formulated carefully. Some aspects of the task specification, such as the use of domain knowledge, are discussed in the last part of this section.

In Section 3, *description algebras* are introduced by

[1]The reason I use Data Surveying rather than the acronym IQL, is that this acronym has already been used for the *Identity Query Language* in (Abiteboul & Kanellakis 1989).

augmenting description languages with sets of operators. These sets of operators are in turn extended with *partial* operators to encode (factual) domain knowledge. These sets of operators form the interface between the discovery (the how) and the task (the what), the *discovery algorithms* are defined relative to them. To enforce efficient discovery, the notion of *polynomial convergence* is defined for these algorithms.

In the fourth and final section, the conclusions are formulated and future directions of research are indicated.

Data Surveying

The definition

In the introduction, data surveying is described as "the discovery of interesting subgroups". Clearly, the result of a data mining session should never be a listing of the members of such a subgroup. Rather, it should result in a (characteristic) *description* of this subgroup.

Such a description can be interpreted as a selection query on the database, i.e., the tuples in the result of this query are exactly the members of the associated subgroup of the database.

To simplify the discussion, we assume that our database has one relation DB, with schema $\mathcal{A} = \{A_1, \ldots, A_n\}$, with associated domains D_i. The set of all possible databases over $\mathcal{A}$ is denoted by $inst(DB)$. In line with the usual abstract definition of queries (Abiteboul, Hull, & Vianu 1994), descriptions and description languages[2] are defined as:

Definition 1 (Description Language) *A description ϕ for DB is a mapping from $inst(DB)$ to $inst(DB)$ that is generic and computable, such that $\forall db \in inst(DB) : \phi(db) \subseteq db$.*

A description language Φ for DB is a language whose expressions are descriptions for DB

Recall that *generic* means roughly that the description (query) cannot distinguish between constants that are indistinguishable in the input. The requirement $\phi(db) \subseteq db$ ensures that all descriptions are always associated with subgroups in the database.

The relational calculus is an example of a description language. An example of a description in that language is:

$$age = 25 \wedge sex = male$$

Descriptions can also be defined recursively. Due to the fact that all descriptions are meant as selection queries, we cannot use the Datalog syntax directly, however, the following description describes all the ancestors of "John" in the database:

$$t \in \phi(db) \leftarrow \left[\begin{array}{c} (t.name = X \wedge t.child = John) \\ \vee \left(\begin{array}{cc} & t.name = X \wedge t.child = Y \\ \wedge & (\exists s \in \phi(db) : s.name = Y) \end{array} \right) \end{array} \right]$$

For a more unusual example of a description language, let X and Y be real-valued attributes. The set of all lines in the X, Y-plane is also a description language. That is,

$$3X + 4Y = 5$$

is an example of a description in that language.

There are two notational conventions on the subgroups associated with descriptions that are used in this paper:

Definition 2 (Cover and Support) *Let Φ be a description language for DB and $\phi \in \Phi$ a description:*

1. *the* cover *of ϕ, denoted by $\langle\phi\rangle$, is the subset of all tuples in db that satisfy ϕ;*
2. *the* support *of ϕ, denoted by $\lfloor\phi\rfloor$ is the projection of $\langle\phi\rangle$ on the domain of the attributes used in ϕ.*

The cover of $age = 25 \wedge sex = male$ is the set of all 25 year old males in the database. Its support is simply $(age = 25, gender = male)$, if there is at least one 25 year old male in the database. The cover of our recursive description are all (connected) ancestors of "John". It's support is the set of all *(name, child)* pairs of all these ancestors. The support of $3X + 4Y = 5$ is the set of all pairs $(X = c_1, Y = c_2)$ for which there are tuples t in the databse with $t.X = c_1 \wedge t.Y = c_2$ and $3c_1 + 4c_2 = 5$.

I will slightly abuse notation and write $\lfloor t \rfloor_\phi$ to denote the projection of a tuple $t \in \langle\phi\rangle$ on the domain of the attributes used in ϕ.

The notion of an interesting subgroup is formalised through a *quality function*. This is a function that assigns a *quality* (a real number) to each description. The higher the quality of a description, the more interesting the corresponding subgroup is.

In general, the quality of a description is based on some property of its cover. For example, if an insurance company wants to find subgroups with a high risk of causing an accident, the quality of a description is related to the average number of accidents registred for its cover. This property of the cover can also be characterised through one or more descriptions: the *target descriptions*. In the insurance example, the target descriptions would be: $accident = yes$ and $true$. The quality of a description ϕ is than computed using:

$$\frac{|\langle\phi\rangle \cap \langle accident = yes\rangle|}{|\langle\phi\rangle \cap \langle true\rangle|}$$

where, as usual $|A|$ denotes the number of tuples in the set A.

In data surveying, all quality functions are based on such counts. More precisely, they are computed from *histograms*, which are defined as a straightforward generalisation of usual notion of a histogram:

[2]The reason to use the term description rather than query is to avoid confusion. In this case the descriptions are the results we are after, not the evaluation of the description as a query on the database.

Definition 3 (Histogram) *Let $\phi, \psi \in \Phi$ be two descriptions, the* histogram *of ϕ with regard to ψ, denoted by $H(\phi|\psi)$, is defined by:*

$$H(\phi|\psi) = \{(x,y)|\ x \in \lfloor\phi\rfloor \wedge \\ y = |\{t \in db|\ t \in \langle\phi\rangle \cap \langle\psi\rangle \wedge x = \lfloor t\rfloor_\phi\}|\}$$

For example, $H(age \in [19,24]|damage = yes)$ is the, usual, set of pairs (x,y) in which x denotes an age between 19 and 24 and y the number of accidents in that age group.

The histogram function is generalised to sets of descriptions with sets of target descriptions by

$$H(\{\phi_1,\ldots,\phi_k\}|\{\psi_1,\ldots,\psi_l\}) = \\ \{H(\phi_i|\psi_j)|\ 1 \le i \le k \wedge 1 \le j \le l\}$$

Quality functions simply take such a set of histograms as input and yield some real number as output.

Definition 4 (Quality function) *A quality function Q for a description language Φ is a pair $(\{\psi_1,\ldots,\psi_l\}, f)$ in which $\{\psi_1,\ldots,\psi_k\} \subseteq \Phi$ is the set of* target descriptions *and f is a computable real valued function. The quality of a set $\{\phi_1,\ldots,\phi_k\} \subseteq \Phi$ of descriptions is defined by:*

$$Q(\{\phi_1,\ldots,\phi_k\}) = f(H(\{\phi_1,\ldots,\phi_k\}|\{\psi_1,\ldots,\psi_l\}))$$

Examples of quality functions are given in the next subsection. However, before those examples are given the notion of a *survey task* is defined:

Definition 5 (Survey task) *Let db be a database, a survey task for db is a pair (Φ, Q), such that Φ is a description language for db and Q is quality function for Φ.*

The result of a survey task is defined as (a set of) description(s) that maximizes the quality function.

Examples

Data surveying is a, perhaps surprisingly, general framework. In (Siebes 1995; 1994a) it is shown that it encompasses traditional exploratory data analysis techniques such as *Projection Pursuit, Principle Component Analysis,* and *Cluster Analysis.* Moreover, in (Siebes 1995) it is also shown how ID3 (Quinlan 1986) fits in this framework. Finally, above, and in (Siebes 1994b; Holsheimer, Kersten, & Siebes 1995) in more detail but without the terminology, it is shown how risk-analysis fits.

As an example in this paper, association rules, (Agrawal, Imielinski, & Swami 1993; Agrawal *et al.* 1995), are briefly examined. In this case, the n binary valued attributes represent, e.g., the items in the store. The descriptions are of the form:

$$\bigwedge_{i \in I} A_i = \mathit{present}$$

for some $I \subseteq \{1,\ldots,n\}$.

The quality Q of a single description ϕ is 1 if $|\langle\phi\rangle| \ge \sigma$ (the minimal supporting set-size) and -1 otherwise. That is, the quality of a description is 1 is its cover is a "large" set and -1 otherwise. This quality function is easily extended to sets of descriptions by, e.g.,

$$Q^*(\{\phi_1,\ldots,\phi_k\}) = \sum_{i=1}^{k} Q(\phi_i)$$

The set with maximal Q^* is exactly the collection of all "large sets".

The specification of survey tasks

The generality of the data surveying framework is caused by the generality of the definition of a survey task, i.e., of descriptions and quality functions. This generality comes at a price: survey tasks have to be specified carefully. In this subsection we briefly discuss the choice of the description language and the choice of the quality function.

The description language chosen determines the subgroups to be found. Thus, how larger the set of descriptions is, how higher the chance that unexpected results are discovered. However, there are two good reasons to minimize the expressiveness.

To compute the quality of a description, it has to be evaluated as a query. Hence, the *data complexity* (Abiteboul, Hull, & Vianu 1994) of the description language (as a query language) becomes an important factor in the evaluation of a survey task.

Recall that the data complexity of a query ϕ is defined as the complexity of evaluating ϕ for variable database inputs. That is, it is the complexity of the problem:"given a database state *db* and a tuple u, determine whether u belongs to $\phi(db)$".

It is well-known that the more expressive a query language, and thus a description language, is, the higher its complexity is. For example, simple conjunctive queries are in LOGSPACE, whereas Datalog is in PTIME. In other words, the more expressive a description language is the longer it will take to compute the quality of a description.

So, the description language should be chosen as simple as is realistically possible. The second reason for such a tight choice is, of course, that the richer the language, the larger the search space and, potentially, the longer it takes to find the results

Domain knowledge plays an important role in this choice. For example, insurance companies require that a distinction in age is based on continuous intervals. Using such structural knowledge means a substantial reduction in the size of the search space without a reduction in the accuracy of the results.

The second component of the specification of a survey task is the specification of the quality function. Because the quality of a description determines whether it is strategic information or not, the quality function encodes *inductive inference.* For, we conclude generalities from a finite number of examples.

The epistemological problems of induction and its conclusions have been discussed by philosophers since at least the time of Hume. Some interesting points of view pertaining these problems can be found in (Holland *et al.* 1986). Since a long time, statistics is the most successful approach to asssess the validity of inductive conclusions. Hence, the quality function should be chosen such that the discoveries are *statistically valid.*

For example, if we discover that young males cause on average more accidents then other insurants, the validity of this claim has to be checked before this "risk-group" is reported. Space limitations preclude a further discussion of this important aspect in the specification of quality functions. In (Siebes 1995), the reader can find such a discussion.

The Discovery

Description algebra

Abstractly, the discovery of the descriptions that maximize the quality function[3] Q is independent of the actual survey task. Because, for a given database state, Q forms a *quality landscape* over the set of descriptions Φ. This quality landscape is analogous to the fitness landscape of genetic and evolutionary programming (Michalewicz 1992). All the discovery algorithm has to do is to find the peaks in this landscape and report the global maxima.

There is, however, one component missing in the search space to exploit this idea. Before one can genuinely speak of a landscape, one needs a topological or even metrical structure on Φ.

However, to make the independence between the "what" and the "how" concrete a complete metric structure on Φ is not necessary. All we have to add to Φ are those components of the structure that a search algorithm actually exploits.

For example, all what hill-climbing and simulated annealing algorithms need to know about the structure is the notion of a *neighbour.* That is, all they need on Φ is an *operator* n, that given a description $\phi \in \Phi$ returns the set of all its neighbours.

Similarly, all a genetic search algorithm needs are an operator that mutates descriptions and an operator that performs a cross-over on a pair of descriptions.

In other words, if we turn the description language into a *description algebra,* the search *is* independent of the actual survey task! Description algebras are defined as follows:

Definition 6 (Description Algebra) *A description algebra is specified by a pair* $(\Phi, \mathcal{O})$*, in which* Φ *is a description language and* $\mathcal{O}$ *a set of operators on* Φ*.*

[3]To simplify the discussion we assume in this section that we are searching for a single description. The generalisation to sets of descriptions is straight-forward.

Domain-knowledge

In the choice of a suitable description language "structural" domain knowledge plays an important role. For example by stipulating that an age-attribute has to be partitioned in continuous intervals. Clearly, there is more domain knowledge, viz., "factual" domain knowledge. For example, taxonomical information such as the fact that New York, San Fransisco, and Seattle are Big Cities; see also (Han, Cai, & Cercone 1992).

Such factual knowledge fits in our algebraic framework: it can be represented by *partial operators.* For example, a description ϕ that uses the value Seattle has the description[4] $\phi[Seattle \setminus BigCity]$ as a neighbour and vice versa.

Previous discovered knowledge implies partial operators in the same way. If for example ϕ maximises a quality function Q which has ψ as a target description, ϕ and ψ are neighbours.

These observations inspire the following definition:

Definition 7 Knowledge Base and Algebraic Survey Tasks
Let $D = (\Phi, \mathcal{O})$ *be a description algebra, a* knowledge base $\mathcal{KB}$ *for* D*, is a set of* partial operators *on* ϕ*.*

An algebraic survey task *for a database db is specified by a triple* $(D = (\Phi, \mathcal{O}), \mathcal{KB}, Q)$ *in which* D *is a description algebra for db,* $\mathcal{KB}$ *a knowledge base for* D *and* Q *a quality function on* Φ*.*

The result of an algebraic survey task is still defined as (a set of) description(s) that maximizes the quality function. The intention, however, is that the search algorithm uses only the (partial) operators in $\mathcal{O}$ and $\mathcal{KB}$ and the quality of the descriptions generated by these operators.

Different from conventional knowledge bases, *consistency* is not an issue for our knowledge bases. For, the partial operators in $\mathcal{KB}$ only augment the standard operators in $\mathcal{O}$. In fact, we assume that the partial operators in $\mathcal{KB}$ are associated with the operators in $\mathcal{O}$. That is, we assume that each of the partial operators in $\mathcal{KB}$ has the same name and arity as one of the operators in $\mathcal{O}$.

For example, neighbours are computed using the neighbour operator in $\mathcal{O}$ and the result is augmented by the descriptions that the applicable partial operators called neighbour in $\mathcal{KB}$ yield. The database is the queried for the quality of these descriptions, and the search algorithm bases its decision of the next step on this information only.

The search

To make the notion of a discovery algorithm more precise, we first have to discuss its type. Since we want to make the "how" (the discovery) independent of the "what" (the task), a discovery algorithm has to be

[4]Replace all occurences of the string "Seattle" in ϕ by the string "BigCity".

polymorphic in the search task and the database, under the requirements that the search task supplies the operators needed by the algorithm and the search task is applicable to the database.

To formalise this, let an *abstract operators* be a *(operator name, arity)* pair. If $\mathcal{O}_1$ is a set of abstract operators, and $\mathcal{O}_2$ a set of concrete operators on some description language Φ, $\mathcal{O}_1 \subseteq \mathcal{O}_2$ denotes that $\mathcal{O}_2$ has a concrete operator for every abstract operator in $\mathcal{O}_1$; i.e., one with the same name and arity.

Let $\mathcal{O}$ be a set of abstract operators, the type of a *discovery algorithm D for $\mathcal{O}$* is given by:

$$\begin{aligned} D \quad : \quad & \forall \tau[\tau = ((\Phi, \mathcal{O}_\tau), \mathcal{KB}, Q) : \mathcal{O} \subseteq \mathcal{O}_\tau] \\ & \forall \, DB \text{ for which } \Phi \text{ is a description language} \\ & \forall db \in inst(DB) : (\tau, DB, db) \rightarrow \Phi \end{aligned}$$

In general, the search space will be too large to search through exhaustively. In other words, an effective discovery algorithm can only "visit" a fraction of the descriptions in Φ.

For a simple search algorithm such as a hill-climber, the distribution of the quality function over the descriptions could be such that the algorithm is forced to visit all descriptions before it ends at the (global) maximum.

Therefore, we should not require that the algorithm visits only a limited number of descriptions in all runs. However, for a hill-climber the worst-case sketched above will be exceptional rather than the rule. Therefore, we can require that D will vist only a limited set of descriptions on the average over all runs for all possible tasks.

The larger the database, the larger the set of descriptions with a non-empty cover will be. In general, the size of the set of non-trivial descriptions will be exponential in the size of the database. Therefore, "a limited number of descriptions" is formalised as "polynomial in the size of the database" (and, thus, logarithmic in $|\Phi|$):

Definition 8 (Discovery Algorithm) *Let $\mathcal{O}$ be a set of abstract operators, a* discovery algorithm *D for $\mathcal{O}$ is a computable function, polymorphically typed as indicated above, such that*

1. *the average number of descriptions visited in a run is polynomial in $|db|$ and*
2. *if $\Psi = \{\phi_1, \ldots, \phi_n\}$ is the set of descriptions visited in one run, D reports a $\phi_i \in \Psi$ with maximal Q value.*

Clearly, this definition does not require that D reports a global maximum. Indeed, it should not, since heuristic algorithms may report a local maximum on a given run. What we *can* require is that the more runs we make, the higher the chance that we find a global maximum. This requirement can be formalised as follows:

Definition 9 (Convergence) *Let D be a discovery algorithm for a set of abstract operators $\mathcal{O}$. For a suitable task τ, denote by R_τ^n the set containing all sets of n runs for τ. Moreover, let ϕ_τ be the description that has maximal quality over all descriptions visited in a run $r \in R_\tau^n$. Finally, let ϕ_0 denote a description with maximal quality in Φ. D* converges *iff:*

$$\forall \tau \, \forall DB \lim_{n \rightarrow \infty} \frac{|\{r \in R_\tau^n | Q(\phi_\tau) = Q(\phi_0)\}|}{|R_\tau^n|} = 1$$

Note that even the discovery algorithm that reports the quality of one randomely chosen description satsifies this requirement. However, it will require a number of runs in the order of $|\Phi|$. For more efficient algorithms, note that if we spell out this definition, we get the usual $\forall \epsilon > 0 \, \exists N \, \forall n > N \cdots$. If this N is polynomial in the size of the database (and thus logarithmic in $|\Phi|$), we say that D *converges polynomially.*

Clearly, polynomial convergence is a desirable property for a discovery algorithm, for only in that case we know that we can find our interesting subgroups effectively.

Conclusions

In this paper the foundations for an inductive query language, Data Surveying, have been laid. The "queries" in this language have been formalised through survey tasks and the discovery of the results of such a task were formalised through discovery algorithms. The wide applicability of data surveying has been indicated by examples from data mining and statistics. The required independence of the how (the discovery algorithm) and the what (the actual survey task) has been achieved by defining the discovery algorithms relative to a set of operators on the search space. To enforce efficient discovery of the results, the notion of *polynomial convergence* of discovery algorithms has been introduced. Finally, the important role domain knowledge plays in the specification of both the survey task (structural knowledge) and the operators (factual knowledge in the knowledge base) has been emphasized.

While we have seen that polynomially converging discovery algorithms are well-suited for an efficient implementation of Data Surveying, we have not shown that they exist! This is the current topic of our theoretical research. The current practical goal is the construction of a new version of our tool Data Surveyor, one that supports the general framework of data surveying; see (Holsheimer *et al.* 1995).

Acknowledgements Many discussions with Marcel Holsheimer, Martin Kersten, Willi Klösgen, and Heikki Mannila have taught me what Data Mining is all about. Data Surveying is a brain-child that belongs to all of us.

References

Abiteboul, S., and Kanellakis, P. 1989. Object identity as a query language primitive. In *Proc. ACM SIGMOD Symposium on the Management of Data*, 159–173.

Abiteboul, S.; Hull, R.; and Vianu, V. 1994. *Foundations of Databases.* Addison Wesley.

Agrawal, R.; Mannila, H.; Srikant, R.; Toivonen, H.; and Verkamo, A. I. 1995. Fast discovery of association rules. In Fayyad et al. (1995). To appear.

Agrawal, R.; Imielinski, T.; and Swami, A. 1993. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 International Conference on Management of Data (SIGMOD 93)*, 207 – 216.

Fayyad, U. M., and Uthurusamy, R., eds. 1994. *AAAI-94 Workshop Knowledge Discovery in Databases.*

Fayyad, U. M.; Piatetsky-Shapiro, G.; Smyth, P.; and Uthurusamy, R., eds. 1995. *Advances in Knowledge Discovery and Data Mining.* AAAI/MIT Press. To appear.

Han, J.; Cai, Y.; and Cercone, N. 1992. Knowledge discovery in databases: An attribute-oriented approach. In *Proceedings of the 18th VLDB Conference*, 547 – 559.

Holland, J. H.; Holyoak, K. J.; Nisbett, R. E.; and Thagard, P. R. 1986. *Induction: processes of inference, learning and discovery.* Computational models of cognition and perception. Cambridge: MIT Press.

Holsheimer, M., and Siebes, A. 1994. Data mining: the search for knowledge in databases. Technical Report CS-R9406, CWI.

Holsheimer, M.; Klösgen, W.; Mannila, H.; and Siebes, A. 1995. A data mining architecture. In preparation.

Holsheimer, M.; Kersten, M.; and Siebes, A. 1995. Data surveyor: Searching the nuggets in parallel. In Fayyad et al. (1995). chapter 4. To appear.

Michalewicz, Z. 1992. *Genetic Algorithms + Data Structures = Evolution Programs.* Artificial Intelligence. Springer-Verlag.

Piatetsky-Shapiro, G., and Frawley, W. J., eds. 1991. *Knowledge Discovery in Databases.* Menlo Park, California: AAAI Press.

Quinlan, J. 1986. Induction of decision trees. *Machine Learning* 1:81–106.

Siebes, A. 1994a. Data mining: Exploratory data analysis on very large databases. In Apt, K.; Schrijver, L.; and Temme, N., eds., *From Universal Morphisms to Megabytes: A Baayen Space Oddysey (Liber Amicorum for Prof. P.C. Baayen).* CWI. 535–558.

Siebes, A. 1994b. Homogeneous discoveries contain no surprises: Inferring risk-profiles from large databases. In Fayyad and Uthurusamy (1994), 97 – 108.

Siebes, A. 1995. On the inseparability of data mining and statistics. In *Proceedings of the Mlnet Familiarization Workshop: Statistics, Machine Learning and Knowledge Discovery in Databases.*