Data Mining: Exploratory Data Analysis on Very Large Databases

To professor Baayen at the occasion of his retirement

Arno Siebes (arno@cwi.nl)

CWI

Artificial Intelligence and Database research are recognised parents of Data Mining research. Statistics is only considered related in as far as it allows the assessment of the quality of the results of mining. In this expository paper it is shown that Statistics can lay legitimite claims of parenthood. More in particular, it is shown how Data Mining can be seen naturaly as a generalisation of both Projection Pursuit and Cluster Analysis. Subsequently it is discussed how this link can help to give Data Mining firm mathematical foundations.

1 Introduction

One of the younger branches of Computer Science, called *Data Mining* or *Knowledge Discovery*, was born out of a, partial, merger of Database and Artificial Intelligence research.

1.1 What is Data Mining?

The goal of Data Mining is to discover information in large databases. Both large and small organisations have set up and maintained databases for years, often for pure accounting reasons. The mountains of data accumulated this way, form potential treasure-troves of strategic information.

For example, consider an insurance company. From your own car insurance policy you can deduce that such a company does not associate the same risk with all of its clients. Rather, this risk depends on where you live, your type of car, your age, and many other factors. If the insurance company has registred all the relevant information of its insurants in databases, it should be able to derive precise rules that tell which risk to assign to which client. The derivation of such *risk-profiles* is an example of data mining.

Many production processes are partly or completely automated. A side effect of this automation is that many aspects of the production process, such as the

quality of the end product and the parameter settings of the machinery along the way, are recorded electronically. The optimal paramater settings, those that one can be confident of the quality of the end product, are hidden in these databases. Data mining intends to facilitate unearthing this knowledge.

1.2 The Roots

Data Mining is based on techniques inherited from both AI and database research. Statistics is used to assess the validity of the results. The roots of Data Mining in these three areas is discussed briefly in this subsection. In the last part, on Statistics, the goal of this paper is set out.

1.2.1 Artificial Intelligence

The AI parent of Data Mining is without a shadow of doubt *Machine Learning.* One of the aspects of intelligent beings is that they adapt their behaviour to their environment. So, it is only natural that early AI researchers developed systems that mimicked this behaviour.

One of the oldest examples of such systems is the *Perceptron* by Rosenblatt [36], a system for *pattern recognition.* The object of pattern recognition is to sort patterns into different classes so that patterns which belong to a class share features. If we call the set of all possible feature combinations the feature space, Perceptron performes well for those patterns that are linearly separable in feature space. Minsky and Pappert showed the limitations of the Perceptron if the patterns are not linearly separable in [29]. *Neural networks* are a way to overcome these limitations, see, e.g., [10] for an introduction in this area.

Neural networks are by far not the only attempt at building learning automata. In fact, an overview of machine learning research is far beyond the scope of this article, if not beyond the scope of a single book. The interested reader is referred to the collection of papers bundeled in [38] and the books edited by Michalsky, [27, 28, 23] to get a feeling for the area. The more theoretically inclined reader might enjoy [1].

The most important development in machine learning for current Data Mining research is the introduction of rule induction systems, [14]. Rule induction is similar to neural networks in that it seeks to separate patterns. The major difference is that it seperates using *descriptions* rather than weights in a network. The descriptions are expressions in the attributes or features of the objects. Hence, the results of rule induction are directly interpretable by human beings.

1.2.2 Databases

The problems in database research that gave rise to Data Mining are more diverse and less well-documented than those in AI. Rather than attempting to describe briefly all these seemingly unrelated problem areas, we describe one in somewhat more detail.

One of the main problems in maintaining large data sets, electronically or otherwise, is to keep them error-free. One of the contributions of database research towards the resolution of this problem is the notion of *integrity constraints.* The constraints on a database describe which entries in a database and which database states are to be considered legal. The more accurate the constraints are, the more errors at, say, data entry can be obviated.

The traditional way to discover constraints is to elicit them from domain experts. Thus, inherently, there is the risk that some constraints are missed. One way to alleviate this risk is by searching for additional constraints when the database is in existence. By confronting the domain experts with constraints that are satisfied by the current database state, these missing constraints can be identified. Pioneering papers in this area are [30, 2, 3].

In theory, a constraint is simply a logical expression. In practice, however, database management systems support only the enforcement of a restricted set of constraints, such as *functional dependencies.* In table R attribute A functionally determines attribute B, denoted by $A \rightarrow B$, if whenever two entries share the same A-value they also share the same B-value.

For this restricted class of constraints, the problem is solved. Efficient algorithms can be found in, e.g., [24]. While Manilla gives precise bounds on the sample sizes needed to conclude the constraints with sufficient confidence in [21].

1.2.3 Statistics

As should be clear from the examples given before, Data Mining is based on *inductive inference.* In other words generalities, such as *rules* or *laws*, are induced from a finite number of examples. Such a conclusion is, of course, never *logical,* the logical conclusions can be inferred using *deduction.* 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 [12].

Since a long time, Statistics is the most successful approach to asssess the validity of inductive conclusions. It is therefore to be expected that Statistics is used in Data Mining precisely for this reason. In other words, Statistics is related to Data Mining.

However, Statistics offers more than is currently used. An introductory course in Statistics and Probability is sufficient to read almost all the literature on Data Mining. Curiously, all Statistics that comes under the name of *Exploratory Data Analysis* is absent in these requirements.

It is the intention of this paper to show that Statistics is more than related to data mining. It could have been, and perhaps should be considered as, one of its parents. More in particular it is shown that Data Mining can be seen as a natural generalisation of a statistical techniques known as *Projection Pursuit* and *Cluster Analysis*

1.3 A Roadmap

The object of this paper is expository, the reader is neither expected to be a statistician nor a data miner. The only new fact in this paper is the surprisingly strong link between Exploratory Data Analysis techniques and Data Mining.

In Section 2, we give a brief review of the classical techniques Regression Analysis, Principle Component Analysis and Cluster Analysis. In the next section Projection Pursuit is introduced and, following Huber [15] it is shown how this subsumes the first two techniques of Section 2.

In the fourth section Data Mining is defined and it is shown how it generalises both Projection Pursuit and Cluster Analysis. In Section 5, the contribution of AI and databases is discussed in the light of this new viewpoint.

In the final section of this paper it is discussed how this link might help to give data mining firm mathematical underpinnings. Since the discovery of such underpinnings needs guidance from experimentation, the architecture of a data mine tool is also briefly discussed.

2 Classical Exploratory Data Analysis

In many laboratory experiments parameters can be individually set. Consequently, hypotheses underlying these experiments can be tested with straightforward statistical techniques. Not all sciences are so lucky, however. In the life sciences and in the social sciences the parameters cannot even be set by the scientist. To analyse this kind of data Statistics developed *Multivariate Analysis*.

Tukey coined the name *Exploratory Data Analysis* (EDA) [40] for, a subset of, these techniques to indicate that the analysis is only part of the work. The interpretation of the results, the formulation of hypotheses and their subsequent testing are equally important. Since I agree with this observation, I have adopted this catchy name.

In this section three "classical" techniques, Regression Analysis, Principle Component Analysis, and Cluster Analysis, are briefly reviewed. The motivation for this section is twofold. In the first place it may serve as a reminder for the avarage data miner. In the second place, the power of Projection Pursuit is argued in the next section by discussing how it subsumes the first two techniques. Subsequently it is argued that Data Mining subsumes both Projection Pursuit and Cluster Analysis. Far more information on EDA can be found in standard textbooks such as [40, 25].

2.1 Regression Analysis

Suppose that the insurance company from the introduction has d real valued attributes in its clients database. If it assumes that, say, the expected claim amount is a function of these variables, it can use Regression to determine this function.

In formal terminology, let (X, Y) be a pair of random variables such that X is $\mathcal{R}^d$ valued while Y is $\mathcal{R}$ valued. The problem is to estimate the *response*

surface

$$f(x) = E(Y|X = x)$$

from n observations $(X_1, Y_1), \cdots, (X_n, Y_n)$ of (X, Y).

A simple way to fit a function to these n observations is through *least squares estimation.* First a parametric form for f is chosen, e.g., if f is assumed to be a linear surface, we have $f(\vec{x}) = \sum_{i=1}^{n} a_i x_i + a_o$. Following, the parameters a_i are estimated by minimising

$$\sum_{i=1}^{n} (Y_i - f(X_i))^2.$$

This can be generalised by assuming Y to be $\mathcal{R}^k$ valued rather than $\mathcal{R}$ valued. If f is then assumed to be linear we get what is known as *multivariate regression.* A generalised least squares estimation exists for this case.

Regression analysis is an example of EDA, if only because one can try different parametric forms for f and choose the one that fits best. Of course, the number of parameters should be small compared to the number of observations. In the terminology of Machine Learning, one should beware of *overfitting.*

2.2 Principle Component Analysis

With Principle Component Analysis (PCA), one hopes to explain most of the variability in the data using only the *principle components* with the highest variability. In other words, PCA is a technique to reduce the dimensionality of the data.

Let X be an $\mathcal{R}^d$ valued random variable and let $X_1, \cdots, X_n$ be a set of n observations of X. In statistical terminology, $(X_1, \ldots, X_n)^T$ is a data matrix. For example, we have a group of n students who all participated in d examinations and X_{ij} denotes the score of student i for examination j. The sample mean vector $\bar{X}$ is simple defined by

$$(\frac{1}{n}\sum_{i=1}^{n} X_{i1}, \cdots, \frac{1}{n}\sum_{i=1}^{n} X_{id})^T,$$

In other words, $\bar{X}_i$ denotes the mean score for examination i. The sample covariance matrix is the $d \times d$ matrix S with entries

$$s_{ij} = \frac{1}{n}\sum_{r=1}^{n} (X_{ri} - \bar{X}_i)(X_{rj} - \bar{X}_j).$$

The covariance matrix S can be written in the form $S = GLG^T$ in which G is an orthogonal matrix and L a diagonal matrix of the eigenvalues of S, with $l_1 \geq l_2 \geq \cdots \geq l_p \geq 0$.

The principle component transformation is defined by rotation

$$W = (G^T(X_1 - \bar{X}), \ldots G^T(X_n - \bar{X}))$$

the columns of W represent *uncorelated* linear combinations of the variables; they are called the *principle components.*

The importance of PCA lies in the observation that $(l_1 + \cdots + l_k)/(l_1 + \cdots + l_d)$ represents the "proportion of the total variation" explained by the first k principle components. So, if in our examination example $l_1/(l_1 + \cdots + l_d) = 0.75$ and its eigenvector is $(1, 0, \ldots, 0)$, the we can conclude that 75% of the variation of the scores of the students is due to the first examination.

2.3 Cluster Analysis

Cluster Analysis (CA) is similar to pattern recognition discussed before. Again we try to classify based on similarity. Different from the previous two techniques, CA does not require the data to be real valued. To simplify our brief discussion we, however, make this assumption.

Again, let X be an $\mathcal{R}^d$ valued random variable and let $\mathcal{X} = X_1, \cdots, X_n$ be a set of n observations of X. A clustering of $\mathcal{X}$ is a cover of $\mathcal{X}$ by disjoint subsets $C_1, \ldots, C_k$. The goal is that the observations in the same class are similar while observations in different classes are different.

For example, if the X_i are observations on flowers, recording the length of the stem, the number of petals, et cetera, a clustering should put observations of flowers of the same kind in the same class.

Inherent in this statement of the cluster problem is the concept of an optimality criterion which dictates when a desirable partitioning has been found. This criterion can be phrased using a *quality function.* The higher the quality of a partitioning, the better it is.

More in particular, we need a measure of the *homogeneity* within a cluster and the *disparity* between clusters. Both measures can very well be based on a *distance function* or *metric* on $\mathcal{R}^d$.

For example, in *complete linkage* one of the restrictions on a class is that the distance between two observations may not exceed some threshold value r. In the *centroid method*, the distance between classes is defined as the distance between their centroids. One of the objectives of this method is to maximise the distance between classes.

There are way to many clustering algorithms to attempt even the shallowest of surveys here. An old, but very readable survey, can be found in [5]. This brief description ends with the observation that *clustering by complete enumeration* is comletely out of the question.

Briefly, this technique would simply enumerate all possible clusterings, evaluate the quality of all of them and report the one(s) with the highest quality. This approach is infeasible simply by the sheer number of possible clusterings. The number of partitions of n objects in m non-empty subsets is given by *Stirling's numbers of the second kind:*

$$S(n, m) = \frac{1}{m!} \sum_{j=0}^{m} \binom{m}{j} (-1)^j (m - j)^n.$$

So, since the number of classes is in general not specified, the total number of clustering alternatives is given by:

$$\sum_{m=1}^{n} S(n,m).$$

3 Projection Pursuit

Mapping multivariate data into low dimensions for visual inspection is a commonly used technique in data analysis; if only because of the uncanning ability of humans to discover structure in two-dimensional plots. The discovery of such mappings that reveal the salient features of the multidimensional data set is in general far from trivial. *Projection Pursuit* (PP) introduced by Friedman and Tukey in [9] is a technique to discover such mappings.

In a nutshell, PP works as follows. We have a p-dimensional dataset X and we examine "all", say, two-dimensional projections of X. We are given some quality function, called the projection index, with which we calculate the quality of all the projections. PP then reports the projection with the highest quality.

Stated as such, PP sounds like just another EDA technique which might as well have been discussed in the previous section. After a brief discussion of PP, however, it is shown, following Huber [15], that PP subsumes many EDA techniques.

3.1 What is Projection Pursuit?

The simplest mappings from higher to lower dimensions are, linear, projections. That is, linear maps A of, say, rank 1 or 2. By definition, PP searches for a projection A that maximises a quality function, in this context it is called the *projection index.*

To get more concrete, let X be a $\mathcal{R}^d$ valued random variable and let $\mathcal{X} = \{X_1, \cdots, X_n\}$ be a set of n observations of X. A 1-dimensional projection A is then a $1 \times d$ matrix of rank 1. The quality of A should be determined from the data set $A(\mathcal{X}) = \{AX_1, \cdots, AX_n\}$.

Many projection indices are possible, an important observation by Huber is that the index should measure how far the projection is away from a set of data points sampled under a normal distribution. The heuristic arguments underlying this claim are:

- A multivariate distribution is normal iff all its one-dimensional projections are normal. Thus, if the least normal one dimensional projection is normal, we need not look at any other projection.

- For most high-dimensional data sets most low-dimensional projections are approximately normal.

A simple projection index in this case is, thus, a χ^2-test. Another example is the sample entropy, i.e.,

$$\frac{1}{n}\sum_{i=1}^{n}\log(\hat{f}(AX_i))$$

in which $\hat{f}$ is the density estimate of the projected points. Friedman and Tukey's original index I is the product of two functions s and k, where s measures the spread of the data and k describes the "local density"of the data after projection.

Defining the index is only part of the work. The question is, of course, how to find the projection A that maximizes the index. Friedman and Tukey mention that their projection index is sufficiently continuous to allow the use of hill-climbing algorithms for the maximization.

A simple form of hill-climbing is as follows. First we choose a random projection matrix $A = (a_1, \ldots, a_d)$ and compute its quality. Subsequently, we construct a set $\{A_1, \ldots, A_N\}$ by adding small vectors to A in "all possible directions". Then we compute the quality of all these projections. The new projection A' is that projection from the set $\{A, A_1, \ldots, A_N\}$ that has maximal quality. If $A = A'$ we stop, else we iterate.

This form of hill-climbing will always end in a local maximum. To find a global maximum the algorithm should be repeated with different initial projections. Moreover, in fact the search is not so much for the global maximum as well as for a projection that gives the analyst insight in the distribution of the data. In other words, we can stop as soon as we find a local maximum that satisfies this criterium.

Besides hill-climbing many more search algorithms exist, we return to this topic later in this paper.

3.2 Projection Pursuit subsumes classical techniques

It is straightforward that PP is a generalisation of PCA. For, in PCA we simply calculate the eigenvalues and eigenvectors of the covariance matrix and project the data orthogonally into the space spanned by the eigenvectors belonging to the largest eigenvalues. This projection clearly fits into our description of PP above.

3.2.1 Regression

The subsumption of Regression by PP is less straightforward than that of PCA above. A central role is played by the "curse of dimensionality" caused by the fact that a high-dimensional space is mostly empty. To give an example let $d = 20$, which is actually low in most data mining examples. Assume that we have a large number of points uniformly distributed in a 20-dimensional unit ball. Then the radius of a ball containing 5% of the data is $(0.05)^{(0.05)} = 0.86$. So, if we want to pick out small features the sample size has to be gigantic. In

other words, for high-dimensional data sets standard Regression is not likely to produce good approximations.

In this case [15], it is often attractive to approximate the response surface by a sum of *ridge functions:*

$$f(x) \approx \sum_{i=1}^{m} g_i(a_i^T x)$$

In other words, we assume that f can be approximated by the sum of a set of $\mathcal{R}$-valued functions, each of which is defined on a 1-dimensional projection of $\mathcal{X}$. The idea is now to use PP to find the "optimal projections" for this approximation. More in particular, Friedman and Stuetzle's Projection Pursuit Regression process [8] works as follows. Assume we have already determined the first $m-1$ vectors a_i and functions g_i. Let

$$r_i = Y_i - \sum_{i=1}^{m-1} g_i(a_i^T x)$$

be the residuals of this approximation. Choose a unit vector $a \in \mathcal{R}^d$ and fit a smooth function g through the data set formed by the pairs $(a^T X_i, r_i)$. Calculate the sum of squared residuals relative to this g,

$$\sum_{i=1}^{n} (r_i - g(a^T X_i))^2$$

and then minimise this sum over all possible choices for a. The resulting a and g are then inserted as the next term in the approximating sum. This iterative procedure stops if the improvement becomes small.

In a similar sense, PP can be said to subsume density estimation. That is, in cases of high-dimensionality Projection Pursuit Density Estimation yields an acceptable approximation.

3.2.2 Clustering

If stating that PP subsumes Regression was already stretching the limits, stating that it subsumes Clustering certainly oversteps these limits. However, PP can certainly help to detect clusters; one might say that this was the motivation for developing PP. In fact, Huber presents the following list of as possible actions after one has found some interesting projections:

1. Identify clusters, isolate them and investigate them seperately.
2. Identify clusters and locate them (i.e., replace them by, say, their center and classify points according to membership to a cluster).
3. Find a parsimonious description (seperate structure from random noise in a nonparametric fashion).

Data Mining not only generalises PP, it *does* generalise Clustering. How it achieves this, is discussed in the next section.

4 Data Mining

For some researchers, Data Mining is simply the application of Machine Learning techniques to large databases. This point of view, however, is far too broad; if only because some techniques simply do not scale up to the massive amounts of data available in databases.

Klösgen and Zytkow define KDD, one of the many aliases of Data Mining, in [22] as

> **Knowledge Discovery in Databases (KDD)** is a major direction in machine discovery dealing with knowledge discovery processes in databases. KDD applies to the ready data available in all application domains of science and in applied domains of marketing, planning, controlling, etc. Typically, KDD has to deal with inconclusive data, noisy data, and sparse data.

where *machine discovery* and *knowledge discovery process* are defined by respectively:

> **Machine Discovery** is a subfield of Artificial Intelligence which develops discovery methods and discovery systems to support knowledge discovery processes.

> **Knowledge Discovery Process** aims at finding out new knowledge about an application domain. Typically, a discovery process consists of many discovery steps, each attempting at the completion of a particular discovery task, and accomplished by the application of a discovery method. A discovery process emerges iteratively and depends on the dynamic, result dependent discovery goals. The process iterates many times through the same domain, typically based on search in various hypotheses spaces. New knowledge is inferred from data often with the use of old knowledge. Domain exploration and discovery focussing are discovery processes applied in new domains, where old knowledge is not available.

For the definition of the unfamiliar terms in these definitions, the reader is referred to [22]. In this paper we use a, slightly, formalised restricted version of this general definition. It is not meant as a general introduction to Data Mining. Again, this is far beyond the scope of this paper. The interested reader is referred to [14, 31, 32].

4.1 Descriptions and Quality

Central in Data Mining is the notion of a *description*. Recall that a database table consists of a *schema* and a *state*. A schema is a set of *attribute names* $\mathcal{A} = \{A_1, \ldots, A_p\}$ together with a set of *attribute domains* $\{D_1, \ldots D_p\}$, such that D_i is the *domain* of A_i. A state of the table can be seen as a finite subset of $D_1 \times \cdots \times D_p$.

Usually, databases have more than one table and the tables are subject to constraints etcetera, but these nuances are unimportant for our present purposes. In other words, we will equate databases with tables as defined above, i.e., a database state $db \subseteq_{fin} D_1 \times \cdots \times D_p$. By DB we will denote the set of all possible database states.

A *tuple* t is simple an element of a database state, i.e., $t \in db$. Rather than using a projection-notation like $\pi_{D_i}(t)$, $t.A_i$, or even t_i if $\mathcal{A}$ is understood, is used to denote the value of t for attribute A_i.

With these conventions, we can define a *description language* Φ as a first order language such that $\forall \phi \in \Phi \, \forall db \, \forall t \in db$ it can be decided whether ϕ holds for t in db. Note that usually the attribute names in $\mathcal{A}$ will be among the non-logical symbols of Φ.

A popular description language is that of *set-descriptions*, these are descriptions of the form:

$$A_i \in V_i \wedge \cdots \wedge A_k \in V_k, \textit{where } A_j \in \mathcal{A} \wedge V_j \subseteq D_j \wedge V_j \textit{ is finite.}$$

The description $age \in [19, 24] \wedge gender = male$ is an example. Since the V_i are assumed to be finite, this is a first order language in disguise.

The cover of a description ϕ, denoted by $\langle \phi \rangle_{db}$, in a database state db is the set of all tuples in db that satisfy ϕ; if db is clear from the context, this subscript is often ommited. For example, $\langle age \in [19, 24] \wedge gender = male \rangle$ denotes all tuples in the database that describe young men.

Besides descriptions, a central role is played by *quality functions*, similar to those encountered in EDA. In fact, there are three classes of quality functions that are used in Data Mining:

Class 1 this are quality functions that assign a quality to a single description for a given database state. That is, it are functions of type $\Phi \times DB \rightarrow \mathcal{R}$.

Class 2 this are quality functions that assign a quality to a finite set of descriptions for a given database. That is, they are of type $\mathcal{P}_{fin}(\Phi) \times DB \rightarrow \mathcal{R}$.

Class 3 this are quality functions that assign a quality functions that assign a quality to a set of descriptions for a given database state based on a combination of a Class 1 and a Class 2 quality function. In other words, a quality function of this class is specified by three functions:

1. $Q_1 : \Phi \times DB \rightarrow \mathcal{R}$;
2. $Q_2 : \mathcal{P}_{fin}(\Phi) \times DB \rightarrow \mathcal{R}$:
3. $f : \mathcal{P}_{fin}(\mathcal{R}) \times \mathcal{R} \rightarrow \mathcal{R}$;

   and $Q_3 : \mathcal{P}_{fin}(\Phi) \times DB \rightarrow \mathcal{R}$ is defined by $Q_3 = f(\{Q_1\}, Q_2)$.

Given the set of descriptions Φ and the quality function(s), Data Mining is simply: "find the (set of) description(s) with the highest quality". Simple variations are of the form: "give me the n best descriptions" etcetera.

It is now easy to see that both Cluster Analysis and Projection Pursuit are examples of Data Mining.

4.1.1 Cluster Analysis

Define the description language Φ such that all finite subsets of $\mathcal{P}(D_1 \times \cdots \times D_p)$ can be described. Moreover, define Q_1 as a function that measures the homogeneity of the clusters, i.e., of the $\langle \phi_i \rangle$, Q_2 as a function that measures the disparity between the clusters, and define f as a function that combines Q_1 and Q_2. The resulting Class 3 quality function and the description language Φ together form a specification of the clustering problem as defined before.

4.1.2 Projection Pursuit

This one is even more simple, define Φ such that all and only all projection planes can be described. Moreover, define the Class 1 quality function as your favourite PP index. The result is PP as a Data Mining problem.

4.1.3 Subsumption

It is disputable whether Data Mining with Class 3 quality functions is a more general problem than Cluster Analysis. However, although the two problems may be close in theory, they are widely different in practice. Most often in Cluster Analysis, the quality is somehow related to a distance function. In Data Mining, however, the quality function is simply part of the specification of the kind of information one is interested in.

The fact that Data Mining with Class 1 quality functions is more general than Projection Pursuit is far less disputable. There is at least one paper that studies PP on discrete data rather than continues data, [4], but in Data Mining one does not fix an a priori "projection dimension", rather one lets the system find the most striking projection.

The generality of Data Mining does have its price, however. In the first place, one has to specify each search task. That is, one has to choose an appropriate description language and a reasonable quality function. This specification comes at the price of a thorough analysis of the problem. In other words, Data Mining is not "plug and play".

The second down-side lies in the search algorithms. The generality of the Data Mining problem implies that it is difficult to use the structure in a problem to speed up the search. In other words, the search algorithms should be able to cope with a large collection of widely different quality functions which, e.g., do not have to depend on a metric as in Cluster Analysis.

In the next subsection we give an example on the definition of quality functions. In the next section we return to the problem of search algorithms.

4.2 Risk Profiles: an example of quality functions

One of the Dutch insurance companies has asked us to derive *risk-profiles* from their car-insurance databases. A set of risk-profiles is a classification of the

insurants such that the insurance company can expect all clients in the same class to cause the same claim-amount per year. The relevance of this knowledge for the insurance business is obvious.

As a first approximation, we derive risk-profiles for the probability that someone will cause a claim, rather than for the expected claim-amount. In this section we briefly explain how these risk-profiles were found; more information can be found in [39].

4.2.1 The Problem

The assumptions underlying this task are as follows. First, we assume that there only a few groups of clients, such that clients in the same group share the same probability of causing a claim. Secondly, we assume that these groups can be distinguished using only a few, say 80, properties of the clients and their cars; moreover, these properties are present in the database as attributes. Finaly, we assume that these groups can be distinguished by our description language Φ.

A precise definition of Φ is not important here. It is a sublanguage of the language of set-descriptions that satisfies the following properties:

1. Φ should be *sparse*, this more or less means that $\langle\phi\rangle$ should be large and with attributes such as *area* and *age* there should be no gerrymandering;

2. If $\langle\phi\rangle \cap \langle\psi\rangle$ is large for $\phi, \psi \in \Phi$, then $\phi \wedge \psi \in \Phi$.

To state our problem in terms of descriptions, define a set $\{\phi_1, \ldots, \phi_k\}$ of descriptions to be a *disjunctive cover*, abbreviated to *discovery*, if:

1. $\forall i, j \in \{1, \ldots, k\} : i \neq j \rightarrow [\phi_i \wedge \phi_k \rightarrow \bot]$

2. $[\bigvee_{i=1}^{k} \phi_i] \rightarrow \top$

The problem can then be restated as: find a discovery $\{\phi_1, \ldots, \phi_k\}$ such that

$$\forall v_1, v_2 \in D_1 \times \cdots \times D_n : [p(v_1) = p(v_2)] \leftrightarrow [\forall i \in \{1, \ldots, k\} : [\phi_i(v_1) \leftrightarrow \phi_i(v_2)]].$$

4.2.2 Analysis of the problem

A set of clients is called *homogeneous* if all members have the same probability of causing a claim. A description ϕ is homogeneous, if the set of all clients that satisfy ϕ is homogeneous. Clearly, the discovery we want to find should be homogeneous, i.e., all its descriptions should be homogeneous.

For a homogeneous description ϕ, the probability of causing a claim of the clients that satisfy ϕ can easily be estimated from the database. Since, all tuples in $\langle\phi\rangle$ can be seen as records of trials of the same Bernoulli experiment. The outcome of this experiment is 1 (a success (sic)) if there was an accident and 0 otherwise.

So, using standard probability theory, [7], we can compute the, say 95%, confidence interval CI_ϕ for the probability of causing a claim of the clients that satisfy ϕ.

In fact, we will compute CI_ϕ in this way for all descriptions ϕ, regardless of whether they are homogeneous or not. Since our end-result is a homogeneous discovery this does not introduce errors.

The question is now, how do we decide whether a description is homogeneous or not. Intuitively, ϕ is homogeneous, if all subsets of $\langle\phi\rangle$ have the same associated probability. But this cannot not work, in a vase with with n blue and m red marbles one can find subsets with fractions of blue marbles varying from 0 to 1.

However, we are not interested in random subsets, but only in subsets that can be described by Φ and Φ is assumed to be sparse. Therefore, we define a description $\phi \in \Phi$ to be homogeneous[1] if:

$$\forall \psi \in \Phi : \phi \wedge \psi \in \Phi \rightarrow CI_\phi \cap CI_{\phi\wedge\psi} \neq \emptyset$$

In other words, if we call $\phi \wedge \psi$ an *extension* of ϕ, a description is homogeneous if its associated probability cannot be distinguished, with 95% certainty, from those of its extensions

Not all homogeneous discoveries are answers to our question, because not all homogeneous discoveries satisfy the condition that the associated probabilities are distinct. Those homogeneous discoveries that do satisfy this condition are said to *split* the database. In other words, a homogeneous discovery $\{\phi_1, \ldots, \phi_l\}$ splits the database if:

$$\forall i, j \in \{1, \ldots, l\} : i \neq j \rightarrow CI_{\phi_i} \cap CI_{\phi_j} = \emptyset$$

All such discoveries are potential answers to our question.

4.2.3 Existence and Quality

If Φ is carefully defined, many homogeneous discoveries will exist. For example, from a list $\Psi = [\phi_1, \ldots, \phi_n], \phi_i \in \Phi$ of descriptions we can generate the list $\Psi' = \{\phi_1, \neg\phi_1 \wedge \phi_2, \neg\phi_1 \wedge \neg\phi_2 \wedge \phi_3, \ldots, (\neg\phi_1 \wedge \cdots \wedge \neg\phi_n)\}$. Ψ' is potentially a homogeneous discovery if it is, Ψ is called a *decision list*, [35].

Whether there exist homogeneous discoveries that split the database depends more on the actual database state than on the design of Φ. In other words, there might be 0, 1 or many.

If there are 0, we are out of luck. The database simply does not contain enough information to partition the clients through risk-profiles. If there are many, we seem to be in similar straits because we can assign many different risks to the same client. However, the *quality* of the different discoveries may differ considerably. In other words, one might be naturally the best.

[1] A related notion of homogeneity has been introduced independently in [37]

A detailed discussion of quality measures on discoveries is outside the scope of this paper. One aspect, however, is interesting to note. One reason for having many homogeneous discoveries is that many descriptions are homogeneous by definition, i.e., all those descriptions which have no extensions in Φ.

These trivially homogeneous descriptions are in a sense too small to count. In other words, a homogeneous description with a large cover is better than one with a small cover. Extending this to discoveries, a discovery that partitions the database in large subsets is better than one that partitions it into smaller subsets.

Similarly, the better a set of descriptions distinguishes between its components, the better it is. To formalise this, define that a homogeneous set of descriptions $\{\phi_1, \ldots, \phi_l\}$ *strongly splits* the database if its descriptions differ in all aspects:

$$\forall \psi \in \Phi \; \forall i, j \in \{1, \ldots, l\} : \left[\begin{array}{ll} i \neq j & \wedge \\ \phi_i \wedge \psi \in \Phi & \wedge \\ \phi_j \wedge \psi \in \Phi & \end{array} \right] \rightarrow CI_{\phi_i \wedge \psi} \cap CI_{\phi_j \wedge \psi} = \emptyset$$

Let $\{\phi_1, \ldots, \phi_k\}$ and $\{\psi_1, \ldots, \psi_l\}$ be two homogeneous discoveries that strongly split the database and such that all $\langle \phi_i \rangle$ and $\langle \psi_j \rangle$ are large. Then there is for each ϕ_i at least one ψ_j such that $\langle \phi_i \rangle \cap \langle \psi_j \rangle \gg \emptyset$ and thus $\langle \phi_i \wedge \psi_j \rangle \in \Phi$. But since both discoveries strongly split the database, there can be at most one. So, $\langle \phi_i \rangle \approx \langle \psi_j \rangle$ and $CI_{\phi_i} \approx CI_{\psi_j}$. In other words, in this case there is essentially only one way to partition the database in a good way.

The fact that the discoveries are not unique is simply caused by the fact that a set of tuples can have more than one description. For example, it could happen that almost all young clients are male and vice versa. In that case the descriptions *age* = *young* and *gender* = *male* are equally good from a theoretical point of view. Not necessarily from a practical point of view. For, it is very well possible that the description *age* = *young* makes sense to a domain expert while *gender* = *male* does not. Hence, both options should be presented to the domain expert.

4.2.4 The Search

If a homogeneous discovery exists that splits the database, it must contain a homogeneous description with the highest associated probability. This suggests a simple algorithm to find such a discovery:

Make a list of homogeneous descriptions as follows:
    find a ϕ that has the maximal associated probability.
    remove $\langle \phi \rangle$ from *db* and add ϕ to the list.
    continue with this process until $\top$ is homogeneous on the remainder of *db*;
Check whether the decision list splits the database.

In other words, we can use the associated probability of a rule as a measure of its quality.

5 What Computer Science offers Data Mining

If Data Mining can be considered as a generalisation of more or less standard statistical techniques, what has Computer Science to offer? In other words, how can Computer Science help to solve the Data Mining problems? In this section we discuss how the two Computer Science parents, AI and database technology help to solve Data Mining tasks with a reasonable performance.

5.1 AI: Search Techniques

Much effort in Machine Learning and in AI in general has been invested in efficient and/or robust search techniques. The range of these often problem specific techniques is far too large to discuss in this paper. Rather, we will concentrate on one technique, viz., *genetic search* [11, 26, 18]. To simplify our discussion, we start with the assumption that we have a Class 1 quality function.

Genetic search, like all genetic algorithms, is defined in analogy with biological evolution, i.e., it is based on the survival of the fittest. It maintains a population of proposed solutions (*chromosomes*) for a given problem. Iteratively, the population undergoes a *simulated evolution:* relative "good" solutions produce offspring, which subsequently replace the "worse" ones.

Each iteration, called a *reproduction cycle*, is performed in three steps. During the selection step a new population is formed from stochastically best samples (with replacement). Then, during the *recombination* step some of the members of the newly selected population are altered. Finally, all such altered individuals are evaluated.

The recombination is based on two operators: *mutation* and *crossover*. Mutation introduces random variability into the population, and crossover exchanges random pieces of both chromosomes in the hope of propagating partial solutions. Schematically, we have the following algorithm:

```
t := 0
initialise P(t)
evaluate P(t)
while (not termination-condition) do
        t:= t+1
        select P(t) from P(t-1)
        recombine P(t)
        evaluate P(t)
od
```

Hence, for the specification of a genetic algorithm for a particular problem we must have the following five components:

1. a "genetic" representation for potential solutions to the problem,
2. a way to create an initial population of potential solutions,

3. an evaluation function that plays the role of the environment, rating solutions in terms of their "fitness",

4. genetic operators that alter the composition of children during reproduction,

5. values for various parameters that the genetic algorithm uses (population size, probabilities of applying genetic operators, etc.).

For our search problem, the items 1, 3, and 4 can be defined as follows. The chromosomes are simply the descriptions in our description language Φ, say slightly modified set-descriptions. More in particular set-descriptions of the form:

$$A_1 \in V_1 \wedge \cdots \wedge A_p \in V_p, \textit{ where } A_j \in \mathcal{A} \wedge V_j \subseteq D_j \wedge V_j \textit{ is finite or } V_j = D_j.$$

In other words all set-descriptions cover all attributes, the cases where $V_j = D_j$ simply cover the attributes on which one doesn't select.

The evaluation function is simply our quality function. The genetic operators can be defined as follows:

Crossover For two descriptions $A_1 \in V_1 \wedge \cdots \wedge A_p \in V_p$ and $A_1 \in W_1 \wedge \cdots \wedge A_p \in W_p$, choose two elements $i, j \in \{1, \ldots, p\}$ and conclude the descriptions:

$$\begin{aligned} &A_1 \in V_1 \wedge \cdots \wedge A_{i-1} \in V_{i-1} \wedge \\ &\quad A_i \in W_i \wedge \cdots \wedge A_j \in W_j \wedge A_{j+1} \in V_{j+1} \wedge \cdots \wedge A_p \in V_p \\ &A_1 \in W_1 \wedge \cdots \wedge A_{i-1} \in W_{i-1} \wedge \\ &\quad A_i \in V_i \wedge \cdots \wedge A_j \in V_j \wedge A_{j+1} \in W_{j+1} \wedge \cdots \wedge A_p \in W_p \end{aligned}$$

Mutation For a description ϕ, choose an $i \in \{1, \ldots, p\}$ and a random $W_i \subseteq D_i$, and replace $A_i \in V_i$ in ϕ by $A_i \in W_i$.

Alternatively, one might execute one step of the Hill-climber algorithm as a mutation step.

The good parameters for the algorithm can hardly be defined in advance, they have to be found by experimentation. It is well-known, however, that the population size should be relatively large, say a few hundred, and that quite some iteration steps, again say a few hundred, are needed before such a system will converge.

If we consider Class 2 or Class 3 quality functions we have to deal with *genomes*, i.e., sets of descriptions rather than with descriptions. In principle, this only changes the possible genetic operators. For example, genomes can switch complete chromosomes or they can pair their chromosomes and combine these pairs as above. In mutation, one might also consider simply dropping chromosomes are change one of the chromosomes with a completely new, arbitrary, chromosome. This freedom of choice implies that we simply should take

a larger collection of operations with a varying probability of being actually chosen. For more information and other possible choices, see [14].

As explained at the beginning of this subsection, there are many more search algorithms than genetic search. One of the distinct advantages of genetic search, however, is its inherent parallelism. All recombinations and all quality evaluations can be done in parallel. This promises a considerable speed-up of the process.

5.2 Databases: Handling massive volumes of data

Large in Statistics is a different term from large in Databases. In statistics a large sample consists of a few thousand records. Large Databases have hundreds of thousands if not millions of records. Moreover, the number of possibly relevant attributes in Data Mining count easily up to 50 or 60. Using samples to cope with these large volumes of data means invariably a loss of resolution in our search. It is far easier for a group of 20.000 to stand out significantly in a crowd of a million than it is for a group of 20 to stand out significantly in a crowd of a thousand.

However, standard database technology is not the answer to the problem of massive amounts of data either. For, the discovery process queries the database severely, since:

1. dbms's are tuned to a variety of uses including transactions,
2. discovery is in principle a read-only process on the database; having access, during the search, to the newest data does no improve the quality of the information significantly,
3. during the search, old results can often be reused,

it is profitable to have a knowledge discovery tool with its own data-server, geared specially towards discovery.

Such a data server is a dbms-kernel tailored for data mining purposes. That is, it contains no transaction management functionality nor write protection. In fact, one can only store new, derived, data, one cannot update data.

What it does contain however, are various mechanisms to speed up query processing as much as possible. For data mining causes an avalanche of queries posed to the database as is witnessed by the description of genetic search.

First and foremost, the data server is a parallel system, since it has been proven that parallel systems can answer bursts of queries far more efficiently than mono-processor systems.

Secondly, it contains a query-optimisation module that optimizes queries both statically and dynamically. Static optimisation is rewriting a query into the most efficient form given database characteristics. Dynamic optimisation means that the query is processed as efficiently as possible given all the other queries that are processed concurrently, [41].

Finally, it contains a browsing optimization module. Many queries in a data mining search are related. That is, later queries can be executed much more efficiently if some previous results are stored temporarily than when they are executed against the complete database. The browsing optimization module tries to optimize query processing by storing such intermediate results [20].

Another aspect of efficient query processing is using the most suited data structure. Therefore, the data-server can dynamically adapt its data-layout (in main memory) to suit the current search process as much as possible [19].

All these techniques are either well-studied in database research or are currently under vigorous investigation

6 Foundations of Data Mining: Theory and Experiment

Data Mining becomes a mature tool for the exploratory data analyst only if one can trust the results. In other words, Data Mining should be given sound mathematical foundations. These foundations comprise two aspects, viz., the quality functions and convergence of the search process.

The primary goal of Data Mining is the discovery of strategic information. In other words, the results will be used to predict the, near, future. The quality functions should be chosen in such a way that such extrapolations are, at least statistically, valid.

Given a description language, a quality function, and a database state, we get a so called *fitness landscape*; a multi dimensional graph of the quality function over the descriptions. The task is to list the descriptions of high quality.

Most often, the size of the set of descriptions makes an exhaustive search over the fitness landscape intractable, as discussed above. Heuristic searches are the only viable option. The immediate question is then, of course, how well do the results found by heuristic search compare with the, almost hypothetical, results of exhaustive search.

The only way in which a heuristic search can consistently outperform random search is by exploiting the shape of the fitness landscape. The shape of this landscape is governed by the, perhaps implicit, structure in the set of descriptions and the behaviour of the quality function on this structure.

In other words, to design good search algorithms one should study the structure of the set of descriptions, e.g., does it form a lattice, is it a topological space or even a metrical space? Moreover, one should study the behaviour of quality functions on this structure, e.g., are they continuous or monotonic.

6.1 What Statistics might offer

Since we have argued that Statistics should be considered as one of the parents of Data Mining, it is only natural to ask what Statistics might offer towards the resolution of these two foundational problems.

For the first problem, the quality function, this is rather obvious. If only because of the quality functions defined for Projection Pursuit. More in general,

if we want statistically valid results, we should use Statistics to test the validity of our results.

For the second problem, the convergence problem, the situation is less clear. There are some results on convergence properties for genetic algorithms in a framework of stochastic processes [33, 34]. However, these results apply to the case of an infinite population size in continuous space. To make this results usefull as foundations for Data Mining, they should be extended to finite population sizes in mixed continuous and discrete spaces.

For a different type of search algorithms, viz., Neural Networks, there are more results. In [10], for example, techniques from Statistical Mechanics are used to analyse the behaviour of Neural Networks. Moreover, in [17], the authors derive the *Information Geometry* of *Boltzmann Machines*, a special class of Neural Networks, along the lines of [16]. These results still depend on continuous space, but no longer do they depend on an infinite population size. The problem with these results, however, is that it far from clear how Neural Nets can be used in a search for descriptions.

6.2 Data Surveyor: experimental guidance

The development of the mathematical underpinnings of Data Mining cannot be the result of theoretical studies alone. Consider, e.g., the quality functions. Although different functions may be more or less mathematically equivalent, their usablity in practice might differ considerably. Similar remarks are valid for the convergence problem. One can make synthetic databases in which known results are hidden. By testing the search strategies on such examples, some insight in the convergence process may be gained.

To get the experimental guidance for the theoretical development, a data mine tool called *Data Surveyor* is currently under development at CWI, [13]. Currently, it has a two-level architecture consisting of a data server on top of which the Surveyor kernel is executed. In the near future, it will be extended to a three level architecture.

The bottom layer will still consist of the data server. The middle layer will consist of a set of different search modules. The top layer will be the user-interface with which the user can formulate data mining tasks and guide the search task. The rationale for have several different search modules is twofold. First, it is very well posible that different algorithms perform better for different data mining problems. Second, by using different search algorithms on the same real world database states more insight in the convergence properties of the various algorithms can be gained.

Acknowledgements: The author wishes to thank Marcel Holsheimer, and Johan van den Akker for stimulating discussions and constructive criticism on an earlier version of this paper.

References

1. Martin Anthony and Norman Biggs. *Computational learning theory: an*

introduction, volume 30 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1992.
2. A. Borgida, T.M. Mitchell, and K. Williamsson. Learning improved integroty constraints and schemas from exceptions in databases and knowledge bases. In J. Mylopoulos M.L. Brodie, editor, *On Knowledge Base Management Systems: Integrating Artificial Inteligence and Database technologies*. Springer-Verlag, 1986.
3. James P. Delgrande. Formal limits on the automatic generation and maintainance of integrity constraints. In *Proc. 6th ACM Sigact-Sigmod-Sigart Symposium on Principles of Database Systems*, 1987.
4. P. Diaconis. Projection pursuit for discrete data. Technical Report 198, Stanford University, 1983.
5. Benjamin S. Duran and Patrick L. Odell. *Cluster Analysis, A Survey*. Lecture Notes in Economics and Mathematical Systems, vol 100. Springer-Verlag, 1974.
6. Usama M. Fayyad and Ramasamy Uthurusamy, editors. *AAAI-94 Workshop Knowledge Discovery in Databases*, Seattle, Washington, 1994.
7. William Feller. *An introduction to probability theory and its applications, Vol 1*. Wiley, 1950.
8. J.H. Friedman and W. Stuetzle. Projection pursuit regression. *Journal of the American Statistical Association*, 76:817–823, 1981.
9. J.H. Friedman and J.W. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computing*, C-23:881–889, 1974.
10. John Hertz, Anders Krogh, and Richard G. Palmer. *Introduction to the Theory of Neral Networks*. Santa Fe Institute Lecture Notes vol 1. Addison-Wesley, 1991.
11. John H. Holland. *Adaptation in natural artificial systems*. University of Michigan Press, Ann Arbor, 1975.
12. John H. Holland, Keith J. Holyoak, Richard E. Nisbett, and Paul R. Thagard. *Induction: processes of inference, learning and discovery*. Computational models of cognition and perception. MIT Press, Cambridge, 1986.
13. Marcel Holsheimer, Martin Kersten, and Arno Siebes. Data surveyor: Searching the nuggets in parallel. In Piatetsky-Shapiro and Frawley [32], chapter 4.
14. Marcel Holsheimer and Arno P.J.M. Siebes. Data mining: the search for knowledge in databases. Technical Report CS-R9406, CWI, January 1994.
15. Peter J. Huber. Projection pursuit. *The Annals of Statistics*, 13(2):435–475, 1985.
16. Shun ichi Amari. *Differential-Geometrical Methods in Statistics*. Lecture Notes in Statistics, vol. 28. Springer-Verlag, 1985.
17. Shun ichi Amari, Koji Kurata, and Hiroshi Nagaoka. Information geometry of boltzmann machines. *IEEE transactions on Neural Networks*, 3(2):260–271, 1992.
18. Cezary Z. Janikow. *Inductive Learning of Decision Rules from Attribute-*

Based Examples: A Knowledge-Intensive Genetic Algorithm Approach. PhD thesis, University of North Carolina at Chapel Hill, 1991.
19. Martin L. Kersten. Goblin: A DBPL designed for Advanced Database Applications. In *2nd Int. Conf. on Database and Expert Systems Applications, DEXA'91*, Berlin, Germany, August 1991.
20. Martin L. Kersten and Michiel de Boer. Query optimization strategies for browsing sessions. In *Proc. IEEE Int. Conf. on Data Engineering*, Houston, 1994.
21. Jyrki Kivinen and Heikki Mannila. Approximate dependency inference from relations. In *Proc. 4th Int. Conf. on Database Theory*, 1992.
22. Willi Klösgen and Jan Zytkow. Machine discovery terminology. In Fayyad and Uthurusamy [6], pages 463–473.
23. Yves Kodratoff and Ryszard S. Michalski, editors. *Machine Learning, an Artificial Intelligence approach*, volume 3. Morgan Kaufmann, San Mateo, California, 1990.
24. Heikki Mannila and Kari-Jouko Räihä. Algorithms for inferring functional dependencies from relations. *Data and Knowledge Engineering*, 12:83–99, 1994.
25. K.V. Mardia, J.T. Kent, and J.M. Bibby. *Multivariate Analysis.* Probability and Mathematical Statistics. Academic Press, 1979.
26. Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Artificial Intelligence. Springer-Verlag, 1992.
27. Ryszard S. Michalski, Jaime G. Carbonell, and Tom M. Mitchell, editors. *Machine Learning, an Artificial Intelligence approach*, volume 1. Morgan Kaufmann, San Mateo, California, 1983.
28. Ryszard S. Michalski, Jaime G. Carbonell, and Tom M. Mitchell, editors. *Machine Learning, an Artificial Intelligence approach*, volume 2. Morgan Kaufmann, San Mateo, California, 1986.
29. M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry.* MIT Press, 1969.
30. J.C. Mitchell. Inference rules for functional and inclusion dependencies. In *Proc. 2nd ACM Sigact-Sigmod Symposium on Principles of Database Systems*, 1983.
31. Gregory Piatetsky-Shapiro and William J. Frawley, editors. *Knowledge Discovery in Databases.* AAAI Press, Menlo Park, California, 1991.
32. Gregory Piatetsky-Shapiro and William J. Frawley, editors. *Knowledge Discovery in Databases*, volume II. MIT Press, Menlo Park, California, forthcoming.
33. Xiaofeng Qi and Francesco Palmieri. Theoretical analysis of evolutionary algorithms with an infinite population size in continuous space part i: Basic properties of selection and mutation. *IEEE transactions on Neural Networks*, 5(1):102–119, 1994.
34. Xiaofeng Qi and Francesco Palmieri. Theoretical analysis of evolutionary algorithms with an infinite population size in continuous space part ii: Analysis of the diversification role of crossover. *IEEE transactions on Neural*

Networks, 5(1):120–129, 1994.
Ronald L. Rivest. Learning decision lists. *Machine Learning*, 2:229 – 246, 1987.
F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 96(6):386–408, 1958.
Richard Segal and Oren Etzioni. Learning decision lists using homogeneous rules. In *Proceedings of the 12th National Conference on Artificial Intelligence*, pages 619–625. AAAI/MIT Press, 1994.
Jude W. Shavlik and Thomas G. Dietterich, editors. *Readings in Machine Learning*. Morgan Kaufmann, 1990.
Arno Siebes. Homogeneous discoveries contain no surprises: Inferring risk-profiles from large databases. In Fayyad and Uthurusamy [6], pages 97 – 108.
J.W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977.
Carel Arie van den Berg. *Dynamic Query Processing in a Parallel Object-Oriented Database System*. PhD thesis, Twente University, 1994.