Detecting Interesting Instances

Katharina Morik

Univ. Dortmund, Computer Science Department, LS VIII
morik@ls8.informatik.uni-dortmund.de,
http://www-ai.cs.uni-dortmund.de

Abstract. Most valid rules that are learned from very large and high dimensional data sets are not interesting, but are already known to the users. The dominant model of the overall data set may well suppress the interesting local patterns. The search for interesting local patterns can be implemented by a two step learning approach which first acquires the global models before it focuses on the rest in order to detect local patterns. In this paper, three sets of interesting instances are distinguished. For these sets, the hypothesis space is enlarged in order to characterize local patterns in a second learning step.

1 Introduction

Most valid rules that are learned from very large and high dimensional data sets are not interesting, but are already known to the users. For instance, we have learned from a very large data set on cars and their warranty cases that the production date of a car is preceding its sales date, which is what we expected [14]. However, there were some exceptions to the rule and these are interesting. Which customers order cars before they are offered? Are the exceptions typing errors? The decision can only be made by domain experts. Either we use the outliers for data cleaning or we find interesting instances by learning a general rule and outputing its exceptions. This idea has been put forward by Vladimir Vapnik [7]. In some data bases, we are looking for fraudulent use of cellular phones as has been reported by Fawcett and Provost [4]. There, finding the instances that are exceptions to generally reliable rules corresponds to finding the criminal indiviuals. In other applications we learn general models for customer behavior, but are interested in exceptionally lucrative customers. Again, these customers are found by collecting the exceptions to accepted rules. Hence, one reason for a set of instances being interesting is that they are exceptions to a general model.

Many learning algorithms aim at covering all positive examples. However, there might be some instances that cannot be covered by any rule. Why do they not fit into any general model? What makes them so exceptional? Maybe these uncovered instances are the most interesting ones and lead us to a local pattern.

A third phenomenon can again be illustrated by the database of cars and their parts. We encountered valid rules not being learned. For instance, it could not be learned that each car has at least one axle. Since we know that this rule should be

verified by the data, we inspected the evidence against the rule and found many missing attribute values in the database. We could clean the data base on the basis of the refutation of valid rules. In a medical application, we looked for effects of drugs. We knew from the domain expert that a particular drug should decrease blood pressure. However, the rule was not learned that after the intake of the drug, the blood pressure showed a clear level change. Too many patients either showed no level change or their blood pressure even increased. We inspected the patient records that formed negative evidence against the rule. There were patients with arhythmic heart beat, patients where the blood pressure varied more than usual, patients where the rate of increasing blood pressure decreased but not the blood pressure itself. On one hand, the hypothesis can be made more precise: the drug decreases the rate of increasing blood pressure. On the other hand, the variance of blood pressure and its relation to arythmic heart beat forms a local pattern. Again, the user has to decide whether the data are wrong or the instances allow us to form an interesting local pattern. Hence, another reason for a set of instances being interesting is that they are negative evidence for a valid rule.

Without the claim of completeness, we may state that there are three reasons for a set of instances to be interesting:

- exceptions to an accepted general rule
- examples that are not covered by any valid rule
- negative examples that prevent the acceptance of a rule.

The acceptance of a rule hypothesis depends on the acceptance criterion or quality measure. In the preceding paragraphs we argued on the basis of a general quality measure which combines positive and negative evidence for and against the validity of a rule in some manner. Such a measure is tailored to finding general models. The pattern detection then means to find the deviations from the general rules. There is an alternative, though. The quality measure can be tailored to finding interesting sets of instances directly. Subgroup detection is the learning task of finding parts of the overall instance space, where the probability distribution differs significantly from the global one [12]. The subgroups can be considered local patterns which are selected by a quality criterion that implements the refutation of the null hypothesis, namely that the distribution in the subgroup equals that in the entire population. A logical approach to subgroup detection has been developed by Stefan Wrobel [18]. The same subgroups can be found by the collection of negative or uncovered examples and by a well-suited quality measure for subgroup detection. Hence, in principle the two approaches fall into the same category. The algorithms, however, are different. Subgroup detection explores the overall search space and uses pruning methods and language restrictions in order to become tractable. The two-step approach allows to use different search spaces and learning methods for the general model and the local patterns. Since the number of negative or uncovered examples is comparatively small, a larger search space can be explored by a more demanding procedure.

In this paper, we describe an inductive logic programming algorithm which first finds all valid rules in a very restricted search space. These rules are used to

collect interesting instances as described above. The second learning step then finds definitions for the small samples in an enlarged search space. There are three ways to enlarge the hypothesis space for the learning algorithm:

- the synactic language restriction of hypotheses is weakened
- the dimensionality of the examples is increased, i.e. more (possibly finer grained) attributes are taken into account
- a more expensive learning strategy is chosen

We investigate these three options on the different sets of interesting instances. For illustration we use the movie database which is freely available in the internet (www.imdb.com). The paper is organised as follows. First, we describe the learning algorithm RDT/DM and indicate the size of the hypothesis space depending on the syntactic language restrictions, the dimensionality of examples and the learning strategy. Second, we shortly present results of learning global models (i.e. all valid rules) and discuss some quality measures for accepting hypotheses. Third, the sets of interesting instances and the detection of local patterns is illustrated by our movie application. A discussion relating the approach to others concludes the paper.

2 Inductive logic programming using RDT/dm

The rule learning task has been stated within the inductive logic programming (ILP) paradigm by Nicolas Helft [9] using the notion from logic of minimal models of a theory $\mathcal{M}^+(Th) \subseteq \mathcal{M}(Th)$. Of course, in general there may well exist many minimal models. However, for first-order logic there exist restrictions such that there is exactly one minimal model to a theory.

Definition 1. *(Minimal model) An interpretation I is a model of a theory Th, $\mathcal{M}(Th)$, if it is true for every sentence in Th. An interpretation I is a* minimal model *of Th, written $\mathcal{M}^+(Th)$, if I is a model of Th and there does not exist an interpretation I' that is a model of Th and $I' \subset I$.*

Rule learning
Given observations $\mathcal{E}$ in a representation language $\mathcal{L}_\mathcal{E}$ and background knowledge $\mathcal{B}$ in a representation language $\mathcal{L}_\mathcal{B}$,
find the set of hypotheses $\mathcal{H}$ in $\mathcal{L}_\mathcal{H}$, which is a (restricted) first-order logic, such that

(1) $\mathcal{M}^+(\mathcal{B} \cup \mathcal{E}) \subseteq \mathcal{M}(H)$ (validity of H)
(2) for each $h \in \mathcal{H}$ there exists $e \in \mathcal{E}$ such that $\mathcal{B}, \mathcal{E} - \{e\} \not\models e$ and $\mathcal{B}, \mathcal{E} - \{e\}, h \models e$ (necessity of h)
(3) for each $h \in \mathcal{L}_\mathcal{H}$ satisfying (1) and (2), it is true that $\mathcal{H} \models h$ (completeness of $\mathcal{H}$)
(4) There is no proper subset $\mathcal{G}$ of $\mathcal{H}$ which is valid and complete (minimality of $\mathcal{H}$).

The first sentence states that the minimal model of the observations and the background knowledge must be a subset of the model of the learned rules. There is one interpretation for learning result, examples, and background knowledge. This sentence expresses the *correctness* of the learning result. The other sentences approximate the completeness and minimality of theories. Since Kleene's proof that we cannot find the minimal set of axioms to a set of observations (facts), the minimality had to be reduced to not including rules that could be removed without any loss. It is still possible that by combining some rules into one rule, the set of rules that is valid and complete is smaller than the learning result. We cannot escape this possible shortcoming of too large a learning result. What we can escape is having redundant rules in the learning set that do not contribute to covering observations. This is stated in (2) and (4). The third sentence states that all rules that are valid and necessary in the sense of (2) are included in the learning result. Again, this property does not exclude that there are more elegant and concise rules that are not learned, since this would contradict Kleene's proof. It only states that all correct rules that are necessary for covering examples are included in the learning result. This learning task has been taken up by several ILP researchers, e.g., [10], [5], [3]. Rule learning is more difficult than the concept learning task because it is possible that all results of concept learning could also be found by rule learning, but not vice versa [11]. Since the learning task is hard in terms of computational complexity, it must be constrained. Constraining the hypothesis language $\mathcal{L}_\mathcal{H}$ to less expressive power than first-order logic is the key to making rule learning efficient. The clear learnability border has been shown for restrictions of first-order logic [11].

In order to restrict the hypothesis space, RDT/DM uses a declarative specification of the hypothesis language, just as its predecessor RDT does [10]. The specification is given by the user in terms of rule schemata. A rule schema is a rule with predicate variables (instead of predicate symbols). In addition, arguments of the literals can be designated for learning constant values. Examples of rule schemata are:

$mp1(P1,P2,P3) : P1(X1)\&P2(X1) \rightarrow P3(X2)$
$mp2(P1,P2,P3) : P1(X1,X2)\&P2(X2) \rightarrow P3(X1)$
$mp3(P1,P2,P3,P4) : P1(X1)\&P2(X1)\&P3(X1) \rightarrow P4(X1)$
$mp4(C,P1,P2,P3) : P1(X1,C)\&P2(X1) \rightarrow P3(X1)$

Where the first and third rule schema restricts $\mathcal{L}_\mathcal{H}$ to propositional learning, the second and fourth schema expresses a simple relation. In the last schema, the second argument of the first literal is a particular constant value that is to be learned. This is indicated in the meta-predicate by C.

For hypothesis generation, RDT/DB instantiates the predicate variables and the arguments that are marked for constant learning. The arity of a predicate as well as the sorts of arguments are taken into account, so that only sort-correct and fitting predicates instantiate the predicate variables. A fully instantiated rule schema is a rule. An instantiation is, for instance,

mp1(america, drama,top) $america(X1)\&drama(X1)\& \rightarrow top(X1)$

The rule states that a movie that was produced at the American continent and is of genre "drama" belongs to the top hundred of the movie rating as given by the movie database.

The rule schemata are ordered by generality: for every instantiation of a more general rule schema there exist more special rules as instantiations of a more special rule schema, if the more special rule schema can be instantiated at all. Hence, the ordering of rule schemata reflects the generality ordering of sets of rules. This structure of the hypothesis space is used while doing top-down search for learning. If a rule is learned its specialization w.r.t. the generality ordering of rule schmeata will not be tried, since this would result in redundant rules. Hence, RDT/DM delivers most general rules. The user writes the rule schemata in order to restrict the hypothesis space. The user also supplies a list of the predicates that can instantiate predicate variables (i.e. determines the dimensionality of the examples). This list can be a selection of all predicates in $\mathcal{L}_\mathcal{E}$.

Another kind of user-given control knowledge is the acceptance criterion. It is used to test hypotheses. The user composes an acceptance criterion for a rule $premise \rightarrow conclusion$ out of four terms which count frequencies in the given data:

$pos(h)$ the number of supporting instances: $fr(premise \wedge conclusion)$;
$neg(h)$ the number of contradicting instances: $fr(premise \wedge \neg conclusion)$;
$concl(h)$ the number of all tuples for which the conclusion predicate of the hypothesis holds: $fr(conclusion)$; and
$negconcl(h)$ the number of all instances for which the conclusion predicate does not hold: $fr(\neg conclusion)$.

This general form for criteria directly corresponds to interestingness criteria as presented in [8]. Using the four terms, one can easily express different acceptance criteria. A typical one (similar to that of APRIORI [1]) is:

$$\frac{pos(h)}{concl(h)} - \frac{neg(h)}{concl(h)} \geq 0.8$$

The acceptance criterion can also be written in a Bayesian manner. If there are two classes for the conclusion predicate (e.g., top 100 movies and bottom 100 movies), the following criterion is similar to the requirement that the a posteriori probability must equal or exceed the a priori probability:

$$\frac{pos(h)}{pos(h) + neg(h)} \geq \frac{concl}{concl + negconcl}$$

RDT/DM is similar to RDT/DB [2] in that it directly accesses the ORACLE database system. It is a re-implementation using the JAVA programming language. The main difference is the search stratey in the hypothesis space $\mathcal{L}_\mathcal{H}$. Where RDT/DB performs a breadth-first search which allows safe pruning, RDT/DM performs a depth-first search in order to minimize the database accesses and to exploit already selected subsets of records. For each accepted hypothesis h its instances $ext(h)$ are collected as $pos(h)$ (support) and $neg(h)$

(outliers). The uncovered instances of all rules $S - ext(H)$ are also stored [1]. The data dictionary of the database system contains information about relations and attributes of the database. This information is used in order to map database relations and attributes automatically to predicates of RDT's hypothesis language. For hypothesis testing, SQL queries are generated by the learning tool and are sent to the database system. The counts for $pos(h), neg(h), concl(h)$, and $negconcl(h)$ are used for calculating the acceptance criterion for fully instantiated rule schemata.

The size of the hypothesis space of RDT/DM does not depend on the number of database records, but on the number of rule schemata, r, the number of predicates that are available for instantiations, p, and the maximal number of literals of a rule schema, k. For each literal, all predicates have to be tried. Without constant learning, the number of hypotheses is $r \cdot p^k$ in the worst case. As k is usually a small number in order to obtain understandable results, this polynomial is acceptable. Constants to be learned are very similar to predicates. For each argument marked for constant learning, all possible values of the argument (the respective database attribute) must be tried. We write c for the number of constants that are marked for learning in a rule schema. Let i be the maximal number of possible values of an argument marked for constant learning; then, the hypothesis space is limited by $r \cdot (p \cdot i^c)^k$. The size of the hypothesis space determines the cost of hypothesis generation. For each hypothesis, two SQL statements have to be executed by the database system. These determine the cost of hypothesis testing.

2.1 The movie database and the chosen hypothesis spaces

The movie database `imdb.com` stores millions of movies with their genre, actors, producers, directors, the production country, keywords, and year of publication. For actors and directors tables with further information about them exist. What makes the database of interest to us is the voting of movie visitors and the resulting ranking of movies into the top 100 movies and the bottom 100 movies. For our experiments we selected only the top and bottom 100 movies. The first task was to learn all valid rules about the top 100 movies. Hence, we reduced the first learning task more or less to concept learning, here, the classification of top movies. This exercise in automatic modeling or learning global models is meant to be the basis for detecting local patterns in the second step. Hence, the restriction should not be a problem.

In order to map database relations and attributes to predicates, the system offers a tool which constructs predicates using the data dictionary of the database and exploiting foreign key relations. In our experiments we used two mappings.

Mapping 1: For each relation R with attributes $A_1, \ldots, A_n$, where the attributes $A_j, \ldots, A_l$ are the primary key, for each $x \in [1, \ldots, n] \backslash [j, \ldots, l]$ a predicate $rn\_AX(A_j, \ldots, A_l, A_x)$ is formed, where AX is the string of the attribute name.

[1] S is the set of all instances.

Since the primary key of the relations is a single attribute, we get two–place predicates. The number of predicates is bound by the number of relations times the maximal number of attributes of a relation (without key attributes).

The other mapping reduces the expressiveness to propositional logic. Of course, this means that the size of the hypothesis space is reduced.

Mapping 2: For each attribute A_i which is not a primary key and has the values $a_1, \ldots, a_n$ a set of predicates $rn\_AI\_ai(A_j, \ldots, A_l)$ are formed, $A_j, \ldots, A_l$ being the primary key.

Mapping and rule schemata together determine the size of the hypothesis space. For learning the general model of top movies, we used the first two rule schemata shown above and the first and second mapping. 22 predicates were selected for learning. In the propositional case this corresponds to 22 dimensions of movies. In the worst case only $2 \cdot 22^2 = 968$ hypotheses need to be tested [2]. For the detection of local patterns in selected instance sets, we enlarged the hypothesis space by including the third rule schemata shown above and by forming more predicates. Changing the learning strategy to constant learning using the fourth rule schema further enlarges the hypothesis space.

3 Learning the global model for movie ranking

Given the top 100 movies classified $top(MovieID)$ and the bottom 100 movies classified $not(top(MovieID))$, we instantiated the conclusion predicate of the first three meta-predicates by $top(X)$. Each movie was characterised by predicates with arity 1 for genre and production country or continent and by predicates with arity 2 for the relation between a movie and any actor of it as well as the relation between a movie and its director. Actors and directors were described by a one–place predicate stating, whether the actor performed in or the director made at least 2 of the top 100 movies, namely $topActor(X)$ and $topDir(X)$. Similarly, $noBotActor(X)$ respectively $noBotDir(X)$ states that an actor or director was never involved in one of the bottom movies. The acceptance criterion was set to

$\frac{pos(H)}{concl(H)} - \frac{neg(H)}{concl(H)} \geq 0.3$

These are the learned rules:

$h_1 : usa(X) \& drama(X) \rightarrow top(X)$
$h_2 : director(X,Y) \& topDir(Y) \rightarrow top(X)$
$h_3 : actor(X,Y) \& topActor(Y) \rightarrow top(X)$
$h_4 : director(X,Y) \& noBotDir(Y) \rightarrow top(X)$
$h_5 : actor(X,Y) \& noBotactor(Y) \rightarrow top(X)$

[2] Since the conclusion predicate is always instantiated by $top(X)$, the conclusion literal is not counted.

Of course, the rules do not make sense for cineasts, but only for the ranking of the movie database in the excerpt we used. The ranking clearly favours American movies: 68 of the top movies and 82 of the bottom movies are American. Taking the genre into account, 40 of the top movies are American dramas, but only 7 of the bottom movies. The tolerant acceptance criterion allows us to collect 7 outliers. 15 top movies are uncovered by the rules. These are candiates for local patterns hidden by the overwhelming dominance of American movies.

We wonder whether another acceptance criterion would allow us to describe the uncovered instances already in the first step. It should be a criterion which focuses on subgroups within the data. Hence, we applied the criterion based on the Binomial test heuristic:

$\sqrt{\frac{pos(h)+neg(h)}{concl+negconcl}} \cdot \mid \frac{pos(h)}{pos(h)+neg(h)} - \frac{concl}{concl+negconcl} \mid \geq 0,05$

The first factor weights the size of $ext(h)$ in relation to the samples size and the second factor compares the distribution in $ext(h)$ with that of the overall population. A theoretical investigation of quality functions as this one and its distinction from averaging functions can be found in [16]. Here, we identify the data set with the overall population and consider $ext(h)$ a sample. The significance of a difference in the distribution is determined with respect to the null hypothesis. In fact, additional rules are found that cover some of the previously uncovered instances:

$h_6 : italy(X) \& drama(X) \rightarrow top(X)$
$h_7 : denmark(X) \& drama(X) \rightarrow top(X)$

However, many more rules were also found and cover instances redundantly. If we narrow the pruning criterion, we get exactly the rules learned in the first step. Hence, we either receive too many rules, among them senseless ones, or we end up with the global rules.

4 Interesting instances in the movie database

Our aim is to detect local patterns. We have learned general rules in order to get rid of the dominant process. We now want to inspect the remaining instances using a larger hypothesis space. We used additionally the third rule schema and changed the acceptance criterion to

$\frac{pos(h)-neg(h)}{pos(h)+neg(h)} \geq 0.6$

We enlarged the number of predicates for learning. The table "keywords" contains up to 200 words for each movie. We formed 286 one-ary predicates out of these. That is, for the few interesting instances we offered a hypothesis space of a hundred million hypotheses ($3 \cdot 330^3 = 107811000$).

The first potentially interesting set of instances are the *outliers* to accepted rules. Do these American dramas which are ranked within the bottom movies have something in common? We switched the learning strategy to learning constant values for actors and directors. The possible values for the argument marked for constant learning are 2544. Hence, the hypothesis space consists

of about 10^{17} hypotheses, $4 \cdot (330 \cdot 2544^1)^3$. 6 rules were found which characterize bottom movies covered by general rules for top movies. Three rules blamed certain actors which performed in none of the top movies but only in the bottom ones[3]. Another rule found the combination of "drama" and "musical", two other rules refer to directors who made bottom movies or top movies. The rules covered only about half of the exceptions. Decreasing the threshhold value of the acceptance criterion led to 34 actors who play in bottom movies and to some keywords for American bottom films, among them "police", "screwball", and "independent". However, more rules were learned than there are exceptions. Decreasing the selectivity of the criterion leads to arbitrary rules. Hence, using Occam's razor, we prefer the 6 rules and list the remaining instances.

The second possibly interesting set of instances are the *uncovered examples*. With the enlarged set of rule schemata but without the additional predicates, h_6 and $h_7$were found, indicating the subgroups of Italian and Danish dramas. A third learned rule states that those European movies are top, which are classified as "drama" and as "romance" as well. However, these rules still leave some instances uncovered. Using the enlarged set of predicates 7 rules were found. Rule h_6 covering particularly the famous movies "bicycle thieves" (1948) and "La vita e bella"(1997) is again found. The keywords "independent film" or "family" make a European movie top. Independent films such as the Danish dogma movies that led to rule h_7 are covered. The keyword "love" together with the genre "drama" also characterise the uncovered instances. Funny enough, the keyword "bicycle" is also characteristic for European top movies as well as for those of genre "drama". All rules do make sense and it is not surprising that 6 of the 7 rules deal with European movies. These seem to be the ones that are dominated by the American movie population[4].

The third collected set of instances was $neg(h)$ where h was rejected in the first learning step. In our movie application, this set of instances was quite large (85) and uninteresting. It shows that the general model is quite appropriate and learning the global model excludes successfully senseless rules. No further learning was performed on this sample.

5 Conclusion

Instance selection usually means to reduce the original data set such that the reduced set allows to find the general model underlying the original data. Sampling is its most popular technique, search for critical points or the construction of prototypes are others [13].The detection of outliers in such a context aims at easing learning from the reduced set [6].In this paper, we are concerned with the opposite goal of finding instance sets that allow us to find local patterns

[3] What the learning algorithm did not find because we did not apply constant learning to movie titles, is that these actors play in "Police Academy"...

[4] It is a pity that we could not find the nationalities of the voting population in the database, because it could well be that the aubgroups actually do not refer to the producing nation of the movies but to the nation of the voters.

that contrast the general model. The instance sets do not mirror the overall distribution as do (random) samples. The detection of outliers in this context means to focus on small abnormal subsets of the data. Similarly, the collection of uncovered instances focuses on subsets which do not fit the general model. These subsets of data are candidates for being the extension of local patterns. We consider local patterns the effect of behavior of a distinct, small population that is dominated by the large population. In other words, we assume a mixture distribution made of distinct components, where one is overwhelmingly large. Therefore, usual methods like, e.g., Markov switching models or models of finite mixture distributions do either not detect these small groups or detect overly many small groups.

The two-step approach presented here, excludes the large population in its first step and hence prevents us from finding too many subgroups. In spirit, our first experiment on outliers is similar to the demand-driven concept formation of Stefan Wrobel [17]. In an incremental setting, he collected the exceptions of strongly supported rules and learned their definition. The hypothesis space remained the same for rule learning and defining new concepts over exceptions. He did not deal with uncovered instances. In our movie application, the first step characterises American movies (the majority group of the population). The collection of uncovered positive examples worked particularly well. The second step then concentrates on European movies (the minority group of the population). From an algorithmic point of view, the two-step procedure allows us to change the search space and search method within a learning run. The first step processes the overall data set using a small hypothesis space, i.e. it estimates the overall distribution on the basis of low-dimensional instances and a tight restriction of hypotheses' complexity. The second step processes the selected abnormal instances on the basis of high-dimensional instances and more complex hypotheses. Here, we applied the same learning algorithm in both steps. Further work could apply complex algorithms in the second step such as the *lgg* [15] in order to bottom-up generalise interesting instances. We have illustrated our approach by the movie database with an excerpt of four tables. Further experiments and – more important – theoretical considerations are still needed.

Acknowledgment Dirk Münstermann has developed RDT/DM and has run the experiments on the movie database. His diploma thesis which explains the new search strategy and data management is forthcoming.

References

1. Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, chapter 12, pages 307–328. AAAI Press/The MIT Press, Cambridge Massachusetts, London England, 1996.
2. Peter Brockhausen and Katharina Morik. Direct access of an ILP algorithm to a database management system. In Bernhard Pfaringer and Johannes Fürnkranz,

editors, *Data Mining with Inductive Logic Programming (ILP for KDD)*, MLnet Sponsored Familiarization Workshop, pages 95–110, Bari, Italy, jul 1996.
3. L. DeRaedt and M. Bruynooghe. An overview of the interactive concept–learner and theory revisor CLINT. In Stephen Muggleton, editor, *Inductive Logic Programming.*, number 38 in The A.P.I.C. Series, chapter 8, pages 163–192. Academic Press, London [u.a.], 1992.
4. T. Fawcett and F. Provost. Adaptive fraud detection. *Data Mining and Knowledge Discovery*, 1(3):291 – 316, 1997.
5. P. A. Flach. A framework for inductive logic programming. In Stephen Muggleton, editor, *Inductive Logic Programming.*, number 38 in The A.P.I.C. Series, chapter 9, pages 193–212. Academic Press, London [u.a.], 1992.
6. Dragan Gamberger and Nada Lavrac. Filtering noisy instances and outliers. In Huan Liu and Hiroshi Motoda, editors, *Instance Selection and Construction for Data Mining*, pages 375 – 394. Kluwer, 2001.
7. Isabelle Guyon, Nada Matic, and Vladimir Vapnik. Discovering informative patterns and data cleaning. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, chapter 2, pages 181–204. AAAI Press/The MIT Press, Menlo Park, California, 1996.
8. David Hand, Heikki Mannila, and Padhraic Smyth. *Principles of Data Mining.* Massachusetts Institute of Technology, 2001.
9. Nicolas Helft. Inductive generalisation: A logical framework. In *Procs. of the 2nd European Working Session on Learning*, 1987.
10. J.-U. Kietz and S. Wrobel. Controlling the complexity of learning in logic through syntactic and task–oriented models. In Stephen Muggleton, editor, *Inductive Logic Programming.*, number 38 in The A.P.I.C. Series, chapter 16, pages 335–360. Academic Press, London [u.a.], 1992.
11. Jörg Uwe Kietz. *Induktive Analyse relationaler Daten.* PhD thesis, Technische Universität Berlin, Berlin, oct 1996.
12. Willi Klösgen. *Handbook of Knowledge Discovery and Data Mining*, chapter Subgroup patterns. Oxford University Press, London, 2000. 2000 to appear.
13. Huan Liu and Hiroshi Motoda. *Instance Selection and Construction for Data Mining.* Kluwer Publishers, 2001.
14. Katharina Morik and Peter Brockhausen. A multistrategy approach to relational knowledge discovery in databases. *Machine Learning Journal*, 27(3):287–312, jun 1997.
15. Gordon D. Plotkin. A note on inductive generalization. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, chapter 8, pages 153–163. American Elsevier, 1970.
16. Tobias Scheffer and Stefan Wrobel. A Sequential Sampling Algorithm for a General Class of Utility Criteria. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 2000.
17. Stefan Wrobel. *Concept Formation and Knowledge Revision.* Kluwer Academic Publishers, Dordrecht, 1994.
18. Stefan Wrobel. An algorithm for multi–relational discovery of subgroups. In J. Komorowski and J. Zytkow, editors, *Principles of Data Minig and Knowledge Discovery: First European Symposium (PKDD 97)*, pages 78–87, Berlin, New York, 1997. Springer.