Pre–processing for data mining.

Luboš Popelínský

Habilitation thesis

Masaryk University in Brno,
Faculty of Informatics

December 2003

Abstract

This text refers to automatic data pre-processing methods for data mining, in particular to data transformation for classification. The introductory part brings an overview of automatic data pre-processing methods. It focus on several methods for feature construction that has been investigated in the last years. The main part of this thesis brings a summary of results obtained by the author and his collaborators, mainly PhD. students, in the last four years. This thesis makes three original contributions to data pre-processing. The new method for combination of principal components analysis and machine learning is the first one. Then the maximal frequent patterns are shown to be very useful as new boolean features. At last the new selective sampling method is introduced.

Acknowledgement

Let me express my thanks to many colleague - it is honor for me to call most of them friends - who helped me to see beauty (and nightmares) of scientific work. First of all, I am grateful to my colleagues at Faculty of Informatics, namely to Mojmír Křetínský and Karel Pala for their long-term encouragement, and especially to Luděk Matyska. Without his support not only the Knowledge Discovery Laboratory would be born.

Once in Porto it was Pavel Brazdil who came with idea of combining statistical methods and machine learning. Luc De Raedt brings me to the exciting research in the field of learning first-order maximal frequent patterns. Jan Blaťák and Miroslav Nepil have been more than my collaborators in the last years and their contribution to the results presented here has been enormous.

Discussion with many colleagues when preparing a proposal of a grant on data pre-processing contributes to this text. My thanks are due to Olga Štěp'anková (for much more than this), Jaroslav Zendulka and Jana Šarmanová. Fruitful discussions with my colleagues and students - either in Czechia, Portugal, France or in Germany helps me a lot. Let me express my thanks to Stefan Kramer, Petr Kuba, Carlos Soares, Luis Torgo among others.

I declare that this thesis was composed by myself, and all presented results are my own, unless otherwise stated.

Some of the material previously appeared in conference proceedings. The part on principal components as new features is an extended version of [74]. The first results results obtained with the RAP system has been published in Hinterzaarten, Germany in 2002. Besides those text notes, Part III is also based on [11, 12]. Some of results introduced in Part IV has been published in [68].

Contents

List of Figures

List of Tables

Part I

General

Chapter 1

Introduction

1.1 Knowledge discovery in databases

Searching for useful information such as patterns, associations, changes, anomalies and other important structures in huge amount of data is usually called knowledge discovery in databases (KDD). The knowledge discovery process can be defined as a nontrivial process identifying new, valid, potentially useful, interpretable patterns in data [37]. This process can be divided into three main phases, data preparation, data mining and patterns evaluation and presentation. *Data preparation* then consists of the following steps. The first steps are *integration and selection*: integration of several heterogeneous sources into one data source and selection of the data required for analysis. The next step is *cleaning*: processing wrong, missing, irrelevant and noisy data. *Data transformation* aims at a form of data which is more suitable for mining itself. Some data attributes can be excluded and new attributes can be computed from the original ones. Another goal of the transformation is also data reduction which enables one to mine only a sample of data. *Data mining* represents the key step of the whole KDD process. A learning algorithm is employed to receive a generalization of the data. It must be stressed here that without good preparation it is almost impossible to achieve good results whatever advanced the mining algorithm is.

Most of *data mining* tasks can be seen as search for frequent patterns [63]. Let r be a database, $\mathcal{L}$ a language for expressing properties (defining subgroups), q a frequency criterion (true = frequent, false = infrequent). Then the data mining task lies in finding the theory Th of r with respect to

$\mathcal{L}$ and q such that $Th(\mathcal{L}, r, q) = \{\phi \in \mathcal{L} \mid q(r, \phi)\}$. In the following text we will always assume that $\mathcal{L}$ is finite.

To illustrate this definition have a look at the following example. Let r be a supermarket basket database. Each basket may contain some of items from the set Σ, $\Sigma =$ { `a(pricots), b(iscuits), c(offee), t(urkey), w(ine)` }, i.e. the $\mathcal{L}$ language is a set of all subset of Σ $\mathcal{L} = 2^{\Sigma}$. The frequency criterion $q(r, \phi)$ holds if ϕ appears at least in $min\_supp = 3$ baskets. In the context of mining frequent patterns, the threshold min_supp is called minimal support.

Let the r database contains baskets

```
abcde, abe, abce, abcd, ace   .
```

In the Table there are the frequent patterns, i.e. sets of cardinality 1,2,3.

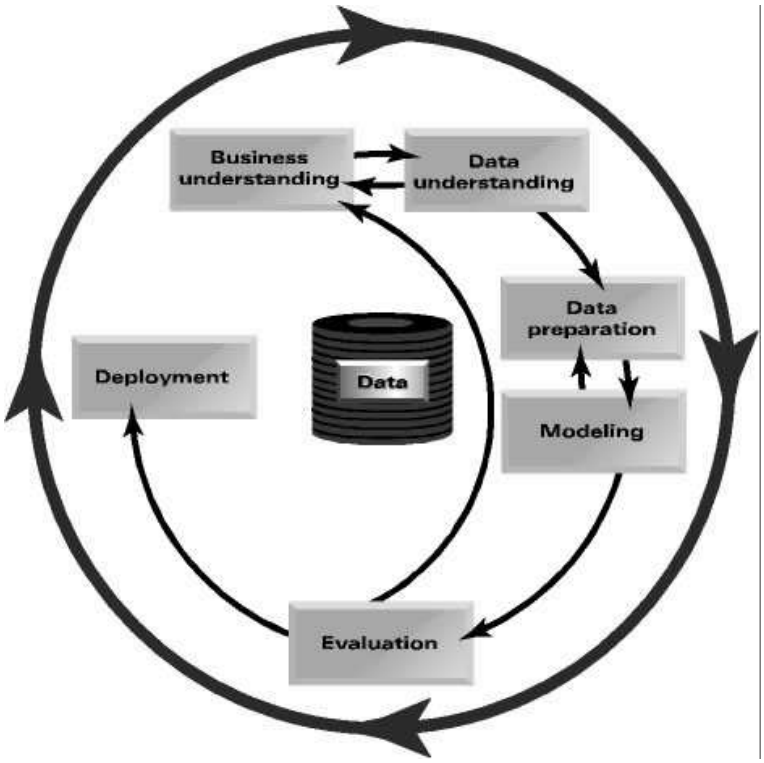
```
1 a,b,c,e
2 ab, ac, ae, bc
3 abc, abe, ace
```

1.2 Standards for KDD

Data mining and KDD have already become widely accepted information technologies, defining their standards. The best- known industrial standard which defines a mining task, including data preparation and a resulting model obtained by data mining, is *Predictive Model Markup Language* (PMML) [1] developed by Data Mining Group (DMG). The main emphasis is put on the mining task itself and on the knowledge acquired. From the viewpoint of preprocessing description it supports only simple transformations of structured data provided by commercial systems. From other industrial standards we mention two. *Common Warehouse Model for Data Mining* (CWM DM)[2] deals with a model in UML which shows objects related to mining. Data pre-processing step has been incorporated into the*Cross-Industry Standard Process for Data Mining* (CRISP-DM, or CRISP) standard[3]. This standard deals with a description of KDD process that deals with huge amount of real data. An XML based language named DMSL (Data Mining Specifica-

[1] http://www.dmg.org/pmml-v1-1.htm
[2] http://www.omg.org/cwm/
[3] http://www.crisp-dm.org

Figure 1.1: Phases of the CRISP-DM Process Model



tion Language) [51] is being developed at FIT VUT. This language defines a KDD process framework with especial emphasis on the data preprocessing step. Nevertheless, there is no commonly accepted standard which would offer tools for detailed and extensible preprocessing description. The situation in formal description of overall preprocessing and KDD process is similar.

1.3 Approaches to data pre-processing

Data preparation is supposed to be the most laborious part of KDD since it consumes most of the project time. Before the wide spread of KDD in practice, data pre-processing methods did not attract much attention of researchers. The situation changed at the end of 90th [33, 20, 50, 52].

There are in principal two ways of preparation. In the first one, called

here *computer-aided data preparation*, a user uses tools that which make the process as easy as possible but need a human expertise. E.g. in MineSet (originaly developed in SGI [4], now in Purple Insight [5]) or SumatraTT [97] it is human who has to decide which tools to use and in what manner. Namely it is up to user which attributes to remove, which new attributes build from old ones, or how to sample the data. Three variants of this approaches has been recognised in the last years to improve efficiency of the whole process. The top-down approach in the MiningMart EU project [6] focuses on creating a collection of well-described solutions for individual mining tasks. When a new problem is being solved, the most suitable solution is searched for among the existing solutions. The second approach used e.g. in the MineSet or Dminer (former Kepler) projects provides a user with a set of tools prepared for individual mining techniques and preprocessing. Then the user can successively choose those tools which he/she considers to be most suitable. These systems usually do not enable a common user to enrich the system with his/her own functions easily. The third approach tackles the data mining task by a bottom-up method [97][7] . It enables a user to define a class of elementary transformation operations, thus extending the repertoire of available preprocessing methods. The commercial systems such as DBMiner, IBM Intelligent Miner, SAS Enterprise Miner, Microsoft SQL Server, or Bayesian Knowledge Discoverer offer only the simplest preprocessing tools such as attribute exclusion according to user requests, but they do not support automatic detection of irrelevant attributes. Similarly, they allow a user to define new attributes but they do not include their automatic construction from data. If they are capable of text mining, then they do not support Czech language.

1.4 Automatic data pre-processing

In this text we focus on another way, on automatic methods for data preprocessing. Some techniques can help user to create good data set, namely automatic feature selection, feature construction and sampling. Nevertheless, it must be stressed at the beginning that the whole process of data pre-

[4] http://www.sgi.com/software/mineset.html
[5] http://www.purpleinsight.com/
[6] http://www-ai.cs.uni-dortmund.de/MMWEB/
[7] See also SolEuNet project, http://soleunet.ijs.si/

processing is impossible to fully automatize. It is also necessary to say that the techniques discussed in this thesis cannot play their role before the data have been collected, selected and cleaned.

Feature selection techniques are the most developed from these three pre-processing methods (see e.g. [20]). This thesis concerns the second and the third methods, feature construction and sampling. There are several reasons why we focused on them. It is well-known that a serious drawback of propositional learners like C4.5 [79] or C5.0 [8] is its very restricted language. A good data transformation technique may result in new attributes that are more appropriate for such algorithms.

The amount of data that are to be processed by data mining systems is increasing and it is not possible to process all the data because of limited computation resources. Fortunately it is not necessary to process all of them. Sampling techniques aim at finding a small fraction examples that represents well the full data set.

1.5 Outline of this thesis

A wide range of feature construction techniques, also called attribute construction or constructive induction techniques, has been explored to enrich the language of propositional learners [7, 15, 92, 98]. The rest of Part I contains a survey of different techniques for feature construction including principal components analysis, frequent patterns and propozitionalization. The last section of the Part I serves as an introduction to sampling techniques.

Principal components are of course not new. They are used sometimes when studying a particular data analysis problem [84]. They are much more widespread among statisticians and data analysts than among machine learning researchers. To our best knowledge few systematic studies have been carried out to evaluate their benefits as new features and the associated computational costs. The aim of Part II of this thesis is to contribute to this issue.

Part III deals with boolean features construction. We describe shortly RAP, a system for mining first-order maximal patterns and present results of feature construction received with this system. We show that partial search for maximal patterns is competitive in term of accuracy to other approaches.

[8] `http://www.rulequest.com`

The main advantage of the current version is that RAP usually generates less number of candidates then other systems.

Selective sampling aims at reduction of the training set without a big decrease of the accuracy of a model trained. In Part IV we present a parametric variant of committee-based selective sampling and a meta-learning technique for setting its parameters.

Chapter 2

Pre–processing techniques in data mining

2.1 Data format and data transformation

In this thesis we focus on mining in a single relation (table). It is the most frequent format of data explored in data mining research. We will assume data are expressed in the form of a single table with M attributes and N lines.

It does not mean at all that the techniques mentioned in this thesis cannot be applied into more complex data format. E.g. frequent patterns discovery is actually a method of propositionalization and any data that can be described in first order logic can be pre–processed by this way.

Each of the pre–processing method can be characterized as *data reduction* or *data extension*. Two main techniques aims at reducing data. *Feature selection* aims at removing attributes that are unuseful for mining. It results in the M_1 x N table where $M_1 < M$. It can be seen as a kind of a vertical reduction of data. *Sampling* enables one to narrow the number of lines in the table. The resulting table has the dimension M x N_1 where $N_1 < N$. It is a horizontal reduction.

Feature construction aims at enriching te initial model. The newly constructed attributes are usually added to the original data. Other possibility is to replace (some of) the initial attributes with the newly constructed features. It is the case of principal components which are frequently used for replacing all the numeric attributes e.g. in image analysis.

2.2 Feature construction

Feature construction [53, 102] (or constructive induction [15]) is one of the most sophisticated data transformation steps in data mining. Automated methods for attribute construction enables one to enrich the set of attributes with new ones. We look for new attributes that are supposed to be more suitable for the mining task at hand than the original ones. More formally, for the existing attributes $x_i, i = 1..M$ we look for new attributes of the form $f_k(x_1, ..., x_l), k = 1, ..., K,\ l \in \{1, 2, ..., M\}$ where f_k is an arbitrary function. Functions most frequently used are logical functions of the form $x_i ? x_j$ for $? =<, =, >=$ etc., or arithmetic functions like $a_i * x_i + b * x_j + c$. E.g. a linear combination computed as a principal component is an example.

Given a repertoir of functions f_k, the goal of any feature selection method is to find *efficiently* a new set of attributes that are useful for improvement of the mining result – accuracy increase, decrease of time or hypothesis complexity etc.

Propositionalization[54, 52, 55] is the most general method for feature construction. Based on inductive logic programming [66] it allows to transform the multi-relational task into a single–relational representation[1]. This transformation enable one to use the common propositional algorithms also for knowledge discovery in multi-relational data. The benefit of propositionalization lies in the reduction of computational complexity of mining. After propositionalization the hypotheses space becomes considerably reduced. Kramer et al. [54] claim that this reduction does not necessarily result in a lower accuracy of the resulting model.

The rest of this chapter contains an introduction into the techniques that the author and his collaborators used for development of new data transformation methods. Only the topics that are relevant to these methods are overviewed.

2.3 Principal components

2.3.1 General

Principal components analysis [67, 81] (or the Karhunen-Loeve expansion) is a statistical method for dimensionality reduction. Principal components

[1] also called the propositional, feature-based, or attribute-value representation

(PCO's) are commonly used to *replace* attributes with smaller number of new attributes. It is an unsupervised projection method that computes linear combinations of the original attributes that are constructed to explain maximally the variance. Let $X = (X_1, ..., X_N)$ be a random vector with variance matrix V. When looking for the first principal component c_1 we look for a vector $c_1 = (c_{11}, ..., c_{1n})$ such that $c_1 c_1^T = 1$ and variance of $c_1 X^T$ is maximal. It means that $c_1 X^T$ describes as much of variablity of data X as possible. The next principal component is computed in a similar way, and must be uncorrelated with the first one, etc. Thus the first few principal components are supposed to contain most of the information implicit in the attributes.

Given M x N data matrix X, N - the number of examples, M - the number of attributes, the idea, following [81], is to take $q < M$ linear combinations XA of the attributes which, in some sense, best represent the original data. "Best" here means that variance is maximized.

Usually the singular value decomposition(SVD) of the data matrix $X = U\Lambda V^T$ is computed where Λ stands for a diagonal matrix with decreasing non–negative values λ_i. U is an M x N matrix with orthonormal columns, and V is p x p orthogonal matrix. The principal components are then the columns of XV. It means [67, 81] that the principal components c_i are equal to eigenvectors of the variance matrix V.

The following properties hold, among others [81]:

1. PCO's display the best possible rank-q approximation to X
2. the first q PCO's minimize the sum of squres of the distances from data points to their projections into the q-dimensional space
3. $min(E[(y - f(x))^T (y - f(x))]\ E'(y|x) = E(y) + \beta^T (x - E(x))$, where β is a solution $Cov(x)\beta = Cov(y, x)$

2.3.2 Kaiser–Meyer–Olkin criterion

Kaiser–Meyer–Olkin criterion (KMO) is usually used to decide that PCO's are useful. Having correlation matrix r_{ij} and partial correlations matrix a_{ij},

$$KMO = \frac{\sum\sum r_{ij}^2}{\sum\sum r_{ij}^2 + \sum\sum a_{ij}^2}$$

If $KMO > 0.5$, than PCO's are recommended to replace the original attributes.

2.3.3 Optimal number of principal components

One question that needs to be answered is: what is the optimal number of PCO's that we should compute? Let $X = (X_1, ..., X_N)$ be again a random vector with variance matrix V. Let $Z_i,\ i = 1, ..., r$ be principal components, r is a number of positive eigenvalues of variance matrix V. Then the following equalities hold:

$$varZ_1 + varZ_2 + ... + varZ_r = varX_1 + varX_2 + ... + varX_N$$

As $varZ_i = \lambda_i$, where λ_i is the i-th eigenvalue of V, the previous equation can be rewritten as:

$$\lambda_1 + \lambda_2 + ... + \lambda_r = varX_1 + varX_2 + ... + varX_N$$

The sum $\sigma^2 = varX_1 + varX_2 + ... + varX_n$ can be seen as a variability measure of vector X. Thus to explain X in terms of Z we need such number q of principal components that $(\lambda_1 + ... + \lambda_q)/\sigma^2$ is close to 1.

Another possibility is to set the number of PCO's to the number of eigenvalues that are significantly greater than one. It was shown [84] that the right approximation does not realy need to compute complex data characteristics like variances. It can be computed using the following criterion:

$$N = \#eigenvalues|eigenvalue > 1 + 2 * \sqrt{\frac{\#attributes-1}{\#examples-1}}$$

2.4 Frequent patterns

2.4.1 General

It was argued [46] that many data mining tasks can be seen as (or based on) a search for frequent patterns in a database. Most of recently-proposed algorithms for mining frequent patterns are variants of Apriori algoritms [2]. *Apriori* uses levelwise strategy (bottom-up breadth-first search) - from frequent patterns of length I patterns of the length $I + 1$ are generated.

It may be hard to mine all the frequent patterns. Instead, mining only *closed patterns* has been proposed as a possible direction [3]. A pattern is closed if it has no extension (there is no longer pattern) with the same frequency. Nevertheless, for some of dense datasets even the set of all dense patterns is too large. The most promising alternative representation are *maximal pattern* that are overviewed in the next paragraph. We conclude with summary of *emerging patterns*.

2.4.2 Maximal patterns

A frequent pattern is called maximal if it is not an extension of any other frequent pattern. It can be easily seen that from the set of all frequent patterns one can generates all frequent patterns. There are many works done on finding maximal patterns in propositional data, mainly in transaction data (e.g. [99, 40]). MaxMiner [82] as well as MAFIA [18] employs look-ahead when searching patterns in depth. MAFIA also prevent to continue search when the new pattern is subsumed by an existing maximal one. When transaction containing X contains as well Y, X is together with Y for the extension. All-MFS [83] uses a generate-and-combine paradigm - first several maximal patterns are generated randomly and then new ones are build as combination of them.

2.4.3 Emerging patterns

Recently, emerging patterns have been introduced [25] and shown to be useful as new features in learning. Having two datasets D_1 and D_2 *the growth rate* of an itemset i *in favour of* D_1 is defined as

$$g = \frac{support\ D_1(i)}{support\ D_2(i)}$$

An *emerging pattern* (EP) is an itemset for which $g > 1$ or infinite. In other words, support of an emerging pattern is changing for different datasets.

ρ*–emerging patterns* (ρEP) are those EPs with growth rate $g \geq \rho$. EPs with high value of ρ can be used for discriminating between D_1 and D_2. When we set D_1 as positive and D_2 as negative examples, it is reasonable to ask about the use of emerging patterns for classification. From this point of view jumping emerging patterns are supposed to be interesting.

A *jumping emerging pattern* (JEP) is a pattern with zero support on D_2 and non-zero support on D_1.[2] Because of this zero/non-zero property JEPs can be mined more efficiently than EPs. In [57] JEPs have been used for building a classifier that is competitive with C4.5.

The used a (cross-product based) *border-diff* algorithm takes randomly one positive example (or one example from a given class for N-class problem) and a set of negative examples (or instances from other classes) and computes for such a training set all JEPs. This process is repeated for all classes. For testing, given a test example all JEPs are applied to the example and score is calculated for each class. That class wins that has the highest score. Score can be computed as a function of support of the JEPs.

Similar approach elaborated Liu et al. in [61]. They generate so called class association rules, the rules that have only a class identifier in the consequent. After employing the Apriori algorithm best class association rules are selected. CBA algorithm based on this concept has been shown to be more accurate on 16 of the 26 UCI datasets then C4.5rules.

Boundary ρ–emerging pattern (ρBEP or BEP) is a ρEP whose proper subset is not a ρEP. An algortihm for mining BEPs is described in [59]. *Plateau EP* (PEP) is a set of patterns having the same support (frequency). *Plateau space* (P-space) consists of all plateau EPs of all boundary EP with the same frequency. *Shadow pattern* (SEP) of a BEP is a pattern one item shorter then the BEP. A classification algorithm for two-class data called prediction by collective likelihood (PCL) that uses BEPs is described in [60]. Frequency of BEPs found in the training set is compared with frequency of these BEPs in test data.

2.5 Propositionalization

2.5.1 General

Learning discriminant functions in first-order logic can be realized by reformulating the problem into an attribute-value form and then applying an efficient propositional learner [54]. This process is usually called *proposi-*

[2] Sometimes (especialy in early papers on emerging patterns), the jumping emerging patterns, as defined above, are named emerging patterns. To prevent from confusion we make the clear distinction here.

tionalization[3]. It has been shown that frequent patterns, actually frequent Datalog queries, can be successfully used as new boolean features [22, 49]. Propositionalization can be either complete (i.e. without loss of information) or partial. It must be stressed that the complexity of propositionalization is in general exponential [30] in respect to the number of background knowledge predicates and their arity. In other words it can result in a propositional representation which contains too many columns. It is also one of the reasons why a higher attention is currently paid to the partial propositionalization [5, 53].

2.5.2 Propositionalization and feature construction

In the propositional representation examples are usualy represented as feature--vectors of a fixed size in the form $f_1 = v_1 \wedge f_2 = v_2 \wedge \ldots \wedge f_a = v_a$, where f_i are the features and v_i the values. Up to our knowledge, all works on propositionalization use only two-valued, boolean features. A feature can be defined, following [54], as a conjunction of (possibly negated) literals

$$f_i(X) \ :- \ Lit_{i,1}, \ \ldots \ , Lit_{i,n_i}. \tag{2.1}$$

where X refers to an individual (a line in the relational table, a molecule etc.). Only literals that are defined in background knowledge can appear in the body of clauses in (1). It seems be useful to extend the set of literals with a predicate of equality [5].

Features may be constructed to describe a subset of the training examples that are in some way interesting. Techniques for finding interesting subgroups [55] are used here. Or they may be constructed to have a same value for a sufficiently large fraction of examples. Methods for finding frequent patterns (itemsets, datalog queries) can be employed. In our work we follow the latter.

2.5.3 Partial propositionalization

Stochastic propositionalization for finding a set of features is introduced in [52]. It employes a strategy similar to random mutation hill-climbing. It was shown that when employing C4.5 learner on this set of features the results are competitive wth other approaches.

[3]We just mention that for some tasks, e.g. recursive predicates, no complete propositionalization exists.

In [89] a method that uses hypotheses returned by Progol is introduced. All subsets of literals that appear in a rule are used as a new features.

WARMR, an algorithm that emplys Apriori-like alorithm for finding a set of first order frequent patters has been used for several propositionalization task. In [49] WARMR was applied to carcinogenicity prediction data and the generated frequent patterns were converted into probabilistic rules. These rules have been found successfull for prediction.

In [5] a selective (lazy) propositionalization for function-free Datalog programs and the *PROPAL* system has been introduced that combines the propositionalization step with the resolution step. *PROPAL* seems be quite fast if compared with these learners, although the running time was measured on a different hardware.

2.6 Sampling

2.6.1 Data reduction

A problem of data mining is volume of data. Even for data of moderate size (in order of hundred thousands examples) the mining process may take too long time with the state-of-the-art tools. As it is not possible to process all the data, data reduction techniques need to be applied to receive a reduced data set. Mining on such data must be efficient enough and to produce the same, or almost the same, result in terms of accuracy, computational time and/or complexity and comprehensibility of the found hypothesis.

Strategies for data reduction can be of different types [43]. *Data cube aggregation*, the technique used in OLAP, employs different aggregation operators in different dimensions of the original data. The found data cube can be much more smaller, and very often also much more suitable, for mining than the unreduced data.

Dimension reduction aims at finding and removing irrelevant, redundant, or insignificant attributes. Feature selection, mentioned earlier, is the main strategy used here. *Data compression* employs, as a rule, lossy compression techniques, e.g. various signal processing techniques like discrete Fourier transform or discrete wavelet transform [43]. After the transformation only the coefficients larger than a threshold are retained. It is a good pre-processing method for data mining algorithms that can take advantage of data sparsity. Another method of data compression is principal components

analysis discussed in Section 2.3.

Discretization and concept hierarchy generation techniques transform attribute values into ranges (e.g. small, medium, large, or $\leq 0, > 0$) or higher conceptual levels (e.g. in the domains term of lectures, physics and chemistry can be replaced by science).

In this thesis we concern with *numerosity reduction*. Data can be replaced by parameters of a data model. These methods are called *parametric methods* and different regression techniques can be used here. The main *non-parametric methods* are histograms, clustering and sampling. Output of good *sampling* is a subset of the original data that represents well the whole data set. Various sampling methods were examined [43, 64], from random selection to selective sampling [19, 38, 56, 94]. The advanced techniques like selective sampling aim at finding representative subsets of training data which would be significantly smaller than the original data set but still would provide enough information necessary for acquiring as accurate model as possible.

2.6.2 Informativness of an example

We define the function $I(x)$ that assigns to each example its informativness. The goal of sampling technique is to choose examples with the highest informativness and only those to use for learning. The most simple sampling technique is a *random sampling* where $I(x) = 1/N$ for N equal to a number of training examples.

2.6.3 Random sampling

Let we have a large data set that contains N examples. The most simple data reduction method is drawing n of the N examples randomly. An advantage of this method is a low complexity which is proportional to the cardinality n of the sample, i.e. potentially sublinear.

Different variants of random sampling has been proposed [43]. The most simple is *random sampling without replacement*. n examples are removed randomly from the data set. In the case of *random sampling with replacement*, an example are only copied into the sample, i.e. in the next step any example already copied into the sample can be drawn again. If the data have been grouped into M mutually disjoint groups, we can use *cluster sampling* applying one of the method already mentioned into the clusters instead the examples itself. The sample then contains m of M clusters. If the data are

divided into several classes (or strata) with unequal frequency, or when the data are skewed, *stratified sampling* should be used. In this case, random sampling is applied to each stratum separately to ensure representativeness of the sample.

Random sampling is especially useful for answering aggregate queries [43] including frequent patterns. Using the central limit theorem we can determine a sufficient sample size n for estimating the value of the query or the support of the frequent pattern.

2.6.4 Selective sampling

When learning a concept usually (1) all training examples are known and classified before starting a learning process, and (2) all these training examples are then used for learning[4]. We will call this setting as *passive learning*. This kind of learning is also called batch learning. *Interactive learning*, in opposite to passive learning, usually does not have an access to a batch of examples. Instead it asks for examples an *oracle* - a human, a database or another program. The learner may be just as a receiver of examples sent by the oracle, or it can ask for specific examples. Overview of queries most commonly used in interactive learning can be found in [6].

Here we focus on the third possibility, *active learning*, also called *selective sampling*. It is up to the learning algorithm to find the best, i.e. the most informative, examples. Selective sampling [19, 38, 56, 94] is an instance selection method which aims at reducing the number of examples needed for learning together with preserving the accuracy of the model learned. In contrast to random sampling selective sampling techniques collect a representative subset of examples that characterize well the target concept.

Most frequently, the reason for selective sampling is incapability of a learning algorithm to learn from whole, huge data set but, following [68], there are two other reasons for data reduction. The one reason could be that a sufficient amount of labeled examples is difficult to obtain; a problem frequently faced in natural language processing. If we cannot rely on unsupervised learning and examples should be annotated manually by a human, then we need to save the annotation costs [19, 94]. The second reason for selective sampling is to obtain more compact and/or more comprehensive result. Sometimes learning on a whole data set leads to a huge model. This

[4]E.g. C4.5 or C5 employs random sampling and windowing

can be caused e.g. by outliers in the training data [39]. For several learning algorithms, including C4.5, it has been observed [69] that increasing the amount of learning data often results in a linear increase in model size without no significant accuracy increase. Therefore, a selection of training data can help to make the model more compact and concise.

Selective sampling proceeds, in general, by measuring the information content of each training example. The objective is to select those examples which could provide the most informative description of a target concept being learned.

There are two main paradigms for measuring the information content: uncertainty-based or committee-based.

Uncertainty-based selective sampling

Approaches to selective sampling based on uncertainty [56] often derive an explicit measure of the expected information gained by using the example. However, the main drawback of these approaches is that they are mostly model-specific.

Committee-based selective sampling

Committee-based selective sampling [38]

Part II

Principal components

Chapter 3

Principal components as new features

Although similar in their goals, statistical methods and machine learning have not been combine very often. We will not discuss here the reason (that seems be more subjective than rational). Instead we bring a positive evidence for such a combination. We show that principal components (PCO's) can be very useful as new attributes in decision tree learning. We show that 3 PCO's is optimal to add them as new features. After a summary of experimental results we introduce a new criterion for recognising that principal are promising for the given data set. We show how time complexity can be significantly lowered for large data sets.

3.1 Computing new features

The PCO's are usually used to replace the original attributes. Another way is to add them to the attributes. In [28, 29] PCO's were used for replacement. Here we test both ways.

3.1.1 Data sets

The datasets used represent a subset of 54 datasets used in comparative studies within project METAL[1]. We explored all data sets that contains at least 3 continuous attributes(columns) and have more than 500 lines. We

[1] http://www.metal-kdd.org/

removed the data sets for which C5.0 reached error rate smaller than 3% as no significant profit can be obtained for such data with any pre–processing technique.

Because of dimensionality problems (too many attributes or too many records ≥ 20000) `musk` and `pyrimidines` data set were removed, too. We evaluated our method on 17 datasets containing both numerical and categorical attributes. The number of columns varied between 8(`diabetes`) to 66 (`optical`), the number of lines was in the range between 537(`fluid`) and 20000(`letter`). More information on the data set can be found in Appendix A.

3.1.2 Learners to explore

We explored propositional learners that are frequently used in machine learning community: decision tree learner C5.0 without boosting – `c50tree`, instance-based learner `mlcib` [87] and naive Bayes classifier `mlcnb` [87, 65]. For all learners the default settings of parameters has been used.

We also performed first experiments with other learners, namely with C5.0 with boosting, with linear discriminant method, and with linear trees (a decision tree that can introduce linear function in its nodes) [47]. However, no significant decrease of error rate has been observed for them. The reason may be that they actually form some kind of linear combination of attributes.

3.1.3 Algorithm

All results below are computed using 10-fold cross validation. In each step the following algorithm was called:

1. Normalise data (i.e. continuous attributes);
2. Compute the first N principal components (linear functions) from the learning set;
3. Employ these linear functions to compute PCO's for the learning set as new attributes;
   ADD the computed values of principal components to the learning set;
   or REPLACE the original attributes;
4. Normalise the test set using means and standard deviations that have been computed from the learning set;

5. Employ the same linear functions as above to compute PCO's for the test set;
   ADD the computed values of the principal components to the test set or REPLACE the original attributes;

6. Run a learning algorithm on the extended learning set ;

7. Test the result on the extended test set.

We tested all possible values of $N = 1..Number\_of\_Attributes$ for each data set.

3.2 To add or to replace?

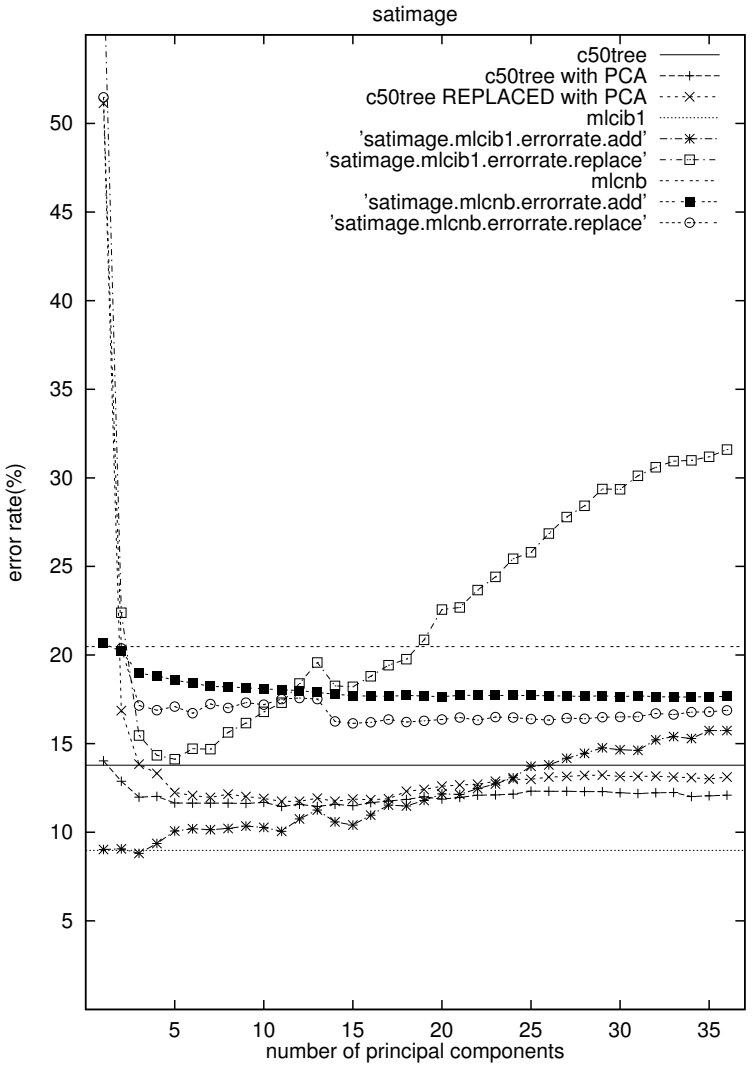
3.2.1 Negative results for replacement

As it was mentioned earlier, the principal components are commonly used to *replace* attributes with smaller number of new attributes. The second way is to *add* principal components to the original dataset. First, we have evaluated both methods on six datasets from UCI repository [10] containing only numerical attributes with no missing values. The datasets `satimage, german_numb, waveform21, waveform40, letter, segment` were used.
All datasets contained at least 1000 examples. For each datasets we first replaced attributes with 1,2,..,N principal components, N is the number of attributes. We employed three learners, `c50tree` (C5.0 with standard settings of parameters, without boosting, 10–cross-validation), `mlcib1` (the instance-based learner IB1) and `mlcnb` (Naive Bayes classifier). Then we ran all the three learners on datasets extended with the same number ofPCO's.

The results obtained by adding attributes were on the whole better then those obtained by replacing attributes. For `satimage` dataset, see Figure 3.1, for instance, the error rate of the decision tree was about 1% better when attributes were added than in the other case. For `waveform` datasets, the replacement worked better, but the difference was quite small. For `letter` and `segment` the error rate with replacement was much worse than for the extended dataset. See Appendix B for the graphs.

Because of that we decided to explore just the way of adding principal components to the original set of attributes.

Figure 3.1: Error rates for satimage



3.2.2 Comparison with Duszak's results

Chapter 4

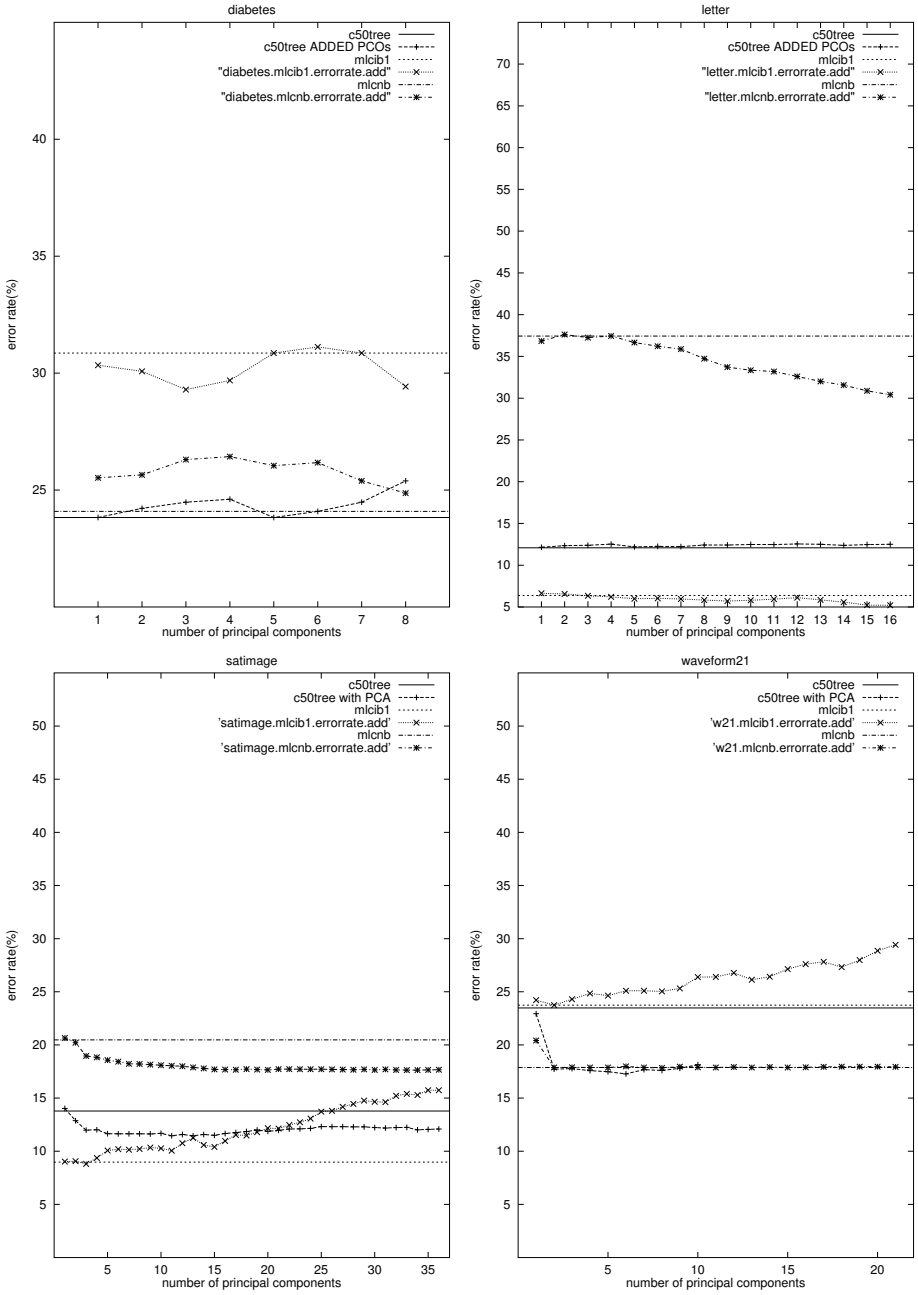
Experimental results

We bring experimental results for the setting when PCO's are being added to the original attributes. We display results error rate, training time (and hypothesis size if relevant) for three propositional learners, the decision tree learner C5.0, the instance-based learner IB1, and the Naive Bayes learner.

4.1 General

In Fig. 4.1 there are results for four data sets and three learners `c50tree, mlcib, mlcnb` that display typical trends of increase/decrease of accuracy. Four typical trends can be observed that are independent on a particular learner. For some data sets PCO's displayed no effect – when increasing their number error rate was not changing (`diabetes` for all three learners; `letter` for `c50tree`). Sometimes the trend of error rate was decreasing but it never decrease below the default level – error rate on the data set without PCO's (`letter` and `c50tree`). The most interesting trends can be seen for `satimage` and `waveform21` data sets. For small number of PCO's – 1 or 2 – the error rate is higher but for 3,4,... PCO's it decrease below the default level (`satimage` for any of the three learners; `waveform21` for `c50tree, mlcnb`). The important fact is that this trend is typical for majority of data sets for which an extension of data with PCO's results in increase of accuracy.

Figure 4.1: Results for `diabetes, letter, satimage, waveform21`



4.2 Decision tree learning

4.2.1 Error rate

Table 4.1 contains the results for c50tree. In the first three columns there are results for the optimal number of PCO's – the number of them, comparison with error rate on the original data (++ means decrease of error rate $> 95\%$, + means decrease of error rate $\leq 95\%$), error rate ratio (error rate for the extended data set divided by the error rate for the original one), and error rate for the extended data. The second and the third triple of columns differ only in the first column. The last column of the tables displays error rates for the original data set.

Table 4.1: Results for C5.0

	c50tree+ best N of PCO's				c50tree+ added 2 PCO's			c50tree+ added 3 PCO's			c50tree
	N		err.r.	err.		err.r	err.		err.r.	err.	
ann	1		1.060	0.35		1.303	0.43		1.303	0.43	0.33
australian	1		1.000	15.51		1.009	15.65		1.000	15.51	15.51
diabetes	1		1.000	23.83		1.016	24.21		1.016	24.48	23.83
fluid	2	++	0.940	2.79	++	0.940	2.79	++	0.940	2.79	2.98
german	3	+	0.980	28.80	+	0.986	29.00	+	0.980	28.80	29.40
letter	1		1.010	12.16		1.021	12.34		1.024	12.38	12.09
optical	1	++	0.937	9.02	+	0.971	9.38	++	0.947	9.15	9.66
page	3	+	0.990	3.05		1.010	3.12	+	0.990	3.05	3.09
pendigits	12	++	0.893	3.24	+	0.986	3.58	++	0.915	3.32	3.63
quisclas	6	+	0.984	37.14		1.014	38.28	+	0.998	37.67	37.75
satimage	11	++	0.838	11.45	++	0.943	12.88	++	0.877	11.98	13.66
segment	3	+	0.964	3.51		1.022	3.72	+	0.964	3.51	3.64
vehicle	2		1.009	26.48		1.009	26.48		1.027	26.95	26.24
vowel	4	++	0.837	18.18	++	0.889	19.29	++	0.870	18.89	21.71
wform21	5	++	0.728	17.10	++	0.732	17.20	++	0.729	17.12	23.48
wform40	6	++	0.689	17.28	++	0.707	17.72	++	0.709	17.78	25.08
yeast	2		1.014	43.87		1.014	43.87		1.025	44.34	43.26

For 11 out of 17 data sets decrease of error rate has been observed. It means that a decrease of error rate appeared for significant number of data sets (one-sample Wilcoxon test on the level 95%). We can also see that the decrease of accuracy is more rare than the increase, and that that decrease is quite small. However, the decrease is not significant on level $\geq 90\%$ (t-test).

4.2.2 Training time

Table 4.2: Training time

	c50tree+		c50tree
	PCO time	train. time	time
ann	0.71	0.56	0.43
australian	0.04	0.07	0.04
fluid	0.18	0.11	0.15
diabetes	0.06	0.08	0.05
letter		10.05	7.68
optical	0.85	4.36	7.42
pendigits	3.96	1.02	3.37
page	0.52	0.82	1.14
quisclas	1.87	2.44	2.53
satimage	4.22	3.73	4.55
segment	0.75	0.57	0.61
vehicle	0.31	0.23	0.24
vowel	0.34	0.74	0.51
waveform21	0.20	1.87	2.55
waveform40	1.79	3.68	5.46
yeast	0.23	0.43	0.60

In our analysis of times (Tab. 4.2), we first compared the training with and without PCO's. For 12 out of 17 data sets the training time for data with PCO's was smaller than for the same data without PCO's. From promising data sets, only `vowel` displayed greater training time for data with PCO's. Then the time to compute PCO's was added to the training time of the decision tree and compared with the time used to construct a decision tree

without PCO's. Only for `ann` data set the ratio of these times overcame 3, and only for 5 out of 16 the ratio was greater 2, i.e. more than twice slower than without PCO's. For `optical` data the sum of PCO time and the training time was smaller than the time without PCO's. Thus, we can conclude that time complexity seems acceptable.

4.2.3 Hypothesis size

We compared the sizes of decision trees (number of nodes) with the corresponding sizes of trees when using PCO's. For 13 out of 17 datasets the size of the decision tree generated decreased when using principal components. From promising data sets only for `fluid` data the tree complexity overcame the complexity of tree without PCO's. Some small increase in hypothesis complexity was observed for `fluid` from 15.4 to 16.3 (5.2%) what corresponds to 1 extra node in the decision tree. More extra nodes appeared for `australian` (increase from 19.9 to 21.8, Δ=9.5%, i.e. approx. 2 extra nodes), `yeast` (166.3, 181.5, Δ=9.1%, 15 nodes) and `diabetes` (22.1, 25.4, Δ=14.9%, approx. 3 extra nodes). When adding the complexity of 3 PCO's (number of attributes * 3) to the complexity of the tree this sum was smaller than `c50tree` complexity only for `waveform21`.

Chapter 5

Answering the main questions

5.1 What number of PCO's to add

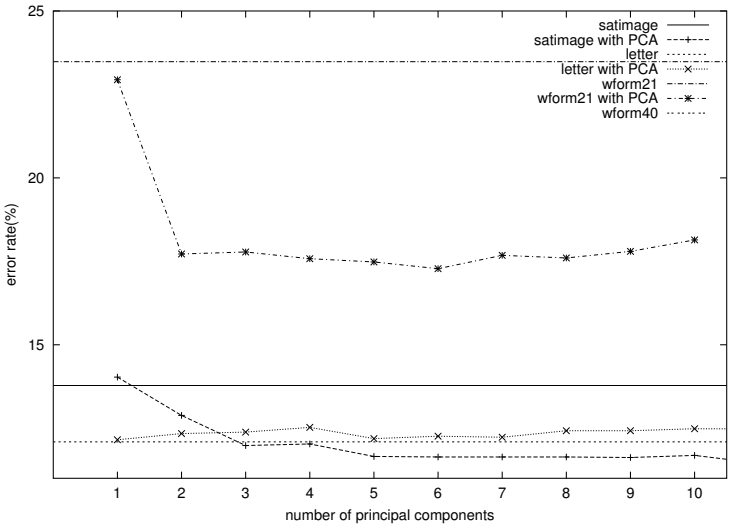
The statement above can be reformulated by such a way: for most of data sets there exists a number N of PCO's that are good candidates for adding to the original set of attributes. A question that needs to be answered now is: what is the optimal number N of PCO's that we should add? In [76] N was set the number of PCO's to the number of eigenvalues that are significantly greater than one. For large data sets it can be computed using the following criterion[84] :

$$N \ = \ \#eigenvalues|eigenvalue > 1 + 2 * \sqrt{\frac{\#attributes-1}{\#examples-1}}$$

Another possibility is to exploit the level of variance that is explained with the first N principal components using eigenvalues of variance matrix [67, 81]. However, none of these criteria worked sufficiently well.

Based on the experiments performed and after analysis of error rate dependency on N we employed the following way of setting N that fits well and needs no additional computation. Looking at Fig. 5.1, the curve `satimage with PCA` (similarly for `wform21`), we can see the typical trend for the data sets that profit from the use of PCO's. (`letter` data set is the counter example; no profit can be obtained with PCO's). For most of the data sets the trend of error rate as a function of N has a similar shape. For small number of PCO's – 1 or 2 – the error rate is higher than or equal to the error rate without use PCO's. Starting from 3 PCO's, the error rate is decreasing below this level. After, the decrease is still continuing but the difference between

Figure 5.1: Results for `wform21, satimage` and `letter`



cases for N ¿ 3 PCO's and for 3 PCO's is not big. Compare now the first and the third triple of columns in Tab. 4.1. We can see that all data sets that are promising for any value of N are also promising for $N = 3$. It does not mean at all that for some $N > 3$ the error rate ratio will be even smaller. However, the expected difference will not be big (see Tab 4.1). Moreover, the C5.0 decision tree learner does not work well when the number of attributes is too big (over-fitting). As we use the PCO's as additional attributes we are tending to add so small number of new attributes as possible. Thus $N = 3$ is a good choice.

5.2 Which data are promising for PCA

Kaiser–Meyer–Olkin criterion (KMO) is usually used to decide that PCO's are useful. If KMO ¿ 0.5 PCO's can be used to replace the original attributes. We also tested this criterion. We observed that for our task the threshold must be higher. We obtained the best results for the threshold 0.6. But even in that case the criterion KMO ¿ 0.6 still hold for some of non–promising data sets. Other possibility was to check the variance covered with 3 PCO's. However, neither that criterion was significant.

We employed a new method based on sampling. 10% examples (or at least 200 examples) was randomly selected for the data set and PCO's were computed for the sample. Than the error rates obtained with C5.0 learner for that sample and the sample extended with PCO's were compared. If the addition of PCO's displayed decrease of error rate, the data set was signed as promising.

This criterion holds for all promising data sets displayed in Table 4.1 but `segment`. It is not time consuming (1.573 sec for `pendigits` data set that contains 10992 examples and 16 continuous attributes). Moreover it can be used with any other learner.

5.3 Where PCO's appear in a decision tree?

We observed that at least one of PCO's appeared in the root (on level 1) of the decision tree or not deeper than on level 4 for the promising data sets. It holds for 15 out of 17 data sets (`australian` was removed because no PCO's appeared in the tree) (significant for the Sign test on level 99%) It is the 2nd principal components that appears "sooner" : for 7 out of 11 such data sets it appeared 7-times (1st PCO three-times, 3rd PCO twice) as the first in the decision tree). It must be, however, noticed that this statement holds, unfortunately, for all the data sets explored, even for those that are not "promising" for adding principal components.

5.4 How to lower computation time

Computation of PCO's is anyway time consuming. However, if the data set becomes smaller the time needed decreases. We checked whether PCO's (i.e. linear functions) computed from a small sample could be used for whole data set. As we worked with quite large data sets – the smallest, `fluid`, contained 537 examples – it seemed reasonable to verify this hypothesis.

We used the PCO's (linear functions) that were computed for the 10% sample. We observed that time varied from 16% (`satimage`) to 61% (`german`) of the time when PCO's were computed from the whole learning set. It is important to stress that for 8 out of 11 data sets this computation time was smaller than 30%. We can see that this approach leads to a significant decrease of time complexity if compared with [73].

Moreover, no significant increase of error rate has been observed. The biggest increase of error rate was observed for `optical`, from 9.15 to 9.55. For 6 out of 11 promising data sets the error rate even slightly decreased.

5.5 Improved version of the algorithm

Input: Data set, Attribute type declaration file, Number of PCO's, Cardinality of the random sample

1. Normalise data (i.e. continuous attributes);
2. Select the random sample; Compute the first 3 principal components from the sample; Check for the sample whether error rate decreased after adding PCO's; If not, use C5.0 on the whole data set with no PCO's and stop. Otherwise
3. Employ PCO's of the sample(linear functions) to compute PCO's for the learning set as new attributes; Add the computed values of principal components to the learning set;
4. Run C5.0 on the extended learning set.

5.6 Conclusion

We displayed the experimental results showing that principal components are useful as new attributes for decision tree learning for significant number of the data sets explored. There is an explanation of the positive result brought. Each decision tree learned with C5.0 splits the example space with axes-parallel lines. Principal components actually allow to build boundaries between examples with lines of any direction. Although it may sound correct this statement has not been proved yet.

We experimentally proved that the results obtained earlier, when PCO's was used to replace the original attributes, do not hold in general. We introduced the new criterion that allows easily to learn whether a data set is a good candidate for application of PCO's. We showed that for the promising data set 3 principal components is optimal to add to the original attributes.

Further work should be carried out to explore some other way of exploitation of principal components, e.g. pruning them (simplifying the linear

functions) or choosing only some of them. Modification of a decision tree learners so that it is able to compute principal components in each step of expansion of the decision tree is challenging.

Part III

Maximal frequent patterns as new features

Chapter 6

RAP

6.1 General

Another preprocessing method introduced in this chapter is based on the search for maximal frequent patterns. The frequent patterns are those patterns which comply with a sufficient number of training examples. As mentioned earlier, most of the methods established so far generate all frequent patterns by breadth first search. From these patterns found, a subset of relevant patterns is selected. After that, binary attributes are generated from these relevant patterns.

This approach have been shown successful for many domains [23]. However, this strategy is too unefficient for the class of data sets that contain long patterns. Bayardo [8] showed that even for benchmark data from the UCI repository the Apriori-like approach is inadequate. This method is extremely inefficient, or even fails, if there is too many short frequent patterns. In general, it is the case of mining dense data – e.g. biochemical and/or molecular data or in (hyper)text. In this case another search strategy – e.g. heuristic or random search – may be preferable.

The RAP system [11] generates frequent patterns by a heuristic search, which results in a faster acquisition of interesting patterns than the breadth first search can achieve. The problem of designing the heuristic evaluation function deserves especial attention.

Here we show that partial search for maximal frequent Datalog queries as new features is competitive with other approaches. We demonstrate a capability of RAP, a system for learning maximal frequent patterns in first-order

logic, for finding new features. Accuracy reached with propositional learners and features generated with RAP is comparable to other feature construction methods. In most experiments RAP was faster then other methods.

6.2 Algorithm

In this section we shortly describe RAP [11], system for searching frequent datalog queries in first-order logic.

The algorithm RAP-Main outlined below describe computation of one maximal pattern.

Algorithm: RAP-Main
Input: Database **r**, language bias definition $\mathcal{L}$, the key predicate used to address examples (*key*), the threshold *minfreq* and maximal pattern length *maxlength*
Structures: set of infrequent patterns *Infreq* and set of known patterns *Known*
Output: maximal pattern Q

```
 was algorithmic[1]
\STATE {make initial pattern  $Q$ (default
$Q=\textit{key}(\textit{Key})$)} \WHILE {pattern $Q$ is shorter
than  $maxlength$ }
  \STATE {generate set $Q_{SPECS}$ of new refinements of $Q$}
  \STATE {discretize continuous arguments in added literals}
  \IF {$Q_{SPECS} \not= \emptyset$}
    \STATE {select one refinement $Q_s$ from $Q_{SPECS}$} such that \\
    $Q_s \not\in \textit{Infreq}$ and $Q_s \not\in \textit{Known}$
  \ELSE
    \STATE {pattern $Q$ is maximal}
  \ENDIF
  \STATE {compute support of $Q_s$ on database $\mathbf{r}$}
  \IF {support of $Q_s$ is great or equal than \emph{minfreq}}
    \STATE {$Q=Q_s$}
  \ELSE
    \STATE {add pattern $Q$ to \emph{Infreq} and go to Step 5}
  \ENDIF
\ENDWHILE
```

In this section we briefly describe each step of this algorithm.

Refinement generation.

The refine operator uses language bias definition (see section 6.4.) to refine the pattern Q. There are two possible settings, which drive its function. The system can generate all possible extensions (candidates) of pattern Q or it can generate (randomly) only one possible refinement. It is useful in the case, where system RAP uses depth-first or random strategy.

6.3 Discretization.

In system RAP is integrated unsupervised discretization method. It is based on *equal frequency intervals* algorithm [62]. This method is not so good as supervised methods, but it is suitable for frequent pattern discovery task, because we often process unclassified data.

Selecting refinement.

System RAP can use one of three available methods, random choice, full depth-first search and best-first search for selecting one refinement from set Q_{SPECS}. There are implemented three heuristics. First of them selects the candidate with minimal value of $V(Q_e)$ where V is computed as

$$V(Q_e) = freq(Q_e) - |freq(Q_e, Pos) - freq(Q_e, Neg)|$$

$freq(Q_e)$ is frequency of candidate Q_e and $freq(Q_e, X)$ is count of examples which belong to the class X and are covered by candidate Q_e. This method is suitable for domains with two main classes but it may be used on other datasets too.

The second method uses the entropy measure.The system compute entropy for all candidates from Q_{SPECS} and selects pattern with minimal one.

The last method uses the confidence measure. The user specifies minimal confidence and system then selects first pattern whose confidence is greater than given value. The confidence is computed for association rule which has the class identifier in the consequent and Q_e in the antecedent (it is simillar to CARs in the CBA classifier).

Check infrequent patterns.

The system can store information about known infrequent patterns. For some kind of data it is inefficient to use this information because the structure maybe quite huge. Using a setting the saving infrequent patterns can be suppressed.

Check known patterns.

Similar situation can appear when testing whether the new pattern is equal or subsumed by some of known patterns. However, the number of the known patterns is not usually so large. There are three main methods how the patterns are checked. RAP checks if Q_e is not subset of or if it is not eaqual to some known maximal or frequent pattern. This checking can be suppressed for the patterns shorter than a given length.

6.4 Language bias

Our formalism is based on the lower level format of WARMR and TILDE [21] language bias. In this subsection we mention main features and some differences from format of WARMR language definition.

Literals or their conjunction which can occur in the patterns are specified by the `rmode/1` predicate in language definition. User can specify count of occurrences of conjunction in pattern.

System RAP can work with typed language bias only. We must specify the types of arguments of each literal mentioned in rmode definition.

System RAP now uses the key format for identifying examples. It uses special predicate, `key/1` which must be defined in the background knowledge and language bias too (there must be mentioned type of example identifier).

User can use lookahead and constraints on variable bindings [21]. Usage of these definitions can save the time of execution because it eliminates useless refinements.

Chapter 7

Feature construction with RAP

7.1 General

We validate our approach on three tasks. We used RAP on propositional data as a preprocessing system for propositional learners and for propositionalization of two real-world domains from bioinformatics. All the results were obtained using 10-fold cross-validation.

7.2 Propositional feature construction

We tested RAP as the preprocessing system for three propositional learners on 9 propositional datasets from UCI archive [10]. The results are shown in Table 7.1. We used learners C4.5 [79], naive bayes and instance-based learner IB1 [87] (column Orig.). Then 30 maximal patterns with frequency greater than 10 % were added to the original dataset (column All). In RAP we used the integrated method of discretization in all experiments. The threshold was set to 0.5 (splitting into two new intervals). For candidate selection in the RAP-Main algorithm we used the entropy minimization heuristics. We observed that the new features could improve accuracy of these learners. We implemented also a simple wrapper [50] for feature selection. The wrapper added new features to the data one by one and called a learner on a validation set (a small subset of training set which was separated from the training set before generation of patterns). Finally, we used the first N features for which the classifier achieved the highest accuracy (column Fsel). We also tried to use the test set as a validation set for feature selection. The results are in

Table 7.1: Predictive accuracy results of propositional learners on the datasets with new features. In boldface the best accuracy is given. In italics are those results for which the patterns selected with the wrapper were better than all generated patterns. We omitted results of IB1 on `letter` dataset since the MLC++ implementation can process only 10 000 examples while this dataset is larger.

	C4.5				Naive bayes				IB1			
Dataset	Orig.	All	Best	Fsel	Orig.	All	Best	Fsel	Orig.	All	Best	Fsel
australian	15.5	**14.5**	**12.9**	15.6	22.6	**13.9**	**12.5**	**17.3**	19.7	**18.1**	**13.9**	**19.1**
crx	13.1	**12.9**	**11.2**	13.3	21.9	**15.3**	**13.7**	**17.6**	17.4	**17.1**	**13.3**	*16.3*
diabetes	24.1	28.4	**22.7**	*24.6*	24.1	**24.1**	**22.8**	24.5	29.0	30.7	**27.6**	*30.5*
german	29.2	**27.3**	**23.0**	29.9	24.3	25.4	**22.2**	25.5	31.4	31.9	**28.3**	*31.5*
letter	12.2	12.8	**11.5**	*12.4*	35.6	36.1	**34.5**	**35.0**	–	–	–	–
quisclas	37.5	38.3	**35.9**	*38.0*	60.2	**44.9**	**44.8**	**45.2**	41.2	42.2	**39.8**	*42.0*
tic-tac-toe	15.7	**12.8**	**6.4**	**10.1**	30.5	**26.3**	**23.6**	*25.6*	1.2	6.1	1.2	*1.3*
vehicle	26.5	**25.3**	**21.6**	27.3	53.1	**37.6**	**36.6**	**38.0**	30.6	31.0	**27.6**	31.5
vowel	21.8	**20.9**	**18.0**	**20.6**	32.5	33.4	**29.8**	*33.2*	0.8	10.6	**0.7**	*0.9*
yeast	43.6	44.9	**41.4**	*44.7*	47.6	47.6	**46.3**	**47.4**	47.7	48.1	**46.5**	**47.5**
Average	23.9	23.8	20.5	23.7	35.2	30.5	28.7	30.9	24.3	26.2	22.1	24.5

the column Best.
It can be seen that there are patterns which improve performance of all mentioned learners. We must note that the good performance of naive bayes can be due to discretization of continuous values in the process of pattern generation.

In Table 7.2 we compare our results with the results of CBA classifier [61] on the datasets common for our and Liu's studies. The table contains a dataset name, an error of C4.5, the best results achieved with CBA[1], the best results achieved with C4.5 and features generated with RAP (Best), the

[1]We have taken the minimal number from columns CARs+infreq., CARs with and without pruning mentioned in Table 1 in [61].

Table 7.2: Comparison of the error rates achieved by CBA classifier and by C4.5 with new features computed by RAP

	Error rate							Num. of candidates		
Dataset	C4.5	CBA	$\frac{C4.5}{CBA}$	RAP(Best)	$\frac{C4.5}{RAP}$	RAP(All)	$\frac{C4.5}{RAP}$	CBA	RAP	$\frac{CBA}{RAP}$
australian	15.49	13.20	1.17	**12.90**	1.20	14.50	1.03	46 564	2 154	22
crx	13.08	14.10	0.93	**11.20**	1.17	**12.90**	1.01	42 877	2 924	15
diabetes	24.10	24.70	0.98	**22.70**	1.06	28.40	0.85	3 315	1 365	2
german	29.20	25.20	1.16	**23.00**	1.27	27.30	1.07	69 277	4 123	17
vehicle	26.49	31.20	0.85	**21.60**	1.23	**25.30**	1.07	23 446	7 843	3
Average			1.02		1.19		1.01			12

results after all 30 features (All), and the number of candidates generated with the systems.

We took the datasets that appeared in both studies. C4.5/RAP beats CBA for all 5 data sets when (Best) variant has been used. However, the (Best) method is usable only in the case that the test set is known in advance. This result only shows that RAP is capable of finding useful patterns. To find a criterion for selection of the best set of maximal patterns is our future work. For three datasets CBA performed better than C4.5/RAP when all features generated by RAP have been added (see the column RAP(All)). We must stress that RAP is more efficient than CBA. It generates in average 12-times less number of candidates than CBA (see last three columns in Table 7.2).

7.3 First-order feature construction

We also evaluated performance of RAP on (multi-)relational data, namely on two domains – the mutagenicity prediction [91] and carcinogenicity prediction [22] (database used in PTE-1 challenge). We generated maximal patterns – binary features for the propositional learner C4.5 [79]. As in experiments above, all the results were obtained using 10-fold cross-validation.

7.3.1 Mutagenicity prediction.

We tested three types of background knowledge. B3 contains atoms, bonds, charges, and values of $logP$ and ϵ_{LUMO}. B3/4 is an extension of B3 with 2-D structures (rings). This hybrid background knowledge enables only to test the occurrence of a structure in compounds. Finally, B4 comprises B3/4, but allows for testing the connection between 2-D structures. The continuous values of partial charges, $logP$, ϵ_{LUMO} were discretized by RAP into three intervals. As the pruning criterion we employed just the testing for known patterns. Each pattern subsumed by some frequent pattern was considered as known, and therefore it was discarded. 10 patterns were generated for B3 and B3/4, whereas 60 patterns were generated for full B4, in average. The

Table 7.3: Predictive accuracy results for mutagenicity prediction domain.

	B3			B3/4			B4		
	Acc.	Size	Time	Acc.	Size	Time	Acc.	Size	Time
Minfreq	%		sec.	%		sec.	%		sec.
0.15	84.02	5.6	306	86.24	11.0	182	85.19	12.8	775
0.20	84.02	5.0	207	87.30	11.0	127	85.18	12.6	826
0.25	84.02	5.0	145	87.30	11.0	126	87.83	11.0	828

results displayed in Table 7.3 are comparable to those published. In Table 7.3 we show the accuracy (column Acc.), the size of hypothesis (column Size)[2] of C4.5 learner and the time consumed by 10-fold cross-validation.

The pruning criterion was so strong that in some folds the number of constructed patterns was too small and these patterns did not cover the whole training set. Sometimes the patterns covered the same portion of training examples since we did not employ covering paradigm in our experiments.

We also tried to relax the pruning criterion in the following way. The patterns shorter than 2 literals were not tested whether they were known. For the minimal frequency of 15 %, RAP found 88.2 patterns in average whose average length was 5.1 literals. The execution took 18 311 seconds, the resulting accuracy of C4.5 learner was 88.9 % and the size of hypothesis was 20.2.

[2]The size of a decision tree is defined as a number of its nodes.

7.3.2 Carcinogenicity prediction.

Here we also tested two versions of background knowledge. The first version contained all predicates except the high-level (2-D) structures. The second version comprised also these structures. We achieved the accuracy of 61.44 % for 65 patterns with the minimal frequency of 15 % and the first version of background knowledge.

7.3.3 Comparison with other work

All mentioned results can be compared with the results presented in [53]. On the mutagenicity domain, when C4.5 learner was used together with the method which used only minimal frequency threshold, the resulting accuracy was 90.4 %. For the carcinogenesis domain and C4.5 learner they report the best classification accuracy of 64.3 %.

7.4 Comparison with Warmr

There are many interesting domains which contain very long patterns. The systems based on a level-wise algorithm cannot be effectively used for processing these datasets. In this section we shall show some advantages of the system RAP against the level-wise system WARMR.

In order to compare the performance of RAP and WARMR, we carried out an experiment on the mutagenicity domain. The minimum frequency threshold was set to 15 % for both systems. For candidate selection within RAP we used the heuristics which minimized the value of $V(Q_e)$ (as described in Section **??**). Both systems used the same background knowledge: B4 without continuous values[3]. Other settings for RAP were the same as in the second experiment with the mutagenicity domain and the background knowledge B4.

We use three main criteria for the comparison: the count of candidates processed and the time and memory consumed by the systems. The first pattern found by RAP was a maximal pattern long 11 literals with the support of 16 %. This pattern and its characteristics are shown below. Note that this pattern contains two substructures. There are two rings connected with a single bond and one sequence of atoms which looks like a part of

[3]The continuous values were excluded since WARMR cannot process them.

an aromatic ring. This pattern is interesting because it discriminates the positive and negative examples very well (it covers 27 positive examples and only 1 negative one). The system RAP evaluated 360 candidates before this pattern was encountered. RAP found 56 maximal patterns and evaluated 9 082 candidates. The execution took 20 minutes and consumed less than 30 MB of memory.

```
Query:  key(A), has_ring(A,ring_size_5,B),
        connected_to_ring(A,B,benzene,1,C), has_atom(A,c,D),
        connected_to(A,D,c,E,1), connected_to(A,D,c,F,7),
        connected_to(A,E,c,G,7), connected_to(A,F,c,H,7),
        connected_to(A,F,h,I,1), connected_to(A,G,c,J,7),
        connected_to(A,H,c,K,7), connected_to(A,H,h,L,1)
  length: 11
  frequency (relative/absolute): 0.17/28
  class distribution: [(pos,27),(neg,1)]
```

WARMR processed 32 594 candidates and found 26 209 frequent patterns until it reached a pattern long eleven literals. It consumed more than 320 MB of memory and the time required for finding all patterns of the length of 10 literals was greater than 30 hours.

Of course, the systems which employ a depth first search or a heuristics search have some disadvantages, e.g. they cannot pass through a database only once when processing all patterns on the same level. However, in this section we showed that a heuristics search can be preferable when searching for a subset of all maximal patterns.

7.5 Conclusion

We presented first results obtained with RAP for feature mining. We showed that maximal patterns can be successfully used as new features for propositional learnes. We showed that usage of heuristics search together with pruning known patterns can speed up propositionalization. The curent version of RAP computes coverage inefficiently. It can be speeded up by using e.g. query packs [14]. This is one of goals of our future work.

Part IV

Selective sampling

Chapter 8

Committee-Based Selective Sampling

8.1 Committee-Based Selective Sampling

This work was motivated by the fact that for large datasets, which are being treated inside the MetaL project, the experiments took too much time. Therefore, our primary goal was to decrease the learning time and perhaps the model size while keeping the error rate as low as possible; a goal perfectly addressed by selective sampling. Since we have been concerned with a simple selective sampling technique which could be easily applied to many different learning algorithms, we gave precedence to the committee-based approach.

The general scheme of selective sampling driven by committee of classifiers is as follows. In the beginning, we build a set of several cheap classifiers – members of the committee. Then, we let the committee make a decision about each given training example, which means that each member has to classify the example according to its own knowledge about the target concept. Thus, we get several (possibly different) class predictions for each example. Hence, the information content of the example is evaluated as a measure of disagreement among the committee members. In the end, for final training we select a subset of examples with high information content.

If we want to devise a particular variant of committee-based selective sampling, several questions should be answered:

1. How many committee members do we need?
2. How to choose the committee members?

3. How to measure the disagreement among committee members?

4. How to select the resulting subset of training examples?

Our parametric variant of committee-based selective sampling adopts the following solution. It presumes that we have a cheap (fast and simple) learning algorithm $\mathcal{A}_{init}$ which we use for training initial classifiers (committee members) and a proper (slow and robust) learning algorithm $\mathcal{A}_{final}$ which we use for training the final classifier. Our method treats the number of committee members as a fixed parameter N. The committee members are established by learning on small subsets obtained by random sampling from the original dataset. The size of these small subsets is given by another parameter I. Our measure of disagreement is rather rough, since we distinguish only two categories: a complete consensus and a dissension. Those examples on which the committee came to a consensus are considered to be *easy*, while the others are considered to be *hard*. The main idea of our method is to select the resulting training subset in such a way that the ratio of *easy* to *hard* examples in the resulting subset is **computed** as a function of the corresponding ratio which was **observed** in the original dataset. As this function we simply took a multiplication by a coefficient X. Another parameter F determines the size of the final training subset. We can already see that our selective sampling technique is parameterized by four numerical values:

N – a number of initial classifiers (members of the committee)

I – a size of the initial training subset used for learning initial classifiers

F – a size of the final training subset used for learning a final classifier

X – a coefficient for modifying the original ratio of *easy* to *hard* examples

More formally, our example selection works as follows:

1. The number of committee members is given by a parameter $N \in \mathbb{N}$, $N \geq 2$.

2. We draw randomly an initial subset from the given training set. The initial subset's size is determined by a parameter $I \in \mathbb{R}$, $0 < I < 1$, as a fraction of the original dataset. The initial subset is randomly split into N blocks and each block is used for training one initial classifier with a learning algorithm $\mathcal{A}_{init}$.

3. Each initial classifier is applied to the whole training set. Therefore, we obtain N class predictions for each example. Those examples which were classified consistently (it means that all N predictions were identical) are considered as *easy* ones while the others are considered as *hard* ones. Let's denote the ratio of *easy* to *hard* examples as e/h.

4. We select randomly a final training subset so that its ratio of *easy* to *hard* examples is given by the expression $X * e/h$ where the coefficient $X \in \mathbb{R}$, $0 \leq X \leq 1$, is another fixed parameter. The final subset's size is determined by a parameter $F \in \mathbb{R}$, $0 < F < 1$, as a fraction of the original dataset. The final subset is used for training a final classifier with a learning algorithm $\mathcal{A}_{final}$.

It is not difficult to guess that a particular setting of the parameters presented above has an important impact on the method's performance. The appropriate parameter setting is not a trivial task since it depends not only on properties of the dataset at hand, but also on our preferences with regard to the learning time, the size, and the precision of learned model.

8.2 Experimental Results of Selective Sampling

At first, we shall show that the selective sampling method really selects representative subsets of the training data. A better quality of the dataset obtained by selective sampling displays a better accuracy of the learned model when compared to the random sampling. In our experiments we tried c50tree algorithm as an initial learner. As final learners we tested c50boost algorithm and c50rules algorithm: corresponding results are shown in Tables 8.1 and 8.2, respectively. These tables show results concerning the total time, the size of learned model, and finally, its error rate on a test set. All numbers were computed by 10-fold cross-validation.

We can see that the error rate achieved by selective sampling remains in many cases close to the original error rate. For adult dataset it even decreased, when c50boost has been used as a final learner. Furthermore, selective sampling is always better than random sampling in terms of accuracy, except for quisclas dataset (which has an excessive error rate on the whole dataset, either). However, it should be noted that this singularity of quisclas

Table 8.1: The comparison of results achieved on whole dataset (WD), by selective sampling (SS), and by random sampling (RS). The initial algorithm $\mathcal{A}_{init}$ was c50tree and the final algorithm $\mathcal{A}_{final}$ was c50boost. The parameters of selective sampling were set as follows: $N = 2$, $I = 0.2$, $F = 0.3$, and $X = 0.1$. The random sampling was set to select the same resulting fraction of data (30 %).

Dataset	Total Time (sec)			Model Size			Error Rate (%)		
	WD	SS	RS	WD	SS	RS	WD	SS	RS
adult	139.3	22.3	22.1	23768	5977	8751	14.49	14.40	15.33
letter	81.1	21.3	15.3	11691	6966	5484	4.69	7.85	9.73
optical	53.0	11.9	7.8	1771	1009	788	2.47	3.83	4.48
pendigits	32.0	8.8	6.2	1703	1122	904	1.15	1.49	2.27
quisclas	19.2	8.3	6.9	5447	1829	1713	35.24	36.77	36.04
satimage	51.9	12.5	8.5	2652	1346	959	9.54	9.88	11.31

Table 8.2: The similar experiment as above, but for the final algorithm c50rules. The parameters of selective sampling were set here as follows: $N = 2$, $I = 0.1$, $F = 0.3$, and $X = 0.2$. The random sampling was set again to select the same resulting fraction of data (30 %).

Dataset	Total Time (sec)			Model Size			Error Rate (%)		
	WD	SS	RS	WD	SS	RS	WD	SS	RS
adult	89.3	16.2	12.8	327	218	215	13.67	14.34	14.80
letter	231.5	29.1	20.3	1177	721	548	11.03	18.37	19.56
optical	16.0	3.3	1.6	209	108	95	8.64	11.17	13.49
pendigits	18.1	3.6	2.1	188	121	106	3.22	4.80	5.99
quisclas	16.9	3.5	2.8	475	179	180	35.34	37.48	37.16
satimage	24.0	3.9	1.9	281	138	108	13.36	14.65	15.77

dataset does not mean that the selective sampling is not useful for it at all. If we use c50boost as the final learner and choose a different setting, namely $N = 4$, $I = 0.3$, $F = 0.3$, and $X = 0.3$, then we get the following results: Total Time 9.1, Model Size 1831, and Error Rate 35.58. Thus, the accuracy of selective sampling is better than of random sampling for quisclas as well,

but we must hit the correct parameter setting.

As for the reduction of model size, often the selective sampling is almost as successful as the random sampling, even thought our method employs no noise handling. For adult dataset with c50boost as a final learner and quisclas dataset with c50rules as a final learner, the selective sampling produced even smaller model than random sampling did.

Nevertheless, the most significant is the reduction of total time. The total time comprises the time taken by sampling, training and testing together. It means that in the case of selective sampling the total time subsumes also the time spent on learning and application of initial classifiers. Consequently, the random sampling is a bit faster, but the extra time spent on selective sampling seems to be really useful, considering the better preserved accuracy. The main asset of time reduction does not rest in the fact that we are able to shrink the total time from 51.9 to 12.5 seconds, but the important thing is that 1/5-time reduction with no considerable decrease in accuracy can help the learning algorithm to scale up to significantly larger datasets.

Chapter 9

Meta-Learning for Parameter Setting

9.1 Motivation

As we could see in the previous chapter, particularly in the discussion about "abnormal" quisclas dataset, the performance of our selective sampling method strongly depends on the setting of its parameters. Table 9.1 demonstrates the impact of parameter X (the coefficient for modifying the original ratio of *easy* to *hard* examples) on the performance criteria. The table shows that the demand on a fast processing and small model goes against the demand on a high accuracy.

Therefore, it is clear that a search for the best parameter setting needs to take into consideration not only the properties (the meta-characterizations) of the particular dataset, but also our preferences with regard to some performance criteria (time, size, accuracy). This observation naturally leads to ranking techniques. Considering our experimental purposes we have resorted to very simple ranking function:

$$R(K, T_s/T_w, E_s/E_w) = K * (T_s/T_w) + (1 - K) * (E_s/E_w)$$

where T_s and T_w are the total times achieved by learning from a sample and by learning from a whole dataset, respectively. Similarly, E_s and E_w are the error rates achieved by learning from a sample and by learning from a whole dataset, respectively. And finally, $K \in \mathbb{R}$, $0 \leq K \leq 1$ is a balance parameter: $K = 0$ means that we are interested in nothing else than the accuracy and,

Table 9.1: The impact of parameter X on the resulting time, model size and error rate, shown on satimage dataset with c50tree as an initial learner and c50boost as a final learner. The resting parameters are fixed to these values: $N = 2$, $I = 0.2$, and $F = 0.3$. The expression e_1/h_1 denotes the original (observed) ratio of *easy* to *hard* examples whereas the expression e_2/h_2 refers to the resulting (computed) ratio. The following time values are listed: T_1 – sampling time, T_2 – training time, T_3 – testing time, and T – total time. Selective sampling with the setting $X = 1.0$ corresponds to random sampling, therefore the sampling time is set to zero.

X	e_1/h_1	e_2/h_2	T_1	T_2	T_3	T	Size	Error
0.1	1389/348	495/1186	1.9	10.5	0.1	12.5	1346	9.88
0.2	1389/348	771/965	1.7	10.0	0.1	11.8	1259	10.46
0.3	1389/348	946/790	1.5	9.4	0.1	11.1	1176	10.57
0.4	1389/348	1067/668	1.4	9.0	0.1	10.5	1124	10.41
0.5	1389/348	1157/577	1.4	8.9	0.1	10.4	1094	10.69
0.6	1389/348	1225/505	1.3	8.3	0.1	9.7	1056	10.86
0.7	1389/348	1279/456	1.2	8.4	0.1	9.8	1025	11.31
0.8	1389/348	1322/410	1.2	8.1	0.1	9.4	1010	11.20
0.9	1389/348	1358/370	1.2	8.1	0.1	9.4	971	11.73
1.0	1389/348	1389/348	0.0	8.4	0.1	8.5	959	11.31

Table 9.2: A construction of meta-examples.

T_p	S_p	E_p	N	I	F	X	K	$R(K, 22.3/139.3, 14.40/14.49)$
0.56	72	0.152	2	0.2	0.3	0.1	0.0	0.994
0.56	72	0.152	2	0.2	0.3	0.1	0.1	0.911
0.56	72	0.152	2	0.2	0.3	0.1	0.2	0.827
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

on the contrary, $K = 1$ means that we regard the total time only. Of course, we always want to minimize this ranking function's value.

We have run the selective sampling on several datasets with different settings to acquire a corresponding values of total time T_s and error rate E_s for a particular setting. Then, those parameter settings together with the corresponding values of the ranking function $R(K, T_s/T_w, E_s/E_w)$ served us as examples for learning a meta-model capable of predicting the right setting for our selective sampling method, given the balance parameter and some data characterizations. As the data characteristics we have exploited a learning time T_p, a model size S_p, and an error rate E_p of a pilot classifier. The pilot classifier was a classifier attained by learning with the initial algorithm $\mathcal{A}_{init}$ from a random sample of a small, fixed size (10 %).

Let's assume that we have run the selective sampling with c50tree and c50boost as an initial and a final learner, respectively, and with the setting $N = 2$, $I = 0.2$, $F = 0.3$, $X = 0.1$, on adult dataset. Then, we get the following performance: total time $T_s = 22.3$ and error rate $E_s = 14.40$. Further, we know that the total time and the error rate by learning from the whole adult dataset is $T_w = 139.3$ and $E_w = 14.49$, respectively. Finally, we run c50tree pilot learner on adult dataset to get its meta-characterizations. Results of the pilot classifier are: time $T_p = 0.56$, size $S_p = 72$, error $E_p = 15.2\,\%$. ¿From these findings we can construct, for instance, the meta-examples for parameter setting which are shown in Table 9.2.

In fact, the number of meta-examples which can be constructed this way is not limited and depends just on the desired variability of the balance parameter K. As an algorithm for learning the meta-model we utilized the system RT4.1 [96] which generates regression trees. The RT4.1 system provided us with a regression model capable of predicting the best (or close to the best) setting by finding the minimal value of the ranking function for

some of the attributes given (namely the meta-attributes T_p, S_p, E_p and the balance parameter K) and the others unknown (those were the parameters of selective sampling: N, I, F, X).

9.2 Experimental Results of Meta-Learning

Table 9.3 presents the results of our regression meta-model. Ranking Deviation is a relative deviation of the predicted ranked minimum from the real ranked minimum (an average ratio of ratios), Hit Rate is a percentage of those settings which were predicted exactly right, Time Deviation is a relative deviation of the total time of the predicted setting from the total time of the best ranked setting and, similarly, Error Deviation is a relative deviation of the error rate of the predicted setting from the error rate of the best ranked setting. The Total Dev. is negative and Error Dev. is positive, which means that our regression model tends to predict settings resulting in faster processing, but worse error rate. All numbers were computed from leave--one-out validation on six different datasets: adult, letter, optical, pendigits, quisclas, satimage.

As we can see in Table 9.3, the meta-attributes T_p, S_p, E_p made the parameter prediction surprisingly worse. We obtained more accurate meta--model (especially for c50boost) by not using those meta-attributes. This is probably due to the fact that the ranking function has a very similiar curve for different datasets and thus the meta-characteristics do not bring additional information – they mislead the regression model instead. On the other hand, it should be noted that we have learned our regression model from relatively small amount of datasets. In fact, the training set in each fold of the leave-one-out validation consisted of five datasets. However, the presence of meta-attributes in the case of c50rules final algorithm led to a significant increase in hit rate, whereas the ranking deviation stayed at almost the same level. Therefore, we suppose that exploitation of meta-attributes promising.

9.3 Conclusion

We have presented a new parametric variant of committee-based selective sampling and a meta-learning technique for setting its parameters. The se-

Table 9.3: The results of meta-learning for selective sampling with c50tree as an initial algorithm and c50boost and c50rules as final algorithms.

Algorithm	Meta Att.	Ranking Dev.	Hit Rate	Time Dev.	Error Dev.
c50boost	absent	7.1257 %	27.3 %	-0.5651 %	9.7485 %
c50boost	present	9.9497 %	19.7 %	-6.9764 %	13.5287 %
c50rules	absent	5.9469 %	21.2 %	-3.1301 %	7.3770 %
c50rules	present	5.9957 %	30.3 %	-10.6430 %	8.0315 %

lective sampling has been proven useful in reducing the learning time while keeping the accuracy at a better level than random selection does. The main contribution of the meta-learning is that its ranking error is fairly low (7.13% for c50boost and 5.95% for c50rules) which is sufficient for a reliable and immediate prediction of the right parameters setting for selective sampling.

Part V

Concluding remarks

Bibliography

[1] Ramesh C. Agarwal, Charu C. Aggarwal, and V. V. V. Prasad. Depth First Generation of Long Patterns. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 108–118. ACM Press, 2000.

[2] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining Association Rules between Sets of Items in Large Databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D.C., 26–28 1993.

[3] Mohammed Javeed aki. Generating non-redundant association rules. In *Proceedings of ACM SIGKDD Conf.*, 2000.

[4] E. Alphonse and C. Rouveirol. Selective propositionalization for relational learning. In Jan M. Zytkow and Jan Rauch, editors, *Principles of Data Mining and Knowledge Discovery, Third European Conference, PKDD '99, Prague, Czech Republic, September 15-18, 1999, Proceedings*, volume 1704 of *Lecture Notes in Computer Science*, pages 271–276. Springer, 1999.

[5] E. Alphonse and C. Rouveirol. Lazy propositionalization for relational learning. In W. Horn, editor, *Proc. of the 14th European Conference on Artificial Intelligence (ECAI'2000)*, pages 256–260. IOS Press, 2000.

[6] D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.

[7] Wnek J. Bala J.W., Michalski R.S. The principal axes method for constructive induction. In Sleeman D. and Edwards P., editors, *Machine*

Learning: Proceedings of the Ninth International Workshop (ML92), pages 20–29. Morgan Kaufmann, Los Altos/Palo Alto/San Francisco, 1992.

[8] Roberto J. Bayardo, Jr. Brute-Force Mining of High-Confidence Classification Rules. In *Proceedings of the Third Int'l Conf. on Knowledge Discovery and Data Mining, KDD'97*, pages 123–126, 1997.

[9] Roberto J. Bayardo, Jr. Efficiently Mining Long Patterns from Databases. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pages 85–93. ACM Press, 1998.

[10] C. L. Blake and C. J. Merz. UCI repository of machine learning databases, 1998.

[11] J. Blaťák, L. Popelínský, and M. Nepil. RAP: Framework for Mining Frequent Datalog Patterns. In *Proceedings of the first KDID workshop at ECML/PKDD 2002*, pages 85–86, 2002.

[12] J. Blaťák, L. Popelínský, and M. Nepil. ??? In *Works in Progress*, pages 1–2, 2003.

[13] Jan Blaťák. Časté vzory a předzpracování dat. Technical report, Fakulta informatiky, Masarykova univerzita v Brně, December 2002.

[14] Hendrik Blockeel, Luc Dehaspe, Bart Demoen, Gerda Janssens, Jan Ramon, and Henk Vandecasteele. Executing query packs in ILP. *Lecture Notes in Computer Science*, 1866:60+, 2000.

[15] Michalski R.S. Bloedorn E., Wnek J. Multistrategy constructive induction: Aq17-mci. In Michalski R.S. and Tecuci G., editors, *Proceedings of the Second International Workshop on Multistrategy Learning (MSL-93)*, pages 188–206. Harpers Ferry, W.VA., 1993.

[16] Tom Brijs and Koen Vanhoof. Cost Sensitive Discretization of Numeric Attributes. In *Principles of Data Mining and Knowledge Discovery*, pages 102–110, 1998.

[17] Doug Burdick, Manuel Calimlim, and J. E. Gehrke. MAFIA: A Maximal Frequent Itemset Algorithm for Transactional Databases. In *Proceedings of the 17th International Conference on Data Engineering*, April 2001.

[18] Burdick D. and Calimlim M.and Gehrke J. Mafia: a maximal frequent itemset algorithm for transactional databases. In *Proceedings of Int. Conf. on Data Engineering*, April 2001.

[19] Ido Dagan and Sean P. Engelson. Selective sampling in natural language learning. In *Proceedings of the Workshop on New Approaches to Learning for Natural Language Processing at IJCAI 1995*, Montreal, Canada, 1995. Morgan Kaufmann.

[20] M. Dash and H. Liu. Feature selection for classification. *Intelligent Data Analysis*, 1(3), 1997.

[21] L. Dehaspe and H. Toivonen. Frequent query discovery: a unifying ilp approach to association rule mining, 1998.

[22] L. Dehaspe, H. Toivonen, and R. D. King. Finding frequent substructures in chemical compounds. In R. Agrawal, P. Stolorz, and G. Piatetsky-Shapiro, editors, *4th International Conference on Knowledge Discovery and Data Mi ning*, pages 30–36. AAAI Press., 1998.

[23] Luc Dehaspe and Hannu Toivonen. Discovery of Relational Association Rules. In Sašo Džeroski and Nada Lavrač, editors, *Relational Data Mining*, pages 189–212. Springer-Verlag, 2001.

[24] Mukund Deshpande and George Karypis. Using Conjunction of Attribute Values for Classification.

[25] Guozhu Dong and Jinyan Li. Efficient Mining of Emerging Patterns: Discovering Trends and Differences. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 43–52. ACM Press, 1999.

[26] Guozhu Dong, Xiuzhen Zhang, Limsoon Wong, and Jinyan Li. CAEP: Classification by aggregating emerging patterns. In *Discovery Science*, pages 30–42, 1999.

[27] James Dougherty, Ron Kohavi, and Mehran Sahami. Supervised and Unsupervised Discretization of Continuous Features. In *International Conference on Machine Learning*, pages 194–202, 1995.

[28] Z. Duszak and W. W. Koczkodaj. Principal components and the accuracy of machine learning. In *DEXA*, pages 421–426, 1992.

[29] Z. Duszak and W.W. Koczkodaj. Using principal component transformation in machine learning. In *Proceedings of ICSRIC, Baden-Baden Germany*, pages 125–129, 1994.

[30] S. Džeroski, L. De Raedt, and H. Blockeel. Relational reinforcement learning. In D. Page, editor, *Proceedings of the 8th International Conference on Inductive Logic Programming*, volume 1446 of *Lecture Notes in Artificial Intelligence*, pages 11–22. Springer-Verlag, 1998.

[31] Sašo Džeroski. Inductive Logic Programming and Knowledge Discovery in Databases. In Fayyad, U. and Piatetsky-Shapiro, G. and Smyth, P. and Uthurusamy, R., editor, *Advances in Knowledge Discovery and Data Mining*, pages 118–152. MIT Press, 1996.

[32] Sašo Džeroski and Nada Lavrač, editors. *Relational Data Mining*. Springer-Verlag, Berlin, September 2001.

[33] A. Famili, Wei-Min Shen, Richard Weber, and Evangelos Simoudis. Data Preprocessing and Intelligent Data Analysis. *Intelligent Data Analysis Journal*, 1(1), 1997.

[34] Usama M. Fayyad and Keki B. Irani. On the Handling of Continuous-Valued Attributes in Decision Tree Generation. *Machine Learning*, 8(1):87–102, 1992.

[35] Usama M. Fayyad and Keki B. Irani. Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. In *Proceegings of the 13th International Joint Conference of Artificial Intelligence*, pages 1022–1027, Chambery, France, 1993.

[36] Usama M. Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, 17(3):37–54, 1996.

[37] Usama M. Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. Knowledge Discovery and Data Mining: Towards a Unifying Framework. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, page 82, 1996.

[38] Yoav Freund, H. Sebastian Seung, Eli Shamir, and Naftali Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2):133–168, 1997.

[39] Dragan Gamberger and Nada Lavrac. Filtering noisy instances and outliers. In H. Liu and H. Motoda, editors, *Instance Selection and Construction for Data Mining*, pages 375–394. Kluwer Academic Publishers, Boston/Dordrecht/London, February 2001.

[40] Karam Gouda and Mohammed Javeed Zaki. Efficiently Mining Maximal Frequent Itemsets. In *ICDM*, pages 163–170, 2001.

[41] Dimitrios Gunopulos, Heikki Mannila, and Sanjeev Saluja. Discovering all Most Specific Sentences by Randomized Algorithms. Research Report MPI-I-96-1-023, Max-Planck-Institut für Informatik, Im Stadtwald, D-66123 Saarbrücken, Germany, September 1996.

[42] Dimitrios Gunopulos, Heikki Mannila, and Sanjeev Saluja. Discovering all Most Specific Sentences by Randomized Algorithms. In *ICDT*, pages 215–229, 1997.

[43] J. Han and M. Kamber. *Data Mining. Concepts and Techniques*. Morgan Kaufmann Publishers, 2001.

[44] Kaoru Hirota and Witold Pedrycz. Implicitly-supervised learning and its application to fuzzy pattern classifiers. *Information Sciences*, 106(1-2):71–85, 1998.

[45] L. Hirschman and R. Gaizauskas. Natural language question answering: The view from here. *Natural Language Engineering*, 7(4):275–300, December 2001 2001.

[46] H.Manilla and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining nad Knowledge Discovery Journal*, 1(4):319–342, 1987.

[47] Gama J. Probabilistic linear tree. In D. Fisher, editor, *Proceedings of the 14th Int. Conf. on Machine Learning (ICML'97)*. Morgan Kaufman, 1997.

[48] George H. John, Ron Kohavi, and Karl Pfleger. Irrelevant Features and the Subset Selection Problem. In *International Conference on Machine Learning*, pages 121–129, 1994.

[49] Ross D. King, Ashwin Srinivasan, and Luc DeHaspe. Warmr: A data mining tool for chemical data. *Journal of Computer-Aided Molecular Design*, 15(2):173–181, 2001.

[50] Ron Kohavi and George H. John. Wrappers for Feature Subset Selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[51] P. Kotsek and J. Zendulka. Describing the data mining process with dmsl. In *Advances in Database and Information Systems. Vol. 2: Research Communications.*, pages 131–140. STUBA, Bratislava, 2002.

[52] S. Kramer, B. Pfahringer, and C. Helma. Stochastic propositionalization of non-determinate background knowledge. In D. Page, editor, *Proceedings of the 8th International Conference on Inductive Logic Programming*, volume 1446 of *Lecture Notes in Artificial Intelligence*, pages 80–94. Springer-Verlag, 1998.

[53] Stefan Kramer and Luc De Raedt. Feature construction with version spaces for biochemical applications. In Carla E. Brodley and Andrea Pohoreckyj Danyluk, editors, *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001)*, pages 258–265. Morgan Kaufmann, 2001.

[54] Stefan Kramer, Nada Lavrač, and Peter Flach. Propositionalization approaches to relational data mining. In Sašo Džeroski and Nada Lavrač, editors, *Relational Data Mining*, pages 262–291. Springer-Verlag, September 2001.

[55] Nada Lavrač, Filip Železný, and Peter Flach. Rsd: Relational subgroup discovery through first-order feature construction. In Stan Matwin and Claude Sammut, editors, *The Twelfth International Conference on Inductive Logic Programming ILP 2002*, July 2002.

[56] David D. Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In William W. Cohen and Haym Hirsh, editors, *Proceedings of* 11^{th} *International Conference on Machine Learning*, pages 148–156. Morgan Kaufmann, 1994.

[57] Jinyan Li, Guozhu Dong, and Kotagiri Ramamohanarao. Making use of the most expressive jumping emerging patterns for classification. In Takao Terano, Huan Liu, and Arbee L. P. Chen, editors, *PADKK 2000*, volume 1805 of *Lecture Notes in Computer Science*, pages 220–232. Springer-Verlag, 2000.

[58] Jinyan Li, Guozhu Dong, and Kotagiri Ramamohanarao. Making use of the most expressive jumping emerging patterns for classification. In Takao Terano, Huan Liu, and Arbee L. P. Chen, editors, *Knowledge Discovery and Data Mining, Current Issues and New Applications, 4th Pacific-Asia Conference, PADKK 2000, Kyoto, Japan, April 18-20, 2000, Proceedings*, volume 1805 of *Lecture Notes in Computer Science*, pages 220–232. Springer-Verlag, 2000.

[59] Jinyan Li, Kotagiri Ramamohanarao, and Guozhu Dong. The space of jumping emerging patterns and its incremental maintenance algorithms. In Pat Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Standord, CA, USA, June 29 - July 2, 2000*, pages 551–558. Morgan Kaufmann, 2000.

[60] Jinyan Li and Limsoon Wong. Geography of differences between two classes of data. In Tapio Elomaa, Heikki Mannila, and Hannu Toivonen, editors, *Principles of Data Mining and Knowledge Discovery, 6th European Conference, PKDD 2002, Helsinki, Finland, August 19-23, 2002, Proceedings*, volume 2431 of *Lecture Notes in Computer Science*, pages 325–337. Springer, 2002.

[61] Bing Liu, Wynne Hsu, and Yiming Ma. Integrating classification and association rule mining. In *Knowledge Discovery and Data Mining*, pages 80–86, 1998.

[62] H. Liu, F. Hussain, C. L. Tan, and M. Dash. Discretization: An Enabling Technique. *Data Mining and Knowledge Discovery*, 6(4):393–423, Oct 2002.

[63] Heikki Mannila and Hannu Toivonen. Levelwise Search and Borders of Theories in Knowledge Discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.

[64] MetaL Project.

[65] Tom M. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[66] S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19/20:629–679, 1994.

[67] F. Murtagh and A. Heck. *Multivariate Data Analysis*. Kluwer Academic, Dordrecht, 1987.

[68] M. Nepil and L. Popelnsk. Part-of-speech tagging by means of ilp and active learning. In *In Proc. of ECML/PKDD2001 Ws on Active Learning, Database Sampling, Experimental Design*. University of Freiburg, Germany, 2001.

[69] Tim Oates and David Jensen. Large datasets lead to overly complex models: An explanation and a solution. In *Proceedings of The Fourth International Conference on Knowledge Discovery and Data Mining*, pages 294–298, 1998.

[70] Claudia Perlich and Foster Provost. Aggregation-based feature invention and relational concept classes. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 167–176. ACM Press, 2003.

[71] L. Popelínský. Data Mining: Hledání znalostí v databázích. In *15. ročník databázové konference DATASEM'95*, pages 211–219, íjen 1995.

[72] L. Popelínský. Quo vadis, data mining? In *17. ročník databázové konference DATASEM'97 Brno*, pages 87–94, íjen 1997.

[73] L. Popelínský. Combining the principal components method different learning algorithms. In C Giraud-Carier et al., editor, *Proceedings of ECML/PKDD2001 Workshop on Integrating Aspects of Data Mining, Decision Support and Meta-Learning, IDDM-01*, pages 119–128, Freiburg, Germany, August 2001.

[74] L. Popelínský. Combining the Principal Components Method with Different Learning Algorithms. In *Proc. of ECML/PKDD2001 IDDM Workshop (Integrating Aspects of Data Mining, Decision Support and Meta-Learning), Freiburg*, 2001.

[75] L. Popelínský and P. B. Brazdil. Combining the principal components method with decision tree learning. In R. S Michalski and P. B. Brazdil, editors, *Proceedings of Multistrategy Learning Workshop MSL-2000*, pages 105–114, Guimarães, Portugal, June 2000. LIACC UP Porto.

[76] L. Popelínský and P. B. Brazdil. The principal components method as a pre-processing stage for decision tree learning. In P. Brazdil and A. Jorge, editors, *Proceedings of PKDD2000 Workshop on Data Mining, Decision Support, Meta-learning and ILP, DDMI 2000*, Lyon, France, September 2000.

[77] PTE2 Challenge.

[78] D. Pyle. *Data Preparation for Data Mining*. Morgan Kaufmann Publishers, 1999.

[79] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[80] J. Ross Quinlan. Improved Use of Continuous Attributes in C4.5. *Journal of Artificial Intelligence Research*, 4:77–90, 1996.

[81] B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[82] Bayrado R.J. Efficienlty mining long patterns from databases. In *Proceedings of ACM SIGMOD Conference*, June 1998.

[83] Gunopulos D.and Manilla H.and Saluja S. Discovering all the most specific sentences by randomized algorithm. In *Proceedings of Int. Conf. on Database Theory IDBT*, January 1997.

[84] G. Sapporta. Some simple rules for interpreting outputs of principal components and correspondence analysis. In *In: Bacelar-Nicolau H., Nicolau F.C., Janssen J.(eds.): Proceedings of 9th Intl.Symp. on Applied Stochastic Models and Data Analysis ASMDA-99*, University of Lisbon, Portugal, 1999.

[85] R. Sedláček and P. Smrž. Automatic processing of czech inflectional and derivative morphology. In *Proceedings of the Fourth International Conference TSD 2001*, 2001.

[86] Masakazu Seno and George Karypis. LPMiner: An Algorithm for Finding Frequent Itemsets Using Length-Decreasing Support Constraint. In *1st IEEE Conference on Data Mining*, 2001.

[87] SGI. MLC++: Machine Learning Library in C++.

[88] SICStus Prolog.

[89] A. Srinivasan. A study of two sampling methods for analysing large datasets with ILP. *Data Mining and Knowledge Discovery*, 3(1):95–123, 1999.

[90] A. Srinivasan, S. Muggleton, R.D. King, and M.J.E. Sternberg. Theories for mutagenecity: A study of first-order and feature based induciton. Technical report, PRG-TR-8-95 Oxford University Computing Laboratory, 1995.

[91] S Srinivasan, R.D. King, and M. Sternberg. Theories for mutagenicity: a study of first-order and feature based induction. *Artifical Intelligence*, 85(1-2):277–299, 1996.

[92] D.J. Stracuzzi and P.E. Utgoff. Feature compilation. Technical report, TR-8-95 Amherst, MA: University of Massachusetts, Computer Science Department., 2000.

[93] Masahiro Terabe, Osamu Katai, Tetsuo Sawaragi, Takashi Washio, and Hiroshi Motoda. Attribute Generation Based on Association Rules. *Knowledge and Information Systems*, 4, July 2002.

[94] Cynthia A. Thompson, Mary Elaine Califf, and Raymond J. Mooney. Active learning for natural language parsing and information extraction. In *Proceedings of 16th International Conference on Machine Learning, ICML 1999*, pages 406–414, Bled, Slovenia, 1999. Morgan Kaufmann.

[95] M. Tomita. *Efficient parsing for natural language*. Kluwer Academic Publishers, 1986.

[96] L. Torgo. *Inductive Learning of Tree-based Regression Models*. PhD thesis, Faculty of Science University of Porto, 1999.

[97] O. tpnkov, P. Aubrecht, Z. Kouba, and P. Mikovsk. Preprocessing for data mining and decision support. In *To appear in Data Mining and Decision Support:Integration and Collaboration.* Kluwer, 2003.

[98] Paul Utgoff. Feature construction for large domains. Invited talk at ICML-2000, Stanford, CA., July 2000.

[99] Agrawal R.and Aggarwal C.and Prasad V. Depth first generation of long patterns. In *Proceedings of ACM SIGMOD Conference*, August 2000.

[100] E. Žačková. *Parciální syntakická analýza češtiny.* PhD thesis, Masaryk University, 2002.

[101] E. Žáčková, M. Nepil, and L. Popelínský. Automatic tagging of compound verb groups in Czech corpora. In *Text, Speech and Dialogue: Proceedings of TSD'2000 Workshop, LNAI*, 2000.

[102] Zijian Zhang. Constructing conjunctive attributes using production rules. *Journal of Research and Practice in Information Technology*, 32(1):13–38, 2000.