Machine Learning and Association Rules

Petr Berka[1,2] and Jan Rauch[1]

[1] University of Economics, W. Churchill Sq. 4, 130 67 Prague, Czech Republic *berka@vse.cz*, *rauch@vse.cz*

[2] Institute of Finance and Administration, Estonska 500, 101 00 Prague, Czech Republic

Abstract. The tutorial will start by reviewing the similarities and differences between statistics, machine learning and data mining. Then we will take a closer look at the knowledge discovery process as described by the CRISP-DM methodology. Here we will focus on various types of machine learning algorithms used for the modeling step and on the statistical approaches and methods used in these algorithms. Attention will primarily be centered on different types of association rules. We will introduce the basic principles of the GUHA method which combines logical and statistical approaches to association rules mining and we will discuss the observational calculi used, the logic of association rules and their applications. We will also show how these principles have been implemented in the LISp-Miner system and how this system can be used to solve real machine learning and data mining tasks.

Keywords: machine learning, data mining, association rules, GUHA method, LISp-Miner system

1 Statistics, machine learning and data mining

1.1 Basic characterization

Statistics, machine learning and data mining are three disciplines that all deal with data analysis, i.e. with the process of inspecting, cleaning, transforming, and modeling data with the goal of highlighting useful information, suggesting conclusions, and supporting decision making.

Statistics can be defined as the science of making effective use of numerical data relating to groups of individuals or experiments. It deals with all aspects of this, including not only the collection, analysis and interpretation of such data, but also the planning of the collection of data, in terms of the designing of surveys and experiments (Dodge, (2003)).

Statistical data analysis can have three forms. *Descriptive statistics* tries to find basic quantitative characteristics of a data set such as central tendency (average, median, mode) or dispersion (standard deviation or range), *confirmatory data analysis* tries to confirm or reject a statistical hypothesis and *exploratory data analysis* uses data to propose new hypotheses (models) to be tested.

Machine learning, a subfield of artificial intelligence "is concerned with the question of how to construct computer programs that automatically improve with experience" (Mitchell, (1997)). In a very broad sense "things learn when they change their behavior in a way that makes them perform better in a future" (Frank, Witten, (2003)). Machine learning is thus not only related to learning from data. We can distinguish two basic learning activities: *knowledge acquisition* and *skill refinement*. Knowledge acquisition consists of inferring and assimilating new material, composed of concepts, general laws, procedures, etc. The knowledge acquired should allow problem solving, performing new tasks, or improving the performance of existing tasks, explaining situations, predicting behavior, etc. Refinement of skills through practice consists of gradually correcting deviations between observed and desired behavior through repeated practice. This activity of human learning covers mental, motor, and sensory processes. It should be noted that current research in machine learning mainly focuses on knowledge acquisition.

Data mining or knowledge discovery in databases is aimed at acquiring implicit knowledge from data and using it to build classification, prediction, description etc. models for decision support. As more data is gathered, with the amount of data doubling every three years, data mining becomes an increasingly important tool to transform this data into knowledge. While it can be used to uncover hidden patterns, it cannot uncover patterns which are not already present in the data set.

The terms data mining and knowledge discovery in databases are often used interchangeably. We will support the view that knowledge discovery is a broader concept covering the whole process in which data mining (also called modeling or analysis) is just one step in which machine learning or statistical algorithms are applied to preprocessed data to build (classification or prediction) models or to find interesting patterns. Thus we will understand knowledge discovery in databases (KDD) as the

> *Non-trivial process of identifying valid, novel, potentially useful and ultimately understandable patterns from data* (Fayyad et al., (1996)),

or as an

> *Analysis of observational data sets to find unsuspected relationships and summarize data in novel ways that are both understandable and useful to the data owner* (Hand et al.,(2001)),

The KDD process consists (according to the CRISP-DM methodology, Chapman et al., (2000)) of business understanding, data understanding, data preprocessing, modeling, evaluation and deployment steps (Fig. 1).

Having these definitions in mind, we can see certain differences in data analysis performed in statistics, machine learning and data mining.

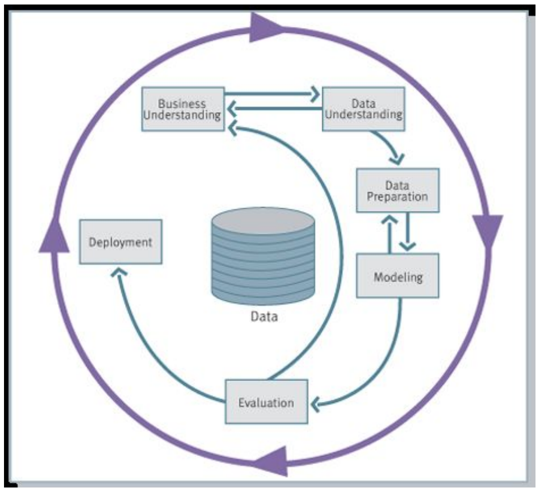

Fig. 1. CRISP-DM methodology (Chapman et al., (2000)).

1.2 Differences and similarities

The conceptual difference between statistics on one side and machine learning and data mining on the other side is the fact that statistical analysis is *hypothesis driven* and *model oriented* while machine learning and data mining are *data driven* and *algorithm oriented* (Hand, (1998)).

When doing statistical data analysis we have to start by formulating one or more hypotheses about a model, then collect the data (in a controlled way that preserves the expected theoretical characteristics of the data), then perform the analysis, and then interpret the results.

Statisticians also usually analyze small volumes of numerical data with known statistical properties (they usually expect data to be drawn from a known distribution and to be stationary).

When doing data mining, we are faced with huge heterogeneous amounts of data with a lot of missings and inconsistencies that may contain some interesting novel knowledge. This data is usually collected for completely different reasons than to provide representative training set for machine learning algorithms. We thus know nothing about the underlying distribution, about its properties (such as stationarity) and we very often do not know what models (dependencies) can be hidden in the data or what are the relevant characteristics (attributes) that will help us to uncover these models.

We usually start by formulating a possible task (rather than a hypothesis), where some tasks (such as classification or prediction) are more related to statistical data analysis, while others (such as pattern discovery) are not. Then we try to find a relevant subset of available data to solve the task, then "play" with various algorithms (machine learning or statistical) and then interpret the results.

Another difference is in the terminology used. Table 1 shows some examples of different terms for the same concepts. Some people also argue that there is a difference in marketing; machine learning methods are much better promoted than statistical ones (Chakrabrati et al. (2008)).

Table 1. Terminological differences

Machine learning	Statistics
attribute	variable
target attribute, class	dependent variable, response
input attribute	independent variable, predictor
learning	fitting, parameter estimation
weights (in neural nets)	parameters (e.g. in regression models)
error	residuum

Nevertheless, there is a lot of common ground in both areas and machine learning and statistics do converge in ideas and procedures. For example, algorithms for building decision trees from data were concurrently proposed in both the statistical community (CART by Breiman et al.) and machine learning community (ID3 by Quinlan), neural networks are very close to regression methods, nearest neighbor methods are used for classification in both areas, and purely statistical methods such as cross-validation or χ^2 test have been incorporated into many machine learning algorithms.

2 Machine Learning Methods and Algorithms

Human learning is one of the most important characteristics of human intelligence. In a similar way, machine learning is one of the most significant fields of artificial intelligence. Machine learning is quite an old discipline; it was a founding branch of artificial intelligence in the late 1950's. However, it has become very attractive, especially recently, thanks to knowledge discovery from databases, data mining, and their developments and applications. Machine learning is one of the feasible means for automatic knowledge acquisition and machine learning algorithms play a key role in the modeling (or analytic) phase of the KDD process described above.

A vast number of machine learning algorithms exist at present. One can categorize machine learning systems along many different dimensions. However, the following dimensions seem to be widely accepted (Berka, Bruha (2000)):

- categorization based on the amount of effort or inference required by the learner,
- categorization based on available feedback during learning,
- categorization based on underlying paradigms, and
- categorization based on the representation of input data and that of acquired concepts.

The taxonomy of learning systems based on the amount of effort (inference) required by the learner to perform its learning task is as follows:

- *Rote learning* consists of simply recording data or pieces of knowledge supplied by the teacher. No inference or transformation is applied to this input by the learner.
- *Learning from instruction* (learning by being told) consists of acquiring knowledge from an external source and integrating it with the knowledge previously acquired. The learner transforms the acquired knowledge from the input language into an internally usable representation.
- *Learning by analogy* consists of acquiring new facts or pieces of knowledge by transforming and augmenting existing knowledge from a similar source idea to an analogous target concept.
- *Explanation-based learning* consists of forming a general characterization of a concept from a very small number of examples and large background knowledge.
- *Learning from examples* consists of acquiring a general concept description (characterization) from a given training set of examples and counterexamples of the concept. The teacher supplies the training set and provides the desired concept for each example. The learner applies induction and can eventually use the background knowledge specific for the given domain. This form of learning seems to be the most investigated in artificial intelligence.
- *Learning from observation and discovery* is a very general form of inductive learning that consists of clustering input examples or observations into different conceptual classes, discovering new concepts and their hierarchical structure. The learner is not supervised by an external teacher, and does not have any information about desired classes (concepts) of input examples; therefore, this form is also called unsupervised learning or learning without a teacher. It corresponds to typical human learning, where the learner elaborates a conceptual structure that will enable him to organize his observations and classify future experience.

Based on the feedback available during learning we can distinguish:

- *supervised learning*, where pre-classified examples are available for learning and the task is to build a (classification or prediction) model that will work on unseen examples,

- *reinforcement learning*, where the teacher observes the learning system (typically a robot) and gives his feedback in the form of rewards (when learning correctly) or punishments (when learning incorrectly),
- *apprenticeship learning*, where the system uses indirect hints derived from the teacherś bahavior,
- *unsupervised learning*, where there is no feedback at all (this is typical for clustering and segmentation tasks).

Two contrasting groups of learning can be distinguished in the above forms of learning:

- *Empirical (similarity-based) learning* involves examining multiple training examples of one or more concepts (classes) in order to determine the characteristics they have in common. Such systems usually have limited background (domain-specific) knowledge. The learner acquires a concept by generalizing these examples through a search process based on inductive inference. Empirical learning comprises both learning from examples and learning from observation and discovery.
- *Analytic learning* formulates a generalization by observing only a single example (or no examples at all) and by exploiting extensive background knowledge about the given domain. Explanation-based learning and learning by analogy belong in this group.

2.1 General overview of empirical concept learning

The most important type of learning for KDD is empirical concept learning. This type of learning can be formally defined as follows:

Let the analyzed (so called *training*) data have the form

$$\begin{array}{c c} & \begin{array}{ccccc} x_1 & x_2 & ... & x_m & y \end{array} \\ \begin{array}{c} 1 \\ 2 \\ ... \\ n \end{array} & \begin{pmatrix} x_{11} & x_{12} & ... & x_{1m} & y_1 \\ x_{21} & x_{22} & ... & x_{2m} & y_2 \\ ... & ... & ... & ... & ... \\ x_{n1} & x_{n2} & ... & x_{nm} & y_n \end{pmatrix} \end{array}$$

where each row corresponds to a single example, also called object (market basket, patient, bank client etc.) and each column corresponds to an attribute (categorial or numerical) describing a property of these objects (so x_{ij} is the value of the j-th attribute for the i-th object).Let us further assume that there is a special attribute y (called target, goal, or class) that divides the examples into classes (for classification tasks) or contains a numerical value we are interested in (for prediction tasks). So, the description of an object $\mathbf{o}_i$ consists of values of "input" attributes $\mathbf{x}_i$ and the value of target attribute y_i

$$\mathbf{o}_i = [\mathbf{x}_i, y_i].$$

Such types of data allow us to define a classification or prediction task. We search for knowledge or a model (represented by the decision function f) $f : \mathbf{x} \to y$, that for input values $\mathbf{x}$ of an object infers the value of target attribute $\hat{y} = f(\mathbf{x})$. During the classification or prediction of an example we can make a wrong decision. We can express the possible error $Q_f(\mathbf{o}_i, \hat{y}_i)$ as

$$Q_f(\mathbf{o}_i, \hat{y}_i) = (y_i - \hat{y}_i)^2 \tag{1}$$

if the target attribute is numerical (i.e. if we are doing prediction), or as

$$Q_f(\mathbf{o}_i, \hat{y}_i) = \begin{cases} 1 & \text{iff} \quad y_i \neq \hat{y}_i \\ 0 & \text{iff} \quad y_i = \hat{y}_i \end{cases} \tag{2}$$

if the target attribute is categorial (i.e. if we are doing classification).

We can compute the total error $Err(f, D_{TR})$ for the whole training data D_{TR} , e.g. as

$$Err(f, D_{TR}) = \frac{1}{n} \sum_{i=1}^{n} Q_f(\mathbf{o}_i, \hat{y}_i). \tag{3}$$

The goal of learning is to find such a knowledge f^*, that will minimize this error.

Induction is the most appropriate form of inference in artificial intelligence, therefore - when we talk about learning - we usually only mean the inductive learning (i.e. learning from examples and from observation and discovery). One assumption of this type of learning has already been mentioned: examples belonging to the same class have similar characteristics. When we display each example as a point in a multidimensional attribute (feature) space, examples belonging to the same class thus form clusters in this space and during learning, we try to find some general description of each cluster. The other assumption is that we infer general knowledge from a finite set of examples; to obtain usable knowledge we thus need sufficiently representative training data. Both these assumptions have a significant impact on practical KDD tasks:

- in the data preprocessing step we need to find such characteristics (attributes) that fulfill the similarity assumption,
- in the modeling and evaluation steps we need to thoroughly test the acquired knowledge on data that has not been used in the learning process (so called *testing* data).

There are two general approaches to inductive learning. Inductive learning can be viewed as:

- *search* in the space of concept descriptions - in this case we learn both the structure and parameters of the model,

- *approximation* within a class of models - in this case we "only" learn the parameters of the model.

If we understand machine learning as the task of finding appropriate clustering of training data in the attribute space, we can search trough the possible clusterings (i.e. models) of the data to find the best one. To perform learning as a search, we must define the ordering of these models according to their generality. The most general model (MGM) would correspond to a situation where all examples belong to the same cluster. Using such a model for classification will result in the assignment of every example to the same (usually the majority) class. The most specific model (MSM) would correspond to a situation where each example creates its own cluster. Such a model would only classify examples from the training data and cannot be applied to any new, previously unseen example. We can say that model M_1 is *more general* than model M_2 (and Model M_2 is *more specific* than model M_1) if we can transform M_1 into M_2 by splitting some of its clusters using a *specialization* operator, and transform M_2 into M_1 by merging some of its clusters using a *generalization* operator. The exact form of these operators is naturally dependent on the machine learning algorithm used; e.g. in learning decision trees we obtain a more specific tree by substituting a leaf with a subtree and obtain a more general tree by substituting a subtree with a leaf.

Ordering between models allows us to use various search methods that can differ in direction (they can be top down - i.e. from a general to specific model, or bottom up - i.e. from a specific to general model), in strategy (they can be blind, heuristic or random), or in width (they can consider a single model or more models in parallel).

Learning as approximation assumes that we "know" a general class of suitable decision functions (e.g. linear) and that we are looking for the best parameters. A typical approach used here is the least squares method. The general learning scheme for finding a model that will minimize the total error

$$Err(f, D_{TR}) \approx \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i))^2 \tag{4}$$

is transformed to solve the equation

$$\frac{d}{dq} \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i))^2 = 0. \tag{5}$$

If the class of functions is known (e.g. if f is linear as in the linear regression method), we can find the analytical solution by solving the set of equations for parameters of the function f. If the class of functions cannot be expressed analytically (e.g. in neural networks), we must use gradient methods to find a numerical solution. In this case we have to compute the gradient of the error as

$$\nabla Err(\mathbf{q}) = \left[\frac{\delta Err}{\delta q_0}, \frac{\delta Err}{\delta q_1}, ..., \frac{\delta Err}{\delta q_k}\right], \tag{6}$$

where $q_0, q_1, ..., q_k$ are parameters of the model f, and then modify the parameters of the model as

$$q_j \leftarrow q_j + \Delta q_j \tag{7}$$

where

$$\Delta q_j = -\eta \frac{\delta Err}{\delta q_j}. \tag{8}$$

There are a number of methods that follow one of the paradigms mentioned above. We will review some of these in the next subsection. However, we must stress here that there is no "best" algorithm that will outperform (in terms of classification or prediction accuracy) the other algorithms on any problem (this fact is known as the no free lunch theorem; see e.g. Wolpert, (1996)).

2.2 Selected methods of empirical concept learning

Decision trees Decision trees belong to the most popular models for solving classification tasks. When building a decision tree, we recursively partition the attribute space in a top-down manner. This method, also known as "divide and conquer" has been implemented in various algorithms. Fig. 2 shows a simplified sketch of the ID3 algorithm by Ross Quinlan (Quinlan, 1986). This algorithm can only process categorial data, numerical attributes must be discretized in advance (the reason for this is in step 2 of the algorithm) and will stop after all examples are correctly classified (step 3). The C4.5 algorithm (a successor to ID3, see Quinlan, (1993)) can process numerical data as well, and has a relaxed stopping condition; so this algorithm can process noisy data containing a mixture of catergorial and numerical attributes.

```
algorithm ID3

1. select an input attribute as the root of a particular (sub)tree
2. divide the data in this root into subsets according to each value of the
   selected attribute and add a new node for each resulting subset
3. if there is a node containing examples of different classes, go to step 1
```

Fig. 2. The ID3 algorithm

The key point of the algorithm is the way a splitting attribute is chosen. A criterion that reflects the ability to correctly separate examples of different classes is computed for every available input attribute at the splitting node.

The attribute with the "best" value is then selected for the splitting node. The ID3 algorithm (Quinlan, (1986)) uses entropy

$$H(A) = \sum_{i=1}^{R} \frac{r_i}{n} \left(- \sum_{j=1}^{S} \frac{a_{ij}}{r_i} \log_2 \frac{a_{ij}}{r_1} \right), \tag{9}$$

the C4.5 algorithm (Quinlan, (1993)) uses information gain

$$InfoGain(A) = - \sum_{j=1}^{S} \frac{s_j}{n} log_2 \frac{s_j}{n} - H(A), \tag{10}$$

the CART algorithm (Breiman et al. (1984)) uses the Gini index

$$Gini(A) = \sum_{i=1}^{R} \frac{r_i}{n} \left(1 - \sum_{j=1}^{S} \left(\frac{a_{ij}}{r_i} \right)^2 \right), \tag{11}$$

the CHAID (Biggs et al. (1991)) algorithm uses χ^2 statistics

$$\chi^2(A) = \sum_{i=1}^{R} \sum_{j=1}^{S} \frac{\left(a_{ij} - \frac{r_i s_j}{n} \right)^2}{\frac{r_i s_j}{n}}. \tag{12}$$

In all these formulae a_{ij} is the number of examples that have the i-th value of the attribute A and the j-th value of the target attribute, r_i is the number of examples that have the i-th value of the attribute A, s_j is the number of examples that have the j-th value of the target attribute and n is the number of all examples.

Algorithms for building decision trees perform a top down greedy search in the space of possible trees. A decision tree divides the attribute space into regions (clusters) of rectangular shape that are perpendicular to the axes.

Association rules An association rule is commonly understood to be an expression of the form

$$X \Rightarrow Y,$$

where X and Y are sets of items such that $X \cap Y = \oslash$. The association rule $X \Rightarrow Y$ means that transactions containing items from set X tend to contain items from set Y . The term association rule was coined by R. Agrawal in the early 90s in relation to a so called market basket analysis (Agrawal et al., 1993). In this analysis, transaction data recorded by point-of-sale systems in supermarkets are analyzed in order to understand the purchase behavior of groups of customers, and used to increase sales, and for cross-selling, store design, discount plans and promotions. So an example of an association rule can be

$$\{\text{eggs, bacon}\} \Rightarrow \{\text{cheese}\}$$

expressing, that customers who buy eggs and bacon also often buy cheese (to make ham-and-eggs).

This idea of association rules was later generalized to any data in the tabular, attribute-value form. So data describing properties (values of attributes) of some examples can be analyzed in order to find associations between conjunctions of attribute-value pairs (categories) called antecedent (Ant) and succedent (or consequent) (Suc):

$$Ant \Rightarrow Suc$$

The two basic characteristics of an association rule are *support* and *confidence*. Let a (for the analyzed data) be the number of examples (rows in the data table) that fulfill both Ant and Suc, b the number of examples that fulfill Ant but do not fulfill Suc, c the number of examples that fulfill Suc but do not fulfill Ant, and d the number of examples that fulfill neither Ant nor Suc. Support is then defined as

$$sup = P(Ant \wedge Suc) = \frac{a}{a+b+c+d}, \tag{13}$$

(the value a can be understood as *absolute support*), and confidence is defined as

$$conf = P(Suc|Ant) = \frac{a}{a+b}. \tag{14}$$

In association rule discovery, the task is to find all syntactically correct rules $Ant \Rightarrow Suc$ (i.e. rules, in which two different values of an attribute cannot occur) so that the support and confidence of these rules is above the user-defined thresholds *minconf* and *minsup*. There are a number of algorithms that perform this task. The main idea of these algorithms is to repeatedly generate a rule in a "top-down" manner by rule specialization (i.e. by adding categories to an existing combination) and test if this rule meets the thresholds *minconf* and *minsup*. The probably best-known algorithm called *apriori* proceeds in two steps. All frequent itemsets are found in the first step (see Fig. 3). A frequent itemset is a set of items that is included in at least *minsup* transactions. Then, association rules with a confidence of at least *minconf* are generated in the second step by splitting a frequent itemset $Comb$ into Ant and Suc (Agrawal et al. (1993)).

Decision rules "If-then" rules belong to the most popular formalism used to represent knowledge either obtained from human experts (as in the case of expert systems) or learned from data. Unlike association rules, where the "then" part of a rule (the succedent) can contain a conjunction of arbitrary attribute-value pairs, decision rules have the form

$$Ant \Rightarrow C$$

```
algorithm apriori

1. assign all items that reached the support of minsup into $L_1$
2. let $k = 2$
3. while $L_{k-1} \neq \emptyset$
   3.1 using the function apriori-gen create a set of candidates $C_k$ from $L_{k-1}$
   3.2 assign all itemsets from $C_k$ that reached the support of minsup into $L_k$
   3.3 increase $k$ by 1

Function apriori-gen($L_{k-1}$)
1. for all itemsets $Comb_p$ , $Comb_q$ from $L_{k-1}$
   if $Comb_p$ and $Comb_q$ share $k-2$ items, then add $Comb_p \wedge Comb_q$ to $C_k$
2. for all itemsets $Comb$ from $C_k$
   if any subset with a length $k-1$ of $Comb$ is not included in $L_{k-1}$ then remove $Comb$ from $C_k$
```

Fig. 3. Finding frequent itemsets using the apriori algorithm

where C is the value of the categorial target attribute.

Again, there are a number of algorithms for building decision rules from data. Fig. 4 shows a sketch of the *set covering algorithm*. One group of algorithms (so called set covering algorithms) proceeds in a way called "separate and conquer"; during each pass of the main cycle of the algorithm some examples of the target concept are described by a single rule and then removed from the training data (Fig. 4). A new candidate rule can be created either by rule specialization (adding a new attribute-value pair to the condition of the rule; this method is used, for example, in the CN2 algorithm by Clark and Niblett (Clark, Niblett, (1989)) or by rule generalization (removing an attribute-value pair from the condition of a rule; this method is used, for example, in the AQ algorithm by Michalski (Michalski, (1969)).

The set covering algorithm thus performs a top-down or bottom-up search in the space of all possible rules. Each rule defines a rectangular region in the attribute space.

```
set covering algorithm

1. create a rule that covers some examples of one class and does not cover
   any examples of other classes
2. remove covered examples from training data
3. if there are some examples not covered by any rule, go to step 1
```

Fig. 4. The set covering algorithm

Another type of rule learning algorithm learns rules in the form

$$Ant \Rightarrow C(w)$$

where w (called weight) expresses the uncertainty of the rule. An example of such an algorithm is the KEX algorithm (Berka, Ivánek (1994)), which performs a heuristic top-down search in the space of candidate rules. In this algorithm the covered examples are not removed during learning, so an example can be covered by more rules. Thus more rules can be used during classification each contributing to the final assignment of an example. KEX uses a pseudo-bayesian combination function borrowed from the expert system PROSPECTOR (Duda, Gashing (1979)) to combine contributions of more rules:

$$w_1 \oplus w_2 = \frac{w_1 \times w_2}{w_1 \times w_2 + (1 - w_1) \times (1 - w_2)}. \tag{15}$$

KEX works in an iterative way, testing and expanding an implication $Ant \Rightarrow C$ in each iteration. This process starts with a default rule weighted with the relative frequency of class C and stops after testing all implications created according to user defined criteria. The induction algorithm inserts only such rules into the knowledge base, for which the confidence (defined in the same way as the confidence of association rules) cannot be inferred (using formula 15) from weights of applicable rules found so far. A sketch of the algorithm is shown in Fig. 5.

```
KEX algorithm
Initialization
  1. forall category (attribute-value pair) $A$ add $A \Rightarrow C$ to $OPEN$
  2. add empty rule to the rule set $KB$

Main loop
while $OPEN$ is not empty
  1. select the first implication $Ant \Rightarrow C$ from $OPEN$
  2. test if this implication significantly improves the set of rules $KB$ built
     so far (using the $\chi^2$ test, we test the difference between the rule validity
     and the result of classification of an example covered by $Ant$) then add
     it as a new rule to $KB$
  3. for all possible categories $A$
     (a) expand the implication $Ant \Rightarrow C$ by adding $A$ to $Ant$
     (b) add $Ant \wedge A \Rightarrow C$ to $OPEN$ so that $OPEN$ remains ordered
         according to decreasing frequency of the condition of rules
  4. remove $Ant \Rightarrow C$ from $OPEN$
```

Fig. 5. Simplified sketch of the KEX rule learning algorithm

Neural networks (artificial) Neural networks are mathematical or computational models of the structure and/or functional aspects of biological neural networks. A neural network is composed of a number of elementary units (neurons) that are interconnected in different ways. A single (artificial) neuron can be understod as a threshold unit that sums up m weighted signals on its input.

$$sum = \sum_{i=1}^{m} w_i x_i \tag{16}$$

The output *out* of the neuron depends on this sum. A common method is to transform the sum using some sigmoidal-shaped function, so

$$out = \frac{1}{1 - e^{sum}} \tag{17}$$

or

$$out = \tanh(sum). \tag{18}$$

This behaviour can easily be used for classification. If inputs into a neuron correspond to the values of input attributes, then the single neuron divides the attribute space using a hyperplane into two sub-spaces each containing examples of one class. Due to the sigmoidal transformation of *sum*, the border between the classes is "soft".

There is a number of different types and topologies of neural networks (Hecht-Nielsen, (1991)). In data mining, the most popular types are *multi-layer perceptron* and *RBF network* used for classification or prediction, and *Kohonen network* (also called SOM, i.e. self-organizing map) used for segmentation and clustering. Let's take a closer look at multi-layer perceptrons. In this type of network, the neurons are organised into layers in such a way, that neurons within one layer are not connected, but every neuron in one layer is connected to all neurons in neighbouring layers. Multi-layer perceptrons have one input layer, several hidden layers and one output layer. As it is known that multi-layer perceptron with one hidden layer is a universal function approximator, the most frequently used topology consists of three layers.

When using multi-layer perceptron for data mining, we bring the values of input attributes to the input layer and obtain the value of the target attribute from the output layer. The knowledge (i.e. the model) is composed of the topology of the network and weights of the connections between neurons. As the topology is given in advance (the topology implicitly encodes the function that transforms the inputs of the networks to it'-s outputs), the learning neural network fits into learning as approximation paradigm. As we cannot express the function between inputs and outputs analytically, we must proceed numerically as shown in formula 6. A simplified version of the algorithm that learns weighs from training data is shown in Fig. 6.

```
Error backpropagation algorithm

1. inicialize the weights in the network with small random numbers (e.g.
   from the interval $[-0.05, 0.05]$)
2. while the stopping condition is not satisfied for every training example
   $[\mathbf{x}, y]$ do
   2.1 compute the output $out_u$ for every neuron $u$
   2.2 for every neuron $o$ from the output layer compute the error
           $error_o = out_o(1 - out_o)(y_o - out_o)$
   2.3 for every neuron $h$ from the hidden layer compute the error
           $error_h = out_h(1 - out_h) \sum_{o \in output} (w_{ho} error_o)$
   2.4 for every connection from neuron $j$ to neuron $k$ modify the weight
       of the connection
           $w_{jk} = w_{jk} + \Delta w_{jk},$
       where
           $\Delta w_{jk} = \eta \, error_k \, x_{jk}$
```

Fig. 6. Error backpropagation algorithm for training multilayer perceptron

Genetic algorithms Another biologically inspired methods are genetic algorithms. Genetic algorithms are used to find the (approximate) solution for optimalization or search problems. As such, they perform (from a data mining perspective) a parallel random search in the space of models. The basic idea of genetic algorithms is to repeatedly apply evolutionary operators (selection, crossover, mutation) to a set of individuals, each representing a possible solution (model). In this process the population of individuals evolves toward better solutions. The basic form of a genetic algorithm is shown in Fig. 7 (deJong et al. (1993)).

For the implementation of the evolutionary principle, the crucial aspect is how to encode the individuals and how to evaluate their quality (called fitness). In the simplest case, the individuals are encoded using bit-strings, and - when doing classification - the fitness reflects the classification accuracy of the individuals.

Genetic algorithms can be used for data mining directly (in this case, each individual encodes a possible model, e.g. decision rule or decision tree), within a "standard" machine learning algorithm (e.g. for generating a single rule in a set covering rule learning algorithm, for an example of this approach see Králík, Brůha (1998)), or in relation to another machine learning algorithm (e.g. to find the optimal topology of a neural network - in this case each individual encodes a single neural network that is separately trained on data).

```
Genetic algorithm(fit,N,K,M)
Initialization
  1. assign t:= 0
  2. randomly create the initial population $Q(t)$ which contains $N$ individuals
  3. compute $fit(h)$ for every individual $h \in Q(t)$

Main loop
  1. while the stopping condition is not satisfied do
     1.1 selection: select individuals $h$ from the population $Q(t)$ that will
         be directly inserted into the population $P(t+1)$
     1.2 crossover: choose pairs of individuals (with probability $K$) from
         the population $Q(t)$, perform crossover on each pair and insert the
         offsprings into the population $Q(t+1)$
     1.3 mutation: choose individuals $h$ (with probability $M$) from the
         population $Q(t+1)$ and mutate them
     1.4 assign t := t + 1
     1.5 compute $fit(h)$ for every individual $h \in Q(t)$
  2. return the individual $h$ with the highest value of $fit(h)$
```

Fig. 7. Genetic algorithm

Bayesian methods Bayesian methods are based on the Bayesian theorem on conditional probabilities. Although originating in the area of statistics, these methods are studied in relation to machine learning and knowledge discovery in databases.

Naive Bayesian classifier is based on the assumption that the evidences (input attributes) $E_1, ..., E_K$ are conditionally independent for a given hypothesis (class) H_j. This simplified assumption allows us to compute the posterior probability of hypothesis H_j based on all evidences $E_1, ..., E_K$ as

$$P(H_j|E_1,\ldots,E_K) = \frac{P(H_j)P(E_1,\ldots,E_K|H_j)}{P(E_1,\ldots,E_K)} = \frac{P(H_j)\prod_{i=1}^{K} P(E_i|H_j)}{P(E_1,\ldots,E_K)}. \tag{19}$$

The numerator in formula 19 corresponds to the joint probability distribution of the naive model. Because this type of model is fixed (by the independence assumption), the learning naive Bayesian classifier fits into learning as an approximation paradigm. The probabilities $P(H_j)$ and $P(E_i|H_j)$ are estimated as relative frequencies from the training data in a similar way to the evaluation of association rules. So

$$P(H_j) = \frac{a+c}{n} \tag{20}$$

and

$$P(E_i|H_j) = \frac{a}{a+c}, \tag{21}$$

where a is the number of examples that fulfill both E_i and H_j, c is the number of examples that fulfill H_j but do not fulfill E_i, and n is the number of all examples in the training set. The probability $P(E_1, ..., E_K)$ is ignored as we are not interested in the exact value of $P(H_j|E_1, ..., E_K)$ but want to know for which hypothesis (class) H_l is the posterior probability maximal.

Bayesian networks offer a more correct solution, that does not assume any independence in advance (see e.g. Jensen, (1996)). This type of models is again based on the Bayesian theorem but we now compute the joint probability distribution as

$$P(u_1, \ldots, u_K) = \prod_{i=1}^{K} P(u_i|parents(u_i)) \tag{22}$$

where u_i are variables (nodes in the Bayesian network) and $parents(u_i)$ are all nodes that directly influence (condition) the node u_i and are thus connected to node u_i in a graph representing the network. When learning Bayesian networks from data, we must infer both the structure of the model (the graph) and the parameters (conditional probabilities) and thus search the space of possible network topologies.

Instance-based learning Instance-based learning or memory-based learning is a family of learning algorithms that, instead of performing explicit generalization, compare instances of new problems with instances seen in training, which have been stored in memory. The crucial concept in instance-based learning is the distance (resp. the similarity) between examples. If all input attributes are numerical, we can easily use some of the metrics known from cluster analysis such as Eucleidian distance

$$d_e(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{\sum_{j=1}^{m} \delta_e(x_{1j}, x_{2j})}, \quad \delta_e(x_{1j}, x_{2j}) = (x_{1j} - x_{2j})^2, \tag{23}$$

or Hamming distance (also called city-block resp. Manhattan distance)

$$d_h(\mathbf{x}_1, \mathbf{x}_2) = \sum_{j=1}^{m} \delta_h(x_{1j}, x_{2j}), \quad \delta_e(x_{1j}, x_{2j}) = |x_{1j} - x_{2j}|^2 \tag{24}$$

If the attributes are categorial, the distance measure must be defined in a different way; the simplest solution, the overlap metrics

$$d_o(\mathbf{x}_1, \mathbf{x}_2) = \sum_{j=1}^{m} \delta_o(x_{1j}, x_{2j}), \quad \delta_o(x_{1j}, x_{2j}) = \begin{cases} 1 & \text{iff} \quad x_{1j} \neq x_{2j} \\ 0 & \text{iff} \quad x_{1j} = x_{2j} \end{cases} \tag{25}$$

is too strict and cannot express subtle differences between examples.

During learning, "suitable" examples are stored to build the model. There are a number of strategies for choosing these examples. In simplest case, every training example is stored; this method is used in the IB1 algorithm (Aha et al. (1991)). A more sophisticated way is to only store examples that are wrongly classified by the current model; this method which corresponds to a heuristic top-down search (wrongly classified example creates a new cluster) can be found in IB2 and IB3 algorithms (also Aha et al. (1991)).

We can also generalize the examples to find some representatives of the respective clusters. Cluster analysis, which can be used in such a case offers two basic approaches: hierarchical clustering (where the number of resulting clusters is not known in advance) can be understood as search, while the k-mean method (where the number of clusters is an input parameter) can be understood as approximation.

3 GUHA Method and LISp-Miner System

A we saw in Sect. 2.2, the term association rule was coined by R. Agrawal in the early 90s and the idea of association rules was later generalized to any data in the tabular, attribute-value form. An association rule is understood as the relation between conjunctions of attribute-value pairs (categories) called antecedent and succedent. There are two basic characteristics of an association rule – *support* and *confidence*.

We must emphasize that the concept of association rules was introduced and studied much earlier than the early 90s. An association rule in the form of a relation between two conjunctions of predicates was introduced and studied in 1968 (Hájek (1968)). The relation was based on Fisher's test. It was done within the framework of the development of the GUHA method of mechanized hypothesis formation (Hájek et al. (1966)).

A milestone in the GUHA method development was the monograph (Hájek and Havránek (1978)), which introduces the general theory of mechanized hypothesis formation based on mathematical logic and statistics. Association rules defined and studied in this book are relations between two general Boolean attributes derived from the columns of an analyzed data matrix. Various types of relations of Boolean attributes are used including relations corresponding to statistical hypothesis tests. However, these relations between Boolean attributes are not called association rules even if the GUHA procedure for mining them is called ASSOC (Hájek (1984)). The concept of association rules has been used for patterns mined by the GUHA ASSOC procedure since the term association rule was introduced in the 90s.

The GUHA ASSOC procedure was implemented several times and applied many times, see e.g. (Hájek (1978)), (Hájek (1981)), (Rauch and Tomečková (2007)). Its last and most frequently used implementation is the 4ft-Miner procedure (Rauch and Šimůnek (2005)). The boom in association rules in the 1990s was the start of a new era in the study of the theoretical properties of association rules. The result could be understood as the logic of association rules (Rauch (2005)) which has various practical applications.

The implementation of the ASSOC procedure does not use the apriori algorithm. It is based on a representation of analyzed data by bit-strings (Rauch and Šimůnek (2005)). This makes it possible to deal with general Boolean attributes, not only attribute-value pairs. A system of software tools for dealing with suitable bit-strings was developed when implementing the 4ft-Miner procedure. It made it possible to suggest and implement six additional GUHA mining procedures for various types of patterns not only concerning Boolean attributes but also general multivalued attributes. Both the 4ft-Miner procedure and six additional GUHA procedures are involved in the LISp-Miner system (Šimůnek (2003)), (Rauch and Šimůnek (2005 A)).

The goal of this chapter is to give an overview of the important features of the GUHA method, to introduce the 4ft-Miner procedure and the logic of association rules and to outline the main features of the six additional GUHA procedures in the LISp-Miner system. This is done in sections 3.1 – 3.5.

3.1 GUHA Method

The monograph (Hájek and Havránek (1978)) opens with two questions:

Q$_1$: Can computers formulate and justify *scientific* hypotheses?
Q$_2$: Can they comprehend empirical data and process it rationally, using the apparatus of modern mathematical logic and statistics to try to produce a rational image of the observed empirical world?

A theory developed in the monograph to answer these questions is based on a scheme of inductive inference:

$$\frac{\textit{theoretical assumptions, observational statement}}{\text{theoretical statement}}\ .$$

This means that if we accept theoretical assumptions and verify a particular statement about observed data, we accept the conclusion - a theoretical statement. It is crucial that an intelligent statement about observed data leads to theoretical conclusions, not the data itself.

The rational inductive inference rules bridging the gap between observational and theoretical statements are based on statistical approaches, i.e. estimates of various parameters or statistical hypothesis tests are used. A theoretical statement is justified if a condition concerning estimates of parameters used or a statement given by a statistical hypothesis test is satisfied in the analyzed data. This leads to special logical calculi formulas which correspond to statements about the observed data based on statistical approaches. Such calculi are called observational calculi. Concise information on this approach can also be found in (Rauch (2005).

Using induction rules based on statistical approaches usually means that there is 1:1 correspondence between observational and theoretical statements.

Thus the task to find interesting theoretical statements can be solved by finding interesting observational statements. GUHA is a method for suggesting interesting observational statements (Hájek and Havránek (1978)). It is realized by using GUHA procedures.

The input of the GUHA procedure consists of analyzed data and a simple definition of a usually very large set of relevant (i.e. potentially interesting) patterns. The GUHA procedure automatically generates each particular pattern and tests if it is true in the analyzed data. The output of the procedure consists of all prime patterns. The pattern is prime if it is true in the analyzed data and if it does not immediately follow from other more simple output patterns. The most commonly used GUHA procedure is the ASSOC procedure which mines for association rules – relations between two general Boolean attributes derived from the columns of an analyzed data matrix.

3.2 Association rules

We understand the association rule as the expression $\varphi \approx \psi$ where φ and ψ are Boolean attributes derived from the columns of an analyzed data matrix. An example of such a data matrix is the data matrix $\mathcal{M}$ in Fig. 8.

object	attributes				examples of Boolean attributes			
i.e.	i.e. columns of $\mathcal{M}$				basic		derived	
row	A_1	A_2	$\dots$	A_K	$A_1(3)$	$A_2(5,7)$	$\neg A_1(3)$	$A_1(3) \wedge A_2(5,7)$
o_1	3	5	$\dots$	6	1	1	0	1
o_2	3	6	$\dots$	7	1	0	0	0
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
o_{n-1}	4	7	$\dots$	4	0	1	1	0
o_n	1	2	$\dots$	1	0	0	1	0

Fig. 8. Data matrix $\mathcal{M}$ and examples of Boolean attributes

Rows of $\mathcal{M}$ correspond to observed objects, there are objects $o_1, \dots, o_n$ Columns of the data matrix correspond to properties of observed objects, they are called *attributes*. There are attributes $A_1, \dots, A_K$ – columns of $\mathcal{M}$. Possible values of attributes are called *categories*. There are a finite number of categories for each attribute.

Basic Boolean attributes are created first. The basic Boolean attribute is an expression $A(\alpha)$ where $\alpha \subset \{a_1, \dots a_k\}$ and $\{a_1, \dots a_k\}$ is the set of all possible values of column A. The basic Boolean attribute $A(\alpha)$ is true in row o of $\mathcal{M}$ if $a \in \alpha$ where a is the value of the attribute A in row o. There are two examples of basic Boolean attribute in Fig. 8 – $A_1(3)$ and $A_2(5,7)$. The basic Boolean attribute $A_1(3)$ is true in row o_1 (we write 1 in the corresponding row and column) and it is false in row o_n (we write 0), analogously for the basic Boolean attribute $A_2(5,7)$ and additional rows.

Boolean attributes φ and ψ used in the association rule $\varphi \approx \psi$ are either basic or derived Boolean attributes. *Derived Boolean attributes* are created from basic Boolean attributes using connectives $\vee$, $\wedge$ and $\neg$ in the usual way. There are derived Boolean attributes in Fig. 8, $\neg A_1(3)$ and $A_1(3) \wedge A_2(5,7)$.

The symbol $\approx$ used in the association rule $\varphi \approx \psi$ is called the *4ft-quantifier*. It defines the relation between the Boolean attributes φ and ψ derived from the columns of the analyzed data matrix $\mathcal{M}$. A condition concerning all possible contingency tables of φ and ψ is associated with each 4ft-quantifier $\approx$. Each contingency table of φ and ψ is related to a given data matrix. The contingency table of φ and ψ in data matrix $\mathcal{M}$ is denoted as $4ft(\varphi,\psi,\ \mathcal{M})$. It is a quadruple $\langle a,b,c,d\rangle$ of natural numbers, where a is

$\mathcal{M}$	ψ	$\neg\psi$
φ	a	b
$\neg\varphi$	c	d

Table 2. 4ft table 4ft(φ,ψ, $\mathcal{M}$) of φ and ψ in $\mathcal{M}$

the number of rows of $\mathcal{M}$ satisfying both φ and ψ, b is the number of rows satisfying φ and not satisfying ψ, etc., see Tab. 2.

The rule $\varphi \approx \psi$ is *true in the data matrix* $\mathcal{M}$ if the condition associated with $\approx$ is satisfied in the contingency table $4ft(\varphi,\psi,\ \mathcal{M})$, otherwise $\varphi \approx \psi$ is *false in the data matrix* $\mathcal{M}$. If the condition corresponding to 4ft-quantifier $\approx$ is satisfied in the contingency table $\langle a,b,c,d\rangle$ we write $\approx (a,b,c,d) = 1$, otherwise we write $\approx (a,b,c,d) = 0$. There are lot of 4ft-quantifiers defined and studied in relation to the GUHA method. Several examples are shown in Tab. 3, we sometimes use $r = a+b$, $k = a+c$ and $n = a+b+c+d$.

We assume $0 < p \leq 1$, $0 < \alpha < 0.5$, $0 < q$ and $B > 0$, χ^2_α is the $(1-\alpha)$ quantile of the χ^2 distribution function. The quantifiers $\Rightarrow_{p,B}$, $\Rightarrow^!_{p,\alpha,B}$ $\sim_{\delta,B}$, $\sim^2_{\alpha,B}$ are defined in (Hájek and Havránek (1978)), the quantifiers $\Leftrightarrow_{p,B}$, $\Leftrightarrow^!_{p,\alpha,B}$, $\equiv_{p,B}$ and $\equiv^!_{p,\alpha,B}$ are defined in (Hájek et al. (1983)) and the quantifier $\approx^+_{\alpha,B}$ is defined in (Rauch (2005)).

Note that the 4ft-quantifier $\Rightarrow_{p,B}$ says that at least $100p$ per cent of rows of the analyzed data matrix $\mathcal{M}$ satisfying φ also satisfy ψ and that there are at least B rows of $\mathcal{M}$ satisfying both φ and ψ. Fisher's quantifier $\approx_{\alpha,B}$ corresponds to the statistical test (on level α) of the null hypothesis of independence of φ and ψ against the alternative one of positive dependence. The 4ft-quantifier $\sim^+_{p,B}$ means that there is at least $100p$ per-cent more objects satisfying ψ in the rows of the analyzed data matrix $\mathcal{M}$ satisfying φ than

4ft quantifier $\approx$		
Name	Symbol	$\approx(a,b,c,d)=1$ iff
Founded implication	$\Rightarrow_{p,B}$	$\frac{a}{a+b} \geq p \wedge a \geq B$
Lower critical implication	$\Rightarrow^{!}_{p,\alpha,B}$	$\sum_{i=a}^{r} \binom{r}{i}^{i} (1-p)^{r-i} \leq \alpha \wedge a \geq B$
Founded double implication	$\Leftrightarrow_{p,B}$	$\frac{a}{a+b+c} \geq p \wedge a \geq B$
Lower critical double implication	$\Leftrightarrow^{!}_{p,\alpha,B}$	$\sum_{i=a}^{n-d} \binom{n-d}{i} p^i (1-p)^{n-d-i} \leq \alpha \wedge a \geq B$
Founded equivalence	$\equiv_{p,B}$	$\frac{a+d}{a+b+c+d} \geq p \wedge a \geq B$
Lower critical equivalence	$\equiv^{!}_{p,\alpha,B}$	$\sum_{i=a+d}^{n} \binom{n}{i} p^i (1-p)^{n-i} \leq \alpha \wedge a \geq B$
Simple deviation	$\sim_{\delta,B}$	$\frac{ad}{bc} > e^{\delta} \wedge a \geq B$
Fisher	$\approx_{\alpha,B}$	$\sum_{i=a}^{\min(r,k)} \frac{\binom{k}{i}\binom{n-k}{r-i}}{\binom{r}{n}} \leq \alpha \wedge a \geq B$
χ^2 quantifier	$\sim^{2}_{\alpha,B}$	$\frac{(ad-bc)^2}{rkls} n \geq \chi^2_{\alpha} \wedge a \geq B$
Above average dependence	$\sim^{+}_{q,B}$	$\frac{a}{a+b} \geq (1+q) \frac{a+c}{a+b+c+d} \wedge a \geq B$

Table 3. Examples of 4ft-quantifiers

among all rows of $\mathcal{M}$ and that there are at least B objects satisfying both φ and ψ, similarly for additional 4ft-quantifiers.

3.3 GUHA 4ft-Miner procedure

The GUHA 4ft-Miner procedure (Rauch and Šimůnek (2005) mines for association rules $\varphi \approx \psi$ and for conditional association rules $\varphi \approx \psi/\chi$. The intuitive meaning of $\varphi \approx \psi/\chi$ is that the Boolean attributes φ and ψ are in a relation given by the 4ft-quantifier $\approx$ when the condition given by the Boolean attributes χ is satisfied. The Boolean attributes φ, ψ and χ are derived from the columns of the analyzed data matrix $\mathcal{M}$.

The conditional association rule $\varphi \approx \psi/\chi$ is *true in the data matrix* $\mathcal{M}$ if there is both a row of $\mathcal{M}$ satisfying χ and if the association rule $\varphi \approx \psi$ is true in the data matrix $\mathcal{M}/\chi$, otherwise $\varphi \approx \psi$ is *false in data matrix* $\mathcal{M}$. The data matrix $\mathcal{M}/\chi$ consists of all rows of the data matrix $\mathcal{M}$ satisfying χ.

The 4ft-Miner procedure is part of the LISp-Miner system (Šimůnek (2003)). 4ft-Miner input consists of the analyzed data matrix and several parameters defining the set of association rules to be automatically generated and tested.

The analyzed data matrix is created from a database table. Any database accessible by ODBC can be used. The columns of the database table are transformed into attributes (i.e. columns) of the analyzed data matrix. There is a special *DataSource* module in the LISP-Miner system intended for these transformations. For example, it is possible to use the original values from a given column of the database table as the categories of the defined attribute. It is also possible to define new categories as intervals of a given length. Moreover the DataSource module can generate the given number of equifrequency intervals as new categories.

There are very fine possibilities for defining the set of association rules to be automatically generated and verified, which are described below. The 4ft-Miner procedure generates and verifies all defined association rules. However, the apriori algorithm introduced in Section 2.2 is not used. Implementation is based on a representation of analyzed data by bit-strings and several optimization tools.

The output of the 4ft-Miner procedure consists of all *prime association rules*. The association rule is prime if it is both true in the analyzed data matrix and if it does not immediately follow from other simpler output association rules. A precise definition of the prime association rule does not fall with the scope of this chapter. However, note that it depends on the properties of the 4ft-quantifier used. Please note that the 4ft-Miner procedure also deals with missing information (Hájek and Havránek (1978)). Again, details dealing with missing information do not fall within the scope of this chapter. There are also many possibilities for filtering and sorting the output set of association rules.

4ft-Miner input The 4ft-Miner procedure mines for association rules of the form $\varphi \approx \psi$ and for conditional association rules of the form $\varphi \approx \psi/\chi$. The Boolean attribute φ is called *antecedent*, ψ is called *succedent* and χ is called *condition*. A definition of the set of relevant questions consists of

- the definition of a *set of relevant antecedents*
- the definition of a *set of relevant succedents*
- the definition of a *set of relevant conditions*
- the definition of the 4ft-quantifier $\approx$.

Antecedent φ is as a conjunction $\varphi = \varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_k$ where $\varphi_1, \varphi_2, \ldots, \varphi_k$ are *partial antecedents*. Each φ_i is chosen from one *set of relevant partial antecedents*. A partial antecedent is a conjunction or disjunction of literals. A literal is a basic Boolean attribute $A(\alpha)$ or negation $\neg A(\alpha)$ of a basic Boolean attribute, see Section 3.2. The set α is a *coefficient* of the literals $A(\alpha)$ and $\neg A(\alpha)$. The *length of the partial antecedent* is the number of literals in the partial antecedent. The length of the antecedent is the sum of the lengths of partial antecedents in the antecedent.

The definition of the set of relevant antecedents is given by a minimal and a maximal length of antecedent and at least one definition of the set of relevant partial antecedents. The set of partial antecedents is given in the following manner:

- one of the options of *a conjunction of literals* or *disjunction of literals*
- the minimum and maximum length of the partial antecedent is defined
- a set of attributes from which literals will be generated is given
- some attributes can be marked as *basic*, each partial antecedent must then contain at least one basic attribute

- a simple definition of the set of all literals to be generated is given for each attribute
- *classes of equivalence* can be defined, each attribute belongs to a maximum of one class of equivalence; no partial antecedent can contain two or more attributes from one class of equivalence.

The *length of the literal* is the number of categories in its coefficient. The set of all literals to be generated for a particular attribute is given by:

- *the type of coefficient*; there are seven types of coefficients: *subsets*, *intervals*, *cyclic intervals*, *left cuts*, *right cuts*, *cuts*, *one particular category*
- the minimum and the maximum length of the literal
- positive/negative literal option:
  - generate only positive literals
  - generate only negative literals
  - generate both positive and negative literals

We use the attribute A with categories {1, 2, 3, 4, 5} to give examples of particular types of coefficients:

- *subsets*: definition of subsets of length 2-3 gives literals A(1,2), A(1,3), A(1,4), A(1,5), A(2,3), . . . , A(4,5), A(1,2,3), A(1,2,4), A(1,2,5), A(2,3,4), . . . , A(3,4,5)
- *intervals*: definition of intervals of length 2-3 gives literals A(1,2), A(2,3), A(3,4), A(4,5), A(1,2,3), A(2,3,4) and A(3,4,5)
- *cyclic intervals*: definition of intervals of length 2-3 gives literals A(1,2), A(2,3), A(3,4), A(4,5), A(5,1) A(1,2,3), A(2,3,4), A(3,4,5), A(4,5,1) and A(5,1,2)
- *left cuts*: definition of left cuts with a maximum length of 3 defines literals A(1), A(1,2) and A(1,2,3)
- *right cuts*: definition of right cuts with a maximum length of 4 defines literals A(5), A(5,4), A(5,4,3) and A(5,4,3,2)
- *cuts* means both left cuts and right cuts
- *one category* means one literal with one chosen category, e.g. A(2).

We must emphasize that even if the coefficient type *subsets* covers all the other types of literals it still makes sense to use specific subtypes. For example, there are more than 10^{13} literals - subsets with length 10 for the attribute *Age* with 100 categories. At the same time, there are only 91 categories of the type *intervals* with length 10.

Definitions of the set of relevant succedents and the set of relevant conditions are analogous. The set of relevant antecedents, the set of relevant succedents and the set of relevant conditions can overlap. However, association rules with more than one literal created from the same attribute are not generated. There are 16 various 4ft-quantifiers, some of them are shown in Tab. 3.

3.4 LISp-Miner System

Six additional GUHA procedures are involved in the LISp-Miner system: SD4ft-Miner, Ac4ft-Miner, KL-Miner, CF-Miner, SDKL-Miner, and SDCF-Miner. They use software tools for dealing with suitable bit-strings developed for the 4ft-Miner procedure. We will outline the main features of these procedures.

SD4ft-Miner The SD4ft-Miner procedure mines for SD4ft-patterns of the form $\alpha \bowtie \beta : \varphi \approx^{SD} \psi/\gamma$. Here α, β, γ, φ, and ψ are Boolean attributes derived from the columns of the analyzed data matrix $\mathcal{M}$. The attributes α and β define two subsets of rows of $\mathcal{M}$. The attribute γ defines a condition. The attributes φ and ψ are *antecedent* and *succedent* of the association rule $\varphi \approx \psi$ in question. The SD4ft-pattern $\alpha \bowtie \beta : \varphi \approx^{SD} \psi/\gamma$ means that the subsets given by the Boolean attributes α and β differ for the validity of association rule $\varphi \approx \psi$ when the condition given by the Boolean attribute γ is satisfied. A measure of difference is defined by the symbol $\approx^{SD}$ which is called the *SD4ft-quantifier*.

The SD4ft-quantifier corresponds to a condition concerning two 4ft-tables $\langle a, b, c, d\rangle$ and $\langle a', b', c', d'\rangle$. An example is the SD4ft-quantifier $\Rightarrow^D_p$ defined by the condition $|\frac{a'}{a'+b'} - \frac{a}{a+b}| \geq p$. The SD4ft-pattern $\alpha \bowtie \beta : \varphi \approx^{SD} \psi/\gamma$ is true in the data matrix $\mathcal{M}$ if the condition corresponding to $\approx^{SD}$ is satisfied for the 4ft-tables $4ft(\varphi, \psi, \mathcal{M}/(\alpha \wedge \gamma))$ and $4ft(\varphi, \psi, \mathcal{M}/(\beta \wedge \gamma))$. The SD4ft-pattern $\alpha \bowtie \beta : \varphi \Rightarrow^D_p \psi/\gamma$ thus means that the absolute value of the difference of confidence of association rule $\varphi \approx \psi$ in data matrix $4ft(\mathcal{M}/(\alpha \wedge \gamma)$ and the confidence of this association rule in data matrix $4ft(\mathcal{M}/(\beta \wedge \gamma)$ is at least p. An example of the application of the SDR4ft-Miner procedure is shown in (Rauch and Šimůnek (2009)).

Ac4ft-Miner The Ac4ft-Miner procedure mines for G-action rules that are inspired by action rules defined in (Ras and Wieczorkowska (2000)). There are only a few, but there has been promising experience with Ac4ft-Miner (Rauch and Šimůnek (2009 A)). A precise description of G-action rules is outside the scope of this chapter. We will only outline a simple example. Let us take a data matrix $\mathcal{P}$ with rows corresponding to patients. We assume $\mathcal{P}$ has columns $S_1, \ldots, S_u$ corresponding to *stable attributes* and columns $F_1, \ldots, F_v$ corresponding to *flexible attributes*. The attribute is stable if its value cannot be changed, e.g. *Sex*, *Year of birth* or *Blood type*. The attribute is flexible if its value can be subject to change, e.g. *BMI* (i.e. *Body Mass Index*) or *Blood pressure*. Let us assume that both *BMI* and *Blood pressure* have categories *low*, *avg* (i.e. *average*), *high*. Then the expression

$$\mathit{Sex}(\mathit{male}) \wedge [\mathit{BMI}(\mathit{high} \rightarrow \mathit{avg})] \Rightarrow_{0.9 \rightarrow 0.8, 50, 50} [\mathit{Blood\ pressure}(\mathit{high} \rightarrow \mathit{avg})]$$

is an example of a G-action rule. Here *Sex*(*male*) is a *stable antecedent*, [*BMI*(*high* $\rightarrow$ *avg*)] is a *change of antecedent* and *BMI*(*high* $\rightarrow$ *avg*) is a *change of coefficient*. Generally, the change of antecedent can be built from several changes of coefficients in the same way as the antecedent of the association rule is built from literals. The same is true for a *change of succedent*. Generally, there can also be a stable succedent.

The above G-action rule says what can happen to male patients described in the analyzed data matrix $\mathcal{P}$ if they change their BMI from high to average. The effect is described by two association rules - the rule $\mathcal{R}_I$: *Sex*(*male*) $\wedge$ *BMI*(*high*) $\Rightarrow_{0.9,50}$ *Blood pressure*(*high*) and by the rule $\mathcal{R}_F$: *Sex*(*male*) $\wedge$ *BMI*(*avg*) $\Rightarrow_{0.8,50}$ *Blood pressure*(*avg*). The rule $\mathcal{R}_I$ describes the initial situation before the change i.e. 90 % of male patients with high BMI also have high blood pressure. The rule $\mathcal{R}_F$ suggests the possible final situation after patients change their BMI from high to average. The effect is that 80 % of them have average blood pressure.

We must emphasize that both rules $\mathcal{R}_I$ and $\mathcal{R}_F$ are evaluated on one matrix $\mathcal{P}$, the sets of rows (i.e. patients) satisfying the antecedents of $\mathcal{R}_I$ and $\mathcal{R}_F$ differ. The G-action rule suggest an action and indicates its possible effect. We have to be careful when interpreting the results of the Ac4ft-Miner procedure. We should note that the Ac4ft-Miner is still under development but seems to be a useful data mining tool.

Patterns mined by the procedures KL-Miner, CF-Miner, SDKL-Miner, and SDCF-Miner are evaluated on the basis of more complex contingency tables than the contingency tables $4ft(\varphi, \psi, \mathcal{M})$ of the Boolean attributes φ and ψ in data matrix $\mathcal{M}$. KL-Miner and SDKL-Miner procedures deal with KL-tables and CF-Miner and SDCF-Miner procedures deal with CF-tables, see Fig. 9.

$$\begin{array}{c|ccc} \mathcal{M} & c_1 & \dots & c_L \\ \hline r_1 & n_{1,1} & \dots & n_{1,L} \\ \vdots & \vdots & \ddots & \vdots \\ r_K & n_{K,1} & \dots & n_{K,L} \end{array}$$

$KL(R, C, \mathcal{M})$

$$\begin{array}{c|ccc} \mathcal{M} & r_1 & \dots & r_K \\ \hline & n_1 & \dots & n_K \end{array}$$

$CF(R, \mathcal{M})$

Fig. 9. KL-table $KL(R, C, \mathcal{M})$ and CF-table $CF(R, \mathcal{M})$

We assume the attribute R has the categories $r_1, \dots, r_K$ and the attribute C has the categories $c_1, \dots, c_L$. The expression $n_{k,l}$ in the KL-table $KL(R, C, \mathcal{M}) = \{n_{k,l}\}_{k=1,\dots K}^{l=1,\dots L}$ denotes the number of rows of the data matrix $\mathcal{M}$ for which the value of attribute R is r_k and the value of attribute C is c_l. The expression n_k in the CF-table $CF(R, \mathcal{M}) = \langle n_1, \dots n_K \rangle$ denotes the number of rows in the data matrix $\mathcal{M}$ for which the value of attribute R is r_k.

KL-Miner, CF-Miner, SDKL-Miner, and SDCF-Miner procedures are briefly introduced in (Rauch and Šimůnek (2005 A)). The KL-Miner procedure is described in more detail in (Rauch et al. (2005)), (Berka (2009)).

KL-Miner The KL-Miner procedure mines for KL-patterns $R \sim C/\gamma$. The *KL-pattern* $R \sim C/\gamma$ means that the attributes R and C are in a relation given by the symbol $\sim$ when the condition given by the Boolean attribute γ is satisfied. The symbol $\sim$ is called the *KL-quantifier*. It corresponds to a condition concerning the *KL-table* $KL(R, C, \mathcal{M}')$ for the attributes R and C in the data matrix $\mathcal{M}'$ in question. The *KL-pattern* $R \sim C/\gamma$ is true in the data matrix $\mathcal{M}$ if the condition corresponding to $\sim$ is satisfied in the KL-table $KL(R, C, \mathcal{M}/\gamma)$ for the attributes R and C in the data matrix $\mathcal{M}/\gamma$. We must remember that the data matrix $\mathcal{M}/\gamma$ consists of all rows of $\mathcal{M}$ satisfying γ.

An example of the KL-quantifier is Kendall's quantifier $\sim_p^{Ken}$ with $0 \leq p \leq 1$ defined by the condition $|\tau_b| \geq p$. Here τ_b is Kendall's coefficient, $\tau_b = \frac{2(P-Q)}{\sqrt{\left(n^2-\sum_k n_{k,*}^2\right)\left(n^2-\sum_l n_{*,l}^2\right)}}$, where $P = \sum_{k,l} n_{k,l} \sum_{i>k} \sum_{j>l} n_{i,j}$, $Q = \sum_{k,l} n_{k,l} \sum_{i>k} \sum_{j<l} n_{i,j}$, $n_{k,*} = \sum_l n_{k,l}$, and $n_{*,l} = \sum_k n_{k,l}$. Kendall's quantifier is used for ordinal attributes. τ_b takes values from $\langle -1, 1 \rangle$ with the following interpretation: $\tau_b > 0$ indicates positive ordinal dependence[1], $\tau_b < 0$ indicates negative ordinal dependence, $\tau_b = 0$ indicates ordinal independence, and $|\tau_b = 1|$ indicates that C is a function of R.

CF-Miner The CF-Miner procedure mines for CF-patterns of the form $\sim R/\gamma$. The *CF-pattern* $\sim R/\gamma$ means that frequencies of categories of the attribute R satisfy the condition given by the symbol $\sim$ when an other condition given by the Boolean attribute γ is satisfied. The symbol $\sim$ is called the *CF-quantifier* here. The CF-quantifier corresponds to a condition concerning the *CF-table* $CF(R, \mathcal{M}')$ for the attribute R in the data matrix $\mathcal{M}'$ in question. The *CF-pattern* $\sim R/\gamma$ is true in the data matrix $\mathcal{M}$ if the condition corresponding to $\sim$ is satisfied in the CF-table $CF(R, \mathcal{M}/\gamma)$ for the attribute R in the data matrix $\mathcal{M}/\gamma$.

SDKL-Miner and SDCF-Miner SDKL-Miner and SDCF-Miner procedures are derived from KL-Miner and CF-Miner procedures respectively in a similar manner to the way the SD4ft-Miner procedure is derived from the 4ft-Miner procedure.

[1] i.e. high values of C often coincide with high values of R and low values of C often coincide with low values of R

3.5 Logical Calculi of Association Rules

Observational calculi are special logical calculi introduced in (Hájek and Havránek (1978)). Their formulas correspond to statements about the analyzed data, see section 3.1. Observational predicate calculi are modifications of classical predicate calculi – only finite models are allowed and generalized quantifiers are added. These calculi are defined in (Hájek and Havránek (1978)), there are open and closed formulas with values defined in Tarski style. Informally speaking, observational monadic predicate calculus with 4ft-quantifiers as the only generalized quantifiers can be understood as predicate calculus of associational rules. We can obtain a logical calculus of association rules (Rauch (2005)) with formulas - association rules $\varphi \approx \psi$ defined in section 3.2 by additional simple modification of the predicate calculus of associational rules.

There are theoretically interesting and practically important results concerning logical calculus of association rules. The results are related to classes of association rules. The important classes of association rules are implicational rules and equivalence rules (i.e. associational) defined in (Hájek and Havránek (1978)), double implicational rules, Σ-double implicational rules and Σ-equivalence rules (Rauch (2005)).

The results deal with missing information, the definability of 4ft-quantifiers in classical predicate calculus and tables of critical frequencies making it possible to avoid complex computation when verifying rules with 4ft-quantifiers corresponding to statistical hypothesis test (e.g. quantifier of lower critical double implication $\Leftrightarrow^{!}_{p,\alpha,B}$ and Fisher's quantifier $\approx_{\alpha,B}$ defined in Tab. 3), see (Rauch (2008)).

Very important results concern deduction rules, for all practically important 4ft-quantifiers there is a criterion making it possible to decide if the deduction rule $\frac{\varphi\approx\psi}{\varphi'\approx\psi'}$ concerning association rules $\frac{\varphi\approx\psi}{\varphi'\approx\psi'}$ and $\frac{\varphi\approx\psi}{\varphi'\approx\psi'}$ is correct or not. Such deduction rules can be used e.g. to deal with domain knowledge, see (Rauch (2009)).

Acknowledgement: The work described here has been supported by the project 201/08/0802 of the Czech Science Foundation.

References

AHA, D.W., KIBLER, D. and ALBERT, M. K. (1991): Instance-based learning algorithms. *Machine Learning, 6, 37-66.*

AGRAWAL, R., IMIELINSKI, T. and SWAMI, A. (1993): Mining associations between sets of items in massive databases. In: *Proc. of the ACM-SIGMOD 1993 Int. Conference on Management of Data*. Washington D.C., 207-216.

BERKA, P. and IVÁNEK, J. (1994): Automated Knowledge Acquisition for PROSPECTOR-like Expert Systems. In: Bergadano, deRaedt (Eds.): *Proc. European COnference on Machine Learning ECML'94*. LNAI 784, Springer, 339-342.

BERKA, P., RAUCH, J., and TOMEČKOVÁ, M. (2009): Data Mining in Atherosclerosis Risk Factor Data. In: BERKA, P., RAUCH, J. and ZIGHED, D. A.: *Data Mining and Medical Knowledge Management: Cases and Applications*. Hershey : IGI Global, 376–397

BIGGS, D., deVILLE, B. and SUEN, E. (1991): A method of choosing multiway partitions for classification and decision trees. *Journal of Applied Statistics, Vol. 18, No. 1, 49-62.*

BREIMAN, L., FRIEDMAN, J.H., OLSHEN, R.A. and STONE, P.J. (1984): *Classification and Regression Trees.* Wadsworth.

BRŮHA, I., and BERKA, P.: Discretization and Fuzzification of Numerical Attributes in Attribute-Based Learning. In: Szczepaniak, P.S., Lisboa, P.J.G. and Kacprzyk, J., (Eds.): *Fuzzy Systems in Medicine*. Physica Verlag, 2000, 112-138.

CLARK, P. and NIBLETT, T. (1989): The CN2 induction algorithm. *Machine Learning, 3, 261-283.*

HÁJEK P. (1968): Problem of a general conception of the GUHA method (in Czech) *Kybernetika* 6, 505–515.

HÁJEK P. (guest editor) (1978): *International Journal of man-Machine Studies*, vol. 10, No 1 (special issue on GUHA).

HÁJEK P. (guest editor) (1981): *International Journal for Man-Machine Studies*, vol. 15, No 3 (second special issue on GUHA).

HÁJEK P. (1984): The new version of the GUHA procedure ASSOC. *Proceedings of COMPSTAT 1984*, pp. 360 – 365

HÁJEK P., HAVEL I., CHYTIL M. (1966): The GUHA method of automatic hypothesis determinantion. *Computing* 1, pp. 293–308

HÁJEK P., HAVRÁNEK T., CHYTIL M. (1983): *GUHA Method.* Academia, 1983, Prague (in Czech)

HÁJEK P., HAVRÁNEK T. (1978): Mechanising Hypothesis Formation – Mathematical Foundations for a General Theory. Springer, Berlin Heidelberg New York

CHAKRABRATI, S., COX, E., FRANK, E., HAN, J., JIANG, X., KAMBER, M., LIGHTSTONE, S., NADEAU, T., NEAPOLITAN, R., PYLE, D., REFAAT, M., SCHNEIDER, M., TEOREY, T., WITTEN, I. (2008): *Data Mining: Know it All.* Elsevier.

CHAPMAN, P., CLINTON, J., KERBER, R., KHABAZA, T., REINARTZ, T., SHEARER, C. and WIRTH, R. (2000): *CRISP-DM 1.0 Step-by-step data mining guide.* SPSS Inc.

DEJONG, K.A., SPEARS, W.M. and GORDON, D.F. (1993): Using genetic algorithms for concept learning. *Machine Learning 13, 161-188.*

DODGE, Y. (2003): *The Oxford Dictionary of Statistical Terms.* Oxford University Press.

DUDA, R.O. and GASCHING, J.E. (1979): Model design in the Prospector consultant system for mineral exploration. In: D. Michie (Eds.): *Expert Systems in the Micro Electronic Age. Edinburgh University Press*, UK.

FAYYAD, U., PIATETSKY-SHAPIRO, G., SMYTH, P. and UTHURUSAMY, R. (Eds.) (1996): *Advances in Knowledge Discovery and Data Mining.* AAAI Press/MIT Press.

HAND, D. (1998): Data Mining: Statistics and More? *The American Statistician, Vol. 52, No. 2, 112-118.*

HAND, D., MANILLA, H. and SMYTH, P. (2002): *Principles of Data Mining.* MIT Press.

HECHT-NIELSEN, R. (1991): *Neurocomputing.* Addison Wesley.

JENSEN, F. (1996): *An introduction to bayesian networks.* UCL Press.

KRÁLÍK, P. and BRŮHA, I. (1998): Genetic Learner: Attribute-Based Rule Inducing Approach. In: *Proc. 4th Int. Conf. On Soft Computing Mendel98.* Brno.

MICHALSKI, R.S. (1969): On the Quasi-minimal solution of the general covering problem. In: *Proc. 5th Int. Symposium on Information Processing FCIP'69.* Bled, Yugoslavia, 125-128.

MITCHELL, T. (1997): *Machine Learning.* McGraw-Hill.

QUINLAN, J.R. (1986): Induction of decision trees. *Machine Learning, 1(1), 81-106.*

QUINLAN, J.R. (1993): *C4.5: Programs for machine learning.* Morgan Kaufman.

RAS, Z., WIECZORKOWSKA, A. (2000): Action-Rules: How to Increase Profit of a Company. In: ZIGHED, D. A., KOMOROWSKI, J., ZYTKOW, J. (Eds.) *Principles of Data Mining and Knowledge Discovery.* Springer-Verlag, 587–592

RAUCH, J. (2005): Logic of Association Rules. *Applied Intelligence* 22, 9-28.

RAUCH, J. (2008): Classes of Association Rules - an Overview. In Lin, T. Y. et al. (Eds.): *Datamining: Foundations and Practice.* Springer, Studies in Computational Intelligence, Volume 118, 283 – 297

RAUCH, J. (2009): Considerations on Logical Calculi for Dealing with Knowledge in Data Mining. In RAS, Z. W. and DARDZINSKA A. (Eds.): Advances in Data Management. Springer, 177 – 202

RAUCH, J. and ŠIMŮNEK, M. (2005): An Alternative Approach to Mining Association Rules. In: LIN, T. Y. et al. (Eds) *Data Mining: Foundations, Methods, and Applications.* Springer-Verlag, 219–238

RAUCH, J. and ŠIMŮNEK, M. (2005 A): GUHA Method and Granular Computing. In: HU X. et al. (Eds.). *Proceedings of IEEE conference Granular Computing 2005.* IEEE, pp. 630–635.

RAUCH, J., ŠIMŮNEK, M. and LÍN, V. (2005): Mining for Patterns Based on Contingency Tables by KL-Miner First Experience. In: LIN, T. Y. et al. (Eds.): *Foundations and Novel Approaches in Data Mining.* Springer-Verlag, 155-167

RAUCH, J. and ŠIMŮNEK, M. (2009): Dealing with Background Knowledge in the SEWEBAR Project. In: Berendt B. et al.: Knowledge Discovery Enhanced with Semantic and Social Information. Berlin, Springer-Verlag, 2009, pp. 89 – 106

RAUCH, J. and ŠIMŮNEK, M. (2009 A): Action Rules and the GUHA Method: Preliminary Considerations and Results. In: RAUCH, J. et al.(Eds.) *Foundations of Intelligent Systems* Berlin, Springer Verlag, 76 – 87.

RAUCH, J. and TOMEČKOVÁ, M. (2007): System of Analytical Questions and Reports on Mining in Health Data - a Case Study. In Roth J. et al. (Eds.) *Proceedings of IADIS European Conference Data Mining 2007.* IADIS Press

ŠIMŮNEK, M. (2003): Academic KDD Project LISp-Miner. In: ABRAHAM, A., FRANKE, K., KOPPEN, K. (Eds.). *Advances in Soft Computing Intelligent Systems Design and Applications.* Heidelberg : Springer-Verlag, 2003, s. 263-272

WITTEN, I. and FRANK, E. (2005): *Data Mining, Practical Machine Learning tools and Techniques with Java.* Morgan Kaufmann.

WOLPERT, D. (1996): The Lack of A Priori Distinctions between Learning Algorithms. *Neural Computation, 1341-1390.*