Knowledge EXplorer: a Tool for Automated Knowledge Acquisition from Data

Petr Berka[1]
Austrian Research Institute for Artificial Intelligence
Schottengasse 3, A-1010 Vienna, Austria
e-mail: berka@ai.univie.ac.at

Abstract

Knowledge EXplorer is a system for exploratory analysis and knowledge acquisition in categorial data. The knowledge acquisition component induces knowledge in the form of weighted rules. These rules can be directly used in an PROSPECTOR like expert system.

1 Introduction

The Knowledge EXplorer is a system for exploratory analysis and knowledge acquisition in categorial data which has been developed at the Dept. of Information and Knowledge Engineering of the Prague University of Economics [7]. The original implementation was done in Turbo Pascal for IBM PC's and compatibles. During my stay at the Austrian Research Institute for AI, substantial parts of this system were implemented in C for the Apollo workstations.

First part of this report describes basic ideas an algorithms of the system, the second part is a brief manual for the new implementation. In the description of the system I concentrated myself mainly on the knowledge acquisition component.

[1]Address for correspondence:
Dept. of Information and Knowledge Engineering, Prague University of Economics,
W. Churchill Sq. 4, 130 67 Prague, CR
e-mail: berka@nb.vse.cs

2 System description

Knowledge EXplorer is designed for

- systematic basic analysis of multidimensional categorial data to find relations of implications and equivalences between pairs of combinations of categories,
- rule (knowledge) base acquisition from multidimensional categorial data without expert.

The first type of tasks, also called *combinational data analysis* was inspired by the GUHA system, which was designed to find all relevant relations in data [6]. Knowledge EXplorer performs several tasks which differ according to desired type of relations:

- specific evaluation,
- complete exploration,
- analysis of conclusions,
- analysis of causes.

The basic idea of this tasks is to find all "interesting" relations in given data. What is interesting specifies the user by selecting the task and input parameters. All these tasks can be viewed as exploratory data analysis methods.

The idea of knowledge acquisition is to find minimal set of rules, which describes given data and which can directly be used for consultation. The knowledge acquisition component consists not only of learning algorithm but also of a testing module and a consultation module.

2.1 Basic notions

This section describes the terminology used in Knowledge EXplorer.

2.1.1 Data

Knowledge EXplorer works with categorial data in the form of a data matrix. Rows correspond to objects, columns correspond to attributes.

2.1.2 Objects

Objects can be patients, animals, experiments, sites in a region etc. Sometimes objects are called cases or observations.

2.1.3 Attributes

Attributes are observed characteristics of objects. Sometimes attributes are called variables or features.

2.1.4 Categories

Values of attributes describe different categories of objects. We will denote jc *category* of objects which have value c in the attribute j. So e.g the attribute-value pair $(sex, female)$ corresponds to the category *woman*.

Knowledge EXplorer assumes, that all atributes are nominal i.e. their values are mutually exclusive and without any ordering (e.g. hair_color). When working with numerical data (e.g. temperature), categorisation must be done before using the system.

2.1.5 Combinations

From logical point of view, *combination* C of categories is conjunction of propositions, which correspond to single categories. Number of categories in a combination is the length of the combination. We will denote

$$C = j_1 c_1 ... j_k c_k$$

a combination of the length k. E.g. $hair\_color = yellow \,\&\, sex = female$ is a combination of the length 2. *Frequency* of a combination C is the number of objects, which fulfil this combination in data D. We will denote it $\parallel C \parallel_D$.

2.1.6 Relations

For two combinations C_1, C_2 with no common attribute we can formulate *implications* $C_1 \Longrightarrow C_2$, $C_2 \Longrightarrow C_1$ and *equivalence* $C_1 \Longleftrightarrow C_2$.

For every pair C_1, C_2 we can create fourfold contingency table in the form:

	C_2	$non(C_2)$
C_1	a	b
$non(C_1)$	c	d

where

a is the number of objects which fulfil both C_1 and C_2 ,

b is the number of objects which fulfil C_1 but don't fulfil C_2 ,

c is the number of objects which don't fulfil C_1 but fulfil C_2 ,

d is the number of objects which don't fulfil C_1 and don't fulfil C_2 .

so

$$\parallel C_1 \parallel_D = a + b$$
$$\parallel C_2 \parallel_D = a + c$$

We can define *validity of the implication* $C_1 \Longrightarrow C_2$ as the relative frequency of C_1 which occur together with C_2 (conditional probability $P(C_2/C_1)$)

$$\parallel C_1 \Longrightarrow C_2 \parallel_D = \frac{\parallel C_1 \& C_2 \parallel_D}{\parallel C_1 \parallel_D} = \frac{a}{a+b}.$$

The *validity of the equivalence* $C_1 \Longleftrightarrow C_2$ is defined as the relative frequency of C_1 and C_2 which occur together with C_1 or C_2

$$\parallel C_1 \Longleftrightarrow C_2 \parallel_D = \frac{\|C_1 \& C_2\|_D}{\|C_1\|_D + \|C_2\|_D - \|C_1 \& C_2\|_D} = \frac{a}{a+b+c}.$$

We can also define *coverage of the implication* $C_1 \Longrightarrow C_2$ as the relative frequency of C_2 which occur together with C_1 (conditional probability $P(C_1/C_2)$)

$$\parallel C_2 \Longrightarrow C_1 \parallel_D = \frac{\|C_1 \& C_2\|_D}{\|C_2\|_D} = \frac{a}{a+c}.$$

So the validity of $C_1 \Longrightarrow C_2$ equals to the coverage of $C_2 \Longrightarrow C_1$ and vice versa. Validity and coverage are in the range $< 0,1 >$ ($< 0\%, 100\% >$). If validity of $C_1 \Longrightarrow C_2$ equals to p then validity of $C_1 \Longrightarrow \neg C_2$ equals to $1-p$ ($100-p\%$).

The validity of an implication expresses how strong the left-hand side combination is related to the right-hand side combination (if all objects which fulfil the left-hand side fulfil also the right-hand side, the validity equals to 1). The coverage of implication expresses how strong the right-hand side combination is related to the left-hand side combination (if all objects which fulfil the right-hand side fulfil also the left-hand side, the coverage equals to 1).

2.1.7 Rules

Rules are in the form of implications with fixed right-hand side combination, the goal given by the user. A weight is assigned to every rule.
The knowledge base can be directly used for consultations (prediction) in an expert system with a PROSPECTOR like inference mechanism (PROSPECTOR like combining function for composing evidences).

2.2 Exploration tasks

Exploration tasks differ in what parts of relations (what combinations) are given by the user and what parts are generated by the system. Within every task, instead of a combination the user can give single attribute. In this case, the system will work with every category of the given attribute as with a combination of the length 1.

2.2.1 Specific evaluation

Most simple task. For two given combinations C_1 and C_2 the system computes validities of implications $C_1 \Longrightarrow C_2$, $C_2 \Longrightarrow C_1$ and equivalence $C_1 \Longleftrightarrow C_2$.

This task has not been implemented in the Apollo version.

2.2.2 Complete exploration

Most complex task. The system finds all implications and equivalences between pairs of combinations, which fulfil user given criteria for length, frequency, validity and coverage.

Every step the system generates a combination C, splits it into all possible pairs of subcombinations C_1, C_2 such that $C = C_1 \,\&\, C_2$ and evaluates the numeric parameters of the relations $C_1 \Longrightarrow C_2$, $C_2 \Longrightarrow C_1$ and $C_1 \Longleftrightarrow C_2$. After this evaluation, the combination C is extended by adding single category and the process repeats. The combinations are generated in descending order of their frequencies.
The generate and evaluate cycle works in following steps:

1. make the most frequent combination in the list the actual combination C,
2. compute validity and coverage for every implication and equivalence created from all possible subcombinations C_1 and C_2,
3. expand the actual combination C by adding a category,
4. store the expanded combination in the list according to its frequency,
5. remove the actual combination C from the list.

This task has not been implemented in the Apollo version.

2.2.3 Analysis of conclusions

In this task, the system finds all implications with fixed (given) left-hand side which fulfil user given criteria for length, frequency, validity and coverage. The resulting implications can be interpreted as description of *conclusions* of the given combination.

The system generates all possible right-hand sides (combinations of categories of attributes not occuring in the left-hand side combination) and evaluates validities and coverages of these implications. This process starts with the most frequent category and procceeds in descending order of frequencies of generated right-hand sides.
The generate and evaluate cycle works in following steps:

1. make the most frequent combination in the list the actual right-hand side combination,
2. compute validity and coverage for the implication with the actual right-hand side combination,
3. expand the actual right-hand side combination by adding a category,
4. store the expanded combination in the list according to its frequency,
5. remove actual right-hand side combination from the list.

2.2.4 Analysis of causes

In this task, the system finds all implications with fixed (given) right-hand side which fulfil user given criteria for length, frequency, validity and coverage. The resulting implications can be interpreted as description of *causes* of the given combination.

The system generates all possible left-hand sides (combinations of categories of attributes not occuring in the right-hand side combination) and evaluates validities and coverages of these implications. This process starts with the most frequent category and procceeds in descending order of frequencies of generated left-hand sides.
The generate and evaluate cycle works in following steps:

1. make the most frequent combination in the list the actual left-hand side combination,
2. compute validity and coverage for the implication with the actual left-hand side combination,
3. expand the actual left-hand side combination by adding a category,
4. store the expanded combination in the list according to its frequency,
5. remove actual left-hand side combination from the list.

2.3 Knowledge acquisition

Knowledge EXplorer performs symbolic empirical multiple concept learning from examples (cases), where the induced concept description is of the form of weighted decision rules. The algorithm can deal with noisy data, unknown values, redundancy and contradictions. The algorithm does not perform incremental learning. Like in analysis of causes, the rules are generated for user-given right-hand side combination C^*.

Knowledge EXplorer works in an iterative way each iteration testing and expanding an implication $Ant \Longrightarrow C^*$. This process starts with "empty rule" with weight as relative frequency of C^* in data and stops after testing all implications which were created according to user defined criteria. The implications are evaluated according to decreasing frequency of Ant, so most reliable implications are tested first.

During testing, the validity (conditional probability $P(C^*/Ant)$) of an implication is computed. If this validity significantly differs from the composed weight (value obtained when composing weights of all subrules of the implication $Ant \Longrightarrow C^*$), then this implication is added to the knowledge base. The weight of this new rule is computed from the validity and the composed weight using inverse composing function. For composing weights we use PROSPECTOR's combining function[2]

$$x \oplus y = (x * y)/(x * y + (1 - x) * (1 - y)).$$

During expanding, new implications are created by adding single categories to Ant. These categories are added in descending order of their frequencies. New implications are stored (according to frequencies of Ant) in an ordered list of implications. So Knowledge EXplorer generates every implication only once and for any implication in question all its subimplications have been already tested.

The system can learn rules for a single concept described as a goal combination (conjunction of categories) or for multiple disjoint concepts, which correspond to different categories of a given attribute.

2.3.1 Algorithm

The algorithm can be described as follows:

Input: Data D, goal combination C^*, required range of lengths $< l_{min}, l_{max} >$ of the left-hand side of a rule, required range of frequencies $< f_{min}, f_{max} >$ of the left-hand

[2] We modify classical PROSPECTOR's approach using correction principle suggested by Hájek [5].

side of a rule, required range of validities $< P_{min}, P_{max} >$ of a rule and required range of coverage $< Q_{min}, Q_{max} >$ of a rule.

Output: Knowledge base KB;

Initialisation:

Let KB be a list consisting of empty implication $0 \Longrightarrow C^*$ with the weight computed[3] from the relative frequency of C^* in data D;

Let CAT be a list of categories jc such that $\parallel jc \parallel_D \in < f_{min}, f_{max} >$, sorted in descending order of $\parallel jc \parallel_D$;

Let OPEN be a list of implications $jc \Longrightarrow C^*$ such that $\parallel jc \parallel_D \in < f_{min}, f_{max} >$, sorted in descending order according to $\parallel jc \parallel_D$;

Computation:

while OPEN is not empty do
begin
  select the top implication $Ant \Longrightarrow C^*$ from OPEN;
  compute its *validity* $P(C^*/Ant)$ and *coverage* $P(Ant/C^*)$;
  if $(P(C^*/Ant) \in < P_{min}, P_{max} > \ \& \ P(Ant/C^*) \in < Q_{min}, Q_{max} >)$ then
  begin;
    compute *composed_weight* $CW(C^*, Ant)$ from the weights of all subrules of $Ant \Longrightarrow C^*$ which are already in KB, using composition function $\oplus$;
    if validity significantly differs[4] from composed weight then add $Ant \Longrightarrow C^*$ to KB with the weight w such that $w \oplus composed\_weight = validity$[5] ;
  end;
  if $length(Ant) < l_{max}$ then
  for each jc from CAT such that $\parallel jc \parallel_D > \parallel Ant \parallel_D$
  begin;
    generate new combination $jc \& Ant$;
    if $\parallel jc \& Ant \parallel_D \in < f_{min}, f_{max} >$ then
    insert $jc \& Ant \Longrightarrow C^*$ into OPEN just after the last implication $C \Longrightarrow C^*$ such, that $\parallel C \parallel_D \geq \parallel jc \& Ant \parallel_D$;
  end;
  delete $Ant \Longrightarrow C^*$ from OPEN;
end;

[3] When learning single concept, this weight *equals* to relative frequency, when learning multiple concept a *correction* must be done so that the weight equals to 0.5 in case of uniform distribution of the goal categories.

[4] We test this difference using χ^2 goodness-of-fit test
$\chi^2 = \sum_{i=1}^{n} \frac{\|Ant\&C^*\|_D * \|Ant\&C^*\|_D}{\|Ant\|_D * CW_i(C^*, Ant)} - \parallel Ant \parallel_D$
where n is the number of learned concepts.

[5] For multiple concept learning, similar correction must be done as in the case of default rule.

2.3.2 Knowledge base evaluation

The acquired knowledge base can be tested on its accuracy. The standard way, how to do this is to use a testing data set. This approach allowes to compare results (classification) given by the expert with results[6] obtained from the learned rules. The resulting performance of the system is expressed as the total successfulness (relative number of correctly classified examples).

Another way how to test the knowledge base, is the *one-leave-out* test. This test also measurres the robustness of the knowledge base. The idea is to remove one example from the training set, learn rules from the remaining examples and then test the rules using the removed example. This process can be repeated for every example in the training set, so acquiring as many (slightly different) knowledge bases as is the number of examples in the training set. This test has not been implemented in the Apollo version.

When visually interpreting the knowledge base, sometimes some "obvious" piece of knowledge cannot be find. This is because the effect of the corresponding "missing" rule can be composed from its (more general) subrules, which are already in the knowledge base. So this rule is redundand and thus not inserted. Therefore, the knowledge base has to be taken into account as a whole.

The generalisation (done by selecting implications using the χ^2 test) is usually very high. Typically, the resulting knowledge base consists only of a small fraction (several percents) of all implications which fulfil the input criteria.

2.3.3 Comparison to another learning algorithms

There are a number of inductive learning programs such as programs of the TDIDT family or the AQ family, that from given examples induce knowledge in the form of decision trees or rules. Several of these programs can also deal with noisy or imperfect data.

As the AQ family algorithms (e.g. CN2 [3]), Knowledge EXplorer learns rules from examples. The basic idea is to construct the knowledge base as a set of implications, which is minimal and which is consistent with the training data set. This process starts with *apriori* knowledge about the distribution of goal combination in data (the empty rule, in CN2 called the default rule) and continues with more specific information (implications found in data), checking the knowledge base on consistency with every new implication.

Unlike AQ systems, Knowledge EXplorer does not remove covered examples from the training data set. So more than one rule can be learned for the same goal combination from an example. This gives the user different descriptions (different points of view) of the same concept.

[6] The predictions of the system are expressed in the form of weights inferred for all goal concepts. The weights are int the range $< 0, 1 >$. Weight 0.5 indicates *undecided*, weight > 0.5 indicates *goal concept predicted* and weight < 0.5 indicates *goal concept not predicted*. The concept which is predicted with the highest weight is presented as the result of the system.

Another difference is that the learning algorithm assigns weights to every rule[7] . The weight expresses the predictive power of a rule. During consultation, the weights are used in a PROSPECTOR like expert system, so more than one goal concept can be predicted for a given case. This can useful if the goal concepts are not mutually exclusive.

Because of the statistical test in the algorithm, Knowledge EXplorer needs "reasonable" number of learning examples to work correctly.[8]

When working with numerical attributes, data preprocessing (categorization) is necessary. This categorization depends heavily on background knowledge about the problem domain.

The results of consultation are given in the form of weights inferred for the goal concepts. For a single case, like in classical expert systems more then one goal concept can be recommended by the system.

3 Conclusion

Knowledge EXplorer is a system which offers both statistical (exploratory) and AI (machine learning) approach to data analysis. If the task is to describe dependencies within given data, exploratory data analysis approach is suitable. If the task is to predict the occurence of a goal combination for new cases, the machine learning approach is needed. These components can interact, so using exploratory tasks we can find optimal parameters for the knowledge acquisition procedure.

The relations obtained by the exploration tasks can be viewed as knowledge which is to be used (visually interpreted) by a human, whereas the rules obtained during knowledge acquisition are to be used by an expert system. In the first case, every single relation can be a basis for human decision making, in the second case, the knowledge base has to be used as a whole.

Knowlege EXplorer has been experimentally tested in various problem domains. The exploration tasks have been used e.g. for evaluation of public opinion pool done during Czechoslovak parliamentary elections in 1992 (together with the Parliament Institute) or for analysis of demographical data about villiage settlements in South Moravia [1]. The knowledge acquisition component has been tested as possible aditional algorithm for the ALEX system [9] or as a system for acquiring knowledge about virological hepatitis tests. Experiments done with the hepatitis data showed that the performance of Knowledge EXplorer is comparable to the performance of TDIDT-like (KnowledgeSeeker) and AQ-like (CN2) systems[9].

[7] When learning unordered rules in CN2, some numerical evaluation of rule using confidence measure is done too, but this is used only to select the best rule in case of possible clashes. Some work was also done in creating rules with certainty factors from decision trees [8].

[8] In the implementation for Apollo, the user can skip the test; in this case *every* implication which fulfil the input parameters will be added into the knowledge base.

[9] This experiments will be described in a separate report.

References

[1] Berka,P. - Ivánek,J. - Jirků,P. - Stejskal,B.: Knowledge Acquisition from Data - a Tool for Regional Planning. In: UNIDO/CSFR Workshop on Application of Interactive Decision Support Systems to Industrial Planning. Bratislava, UNIDO/CSFR 1991.

[2] Boswell,R.: Manual for CN2 version 4.1. TI/P2154/RAB/4/1.3, Turing Institute, 1990.

[3] Clark,P.: Functional Specification of CN and AQ. TI/P2154/PC/4/1.2, Turing Institute, 1989.

[4] Duda,R.O. - Gasching,J.E.: Model Design in the Prospector Consultant System for Mineral Exploration. in: Michie,D. (ed.), Expert Systems in the Micro Electronic Age, Edinburgh University Press, UK, 1979.

[5] Hájek,P.: Combining Functions for Certainty Factors in Consulting Systems. Int.J. Man-Machine Studies 22,1985, 59-76.

[6] Hájek,P. - Havránek,T.: Mechanizing Hypothesis Formation - Mathematical Foundations for a General Theory. Berlin, Springer-Verlag 1978.

[7] Ivánek,J. - Stejskal,B.: Automatic Acquisition of Knowledge Base from Data without Expert: ESOD (Expert System from Observational Data), in: Proc. COMPSTAT'88 Copenhagen (Physica-Verlag Heildelberg 1988), 175-180.

[8] Quinlan,J.R.: Generating Production Rules from Decision Trees, in Proceedings of the 10th International Joint Conference on Artificial Intelligence (IJCAI-87), Morgan Kaufmann, Los Altos, CA, p.304-307, 1987.

[9] Winkelbauer,L. - Berka,P.: New Algorithms for ALEX: Expanding An Integrated Learning Environment. submitted to ECML93 workshop.

A User manual

This part describes the current (January 1993) implementation of the system for the Apollo workstations.

Some new features were introduced in the Apollo version in the comparison with the standard PC implementation. These features are

- coverage of the implication as a new parameter,
- range of numerical input parameters (length, frequency, validity and coverage) instead of thresholds,
- META parameters which control the complexity of searched space.

On the other hand, some standard procedures (specific evaluation, complete exploration and one-leave-out test) have not been implemented yet. This will be done in the near future.

A.1 Control

On the top level, you can see following menu of procedures:

```
**************************************
*                                    *
*          Knowledge EXplorer        *
*     a Knowledge Acquisition Tool   *
*                      January 1993  *
**************************************


KEX>> Main Menu
A - Analysis of causes
C - Analysis of conclusions
K - Create konwledge base
T - Test konwledge base
O - Consultations
P - Knowledge base presentation
X - Exit the program


KEX>> Choise: _
```

To select desired procedure, give in corresponding letter. Every procedure has its own prompt, so you are still informed what procedure is running.

After selecting the procedure you will be asked for the input file. If you want to work with actual input file (its name is displayed) simply press $< Enter >$. $'Q'$ returns you to the main menu.

After selecting the input file you will be asked for the output file. If you give no name for this file, the output will only be displayed on the screen.

Then you have to give in input parameters for the procedure (as described in the next section) and the computation begins. After finishing the computation, the system returns to the main menu.

A.2 Data

The input data file consists of

1. header,
2. attribute definition,
3. data matrix.

The header consists of three lines:

```
description_of_the_data
dscription_of_the_data
number_of_objects  number_of_attributes
```

The attribute definition consists for every attribute of

```
name_of_the_attribute
number_of_categories
```

for every category

```
code_of_the_category  name_of_the_category
```

The categories are coded using one-character symbols (usually letters). *Do not use numbers to code categories, also don't use the character '.'.* Missing values are coded as $'?'$. Missing values are excluded from the analysis.

The data matrix consists of rows each row corresponding to one object. The objects are coded as strings of codes of categories *without any blanks.*

For an example of input data file see the section Example run.

A.3 Combination

Combination is required as input for analysis of causes, analysis of conclusions and for knowledge base creation.
The combination is given in the sequence:

```
number_of_attribute
code_of_category
```

This sequence repeats until you give '0' as the number of attribute.
If you want to do analysis for all categories of an attribute, simply give in the desired number_of_attribute and '.' for the code_of_category.

A.4 Procedures

The procedures for analysis of causes, analysis of conclusions and for knowledge base creation requires a lot of input parameters. These parameters allow to "tune" the system well according to background knowledge about the data. This may be a little confusing for an unskilled user. However, there are some standard strategies how to choose the numerical parameters:

- full analysis (min length = 1, max length = number of attributes not occurring in given combination, min freq = 1, max freq = number of objects, min validity = 0, max validity = 100, min coverage = 0, max coverage = 100),
- minimal length analysis (min length = max length = 1, another parameters as for full analysis),
- "no noise" analysis (min validity = 100, min coverage = 100)[10]

The full analysis strategy *does not ensure best results* in knowledge base creation procedure.

A.4.1 Analysis of causes

This procedure performs the analysis of causes.

Input:

```
Data file as described above
```

When reading data file, the system display a table of frequencies of all categories in data.

```
Name of output file to save the results
Left-hand side combination as described above
```

If no combination is given, the system will generate all right-hand side combinations and compute their frequencies.

[10] Use 100 only if you are 100% sure that the data are without noise and the goal concepts are separable using given attributes. Setting these parameters to 90 or 80 will also work well.)

```
Range for length of the right-hand side combination
Range for frequency of the right-hand side combination
Range for validity of the implication
Range for coverage of the implication

Expand implications with coverage = 100% (n/y) ?
```

When answering $'y'$ the system will generate all implications, some of them redundand because with no new information for the user[11]. Answer $'n'$ to reduce the search space..

Output: (always on the screen, when requested also into the output file)

```
List of the found implications
Statistics of implications
```

Review of the number of all generated implications according to their length (rows) and validities (columns).

A.4.2 Analysis of conclusions

This procedure performs the analysis of conclusions.

Input:

```
Data file as described above
```

When reading data file, the system display a table of frequencies of all categories in data.

```
Name of output file to save the results
Right-hand side combination as described above
```

If no combination is given, the system will generate all left-hand side combinations and compute their frequencies.

```
Range for length of the left-hand side combination
Range for frequency of the left-hand side combination
Range for validity of the implication
Range for coverage of the implication

Expand implications with validity = 100% (n/y) ?
```

When answering $'y'$ the system will generate all implications, some of them redundand because with no new information for the user[12]. Answer $'n'$ to reduce the search space.

Output: (always on the screen, when requested also into the output file)

```
List of the found implications
Statistics of implications
```

Review of the number of all generated implications according to their length (rows) and validities (columns).

[11] Coverage of the expanded implication will again be 100%.
[12] Validity of the expanded implication will again be 100%.

A.4.3 Create knowledge base

This procedure performs the knowledge acquisition.

Input:

```
Data file as described above
```

When reading data file, the system display a table of frequencies of all categories in data.

```
Name of output file to save the results
Right-hand side combination as described above
Range for length of the left-hand side combination
Range for frequency of the left-hand side combination
Range for validity of the implication
Range for coverage of the implication

Insert empty rule (n/y) ?
Test implications (n/y) ?
```

The standard answer (for sufficient number of objects) is $'y'$ to both questions. In this case you will use the whole functionality of the learning algorithm. When answering both questions with $'n'$, you will obtain as many rules as implications in the analysis of conclusions (no generalisation is done). This may be reasonable for small training set. Wether to insert empty rule may also depend on the learning task specification.

```
Expand implications with validity = 100% (n/y) ?
```

When answering $'y'$ the system will generate all implications, some of them redundand because with no new information for the user[13]. Answer $'n'$ to reduce the search space..

Output: (always on the screen, when requested also into the output file)

```
List of the found rules
Statistics of implications
```

Review of the number of all generated implications according to their length (rows) and validities (columns).

A.4.4 Test knowledge base

This procedure performs testing of the acquired knowledge base.

Input:

```
Knowledge base
```

Either a knowledge base file created and saved by the knowledge base creation procedure (in this case give in the name of the file) or internally stored knowledge base which was just created by the knowledge base creation procedure (in this case simply press $< Enter >$). You will see a message, wether an internally stored knowledge base is available.

[13] Validity of the expanded implication will again be 100%.

```
File with test examples
```

This file has the same structure as the data matrix in the data file, i.e. each row corresponds to one object (string of codes of categories), the file has no header.

```
Name of output file to save the results
```

Output: (always on the screen, when requested also into the output file)

```
Results of consultation for every example
```

(Only in the output file.)

```
Review of results of testing
```

This review consists of a table where for every learned concept (a row in the table) the number of classifications done by the system and the number of correct classifications is given. Some examples in the testing set may be unclassified; either the resulting weight was in the range $< 0.45, 0.55 >$ (the row "not decided" in the table, or there was no aplicable rule in the knowledge base (the row "not predict." in the table)[14]. The resulting performance of the system is given in the row "Total".

A.4.5 Consultation

This procedure performs consultations for single cases, given from the keyboard.

Input:

```
Knowledge base
Name of output file to save the results
```

For every consulted case

```
Input case
```

The input case is given as a sequence of codes of categories assigned to displayed names of attributes.. Give the code '?' for unknown values.

Output: (For every consulted case)

```
List of activated rules
List of goal concepts with the inferred weights
```

[14] If the knowledge base contains empty rule, prediction is always done.

A.4.6 Knowledge base presentation

This procedure saves the knowledge base in a file in more legible way.

Input:

```
Knowledge base
Name of output file to save the results
```

Unlike all other procedures the output file must be given.

Output:

```
File with the knowledge base
```

A.5 Example run

To demonstrate, how to work with the system, we will use the example data from [2].

The input data file looks like this:

```
ANIMALS.DAT file
Demo data from CN2
10 7
skin_covering
4
n none
h hair
f feathers
s scales
milk
2
y yes
n no
homeothermic
2
y yes
n no
habitat
3
l land
s sea
a air
reproduction
2
o oviparous
v viviparous
```

```
breathing
2
l lungs
g gills
class
5
m mammal
f fish
r reptile
b bird
a amphibian
hyylvlm
nyysvlm
hyysolm
hyyavlm
snnsogf
snnlolr
snnsolr
fnyaolb
fnylolb
nnnlola
```

Let us start with the knowledge base acquisition for animals recognition. We will use the *no noise* strategy with max. length = 2 and we will *skip both empty rule and the test*[15].

```
KEX>> Choise: k

KB>> Current data are  animals.dat.
KB>> Name of data (<ENTER> for no change, 'Q' for main menu):
KB>> Name of output file (<ENTER> for no file):

      INPUT PARAMETERS

KB>> Attribute (0 for end):  7
KB>>  Category (. for all):  .

KB>>     Combination:  7.
KB>>      Min length (1):      1
```

[15] This will give best results for our data. If we use *full analysis* strategy with empty rule and implication test metaparameters set to *'y'*, because of small number of examples, the performance of the system will be rather poor; we will obtain 20 rules (4 different left-hand sides times 5 goal concepts) and the successfulness of prediction will be 40% (positive rules onply for the concpt mammal). If we increase the number of training examples to 100 (by copying 10 times our 10 examples), for the same input parameters we will obtain 90 (18 times 5) rules and the successfulness of prediction will be 100% .

```
KB>>      Max length (NoAtt):  2
KB>>   Min frequency (1):      1
KB>>   Max frequency (NoObj):  10
KB>>    Min validity (0%):     90
KB>>    Max validity (100%):   100
KB>>    Min coverage (0%):     90
KB>>    Max coverage (100%):   100
```

When reading data file he system displays the table of frequencies of all categories.

```
          FREQUENCIES OF CATEGORIES

 Att.  cat. fr.   cat. fr.   cat. fr.   cat. fr.   cat. fr.
-------------------------------------------------------------
   1    1n    2    1h    3    1f    2    1s    3
   2    2y    4    2n    6
   3    3y    6    3n    4
   4    4l    4    4s    4    4a    2
   5    5o    7    5v    3
   6    6l    9    6g    1
   7    7m    4    7f    1    7r    2    7b    2    7a    1

         META PARAMETERS

KB>> Insert empty rule (n/y) ?  n
KB>> Test implications (n/y) ?  n
KB>> Expand implications with validity = 100% (n/y) ?  n


                 GENERATED RULES

         Frequencies
 no.   left  right  both   Weight   Implication
-------------------------------------------------------------------------
   1     4      4      4   0.8889   2y  ==>  7m
   2     2      2      2   0.8000   1f  ==>  7b
   3     2      2      2   0.8000   3y2n  ==>  7b
   4     2      2      2   0.8000   1s6l  ==>  7r
   5     1      1      1   0.6667   6g  ==>  7f
   6     1      1      1   0.6667   1n5o  ==>  7a
   7     1      1      1   0.6667   1n2n  ==>  7a
   8     1      1      1   0.6667   1n3n  ==>  7a
   9     1      1      1   0.6667   1n4l  ==>  7a
```

```
     NUMBER OF GENERATED IMPLICATIONS

                        validity
length     0   (0,50)   50  (50,100)  100      sum
-------------------------------------------------------
   1       0      0      0      0      3         3
   2       0      0      0      0      6         6
-------------------------------------------------------
total      0      0      0      0      9         9

     NUMBER OF GENERATED RULES   9
```

Because of given meta parameters, the number of rules equals to the number of implications.

```
KB>> End of task.
```

Now we will test the knowledge base. In our demonstration, we use the same examples as for the learning.

```
KEX>> Choise: t

TEST>> Current knowledge base is created from data animals.dat.
TEST>> Name of knowledge base (<ENTER> for no change, 'Q' for main menu):
TEST>> Name of test data file ('Q' for main menu): animals.con
TEST>> Name of file for results (<ENTER> for no file):
```

During computation the system displays the testing examples.

```
                KNOWLEDGE BASE TESTING

   no.  object
-------------------------------------------------------------------------
    1.  hyylvlm
    2.  nyysvlm
    3.  hyysolm
    4.  hyyavlm
    5.  snnsogf
    6.  snnlolr
    7.  snnsolr
    8.  fnyaolb
    9.  fnylolb
   10.  nnnlola
```

```
          RESULTS OF RULE BASE TESTING IN DATA  animals.con

     pred        total           from which        total       from which
                abs    rel       true    false                 true    false
-------------------------------------------------------------------------------
       7m        4    40%          4        0        100%      100%      0%
       7f        1    10%          1        0        100%      100%      0%
       7r        2    20%          2        0        100%      100%      0%
       7b        2    20%          2        0        100%      100%      0%
       7a        1    10%          1        0        100%      100%      0%
-------------------------------------------------------------------------------
     Total      10   100%         10        0        100%      100%      0%
-------------------------------------------------------------------------------
not decided      0     0%     **********************************************
not predict.     0     0%     **********************************************
-------------------------------------------------------------------------------
     Total      10   100%         10        0        100%      100%      0%


TEST>> End of task.
```

Next part shows how to consult with the knowledge base giving cases from keyboard.

```
KEX>> Choise: o

CONSULT>> Current knowledge base is created from data animals.dat.
CONSULT>> Name of knowledge base (<ENTER> for no change, 'Q' for main menu):
CONSULT>> Name of file for results (<ENTER> for no file):


                  CONSULTATION


CONSULT>> New consultation (n/y) ? y

CONSULT>> skin_covering (n h f s  ?): n
CONSULT>> milk (y n  ?): n
CONSULT>> homeothermic (y n  ?): n
CONSULT>> habitat (l s a  ?): s
CONSULT>> reproduction (o v  ?): o
CONSULT>> breathing (l g  ?): l
```

```
Input case = nnnsol?

            Activated rules
 no.   left  right  both   Weight   Implication
-------------------------------------------------------------------------
   6     1      1      1   0.6667   1n5o  ==>  7a
   7     1      1      1   0.6667   1n2n  ==>  7a
   8     1      1      1   0.6667   1n3n  ==>  7a


   no. goal   weight   object
-------------------------------------------------------------------------
    1.  7m    0.5000   nnnsol?
     .  7f    0.5000    . . .
     .  7r    0.5000    . . .
     .  7b    0.5000    . . .
     .  7a    0.8889    . . .
```

From three aplicable rules, resulting concept *amphibian* was inferred (with the weight 0.8889) for the input case.

```
CONSULT>> New consultation (n/y) ? n

CONSULT>> End of task.
```

To save the rules in legible form, we run the presentation procedure.

```
KEX>> Choise: p

KBPRES>> Current knowledge base is created from data animals.dat.
KBPRES>> Name of knowledge base (<ENTER> for no change, 'Q' for main menu):
KBPRES>> Name of file for results ('Q' for main menu): animals.rul
KBPRES>> Storing knowledge base.

KBPRES>> End of task.
```

This is how the rules are saved in the file animals.rul:

```
                    RULES IN THE KNOWLEDGE BASE

RULE   1:   IF    milk  ==  yes
          THEN    class  ==  mammal      (0.8889)
```

```
RULE   2:   IF    skin_covering  ==  feathers
          THEN    class  ==  bird      (0.8000)

RULE   3:   IF    homeothermic  ==  yes
                  AND
                  milk  ==  no
          THEN    class  ==  bird      (0.8000)

RULE   4:   IF    skin_covering  ==  scales
                  AND
                  breathing  ==  lungs
          THEN    class  ==  reptile      (0.8000)

RULE   5:   IF    breathing  ==  gills
          THEN    class  ==  fish      (0.6667)

RULE   6:   IF    skin_covering  ==  none
                  AND
                  reproduction  ==  oviparous
          THEN    class  ==  amphibian     (0.6667)

RULE   7:   IF    skin_covering  ==  none
                  AND
                  milk  ==  no
          THEN    class  ==  amphibian     (0.6667)

RULE   8:   IF    skin_covering  ==  none
                  AND
                  homeothermic  ==  no
          THEN    class  ==  amphibian     (0.6667)

RULE   9:   IF    skin_covering  ==  none
                  AND
                  habitat  ==  land
          THEN    class  ==  amphibian     (0.6667)
```

We will end out tour through the Knowledge EXplorer. The use of the analysis of causes and analysis of conclusions is similar to the use of the knowledge acquisition procedure and therefore is not shown here.

```
KEX>> Choise: x

Thank you for using KEX.
```