Data Mining with the Combinatorial Rule Model: an application in a health-care relational database

Miguel Artur Feldens, José Mauro Volkmer de Castilho
Universidade Federal do Rio Grande do Sul
[feldens, castilho]@inf.ufrgs.br

Abstract: This paper presents a simple and efficient association rule induction algorithm, the Combinatorial Rule Model (CRM), which has been applied to discover knowledge in health care databases, for data quality improvement purposes. In spite of being adapted from an exhaustive search system, its cost is very reasonable, as its architecture is record-oriented, performing the whole learning process with only one pass over the training examples (records). The system has been implemented as a set of reusable objects, designed to support different activities of the knowledge discovery process. The prototype has been designed according to specific classification task requirements, but can be used to discover knowledge in any relational table. Some tests were included in this paper, showing the performance of relevant evidence selection criteria and presenting some discovered rules.

Keywords: knowledge discovery in databases, data mining, machine learning, databases, artificial intelligence

1. Introduction

The technology to collect and store information has experienced great evolution over the last decades, being applied to many domains, generating huge and always growing collections of data. Processing such volumes of information has become a hard task to be tackled by non-automated methods, being valid to say that today the capacities to collect and store data overcome the ability to really use this information. Knowledge discovery in databases (KDD) is one of the areas that contribute to the development of efficient information recovery technology, dealing with systems capable of extracting new, useful and interesting knowledge out of databases.

In order to discover useful knowledge, most KDD systems frequently rely on classification algorithms to perform "data mining", which can be based on a variety of approaches, such as classical statistics, neural networks and machine learning. In spite of this background, extensive experimentation have shown that still there's no algorithm with optimal performance for every kind of data [MIC94]. Experimentations with MLC++ machine learning library [KOH96] recommend testing as many algorithms as possible (when it's possible) with the data to be used in an application, and choose what works the best. This situation keeps the development and analysis of learning algorithms an open research issue.

In spite of being frequently stated that exhaustive search for knowledge in databases is not practical, systems like Brute [RID94] have shown that this kind of approach avoids many of the pitfalls of using greedy techniques. Its empirical analysis has proved that it performs better than greedy algorithms in terms of accuracy, and that its cost is often reasonable. Based on such statements, exhaustive techniques could be better suited for applications where accuracy is the most critical issue. In classification for financial credit analysis, for example, the money savings that could be achieved with the reduction of the error rate could be worth the computational cost.

In the present work, due to the nature of proposed classification tasks, a simple algorithm to perform exhaustive search was developed. The *Combinatorial Rule Model* (CRM) [FEL97] is a data mining-aware, less expensive, simplified version of the *Combinatorial Neural Model* (CNM) [MAC89] [DEN91]. Besides some conceptual differences from the original model that were introduced in CRM, deterministic and heuristic mechanisms for evidence selection

(selection of relevant features from the examples), which significantly reduce the search space, were implemented, along with a set of classes to support data mining algorithms. Some tests are included in this paper, proving CRM to be a very efficient and practical algorithm for the kind of classification tasks it has been applied to.

The prototype has been designed to discover rules from health care data, extracted from databases of the public health care system in Brazil (known *as SUS - Sistema Único de Saúde*). The idea is to find rules relating medical procedures, internment time, special procedures, patient personal characteristics, etc. Discovered knowledge could be useful for data quality improvement [PAR93], as well as hospital administration decision making.

The remainder of this paper is organized as follows: next section (*2. Combinatorial Neural Network*), describes CNM, a hybrid model on which the implemented algorithms were based. Section 3 (*Combinatorial Rule Model*) describes implemented classification algorithms, and some essential details about other objects of this prototype. The prototype itself is briefly described in section 4 (*Implementation - SIDI Data Mining Tool*). Tests with the prototype are presented in the next section (*5. Some Results with health care data*) and, finally, the last section (*6. Conclusions*) discusses the results of the implementation.

2. Combinatorial Neural Model (CNM)

The Combinatorial Neural Model (CNM) [MAC89] [DEN91] integrates symbolic and non-symbolic knowledge, in a simple architecture. This model, which has been successfully used in expert system implementations [REA93] [ROS94] [MEN96], has characteristics which are desirable in a classification system; suitable for data mining:

a) Simplicity of neural learning - due to neural network's generalization capacity
b) Explanation capacity - since this model can map neural network's knowledge to a symbolic representation
c) High-speed training - tackled in only one pass over the training examples
d) Immune to neural network pitfalls like local optima, plateau, etc.
e) Incremental learning possibility - since previously learned knowledge can be mapped to neural network

CNM includes mapping of previous knowledge to the neural network, training algorithms, and "pruning" criteria for trained networks, in order to extract only significant pieces of knowledge. The CNM network, as represented in figure 1, is a 1-hidden layer, feed-forward network. Its special characteristics, are in the way CNM topology is constructed, in neuron characteristics, in the links between neurons, and in its training algorithm.

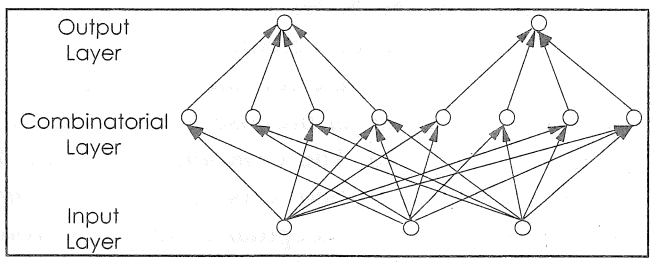

FIGURE 1 - CNM neural network

The input layer of a CNM network represents the set of evidences about the examples. That is, each input is in the [0,1] interval, indicating the pertinence of the training example to a certain concept, or the degree of confidence. The intermediate (combinatorial) layer is automatically generated: a neuron is added to this layer for each possible combination of evidences, from order 1 to a maximum order, given by the user. The output layer corresponds to the possible classes an example could belong.

Combinatorial neurons behave as conjunctions of evidences, or hypothesis of conjunctions of evidences that lead to a certain class. For that reason, combinatorial neurons propagate input values according to fuzzy AND operator [KOS91], taking to its output the minimum value received by the inputs. Output neurons group the classification hypothesis, implementing fuzzy OR, propagating the maximum value received in its inputs. With such characteristics, a set of conjunctive rules, could be easily represented as a CRM network.

The connections between neurons have weights, and also pairs of accumulators (which would contain punishment/reward total values). During the training process without previous knowledge, all weights are initially set to one and all accumulators to zero. As each example is presented, and propagated, in case it is properly classified, all links that led to the classification have their reward accumulators incremented through backpropagation. Similarly, misclassifications increment the punishment accumulators the path that led to mistake outputs. Note that weights remain unchanged during training process, only accumulators are incremented.

The training process is generally done in one pass over the training examples. At the end of this sequential pass, accumulators are used to compute new weights. Based on CNM topology, symbolic knowledge can be easily extracted, and the new weights can be used to compute the confidence of each piece of knowledge.

The main limitation of CNM, in terms of real-world data mining problems, is the possibility of combinatorial explosion, since the intermediate layer grows exponentially with the amount of evidences. In CRM implementation, we attempted to minimize this problem, with the addition of axioms that can reduce search space.

3. Combinatorial Rule Model (CRM)

We attempted to combine CNM architecture to the statistical approach, so that the accumulators, by the end of the training process could be translated to the statistical degree of confidence. The confidence *Conf* of a rule $A \rightarrow C_1$ is calculated as follows [AGR96]:

$$Conf = \frac{|A \wedge C_1|}{|A|}$$

That is, the number of examples of the class C_1 that have the characteristics defined by A, divided by the number of entities with all characteristics in A.

Using this probabilistic approach makes it easy to integrate knowledge from CRM with knowledge from other sources. Any given rule could have its confidence computed from the results of two SQL queries [KOR93] over the set of examples.

Besides the modification described above, the original model has been simplified, to get a minimal support for data mining together with some desirable advantages of the architecture, and also lower cost. Some additional components (classes) have also been implemented, to support data mining with CRM or any other algorithm that may be implemented.

The search process performed with CNM is exhaustive, as all classification hypothesis, represented by the combinatorial neurons, are considered. CNM's advantage when compared to a generate-and-test exhaustive algorithm is that CNM search is naturally record-oriented, and not hypothesis-oriented; i.e.: instead of iteratively generating and testing every possible hypothesis, the whole search space is generated and then trained, presenting an example (represented as a record or group of records) at a time.

Based on the considerations presented above, CRM has been implemented. In this model, a space of *candidate-descriptions* is generated in a combinatorial way. Each of these *candidate-descriptions* includes a conjunction of *cause-literals* (evidences on the IF side) and a *target-literal* (class label), and still a pair of *punishment/reward* accumulators. Below is the definition of a *candidate-description*, written in Delphi [BOR95]:

```
TDescription = class(TObject)
    public
        InputLiterals: TIntList;
        OutputLiteral: LongInt;
        RewardAcc: LongInt;
        PunishAcc: LongInt;

        constructor Create;
        destructor Destroy;
end;
```

Literals represent the evidences in the database. As the prototype has been designed to search for knowledge in any relational table [KOR93], a special class has been implemented (TKDTable), as a specialization of the relational table from Delphi Component Library [BOR95]. Special methods were implemented so that data can be accessed through *literals*, each one encapsulating an operation over a pair *<Attribute,Value>*: for qualitative attributes in the database, all possible values will correspond to a literal; for quantitative attributes, there will be a literal for each possible discretization range for that attribute. This way, rules can be constructed based only on the literals, abstracting its content.

The search process is shown in Algorithm 1. The generation of the *candidate-description* list is based on *search sub-spaces*. Each of these sub-spaces has a list of previously selected relevant literals for a specific class. There's a statistical process that selects the relevant literals for each class (section 3.1), which was significantly cost saving, compared to CNM approach that considers every evidence for every class.

GenerateDescriptions function simply generates all possible combinations of *cause-literals* in a *sub-space*, adding a *candidate-description* to a list, returned by the end of the process. Obviously, it avoids the generation of *candidate-descriptions* with literals that are mutually exclusive, like:

$$Height = Tall \wedge Height = Short \rightarrow Class = X$$

That is, *GenerateDescriptions* doesn't consider *candidate-descriptions* containing literals evaluated over the same attribute, except if this is a multi-valued attribute, and this is possible, since the relational table containing the examples can be the result of a JOIN operation over other relations [KOR93], i.e. a non-normalized table.

```
function TKDCRM.RuleSearch: TList;
Begin
   Let SPACELIST be a previously built list of search sub-spaces
   Let DESCLIST be a list of candidate-descriptions, initially empty
   Let RULELIST be a list of rules, initially empty
   For each SUB-SPACE in SPACELIST
   Begin
      DESCLIST=DESCLIST+GenerateDescriptions(SUB-SPACE)
   End
   Training(DESCLIST)
   RULELIST=ExtractRules(DESCLIST)
End
```

Algorithm 1 - CRM search

3.1 Selection of relevant literals (feature selection step)

As the size of search space grows exponentially with the number of literals to be considered, a special class has been implemented (TKDDependencies), to reduce search space prior to the application of CRM, and also compute simple (order 1) rules. This class accesses a data mining relational table (TKDTable) and can provide a list of search sub-spaces, using two deterministic and one heuristic criteria to select literals.

Let l_i be an input-literal, it is considered relevant to class C if and only if:

a) $|l_i \wedge C| >= SupportThreshold$

b) $Conf(l_i \rightarrow C) < 1$

c) $RelativeError(l_i, C) > ErrorThreshold$

The first condition states that literals are considered relevant to class C if there is enough support of this literal between the members of this class. The second means that if a literal defines a rule with confidence degree 1 to a certain class, it is not relevant, since all higher order rules discovered with that literal would be *covered* by the simple $l_i \rightarrow C$ rule.

The third condition is heuristic, based on the relative error to the expected distribution, based on the hypothesis of independence of attributes. It computes the expected number of occurrences of the literal l_i in class C, if these were independent:

$$Expected(l_i, C) = \frac{|l_i| * |C|}{N}$$

where N is the number of training examples. The relative error is then computed by:

$$RelativeError(l_i, C) = \frac{Actual(l_i, C) - Expected(l_i, C)}{Expected(l_i, C)}$$

Relative error is frequently used to compute the correlation between attributes [ZYT93] [SPI78], with the medium square error for every pair <*cause,target*>. Here, it has been taken individually and used as a heuristic, based on the intuitive notion that a *cause-literal* that occurs

more frequently associated with a certain class is more likely to be part of a rule which describes the members of that class. This supposition doesn't consider the mutual information between literals, but has proved in practice to result in search space reduction of up to 40.11 % in CRM search space, without loss of discovered knowledge.

3.2 Training

The training procedure is a simplification of CNM's training, but the basic idea remains the same. The Algorithm 2 just checks the validity of the *input-literals* for a given training example, just like a propagation in a neural network, and, according to the results, it increments the proper accumulators associated to the descriptions (backpropagation). The procedure consists of two nested iterative loops, so that for each example and each *candidate-description* the "propagation" and "backpropagation" is done.

```
procedure TKDCRM.Training(DESCLIST);
Begin
   Let MSET be a relation, containing the training examples (TKDTable class)
   MSET.FirstEntity
   For each register in MSET
      For each candidate-description DESC in DESCLIST
      Begin
        {Propagation}
        AResult = True
        k = 0
        Repeat
          AResult = MSET.Literals[DESC.InputLiterals.Items[k]]
          k = k +1
        Until (AResult = False) or (k = DESC.InputLiterals.Count)

        {Backpropagation}
        If AResult = True Then
          If MSET.Literals[DESC.OutputLiteral] = True Then
              DESC.RewardAcc = DESC.RewardAcc + 1
          Else
              DESC.PunishAcc = DESC.PunishAcc + 1
      End
   End
End
```

Algorithm 2 - CRM training

3.3 Rule extraction

The function that returns a list of rules from the list of previously trained *candidate-descriptions* is presented in Algorithm 3. For each description, it checks for support, looking for the number of cases for which the rule is valid, represented by *RewardAcc*. If its support goes beyond the threshold given by the user, confidence is computed, and checked with the confidence threshold. If the description's *RewardAcc* and confidence values succeed in comparisons with both thresholds, a new rule is added to the list.

Notice that the content of the accumulators of a trained description is:

$$RewardAcc = |C \wedge T|$$
$$PunishAcc = |C \wedge \neg T|$$

Where C denotes the list of *input-literals* and T the *target-literal* of a given description. So, the confidence can be computed by:

$$Conf = \frac{RewardAcc}{RewardAcc + PunishAcc}$$

```
function TCNM.ExtractRules(DESCLIST): TList;
Begin
   Let RL be a list o rules, initially empty
   For each candidate-description DESC in DESCLIST
   Begin
      Support = DESC.RewardAcc
      If Support >= SupportThreshold Then
      Begin
        Confidence = DESC.RewardAcc/(DESC.RewardAcc+DESC.PunishAcc)
        If Confidence > ConfThreshold Then
        Begin
          Rule =  TKDRule.Create (creates a new rule)
          Rule.Target = DESC.OutputLiteral
          Rule.Support = Support
          Rule.Confidence = Confidence
          Rule.CauseList = DESC.InputLiterals
          RL.Add(Rule)
        End
      End
   End
End
```

Algorithm 3 - CRM rule extraction

4. Implementation - SIDI Data Mining Tool

The prototype runs on IBM-PC compatibles, under Windows environment. This section will present some information about data and classificatory tasks, which guided the prototype design and implementation.

4.1 Data

The databases store AIHs (*autorizações de internação hospitalar* - hospital internment authorizations), which contain data representing any internment, medical procedures and diagnosis in SUS institutions. Data used in the tests correspond to four months of activities in *Hospital de Pronto-Socorro (HPS)*, the general hospital of Porto Alegre city. From the original database schema, some attributes were selected for the validation experiment, based on advices from a specialist, in order to obtain a small dataset, likely to contain useful implicit information from hospital administration standpoint.

Two relations were extracted from HPS data. The first - which is used for the tests in this paper - containing 3519 internment registers, with corresponding primary and secondary

diagnosis, required and actual procedures, internment time, etc. The second, with 7279 registers, contains information about special procedures.

4.2 Some classificatory tasks

To guide the design of the prototype, some discovery goals were previously elaborated. Some examples are:

a) Associations between diagnosis and medical procedures;
b) Associations between required and actual procedures - as these data come from an emergency hospital, required procedures are not always actually done;
c) Associations between patient characteristics and procedures - this could discover what should be done exclusively with patients of an specific sex, age, etc;
d) Associations between procedures and internment time;

4.3 System interface

The prototype's main window is shown in figure 2. It allows the user to configure the data type (*Quality/Number*), to check the attribute that identifies each entity in the table (*Index*), choose the attributes to be searched as causes (IF side of the rules) and define the target (attribute that indicates the class of each entity). Remember that the (*Index*) is necessary since it's assumed that the relation containing the training examples, could be a non-normalized table.

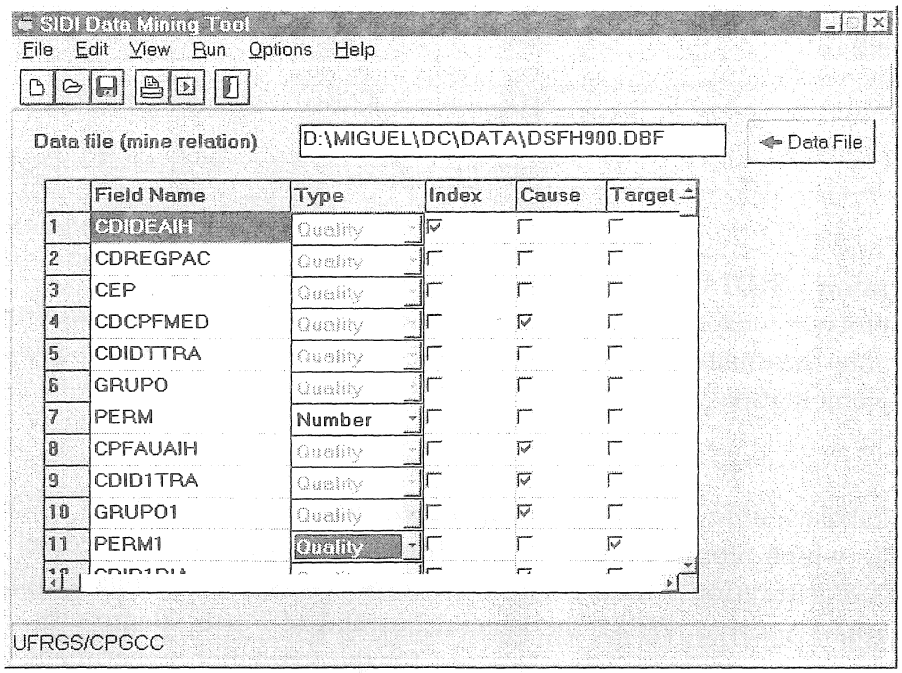

FIGURE 2 - Main window

5. Some results with health care data

First, one of the series of tests with the heuristic criteria for search space reduction is analyzed. Later, a few examples of actual rules are presented.

5.1 Relative error criteria

The series of tests included here corresponds to the classification of "Motivo de cobrança" (charge reason), whose values correspond to patient discharge, transference or death. All relevant attributes have been checked as possible causes, and the following thresholds were set: support of 35 cases and 90% of confidence. The system has been run with different relative error thresholds (*PrunThreshold*). Table 1 shows the results:

TABLE 1 - Testes de PrunThreshold (Motivo de cobrança)

PrunThreshold	CRM Hypothesis	CRM rules	Usefulness ratio (%)
-1	23099	153	0.662
-0.9	23099	153	0.662
-0.8	23099	153	0.662
-0.7	23099	153	0.662
-0.6	23099	153	0.662
-0.5	21359	153	0.716
-0.4	18613	153	0.822
-0.3	15676	153	0.976
-0.2	13835	153	1.106
-0.1	10715	150	1.400
0	6174	103	1.668
0.1	1707	53	3.105
0.2	697	25	3.587
0.3	598	19	3.177
0.4	282	17	6.028
0.5	202	9	4.455

Figure 3 represents the evolution of the usefulness ratio of the hypothesis as *PrunThreshold* is incremented. The first section of the graphic is horizontal, as no literal is eliminated; the second is ascending, where irrelevant literals are avoided (number of rules remains the same) or less relevant literals (number of rules starts to decay). The irregular section means that some relevant literals are being eliminated. The last section, which is descendent, marks the elimination of most of the relevant literals, as the number of rules tends to zero. Usefulness is defined as:

$$Usefu\,ln\,ess(\%) = \frac{NumberOfRules}{NumberOfHypothesys} * 100$$

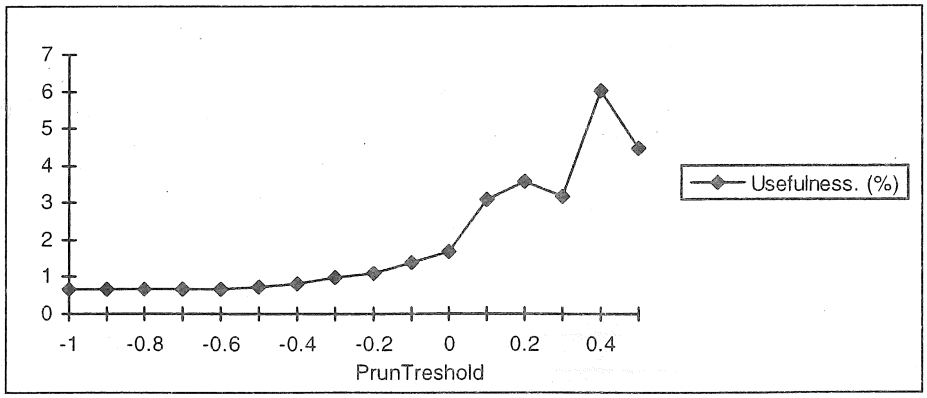

FIGURE 3 - PrunThreshold X Usefulness ratio (Motivo de cobrança)

This test, along with other series that have been done [FEL97] qualitatively shows that the increment of the relative error threshold increases the ratio of useful hypothesis generated by CRM. Further tests must be done, in order to check the loss of accuracy as literals are eliminated.

5.2 A classificatory task

One of the targets which were tested was the internment time. The rules below were selected out of the 75 discovered with this specific target. Notice that the first rule means that an specific treatment does not demand internment and the second relates a certain combination of primary and secondary diagnosis with two day internment:

```
1)
   IF CDID1TRA="31000002"
   THEN PERM1=0
   Confidence level = 100%
   Supported by 67 cases
...
71)
   IF CDID1DIA="094633" AND
      CDID2DIA="000000"
   THEN PERM1=2
   Confidence level = 96.3%
   Supported by 26 cases
...
```

FIGURE 4 - Some rules for the internment time

6. Conclusions

CRM is a simple and efficient system for extracting rules from relational databases, being recommended for applications requiring accuracy, once that exhaustive search avoids the pitfalls of greedy search techniques. Thanks to the selection of relevant literals, the search space for the classification algorithm could be significantly reduced, resulting in a reasonable search cost and making CRM practical for some real world applications. On the other hand, the algorithm is not good for classificatory tasks where there are too many evidences to be considered, specially if there are multi-valued attributes. In these cases, even with proper settings for evidence selection, there is the possibility of combinatorial explosion. A theoretical study of CRM's complexity should be done, to detect such cases. More tests must also be done, to check the loss of accuracy

as evidences are removed from the search space by the heuristic axiom which has been used. Further experimentation must also be done to compare CRM's accuracy and cost with other algorithms.

Ackownedments

We would like to thank CNPq - PROTEM - CC Fase III, for supporting this research, as part of the project "SIDI - Modelagem e Desenvolvimento de Sistemas de Informação Distribuídos e inteligentes: Aplicação ao controle de Informações na Área da saúde" (Modelling and development of intelligent distributed information systems: application to the control of health care information).

7. Bibliography

[AGR96] AGRAWAL, R. et al. Fast discovery of association rules. In: FAYYAD, U. M. et al. **Advances in Knowledge discovery and data mining**. Menlo Park, CA: AAAI Press, 1996.

[BOR95] BORLAND INTERNATIONAL. **Delphi User's Guide**. Scotts Valley, CA: Borland International, 1995.

[DEN91] DENIS, F.A.R.M.; MACHADO, R.J. **O modelo conexionista evolutivo**. Rio de Janeiro: IBM - Centro Científico Rio, 1991. (Relatório Técnico CCR - 128).

[FEL97] FELDENS, M. A. **Engenharia da descoberta de conhecimento em bases de dados**: estudo e aplicação na área de saúde. Porto Alegre: CPGCC da UFRGS, 1997. Mastership dissertation.

[KOH96] KOHAVI, R.; SOMMERFIELD, D. **Data mining using MLC++**. Montain View, CA, USA: Silicon Graphics, Inc, Available via anonymous FTP from starry.stanford.edu, file /pub/ronyk/mlc96.ps (1996).

[KOR93] KORTH, H. F.; SILBERSCHATZ, A. **Sistema de bancos de dados**. 2ª ed. São Paulo: Makron Books, 1993.

[KOS91] KOSKO, B. **Neural networks and fuzzy systems:** A dynamical systems approach. Englewood Cliffs: Prentice Hall, 1991.

[MAC89] MACHADO R. J.; ROCHA A. F. **Handling knowledge in high order neural networks:** the combinatorial neural model. Rio de Janeiro: IBM Rio Scientific Center, 1989. (Technical Report CCR076)

[MEN96] MENDONÇA, E. A. **Hycones III:** Sistema de apoio à UTI cardiológica. Porto Alegre: Instituto de Cardiologia do Rio Grande do Sul e Fundação Universitária Cardiológica, 1996. Mastership dissertation.

[MIC94] MICHIE, D; SPIEGELHALTER, D. J.; TAYLOR, C. C. **Machine learning, neural and statistical classification**. England: Ellis Horwood, 1994.

[PAR93] PARSAYE, K.; CHIGNELL, M. Data quality control with smart databases. **AI Expert**. San Francisco: Miller Freeman, May 1993.

[REA93] REÁTEGUI, E. B. **Um modelo para sistemas especialistas conexionistas híbridos**. Porto Alegre: CPGCC da UFRGS, 1993. Mastership dissertation.

[RID94] RIDDLE, P.; SEGAL, R.; ETZIONI, O. Representation design and brute-force induction in a Boeing manufacturing domain. **Applied Artificial Intelligence**, 8: 125-147, 1994. Available via anonymous FTP from ftp.cs.washington.edu, file /pub/ai/brute-aai94.ps.Z.

[ROS94] ROSA, S. I. V. **Aplicação de sistemas especialistas no processo decisório:** Uma abordagem híbrida. Porto Alegre: Faculdade de ciências econômicas da UFRGS, 1994. Mastership dissertation.

[SPI78] SPIEGEL, M. R. **Estatística**. São Paulo: Mc Graw-Hill, 1978.

[ZYT93] ZYTKOW, J. M.; ZEMBOWICZ, R. Database exploration in search of regularities. **Journal of Intelligent Information Systems**. v. 2, n. 1, 1993.