



Discovering graph structures in high dimensional spaces

V. Dubois & M. Quafafou

IRIN, Université de Nantes, France.

Abstract

This paper introduces a new idea to face this problem by adding the notion of functional dependency to the context of minimum encoding inference. Our analysis is based on a local discovery of relations between binary features. We present a very effective expression of class model local note, which can thus be computed without knowledge of any parameter value. The first method of data analysis described in this paper (also called crude method) is the direct application of our expression of the model class local note, and produce pieces of knowledge discovered in the data, organized in a graph.

1 Introduction

Discovering complex structures in a database is an emerging research area which deals with a hard task. Most of the proposed solutions are developed in a probabilistic context which leads to discovering Bayesian Networks traducing causality phenomena. A mark is computed for each model, and the selection of the best model is generally reduced to a ranking of these models according to their marks. Both Wallace's MML [2] and Rissanen's MDL approaches provide notation formulae and interpretation, and may be used for inducing models which describe the hidden structure of data. In this context, the "best" class of model (structure) is defined as the class which gives the best overall note for some parameter (see [3]). However, discovering such a structure is time consuming [6], especially in high dimensional spaces. We introduce in this paper a new method to deal with this problem. The starting point of our work is the decomposition of the problem by searching different partial "best" models. Each of these latter model represent

a dependency relation between one attribute and others. To achieve this goal, we added the notion of functional dependency to the context of minimum encoding inference. Consequently, searching the best model consists of discovering functional dependencies. Our analysis is based on a local discovery of relations between binary features. We present a very effective expression of class model local note, which can thus be computed without knowledge of any parameter value.

We present a first method of data analysis (also called crude method) as a direct application of our expression of the model class local note, and produce pieces of knowledge discovered in the data, organized in a graph. This method is based on an exploration of the attributes lattice, with some pruning. We refine this crude method using a probabilistic approximation of functional dependency between attributes. Consequently, only small parts of the attributes space is explored. The usefulness of our approach is shown considering the XOR problem.

2 Theoretical context

2.1 Functional and approximate dependencies

Let R be a relation schema.

A *functional dependency* over R is an expression $A \rightarrow b$, where A is a set of attributes, that is $A \subset R$, and $b \in R$ a single attribute.

Given a relation r over R , a *functional dependency* is said to hold *iff* for each pair of row (i, j) , equality over A implies equality on b , which is denoted $i[A] = j[A] \Rightarrow i[b] = j[b]$.

An *approximate dependency* over R is an expression $A \xrightarrow{\alpha} b$ that is satisfied *iff* there exists r' over R that satisfies $A \rightarrow b$, with same value on A ($i[A_r] = i[A_{r'}]$), and $p(i[b_r] = i[b_{r'}]) \geq \alpha$, that is to say α is the relaxation parameter we add to the functional (exact) dependency ([11] and [5]).

For A and b fixed, we define the *error* $\text{error}(A \rightarrow b)$ on the dependency as the least value e such that $A \xrightarrow{\alpha} b$ holds with $\alpha = 1 - e$.

2.2 MML Metrics

Definition of approximate dependencies enable search of dependencies in real world databases, because limited error or exception does not invalidate an approximate dependency. But the functional dependency error is not always a sufficient criteria of interest. For example, $A \subset A'$ implies that the error on $A \rightarrow b$ is higher than the one on $A' \rightarrow b$, even though the interest of A may be higher. As a consequence, the error on $R - \{b\} \rightarrow b$ is the smallest possible error for b fixed.

It is the reason why we propose a measure of interest for approximate dependencies [7] based on MML (Minimum Message Length) Metrics [9]. The principle is to suppose that we want to use knowledge we already have on $r[A]$ to reduce the amount of data required to transmit $r[b]$. Such data reduction may be achieved using huffman code with the distribution of $r[b]$. As $r[A]$ is already known, we use conditionnal probabilities $p(i[b] \mid i[A])$. These probabilities are grouped in a CPT (Conditionnal probability Table). A message describing $r[b]$ is composed of the

conditionnal probability table (CPT) and the encoded data. So the total length of the message is given by

$$L(b | A) = L(b | A, \text{CPT}(A, b)) + L(\text{CPT}(A, b)) \quad (1)$$

If each attribute b of R is expressed using such a message, it defines an oriented graph G by the follpwing way : if b is encoded using knowledge of the set $A = \{a_1, \dots, a_n\}$ of attributes, each a_i is linked to b . A is said to be the set of parents of b , and is denoted by $\text{Pa}(b)$. If the graph G is acyclic, it means that it is possible to transmit attributes in R one at a time, transmitting an attribute only if its parents have already been transmitted (there is no cycle in the graph). The length required is given by :

$$L(G) + \sum_{b \in R} L(b | \text{Pa}(b), \text{CPT}(\text{Pa}(b), b)) + L(\text{CPT}(\text{Pa}(b), b)) \quad (2)$$

The graph G and the set of CPTs related to G and to r is also called a bayesian (or probabilist) network (BN) (see [4] and [8]), and it defines a distribution over R . MML metrics are usually defined on such a network. So $L(b | A)$ may be seen as the local MML cost of a BN.

2.3 MML Metric Approximation using Dependency error

In the equation (1), the local cost is the sum of two terms. The first one denotes the quality of the local approximation, and the second one the cost induced by the structure (the length of the CPT). The structure cost only depends on the number of parents, and can be computed easily, wheras the direct evaluation of $L(b | A, \text{CPT}(b, A))$ is hard to compute. We will show how to use the previously defined dependency error to estimate this term.

If a functional dependency exists between a set A of attributes and a unique attribute b , it is possible to compute b from A without any error or additionnal information. Conversely, if such a computation is possible, it implies a functional dependency : $\text{error}_r(A \rightarrow b) = 0 \Leftrightarrow L(b_r | A_r, \text{CPT}(b_r, A_r)) = 0$

Given $r, A \subset R, b \in R, \alpha = 1 - \text{error}(A \rightarrow b)$, we choose the same r' as in the approximate dependency definition. Then, we introduce a new boolean attribute e defined using r' and r in the following way : $\forall i \in r, i[e] = (i[b_r] \neq i[b_{r'}])$. It follows (by construction) that $p(i[e] = 1) = 1 - \alpha = \text{error}(A \rightarrow b)$

Given $i[b_{r'}]$ and $i[e]$, it is possible to compute $i[b_r] = (i[b_{r'}] \neq i[e])$.

$$L(b_r | A_r, \text{CPT}(b_r, A_r)) = L(b_{r'}, e | A_{r'}, \text{CPT}(b_{r'}, A_{r'})) = L(e | A) \quad (3)$$

It is straightforward that $L(e | A) \leq L(e)$. Thus, $L(b | A, \text{CPT}(b, A))$ may be approximated by $L(e)$. If e does not depend on A (error is uniform or null), no error is committed.

2.4 FD error estimation

In order to compute the $L(e)$, we have to calculate its probability $p(i[e] = 1)$.

A	b_r	$b_{r'}$	e
A_1	0	0	0
	1		1
A_2	0		1
	1	1	0

Abreviation	Full expression
$p(\mathbf{X})$	$p(i[X] = \mathbf{X})$
$p(X = \mathbf{X} \mid \mathbf{A})$	$p(i[X] = \mathbf{X} \mid i[A] = \mathbf{A})$
$X_{=}$	$i[X] = j[X]$
$p(X = \mathbf{X} \mid Y_{=})$	$p(j[X] = \mathbf{X} \mid j[Y] = i[Y])$

From general conditionnal probability properties, we have :

$$p(i[A] = \mathbf{A}) = \sum_{\mathbf{B}} p(i[B] = \mathbf{B}).p(i[A] = \mathbf{A} \mid i[B] = \mathbf{B}) \quad (4)$$

We apply this to $p(i[e] = 1)$:

$$\begin{aligned} p(i[e] = 1) &= \sum_{\mathbf{A}} p(i[A] = \mathbf{A}).p(i[e] = 1 \mid i[A] = \mathbf{A}) \\ p(i[e] = 1) &= \sum_{\mathbf{A}} p(\mathbf{A}).\min(p(b = 0 \mid \mathbf{A}), p(b = 1 \mid \mathbf{A})) \end{aligned} \quad (5)$$

We express $\min(a, b)$ by the following way : $\min(a, b) = \frac{a+b}{2} + \frac{|a-b|}{2}$. Moreover, we have : $p(A \mid B) + p(\neg A \mid B) = 1$. Thus, the expression of $p(i[e] = 1)$ is now:

$$p(i[e] = 1) = \sum_{\mathbf{A}} p(\mathbf{A}). \left(\frac{1}{2} - \frac{|p(b = 0 \mid \mathbf{A}) - p(b = 1 \mid \mathbf{A})|}{2} \right) \quad (6)$$

Remark that in general we have : $(p(b) - p(\neg b))^2 = 2(p(b)^2 + p(\neg b)^2) - 1$

Applied to the previous equation, it gives :

$$p(i[e] = 1) = \sum_{\mathbf{A}} p(\mathbf{A}). \left(\frac{1}{2} - \frac{\sqrt{2.(p(b = 0 \mid \mathbf{A})^2 + p(b = 1 \mid \mathbf{A})^2) - 1}}{2} \right) \quad (7)$$

Using the following general property, $\text{avg}_i(f(i[A])) = \sum_{\mathbf{A}} p(i[A] = \mathbf{A}).f(\mathbf{A})$

The previous equation leads us to :

$$p(i[e] = 1) = \frac{1}{2} - \frac{\text{avg}_i(\sqrt{2.(p(b = 0 \mid A_{=})^2 + p(b = 1 \mid A_{=})^2) - 1})}{2} \quad (8)$$

Approximating $\text{avg}_i(\sqrt{x_i})$ by $\sqrt{\text{avg}_i(x_i)}$ only generates a second order error. Moreover, the sign of error is known :

$$p(i[e] = 1) \leq \frac{1}{2} - \frac{\sqrt{2.\text{avg}_i(p(b = 0 \mid A_{=})^2 + p(b = 1 \mid A_{=})^2) - 1}}{2} \quad (9)$$

As we only have binary attributes, $p(b_{=} \mid C) = p(b = 0 \mid C)^2 + p(b = 1 \mid C)^2$

The dependency error is now approximated by the following equation :

$$p(i[e] = 1) \leq \frac{1}{2} - \frac{\sqrt{2.p(b_{=} \mid A_{=}) - 1}}{2} \quad (10)$$

Although this formula seems to be computationally harder than direct one, we will show how pre-calculation of a reasonable quantity of values helps in computing the error at low cost on many pairs (A, b) without any additional data access. This result is more easily understood if we remark that explicit or implicit construction of $f : A \mapsto b$ is not required, and that quantity such as $p(A_i = A_j)$ are involved in many different error expressions, and may be computed once and for all.

3 Method

For each attribute b , the best possible set of attributes A is computed. Interest of a given set of parents is evaluated by the following way : From relation r , we compute $p(i[b] = j[b] \mid i[A] = j[A])$. Then, Using equation (10), $p(i[e] = 1)$ is evaluated. $L(e)$ is computed from $p(i[e] = 1)$, and is only slightly greater than $L(b \mid A_r, \text{CPT}(A, b))$ (equation (3))

To find the best set of parents, we will search through the lattice of possible parents, from the smallest set to the largest [10]. So, our reference will be the length related to the empty set $\emptyset$.

We define the *unitary length* as the length divided by the number $l = |r|$ of instances in our relation, and denote it $LU(X) = L(X)/l$. Let $\Delta(A, b) = (L(b \mid A) - L(b \mid \emptyset))/l$, which can be rewritten as follow :

$$\Delta(A, b) = LU(b \mid A) - LU(b) - LU(\text{CPT}(b)) + LU(\text{CPT}(b, A)) \quad (11)$$

$\text{CPT}(b, A)$ is a table with $2^{|A|}$ rows, and values in $[0, 1]$. Let k be the length which allows to code a single value of this table. Hence, the length of such a table is $L(\text{CPT}(b, A)) = k \cdot 2^{|A|}$, and $LU(\text{CPT}(b, A)) = \frac{k}{l} \cdot 2^{|A|}$

Let us introduce a constant $C = LU(b)$ and a function $g : x \mapsto \frac{k}{l} \cdot 2^x$. Δ expression is now : $\Delta(A, b) = LU(b \mid A) + g(|A|) - C$. We consider $\Delta(A, b)$ as a valuable measure (mark) of interest of the approximate dependency $A \xrightarrow{\alpha} b$ for the following reasons :

- $\Delta(A, b)$ is meaningful : it expresses the average data reduction length per row of r we expect if we use the knowledge of A to code b . Structural cost (CPT encoding length) is taken into account.
- Interpretation of the value of $\Delta(A, b)$ is straightforward: either it is negative and the use of $A \xrightarrow{\alpha} b$ reduces the amount of data, or it is not, and $A \xrightarrow{\alpha} b$ is of no interest.
- $A \subset A'$ does not imply $\Delta(A, b) \leq \Delta(A', b)$ because of the structural cost. The greater the set of parents, the smaller the error has to be.
- The acyclic graph G on attributes of R that maximizes $\sum_b \Delta(\text{Pa}(b), b)$ is also the best structure for a bayesian network in the sense of MML. However, local optimisation usually results in a non acyclic graph structure. Hopefully, this degenerated structure is helpful to detect and understand relations between attributes.

- By choosing a value of $\frac{k}{l}$, it is possible to tune structural penalty. It is the only parameter of function Δ . The higher it is, the lower the maximal size of valuable sets of parents are. For $k = 0$, use of Δ as a metric is equivalent to using the error of approximate dependencies.

Following properties may easily be proved :

- $LU(b | A) \in [0..1]$
- $A \subset A' \Rightarrow LU(b|A) \geq LU(b | A')$
- $g > 0$ and g grows strictly.
- $C \geq 0$

Our objective is to minimize $\Delta(A, b)$, for r and b fixed. Let us define $\min_{|A|=x} LU(b | A) = LU_{\min}$. It follows that

$$\min_{|A|>x} \Delta(A, b) \geq \min_{|A|=x} \Delta(A, b) + g(x+1) - g(x) - LU_{\min} \quad (12)$$

Moreover, if we have $g(x+1) - g(x) > LU_{\min}$,

$$\min_{|A|\geq x} \Delta(A, b) = \min_{|A|=x} \Delta(A, b) = LU_{\min} + g(x) - C \quad (13)$$

So, as soon as $g(x+1) - g(x) > LU_{\min}$, the optimal parent set has been found and its local length is $LU_{\min} + g(x) - C$. This result avoids full space exploration.

As g grows strictly, $L_{\min} = 0 \Rightarrow g(x+1) - g(x) > L_{\min}$: If we found a set of parents for which the FD is exact, we know it is the best set of parents possible and search ends.

4 Optimisation and pruning

4.1 Reducing number of instances

If we want to evaluate $p(i[b] = j[b] | i[A] = j[A])$ from the relation r , we need to compute $|r|^2$ differences between instances. This is not acceptable for large databases. For this reason instance complexity is reduced by sampling the set of all possible pairs of instances. A sample size of $|r|$ pairs appears to be sufficient, and is more reasonable than a sample size of $|r|^2$. If relation r is highly redundant, it is possible to lower the sample size without quality loss on results.

4.2 Pruning space of possible parents

Let a and b be two binary attributes. a and b may depend on each other only if $\exists(i, j), i[a] \neq j[a], i[b] \neq j[b]$. Otherwise, $a \notin \mathbf{Pa}(b), b \notin \mathbf{Pa}(a)$, and lattice of possible parents is reduced. Although it is not efficient for random variables, it occurs to be effective for textual data, where most of the words are independant and low frequencies.



5 Algorithms

5.1 General crude algorithm

```
begin
  initialisation (read data)
  for each attribute b
    for each set A of possible parents
      compute error(A,b)
      score A
    choose best set of parents
end
```

The score function used in this algorithm is $\Delta(A, b)$, its only parameter is $\frac{k}{l}$. The lower it is, the more arcs we have in resulting graph. This algorithm is refined by only evaluating sets of parents whose size is lower or equal than :

- the size of best parent set (if not empty)
- the size where CPT encoding cost is greater than attribute b encoding size. (if empty set is the best parent set)

5.2 Initialization

```
begin
  repeat
    read random instance I1
    read random instance I2
    compute D = { i | (i in I1) = (i in I2) }
    number[D] = number[D] +1
  until enough data read
end
```

Initialisation is the only part of the algorithm where we access the data. Further computation is based on results computed by this procedure.

5.3 Error evaluation

```
begin error(A,b)
  nA = 0
  nAb = 0
  for each E such that A subset E
    nA = nA + number[E]
    if b in E then nAb = nAb + number[E]
  return (1 - sqrt(2 * nAb / nA - 1) ) /2
end (error)
```

Error evaluation only uses formulae of approximate dependency error estimation.



6 Xor test

The previously described technique has been implemented. The following tests have been done, and many applications seem to be interesting.

We used a set R of 99 random independant binary attributes and a single computed $c = a \text{ xor } b$ (a and b choosed among random attribute). Our program has been run to choose best parents for each attribute. As a result, it found the following approximative dependency : $a, b \rightarrow c$ and all of its permutations. It is obvious in our case that these three FD are exact and equivalent, and none is better than the other.

The interest of this sample set is to show that no approximative dependency between random independent data is generated by our algorithm. This is a non trivial result, because, on same data, $\text{error}(X \rightarrow y) = 0$ for X large enough. It means that $X \rightarrow y$ holds, but is of no interest because this exact dependency is not relevant (it take place between random independent set of attributes). Such *faked* dependencies occur as soon as the dimension of the relation is high enough. Our measure of interest $\Delta(X, y)$ handles this problem in a correct, reasonable and well-founded way.

Another interest of this test is that a, b, c are independent when we consider only two among them, whereas they are dependent when considered together. Hence, classical tree based MML bayesian networks construction algorithms [1] will not be able to detect the link between attributes (these kind of algorithms only consider mutual information between pairs of attributes). Our algorithm detects them because it does not rely on the relation between attributes, it only looks for dependencies (in a broad meaning). Moreover, if we only consider sets of only one attribute, our metric is equivalent to mutual information.

This test is the same as the previous one, except that noise has been added to attribute a (we changed a value in x percent of instances). The program is still able to find correct dependencies though it requires a longer sample of data to find attributes involved in the noisy xor among random ones. It validates the fact that our method is noise tolerant.

7 Discussion and conclusion

This paper introduces a method for discovering graph structures in high dimensional spaces. Our preliminary research consider only boolean attributes. Our approach considers the functional dependency (fd) notion in the MML context. Moreover, we have focused on the approximative functional dependency (afd) which are generally based on probabilistic frameworks. Hence, we have defined a note of an afd which is based on its encoding lenght. Consequently, we combine the error of the fd and the encoding structure cost of afd.

However, a correct choice of two parameters, i.e., the size of the sample and the weight of the structure, is required to get the appropriate result. The sample size is the number of pairs of rows our program will read in the database. This access to the database is done once and for all during the initialisation. This parameter



needs to be increased in the following cases :

- if $|r|$ increases
- if the noise increases
- when looking for noisy dependencies
- when looking for dependencies with low support
- when an increased accuracy is required

The parameter k/l sets the weight of structural penalty. The higher it is, the smaller the size of search space will be. Reducing this parameter has a dramatic effect on the complexity (related to the number of attributes). The degree of complexity in the number of attributes depends directly on the value of this parameter. For high value of k/l , the complexity will be linear, and for 0, it will be exponential (n^n).

References

- [1] Chow, C. & Liu, C. Approximating probability distributions with dependence trees. *IEEE Trans. on Information Theory*, **14**, 462–468, 1968.
- [2] Chris Wallace, K. K. & Dai, H. Causal discovery via MML. Technical report, Departement of computer science, Monash university, 1996.
- [3] Dai, H. A minimal causal model learner. In *Proceedings of PAKDD '99*, pp. 400–408, 1999.
- [4] Heckerman, D. A tutorial on learning with bayesian networks. Technical report, microsoft, 1996.
- [5] Kivinen, J. & Mannila, H. Approximate dependency inference from relations. *Theoretical Computer Science*, **1**, 129–149, 1995.
- [6] L. E. Sucar, J. P.-B. & Ruiz-Suárez, J. Learning structure from data and its application to ozone prediction. *Applied Intelligence*, **7**, 327–338, 1997.
- [7] M. Ganesh, J. S., N. Jain & Wijesekera, D. Efficient mining for functional dependencies. Technical report, Department of computer science, university of Minnesota, 1996.
- [8] Nir Friedman, M. G. Learning bayesian networks from data (slides).
- [9] Oliver, J. J. & Baxter, R. A. Mml and bayesianism : similarities and differences. Technical report, Department of computer science, Monash University, 1995.
- [10] Savnik, I. & Flach, P. Bottom-up induction of functional dependencies from relations. In Piatetsky-Shapiro, , (ed.), *Knowledge Discovery in Databases, Papers from the 1993 AAAI Workshop (KDD '93)*, pp. 174–185. AAAI, 1993.
- [11] Ykä Hutala, P. P., Juha Kärkkäinen & Toivonen, H. Efficient discovery of functional an approximate dependencies using partitions. Technical report, University Of Helsinki, 1997.