LEARNING BAYESIAN DEPENDENCE MODEL FOR STUDENT MODELLING

Adina COCU

Department of Computer Science and Engineering,
University "Dunărea de Jos" of Galați, 2 Științei, 800146, Romania
Adina.Cocu@ugal.ro

Abstract: Learning a Bayesian network from a numeric set of data is a challenging task because of dual nature of learning process: initial need to learn network structure, and then to find out the distribution probability tables. In this paper, we propose a machine-learning algorithm based on hill climbing search combined with Tabu list. The aim of learning process is to discover the best network that represents dependences between nodes. Another issue in machine learning procedure is handling numeric attributes. In order to do that, we must perform an attribute discretization pre-processes. This discretization operation can influence the results of learning network structure. Therefore, we make a comparative study to find out the most suitable combination between discretization method and learning algorithm, for a specific data set.

Keywords: Bayes network, structure learning, student modelling, intelligent tutoring system.

1. INTRODUCTION

In many fields of artificial intelligence, it is necessary to learn a complete dependence network between a set of variables. The values of the variables have been obtaining from real world and often involve numeric attributes. For Bayesian network learning algorithms, numeric attributes must be pre-processed by discretization in order to apply a searching method and local score metrics. In the situation when data set is complete observable and the structure network is unknown it must be applied a search in models space for construction of graph topology (Murphy, 1998). Then it is applied an estimation procedure for direct determination of the conditional probabilities tables.

We propose to use for learning purposes a hill climbing search algorithm that add and delete edges between network nodes with no pre-ordering of variables. To optimize the search algorithm we use a list, named Tabu, that store the visited steps in order to choose the least worse candidate structure in the neighbourhood when search method hits an local optimum.

Combination of different discretization method with proposed learning algorithm can lead to different network structures. We use equal length interval discretization, equal frequency discretization and empiric adjusting the boundaries of discretization interval. For each learned structure we calculate diverse local score metrics and then compare them with the values of the other learned structures. This can be consider an optimization problem where a local score measure of a network structure, given the training data, needs to be maximize. Local score metrics used are based on probabilistic Bayesian approach, minimum description length (MDL), entropy function metric and Akaike Information Criterion (AIC) (Ying, *et al.*, 2002; Ying, *et al.*, 2003; Perner, *et al.*, 1998).

For experimental trials, we use a data set representing 18 concepts about object oriented programming language and the assessments results for these concepts obtained from 195 students. The aim is to discover through learning process the best dependence network between concepts for diagnostic assessment purposes (Cocu, 2007). The data set values are into interval of minimum $v_{min}=0$ (mean the student do not know anything about concept) and maximum $v_{max}=0.5$ (indicate the student know very well the notions). The learning data set is relatively

This paper was recommended for publication by Luminita Dumitriu

small in comparison with the other benchmarks in the dependence-modeling domain (Cooper, *et al*., 1992).

2. DISCRETIZATION FOR DEPENDENCE MODEL LEARNING

The attributes from our data set have continuous numeric values. The machine-learning algorithm must calculate some score based on the number of possible combination between nodes values. Even for attributes that have a finite, but large number of values, there will be very few training instances for any one value. Therefore, it is often desirable to cumulate a range of values into a single value for the purpose of measurement calculation. This assumes that attribute values must be discrete with a finite number, because it is not possible to assign metrics to all single value of an attribute with an infinite number of values. The discretization is the operation that transforms numeric values of attributes into discrete values (Ying, *et al*., 2002). Through discretization process, the attributes became categorical, with a finite number of values.
The proportion between number of categories of each attribute and number of values from each category influence the results of learning process. In our study, because all attributes takes values from the same interval and represent the same measure (assessment of a concept) we apply discretization conditions in the same time, for all concepts.
In the literature related to learning machines techniques, there are a set of discretization methods like: equal width or equal frequency discretization, entropy based discretization method, iterative discretization, fuzzy and lazy discretization, proportional k-interval discretization, non-disjoint discretization, chi-learning vector quantization based discretization method, merge discretization and histogram based discretization (Ying, *et al*., 2002; Ying, *et al*., 2003; Perner, *et al*., 1998). Each of them is more suitable for using in one situation. In this study, the discretization methods used are equal length intervals, equal frequency and empiric discretization. The others discretization method are useful in classification problems, because they use different measures of class attribute and it is not functional for dependence discovery (Ying, *et al*., 2002).
Suppose there are V_i numeric attributes from interval $[v_{min}, v_{max}]$ with n complete records (for which the value of V_i is known) training instances.

2.1 Equal Length Discretization Method

This method (ELD) (Ying, *et al*., 2002; Ying, *et al*., 2003) divides the interval between v_{min} and v_{max} into m intervals of equal length. Thus the intervals will have length $len = (v_{max} - v_{min})/m$
and the cut points are at:
$v_{min} + len; v_{min} + 2*len; \ldots ; v_{max}-len$.

In our experiments we use m set as 3 (with the cut points 0.17 and 0.33) and 4 intervals (with the cut points 0.125, 0.25 and 0.375).

2.2 Equal Frequency Discretization Method

This method (EFD) (Ying, *et al*., 2002; Ying, *et al*., 2003) divides the sorted values into m intervals so that each interval contains approximately the same number of training instances. In our experiment, because the data set represent the same measure of student assessment, we consider frequencies for all the values of all attributes. Because the interval values are relatively small and many students acquire maxim values for assessment, the obtained cut points are 0.2 and 0.5.

2.3 Empiric Discretization Method

Both ELD and EFD methods can have much attribute information loss since m parameter is determined without reference to the properties of the training data. In this situation, the domain expert makes a visual inspection of the data and changes the cut points for the situation of 3-interval discretization, in order to obtain a fine tune adjustment. The new cut points are 0.2 and 0.4 and frequencies differ for each attribute.

3. LEARNING ALGORITHM

In our case, the data learning set is complete. The algorithm for learning the network structure will use the data set. Because the quantity of learning data is small comparative with the number of attributes, we propose to use a hill climbing method combined, for optimization reasons, with a list of visited solutions. In problems that can have many solutions is usually used the hill climber method. The aim of method is to find the most probable model. It uses a local measure to score each solution in order to determine the best one. The algorithm is often used in classification problems based on Bayesian approach (Bouckaert, 2004; Friedman, *et al.*, 2001). We choose the algorithm because its capacity to find good solutions in acceptable time interval. In classification learning algorithm, the network structure is known (all the attributes have the class attribute like parent). In contrast with classification, in dependence structure learning we do know nothing about links between nodes.
In our situation, the algorithm starts with an empty dependence network (none of the node do not have parents) and successively operate adds or delete or reverse arcs between attributes nodes with the aim of improving the solution. At every step, the algorithm calculates score of each node based on all the possible combination of values that can be take by node with its parents. If the local score is better after new operation, than this operation is saved and then update the network structure. It is possible, that at a moment of search, the algorithm find a local

minimum, where it cannot see any improvement anymore. In this moment, the algorithm ends its searches. Ideally, at that point, solution founded is close to the most favourable, but it is not guarantied that hill climbing will ever come close to the optimal solution. In big search spaces, the optimum can be founded after too many steps or cannot be found at all. This is the reason for we append a maximum number of runs like parameter for search algorithm.
In dependence modelling, unlike in other Bayesian network application (like classification problems) all the nodes have the same priority. However, in order to introduce priory user knowledge we add the possibility to set some node that will not have parents (root nodes).
As follow, we give the learning algorithm steps. At the end of execution, it is obtained the best network dependence structure for user predefined number of runs.

```
Initialize networkStructure, bestStruct=netStruct
Repeat numberOfRuns
    CurrentScore=CalcNetScore(netStruct)
    for all nodes
     currentNodeScore=CalcScore(node)
     operation=Search(OptimalOp=add/del/reverse)
     newNodeScore==CalcScore(node<-operation)
     if(currentNodeScore < newScoreNode &
operation ∉ TabuList)
      netStruct=PerformOperation(operation)
      newScore= CalcNetScore(netStruct)
        TabuList+=operation
        if(newScore<currentScore)
            netStruct=bestStruct
        else bestStruct=netStruct
```

In the literature (Bouckaert, 2004), there are several local score functions that can be used for disseminate the better solution. In the next section, we briefly present the function employed in our experiments.

3.1 Local score metrics

Local score metric is a function that calculates the score for each node and the score for entire network through sum for all nodes.
As follow, we describe these functions based on Bayesian approach, minimum description length (MDL), entropy based metric and Akaike Information Criterion (AIC).
Suppose we have n attributes, each with cardinality notated with c_i ($1 \le i \le n$) and cardinality of parents notated like in equation (1).

$$p_i = \prod_{v_j \in parents(v_i)} c_j \quad (1)$$

If a node do not have parents then $p_i = 1$.
We denote with Cij ($1 \le i \le n, 1 \le j \le p_i$) the number of records in dataset for possible combination of parents(v_i) taking the j-th value. And we use C_{ijk} ($1 \le i \le n, 1 \le j \le p_i, 1 \le k \le c_i$) to denote the number of combination for parents(v_i) taking the j-th value and v_i taking the k-th values. These numbers of values are link through sum in equation (2).

$$C_{ij} = \sum_{k=1}^{c_i} C_{ijk} \quad (2)$$

Let us notate with C the total number of records in database.
The Bayesian metric uses the gamma function and parameter alpha (3). Parameter alpha determines how easily we change our belief about quantitative nature of dependences. If alpha is small we believe any change in data, and if it big we do not believe data easily.

$$Bayes = \prod_{i=1}^{n} \prod_{j=1}^{p_i} \frac{\Gamma\left(\frac{\alpha}{p_i}\right)}{\Gamma\left(\frac{\alpha}{p_i} + C_{ij}\right)} \prod_{k=1}^{c_i} \frac{\Gamma\left(\frac{\alpha}{c_i p_i} + C_{ijk}\right)}{\Gamma\left(\frac{\alpha}{c_i p_i}\right)} \quad (3)$$

The entropy metric is the measure of the amount of information that is missing in dataset (4).

$$Entropy = -C \sum_{i=1}^{n} \sum_{j=1}^{p_i} \sum_{k=1}^{c_i} \frac{C_{ijk}}{C} \log \frac{C_{ijk}}{C_{ij}} \quad (4)$$

For the others metric we define the number of parameters K as (5).

$$K = \sum_{i=1}^{n} (c_i - 1) p_i \quad (5)$$

Akaike Information Criterion (AIC) evaluates goodness of fit in a statistical model (6). A big AIC score implies a good model. AIC penalizes the addition of parameters to the model.

$$AIC = Entropy + K \quad (6)$$

Minimum Description Length (MDL) (7) take into account the model simplicity and model fit to the data by minimizing the length of a joint description of model and data given the model.

$$MDL = Entropy + \frac{K}{2} \log C \quad (7)$$

We observe that the computed score values are useful in search algorithm for choosing the best structure. In addition, we use the difference in score across a range of models obtained with different discretization intervals and different algorithm parameters. Typically, we select among candidate models by choosing the one leading to the biggest score.

4. EXPERIMENTAL RESULTS

In classification problems, we ca verify the accuracy of learning algorithm with a test dataset or with n-fold cross validation methods. Unlike these problems, in which the network structure is known, in dependence model we do know neither structure, nor conditional tables. Therefore, it not exist data for comparison reasons. In association discovery algorithm it is used a confidence factor for choose the best rules. Therefore, based on this observations, in our experiment, for dissemination between structures resulted after run hill climbing method combined with Tabu list, we will use different scores of the entire network for evaluate them. In search algorithm it will be used only the Bayesian score.
The experimental cases are numbered in table 1 and for each case are detailed testing conditions in columns (1-number of cases, 2-discretization method, 3-number of intervals, 4-maximum number of parents, 5-alpha parameter used in Bayesian score metric, 6-total number of runs for search algorithm, 7-if it is considered reversibility of arcs).

Table 1. The experimental cases with conditions and parameters

1	2	3	4	5	6	7
1	ELD	3	3	0.5	200	true
2	ELD	3	3	0.5	500	true
3	ELD	3	3	1.5	200	true
4	ELD	3	5	0.5	200	false
5	ELD	3	5	0.5	200	true
6	EFD	3	5	0.5	200	false
7	EFD	3	3	0.5	200	true
8	EFD	3	3	1.5	200	true
9	ELD	4	5	0.5	200	true
10	ELD	4	3	0.5	200	true
11	ED	3	3	0.5	200	true

The results of experimental cases are grouped through types of score metrics and showed in table 2.

Table 2. The comparison between results obtained for different run conditions

No.	Scores			
cases	Bayes	Entropy	AIC	MDL
1	-3350.044	-3307.951	-3647.951	-4204.361
2	-3350.044	-3307.951	-3647.951	-4204.361
3	-3365.110	-3273.841	-3577.841	-4075.337
4	-3348.586	-3398.807	-3882.807	-4674.873
5	-3348.777	-3644.945	-4416.945	-5680.323
6	-3514.089	-3599.078	-4111.078	-4948.966
7	-3521.012	-3448.528	-3744.528	-4228.932
8	-3528.780	-3444.934	-3800.934	-4383.528
9	-4047.816	-5051.059	-6653.059	-9274.731
10	-4049.273	-4528.788	-5554.788	-7233.837
11	-3409.201	-3372.019	-3752.019	-4373.889

Studying the results from table 2 from learning concepts structure point of view we make next observations:

- For small quantity of experimental data, increasing number of steps run does not improve the quality of results (in cases 1 and 2). We consider that this dataset it is small because it contains 195 records for 18 attributes. For example, the benchmark iris dataset contain 150 instances for five attributes.
- Increasing the value of alpha parameter (means that the user do not invest much trust in data) conduct to a drop of Bayes scores (in cases one compared with three and seven compared with eight). The observation is true independently from selection of discretization method.
- Using the possibility to search reversed arcs in Hill Climbing algorithm do not change to much the Bayesian score, but drop the others scores (cases four and five).
- Searching for more parents improve a little bit the Bayesian score, but substantial fall the other scores (cases one and five). In consequence, we overtake to the conclusion that choosing the parents number depends only by problem specification. In our case, in learning programming concepts, because we have few concepts, degree of connections between them must be maximum equal with three.
- The best results are obtained for ELD discretization method with 3 intervals and the worst for situation with 4 discretization intervals. For object oriented concepts notated with maximum score 0.5, the discretization length interval is small and increasing the number does not improve the results.
- In addition, a discretization based on frequency does not help, because the distribution of learner's knowledge is not uniform. The equal frequency discretization is useful, especially, in classification problems.
- The empirical discretization gives acceptable results, but there are not better than ELD in this case.

Comparing all the learned graphs do not lead to the observation of major differences. The structures differ only by some arcs, but the most important links between nodes are present in all graphs.
In conclusion, in case one we obtain the best results.

5. CONCLUSION

We propose this approach because in the literature exist few dependence learning techniques (B-course, Bayesia) capable to make an optimized search and learn with maximum efficiency a network structure.
Considering the small dimension of dataset, the learning algorithm must be able to learn as much as possible from reality evidence and this is the reason why it is necessary a comparative study between different running conditions. Choosing equal length discretization methods leads to better results in dependence discovery among object oriented programming concepts. In addition, empiric methods give satisfactory results. However, although they may be deemed simplistic, both methods can be used and work surprisingly well for dependency attribute discovery. One reason suggested is that in this kind of problems, with high dependency by human nature, experts experience is an important factor. Another

issue is that the concepts must be equilibrated and be correct scored, without dependence by number of good or lazy students.

The learning domain expert recognize that using a network structure learning methods based on real life experience lead to surprisingly results, usually unobserved in teaching experience.

REFERENCES

Bayesia, http://www.bayesia.com/ B-Course, version 2.0.0, http://b-course.cs.helsinki.fi/obc/

Bouckaert R., (2004), Bayesian Network Classifers in Weka, *Internal report University of Waikato*, New Zeeland.

Cocu Adina, (2007), Knowledge Based Diagnostic System with Probabilistic Approach, *at 13th International Symposium on Modeling, Simulation and Systems' Identification, SIMSIS*, 21-22 september, Galati, ISBN 978-973-88413-0-7, pp. 17-22.

Cooper G.F., Herskovits E., (1992), A Bayesian Method for the Induction of Probabilistic Networks from Data, *Machine Learning, vol. 9,* pp. 309--347.

Friedman Nir, Koller Daphne, (2001), Learning Bayesian Network from Data, Technical report.

Murphy Kevin, (1998), A Brief Introduction to Graphical Models and Bayesian Networks.

Petra Perner, Sascha Trautzsch, (1998), Multi-Interval Discretization Methods for Decision Tree Learning, *Advances in Pattern Recognition, A. Amin, D. Dori, P. Pudil, and H. Freeman (Eds.), LNCS 1451, Springer Verlag*, pp.475-482

Ying Yang, Geoffrey I. Webb, (2003), Weighted Proportional k-Interval Discretizationfor Naive-Bayes Classifiers, *Proceeding of PAKDD 2003*, pp. 501-512.

Ying Yang, Geoffrey I., (2002), A Comparative Study of Discretization Methods for Naive-Bayes Classifiers, In *Proceedings of PKAW, The 2002 Pacific Rim Knowledge Acquisition Workshop*, Tokyo, Japan, pp. 159-173.