A Genetic Algorithm for Learning Bayesian Networks to Match Training Sets

Yingbiao Ling [1] Yunfei Jiang[2] Xiangjun Wu [1]

[1] Department of Computer Science and Technology
Information Science and Technology School at Sun Yat-Sen University
[2] Institute of Computer Software of Sun Yat-Sen University

Abstract

In this paper, a new genetic learning algorithm of Bayesian network, the GLA, is proposed. Several virtues of the GLA are outstanding: 1) the implicit parallelity of the genetic search techniques ensures high efficiency; 2) the scoring metric functions are not necessarily differentiable, various fitness functions can be used as an input of algorithm; 3) it returns the optimal solution for a training set rather than a local optimal one or an average one. The principles and experiment works of the GLA are presented. By solving a sample problem, the GLA shows acceptable efficiency and can return the optimal solution.

Keywords: Data Mining, Bayesian Network, Training Sets, Scoring Functions, Genetic Algorithms.

1. Introduction

Bayesian networks[1] , also called causal networks or belief networks, are very attractive models for knowledge representing and uncertain reasoning in the many applications such as data mining(DM), machine learning(ML) and intelligent agent(IA), but it is difficult and time-consuming to construct Bayesian Networks by the domain experts. As collecting sample data of various domains is getting easier and easier, it attaches more and more emphasis to the algorithms for constructing good Bayesian Networks to match the corresponding domain..

The problem of learning Bayesian Networks is NP-hard. Some heuristics local search approaches are proposed to solve the problem of learning a Bayesian network. The Gradient Descent method [5] is based on greedy blind hill-climbing, but it usually gets trap at a local optimal model and only works with differentiable scoring functions. The structural EM method [7] is also blind hill-climbing one with a bit difference, but it is destined to be trapped at a local optimal or to wander on a plateau. A stochastic approach, the Monte Carlo method [1], is a blind search technique that results in low efficiency, and it is difficult to reach the optimal models but an average one. In this paper, a new genetic algorithm for learning Bayesian network (GLA) is proposed. In the following sections of this paper, we first give a brief introduction to basic knowledge of the Bayesian networks, and then present the principles of the GLA. At last, experiment results of GLA are explained.

2. Bayesian networks and the learning problems

A Bayesian network consists of two components: a network structure of a directed acyclic graph (DAG) and the associated conditional probability tables (CPTs). Bayesian networks are used for describing probability distribution of dependent variables in a given variables set. Let see a simple example. Suppose we have a sample data set as Table 1. There are four random variable G, M, B and L. They are 4 binary-valued variables.

No.	x_1	x_2	x_3	x_4	QTY of instance
1	1	1	1	1	54
2	1	1	1	0	1
3	1	0	1	1	7
4	1	0	1	0	27
5	0	1	1	1	3
6	0	0	1	0	2
7	0	0	0	1	4
8	0	0	0	0	2
Total					100

Table 1. A sample data set

There are 16 joint probabilities over these variables, each of the form $p(x_1=g, x_2=m, x_3=b, x_4=l)$, where g, m, b, l are 0 or 1. From this sample data set, we can compute 8 joint probabilities. For example, we know $p(x_1=1, x_2=1, x_3=1, x_4=1)=54/100=0.54$.

A Bayesian network for this sample data set is as Figure 1. It approximate this sample data set. For example, we can obtain $p(x_1=1, x_2=1, x_3=1, x_4=1)$ by the chain rule:

$p(x_1=1, x_2=1, x_3=1, x_4=1)$
$= p(x_1=1| x_3=1)\, p(x_2=1| x_3=1, x_4=1)\, p(x_3=1)\, p(x_4=1)$
$=0.95\times0.9\times0.95\times0.7$
$=0.569$

The join probabilities obtained from the Bayesian network is approximately equal to the ones from the sample data set. If the Bayesian network matches the sample data set well, the error may be negligible, for $p(x_1=1, x_2=1, x_3=1, x_4=1)$, the error is:

0.54-0.569=0.029.

We are only interested in good Bayesian networks with regarded to a scoring metric.

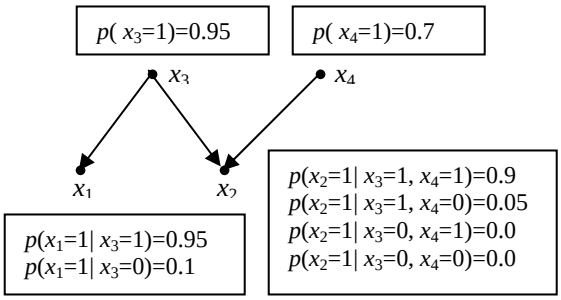

Figure 1. A Bayesian network with the associated CPTs

A problem of learning Bayesian network is to find a Bayesian network ***B*** that minimize a given scoring metric ***S*** in a domain with a set of variables ***V*** and a given training set ***T***. we use a triple <***V***, ***T***, ***S***> to denote a learning problem. Suppose the set of variables ***V*** contains *n* random variables: $x_1, \ldots, x_n$. A training set ***T*** is a table in which each row is a sample and number of the sample. And a sample is an *n*-dimensional vector of values of the *n* variables in the problem domain. A scoring metric ***S*** is a function which maps a set of Bayesian networks to a set of real number, the function value evaluate how well a Bayesian network ***B*** can match the dependency of the variables and whether it's structure is concise for a given sample data set. Several scoring metrics are available to measure competing networks in search space. The RP(Relative Posterior Probability)[3], the BIC(Bayesian Information Criterion)[3] and the MDL(Minimum Description Length)[5] are the most common ones. The metric is based on the Hoffman Encoding Theory. If we found an appropriate Bayesian network, we could use a Hoffman code based on it to encode the training set in minimum length of bits. The MDL has two terms. The first term is the length of bits used to encode the joint probabilities in a Bayesian network ***B***. It is denoted as $\boldsymbol{DL}_{data}$. According to this definition, we know

$$DL_{data} = -\sum_{i=1}^{m} \log\ p(\vec{x}_i)$$

The *m* is the number of sample vectors in the training set. The $\vec{x}_i$ is the *i*-th row of the training set ***T*** and $\vec{x}_i$ is a *n*-dimensional vector of values of the *n* variables, and $p(\vec{x}_i)$ is the join probability of $\vec{x}_i$ in a particular Bayesian network, we also know

$$p(\vec{x}) = \prod_{j=1}^{n} P(x_j \mid \nabla x_j)$$

where ∇x_j is a combination of values of the variables in the parent node set of the node x_j.

The other term of the MDL is the length of bits used to encode the CPTs in a Bayesian network, and denoted as $\boldsymbol{DL}_{srru}$. For a node x_j, the number of the associated condition probabilities is $\frac{1}{2}\|\nabla x_j\|$, where $\|\nabla x_j\|$ is the number of all combinations of values of the variables in the parent node set of the node x_j, and $\log_2 m$ bits is needed for encoding a condition probabilities .So we have

$$DL_{stru}(x_j, \nabla x_j) = \frac{1}{2}\|\nabla x_j\| \log_2 m$$

If the node x_j has no parent nodes, then set $\|\nabla x_j\| = 1$. So for a Bayesian network, the MDL is

$$-\sum_{i=1}^{m} \log p(\vec{x}_i) + \sum_{j=1}^{n} \frac{1}{2}\|\nabla x_j\| \log_2 m$$

From information theory, the best encoding of a training set ***T*** distributed according to the joint probabilities given by a Bayesian network B requires the minimum description length of bits. Obviously, a problem of learning Bayesian network is a global optimization problem of the scoring metric function in the search space of possible Bayesian networks. The number of the possible structures is an exponential function of the number of variables in the data set; and it is so large that we could not even contemplate any kind of exhaustive search to find one that minimizes the given scoring metric.

3. The Genetic Learning Algorithm for Bayesian Networks

The inspiration of genetic algorithm [4] comes from the evolution of biology. We propose a genetic algorithm for learning Bayesian networks (GLA for short). The GLA gets a training set and a scoring metric function as inputs, and prints an optimal Bayesian network and the associated CPTs as outputs. The algorithm outline is as Figure 2.Here we present some implementation details of the GLA.

The data structure of a Bayesian network is represented as an adjacency matrix ***A***, such that ***A***[i][j]=1 if <V_j, V_i> is an arc in the Bayesian network and otherwise A[i][j]=0. In order to find the parent nodes of a node efficiently, the direction of an arc here is reversed.

1. Randomly generate the initial generation of k Bayesian networks $B_1, \ldots, B_k$.
2. Select h individuals from $\{B_1, \ldots, B_k\}$ to yield a survivor set $\{C_1,\ldots,C_h\}$, where $h < k$.
3. Crossover with mutation on $\{C_1,\ldots,C_h\}$ to yield the next generation of k individuals $B_1,\ldots,B_k$.
4. If all structure in $\{B_1,\ldots,B_k\}$ are not the same then go to step 2.
5. Print the structure B_1 with the associated CPTs.

Figure 2. The description of GLA

A Bayesian network must be a DAG, so each new generated individual must been checked that it does not contain a loop. The main process of checking repeats the following two steps until all nodes are output or all node left have parent nodes:

- Select a node that has no parent node and output it.
- Delete the node and all arcs that end at this node from the graph.

If the process ends at all node left have parent nodes, then the graph has a loop.

How to generate the initial generation of individuals? For example, there is a training set of five variables. The graph of any Bayesian Network for it contains only 5 vertexes and less than 20 arcs. The initial generation contains k individuals. To spread equally in the state space, we generate initial generation in following way:

- Set each individual to be a graph that only contains vertexes.
- For all individuals, it is randomly set 1 to 20 arcs, and if it contains a ring, then randomly deletes one arc until it is acyclic.

The selection operation of a genetic algorithm simulates the function of environment that the adaptive individuals survive better. We use the selection operation to create a survivor set of individuals from current generation. Two forms of selection operation are available. The first one proceeds as following:

- All individuals are sorted in ascending order according their values of the scoring function.
- The sorted individuals are equally divided into 4 groups. One of the four individuals in the first group is randomly deleted. One of the two individuals in the second group is randomly deleted. Three of the four individuals in the third group are randomly deleted. All individuals in the last group are deleted.
- The survivors of each group are made together to a survivor set.

The crossover operation is carried out after the selection operation to simulate the reproduction of biology. Two individuals are randomly picked up from the survivor set to crossover and reproduce a child to replace a loser of current generation, until all losers are replaced. The total number of individuals keeps unchanged. The rule of crossover is as follow:

- Find out all common arcs of the parents as child's arcs and count the number (denote as K) of the uncommon arcs of them.
- Randomly select L arcs from the uncommon arcs to be added to child (L is a random number between 0 and K), and regenerate it if it contains a cycle until this repetition reach 100 times.

If the child only contains the common edges of its parents, then it contains no cycle because its parents have no cycle. The advantages of this crossover operation are as follow:

- Keep the number of a child's edges close to that of its parents.
- The child inherits the features of its parents.
- Ensure to produce a valid child individual for each time.
- Generate a child the same as its parents if its parents are the same as each other.

When all individuals are identical, no new individual will be generated and the search stops with convergence.

Theoretically speaking, the program will stop when all individuals are identical, but experiment shows that just most of individuals are identical after 300 iterations of evolution and there exit a few individuals keep different from the majority after 1000 iterations. The algorithm stops if most of individuals are identical.

4. Implementation and Results

The sample data set is a survey data of the high school students of certain region[5]. There are five variables: X_1, X_2, X_3, X_4, X_5, and they get values respectively from the following sets: *Sex*, *IQ*, *DE*, *FE*, *PEU*. We know: *Sex*={male, female}, *IQ*={low, lower-middle, upper-middle, high}, *DE*={low, lower-middle, upper-middle, high}, *FE*={low, high}, *PEU*={yes, no}. The Table 2 shows the statistic result. The first element, e.g. 4, of the table stands for the number of the samples <'male', 'low', 'low', 'low', 'yes'>, and the second, e.g. 349, stands for the number of the samples <'male', 'low', 'low', 'low', 'no'>, and so on.

We have run program 20 times and seven times reach lower values of the scoring function. The corresponding structures are as Table 3 and Figure 3 shows. Most of the resulting graphs have 7 arcs (only one of them is 6 arcs), and their structures a very

similar. No.2 and No.7 have the same score and the positions of their arcs are identical, but four arcs of them have different directions. It is obvious that the MDL scoring function is not sensitive to the causal relations and only reflect whether there exit a relation among the variables. To overcome this problem, it is believed that trying a different scoring function is needed. In this example, there are 7 of the 20 trials that converge.

4	34	13	64	9	20	33	72
1	12	38	54	1	67	49	43
2	23	27	84	7	20	64	95
1	11	93	92	1	79	11	59
8	16	47	91	6	12	74	110
1	92	14	10	6	42	19	73
4	48	49	57	5	47	12	90
9	41	22	65	8	17	41	54
5	45	39	44	5	31	14	47
8	21	20	35	1	96	28	24
1	28	29	61	1	23	47	88
1	16	62	85	1	11	72	50
7	16	36	72	1	19	75	90
1	17	91	10	2	81	14	77
6	50	36	58	5	7	11	76
1	48	12	81	1	49	36	98

Table 2. The Sample Data Set

No.	Structures Description	Scoring
1	3->1,4->1,4->2,5->2,5->3,3->4,5->4	66118
2	4->1,5->1,4->2,5->2,3->4,5->4,3->5	66071.8
3	1->2,4->2,4->3,5->3,1->4,2->5,4->5	66106.1
4	3->1,5->1,4->3,5->3,2->4,5->4,2->5	66177
5	4->1,4->2,5->2,5->3,3->4,5->4	66049.2
6	5->1,5->2,4->3,5->3,1->4,2->4,5->4	66129.8
7	4->2,5->2,4->3,5->3,1->4,5->4,1->5	66071.8

Table 3. Some run-time results

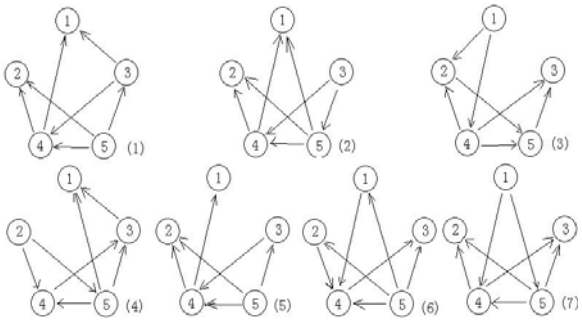

Figure 3. The corresponding graphs of the optimals

5. Conclusion and future work

When the GLA ends, e.g. all or most of the structures are the same, it converges to an optimal structure. The GLA is suitable for various scoring functions, can converge to the optimal structure with arbitrary initial set of structures. The GLA is easier and more efficient to converge because the reproduction of new filial generation is more purposeful rather than a blind search. The computer experiment results show that learning Bayesian networks structure by the genetic type of algorithms is feasible.

To raise the convergence of the GLA, it is suggested that the rate of losers be raised, and we believe that the more amount of the losers is selected, the more convergence of the GLA will get. Other improvements also may be adopted. A mutation mechanism can help search the state space more sufficiently, but may risk losing the convergence. Using domain and experience knowledge to generate child individuals is also attractive. If a new scoring function which can combine experience knowledge with posterior probability, the advantage of the GLA will exert better.

6. References

[1] Nilsson, N.J., "Artificial Intelligence, A New Synthesis," Morgan Kaufmann Publishers, Inc, 1998.

[2] Heckerman, D. and Chickering, D. M., "Learning Bayesian Networks: The Combination of Knowledge and Statistical Data," Machine Learning, 20(1995): 197-243.

[3] LIU Qi-yuan, ZHANG Cong, and SHEN Yi-dong, and WANG Cheng-liang, "An On-Line Structure Learning Algorithm of Belief Network," Journal of software, 13(12) (2002): 2297-2304.

[4] Goldberg, D., "Genetic Algorithms in Search, Optimization, and Machine Learning," Reading, MA: Addison-Wesley, 1989.

[5] Friedman, N., and Goldszmidt, M., "Learning Bayesian Network with local structure," In: Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence, San Francisco: Morgan Kaufmann, 1996: 252-262.

[6] Allen, T. V., Greiner, R., "Comparing model selection criteria for belief networks," In: Proceeding of the 17th International Conference on Machine Learning. San Francisco: Morgan Kaufmann, 1996: 1047-1054.