ELSEVIER

Pattern Recognition Letters 18 (1997) 1269–1273

Pattern Recognition Letters

Analysis of the behaviour of genetic algorithms when learning Bayesian network structure from data

R. Etxeberria [a,*], P. Larrañaga [b], J.M. Picaza [a]

[a] Department of Computer Languages and Systems, University of the Basque Country, E-20009, Donostia-San Sebastian, Spain
[b] Department of Computer Science and Artificial Intelligence, University of the Basque Country, E-20009, Donostia-San Sebastian, Spain

Abstract

In the last few years Bayesian networks have become a popular way of modelling probabilistic relationships among a set of variables for a given domain. For large domains, though, the construction of Bayesian networks is a hard task and the number of possible structures and the number of parameters for those structures can be huge. Trying to solve this, some researchers have studied how this construction can be automated. This work analyzes the behaviour of genetic algorithms when performing such automation. It is shown that the different ways in which genetic algorithms can tackle the problem influence the results. © 1997 Elsevier Science B.V.

Keywords: Bayesian network; Structure learning; Genetic algorithm

1. Introduction

In the last few years Bayesian networks have become a popular way of modelling probabilistic relationships among a set of variables for a given domain. Although sometimes experts can create good Bayesian networks from their own experience, it can be a very hard task for large domains. Therefore, many methods have been investigated to automate the creation of Bayesian networks using cases collected from past experience. Bayesian networks are defined by a directed acyclic graph (DAG) where each node represents a variable of the domain and the arcs the conditional (in)dependencies between the variables. Then, prior probabilities are given for all nodes with no predecessors and conditional probabilities for all other nodes, given all possible combinations of their direct predecessors. Therefore, the automated creation of a Bayesian network can be separated into two tasks, structure learning, which consists of creating the structure of the Bayesian network from the collected data, and parameter learning, which consists of calculating the numerical parameters for a given structure.

This work will focus on the structure learning problem. Although other methods have been described for such learning, the method considered in this paper relies on searching the structure which best fits the collected data. This method can be seen as an optimization problem: having a formula (quality measure) which gives the fitness of the structures, the structure that maximizes that formula must be found using a search procedure. The number of possible structures is huge though (Robinson, 1977), and it has been proved that the search is NP-hard

[*] Corresponding author.

(Chickering et al., 1995). Therefore, heuristic search procedures (Reeves, 1993) have been tried.

This work analyzes a heuristic search procedure, the genetic algorithm, and it discusses its behaviour depending on some of its parameters. The influence of the manipulation of the structures and the effect of some quality measures based on the K2 metric (Cooper and Herskovits, 1992) as well as the influence of the size of the problem have been analyzed.

2. Genetic algorithms

Genetic algorithms are heuristic search procedures based on the mechanics of natural selection. The search space of the problem is represented as a collection of individuals (population) which are encoded by character strings. An objective function measures the quality of the individuals and the genetic algorithm tries to find the individual which optimizes that objective function.

In our problem the individuals are the Bayesian network structures, and the objective function is the quality measure. The Bayesian network structure is represented by the string $c_{11}c_{21}\ldots c_{n1}c_{12}c_{22}\ldots c_{n2}\ldots c_{1n}c_{2n}\ldots c_{nn}$ where n is the number of variables of the structure, and $c_{ij}=1$ if the jth variable is a parent node of the ith variable and $c_{ij}=0$ otherwise.

Starting with an initial population, the genetic algorithm modifies it in the following manner. First, two individuals are selected (the parents). Then, based on those two individuals more individuals are created (the parents have children). These two steps can be performed more than once. Finally a new population is created combining the previous one with the newly created individuals (a generation has passed). This process is repeated until a stop criterion is fulfilled. The algorithm returns the best individual according to the objective function. Therefore, beside the encoding of the individuals and the objective function, some other things must be defined in order to use the genetic algorithm: how the initial population is created, how the parents are selected, how the children are created from the parents, how the new population is created, and when the algorithm is stopped.

When creating the initial population, its size and the way in which the individuals are created must be defined. In this work individuals were created randomly and two population sizes were considered: n and $2n$, where n is the number of nodes of the Bayesian network being searched. The parents were selected with a probability proportional to the rank of their quality, i.e. if $q(\cdot)$ is the objective function, λ is the size of the population, and I_j is the jth individual in the population, then the probability p_j that individual I_j was selected to be a parent was equal to $p_j=\text{rank}(q(I_j))/[\lambda(\lambda+1)/2]$.

Usually two operators are involved in the creation of the children: the crossover operator and the mutation operator. Once the parents have been selected they are subjected to the crossover operation with a given probability, and then, the individuals that are created from that operation are mutated with another given probability. Two crossover operators were considered: the uniform crossover operator and fuse-dags (Matzkevich and Abramson, 1992). The mutation operator consisted of a random addition or deletion of an arc. If the modifications created cycles in the structures, arcs were randomly deleted from those cycles until the structures became DAGs. The crossover probability was 0.9 and the mutation rate 0.01. When no operator was applied the children were exact copies of the parents.

Only one child was generated in each generation and it was introduced in the new population if its quality was better than that of the worst individual in the previous population; otherwise the child was rejected. The algorithms were stopped after 100 000 generations, or when in 1000 successive generations the average quality of the population did not change.

3. Quality measures

The quality measures are expressions that quantify the fitness of the structures. Structures which better fit the data will have better values. However, a usual problem with the quality measures is that, trying to catch all the (in)dependencies among the variables, they favour very complex structures that, although performing well on the data from which they were learned, perform very poorly on other data as those structures are different from the true ones. A common solution to this problem is to reduce the complexity of the learned structures. This complexity reduction can be as simple as limiting the number of parent nodes of the variables (Larrañaga et al., 1996), or more sophisticated as adding a term to the quality measures that penalizes complex structures (Etxeberria et al., 1997).

In this work the influence of using different complexity reduction criteria was analyzed. On the one hand, structures were limited to 7 or fewer parent nodes, using the K2 quality measure (Cooper and Herskovits, 1992). On the other hand, two modifications to K2 involving penalization terms (Etxeberria et al., 1997) were used.

4. Evaluation method

First, some Bayesian networks were created. Then, data sets were generated from those Bayesian networks. Finally, Bayesian network structures were learned from the data sets and compared to the true ones.

Beside the influence of the parameters mentioned in Sections 2 and 3, the effect of the size of the problem was analyzed as well, i.e. the effect of the number of variables of the structures, the effect of the complexity of the structures, and the effect of the number of cases in the given data sets were observed.

When creating the ''true'' Bayesian networks five sizes were considered: 10, 20, 30, 40 and 50 nodes. For each size, three complexity categories were defined (Kjærulff, 1993): dense, middle and sparse. For each size and each complexity category a Bayesian network was randomly created, resulting in 13 Bayesian networks (structures with 10 or 20 variables can never be sparse). For each Bayesian network, four data sets having 100, 500, 2000 and 5000 cases were simulated using PLS (Henrion, 1988), giving a total of $13 \times 4 = 52$ data sets. Taking those data sets as input, 10 runs were done with each of the $2 \times 2 \times 3 = 12$ search procedures (population size $\times$ crossover operator $\times$ quality measure) resulting in $52 \times 12 \times 10 = 6\,240$ ''learned'' Bayesian networks.

In order to analyze the behaviour of the genetic algorithms, three parameters were measured in each run: the similarity between the true and the learned structure, the quality of the learned Bayesian network in relation to the data, and the convergence speed. The similarity between the true and the learned Bayesian network was measured counting the extra and missing arcs between the structures. The quality of the learned structure in relation to the data was calculated using the Kullback–Leibler distance (Kullback and Leibler, 1951) between the probability distribution defined by the frequencies of the cases in the data sets and the probability distribution defined by the learned Bayesian networks (parameters were assigned as shown in (Cooper and Herskovits, 1992)). The convergence speed was measured counting the number of generations until the algorithm stopped.

5. Results

The results were analyzed using analysis of variance (SPSS Inc., 1988). The Kruskal–Wallis tests indicated that, although the learned structures had more extra arcs when the population size was n ($p < 0.0001$), there was no statistically significant difference in the quality of the structure which indicates a clear overfitting. As expected, many more generations were needed with $2n$ individuals ($p < 0.0001$).

Statistically significant differences were found in the number of extra and missing arcs when using the two different crossover operators ($p < 0.0001$). Structures learned using fuse-dags were much more complex than those learned using the uniform crossover operator. However, again, there was no significant difference in the quality of the structures. Algorithms using fuse-dags as the crossover operator converged slightly faster than those using the uniform crossover operator ($p < 0.0006$).

The learned structures had more extra arcs when no penalization term was added to K2 ($p < 0.0001$). However, the learned structures had more missing arcs when the penalization term was used ($p < 0.0001$). Therefore, although it cannot be said that a penalization term learned structures which were closer to the true ones, it is clear that K2 with a penalization term learned simpler structures. Again, there was no significant difference in the quality of the structures. The algorithm converged faster when a penalization term was added to K2 ($p < 0.0001$).

The number of cases had a strong influence on the results. As expected, the learned networks were more similar to the true ones ($p < 0.0001$) and were better when more cases were used ($p < 0.0001$). However, more cases implied more generations ($p < 0.0001$).

The effect of the number of nodes was notorious as well. With more nodes worse results were obtained ($p < 0.0001$ in all cases). This is very normal because, on the one hand, many more cases would be needed when more variables are considered, and on the other hand, the search space would be much bigger.

The effect of the complexity of the original network was quite strange. With sparse structures the genetic algorithms added more extra arcs ($p < 0.0001$), but with dense structures there were more arcs missing ($p < 0.0001$): the genetic algorithms tried to ‘‘balance’’ the complexity of the structures. On the other hand, structures learned using data sets originating from dense Bayesian networks had better quality than others ($p < 0.0001$). Therefore, the quality of the true networks were compared to the quality of the learned ones, and a high correlation ($r = 0.99$) was found. Although the implications of this result are not clear, it seems that its causes must be found in the PLS simulation algorithm.

Discussion

Holz: There are two reasons cited generally for using genetic algorithms that have to do with the global nature of the search. One is that you combine information from multiple hypotheses. The second is that you become less sensitive to the initial starting point. By choosing to replace only one member of the population in each generation, I would be greatly concerned that you have increased the sensitivity to your initial starting point. That is very unusual, at least in practice, in genetic algorithms. Why did you not choose to vary that?

Etxeberria: You mean that we should create more than one individual in each generation?

Holz: The maximum number of trials you can have in your 100 000 generations is 100 000 plus N, where N is the number of starting points. So results that have to do with sensitivity to the complexity of the problem are maybe questionable because you have hard limited the number of trials that you can make.

Etxeberria: But I think it is a quite flexible limit.

Holz: Why is it flexible, if you try one new population member for a generation and you stop at 100 000 generations?

Szirányi: There are instances where you use fifty free variables in genetic learning. This seems a very high number.

Etxeberria: I do not understand what you want to say.

Loew: Your string, your genome, how many parameters do you fit?

Exteberria: Do you think fifty is high, well in a real problem fifty is quite common.

Acknowledgements

This work was supported by Eusko Jaurlaritza under grants PI95/52 and PI96/12, by the University of the Basque Country under grant UPV 140226-EA186/96 and by Gipuzkoako Foru Aldundiko Ekonomia eta Turismo Departamentua under grant OF84/1996.

References

Chickering, D., Geiger, D., Heckerman, D., 1995. In: Proc. 5th Conf. on Artificial Intelligence and Statistics, pp. 112–128.
Cooper, G.F., Herskovits, E.A., 1992. Machine Learning 9, 309–347.
R. Etxeberria, P. Larrañaga, J. Manuel Picaza, 1997. In: Proc. of Causal Models and Statistical Learning, pp. 151–168.
Henrion, M., 1988. In: Proc. 4th Conf. on Uncertainty in Artificial Intelligence, pp. 149–163.

Kjærulff, U., 1993. Aspects of efficiency improvement in Bayesian networks. Ph.D. Thesis. Institute of Electronic Systems, Aalborg University.

Kullback, S., Leibler, R., 1951. Annals of Mathematics and Statistics 22, 79–86.

Larrañaga, P., Poza, M., Yurramendi, Y., Murga, R.H., Kuijpers, C.M.H., 1996. Structure learning of Bayesian networks by genetic algorithms: A performance analysis of control parameters. IEEE Trans. Pattern Anal. Machine Intell. 18, 912–926.

Matzkevich, I., Abramson, B., 1992. In: Proc. 8th Conf. on Uncertainty in Artificial Intelligence, pp. 191–198.

Reeves, C.R., 1993. Modern Heuristic Techniques for Combinatorial Problems. Blackwell Scientific Publications, Oxford.

Robinson, R.W., 1977. Counting unlabeled acyclic digraphs. In: Little, C.H.C. (Ed.), Combinatorial Mathematics V, Lecture Notes in Mathematics, vol. 622. Springer, Berlin, pp. 28–43.

SPSS Inc., 1988. SPSS-X User’s Guide third edition.