http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

Information Technology Journal 5 (3): 540-545, 2006
ISSN 1812-5638
© 2006 Asian Network for Scientific Information

Research on Learning Bayesian Networks by Particle Swarm Optimization

Xing-Chen Heng, Zheng Qin, Xian-Hui Wang and Li-Ping Shao
Research Institute of Computer Software of Xi'an Jiaotong University, 710049,
Xi'an, Shanxi, People's Republic of China

Abstract: A new approach to learning Bayesian networks (Bns) was proposed in this study. This approach was based on Particle Swarm Optimization (PSO). We start by giving a fitness function to evaluate possible structure of BN. Next, the definition and encoding of the basic mathematical elements of PSO were given and the basic operations of PSO was designed which provides guarantee of convergence. Next, full samples for the training set and test set are generated from a known original Bayesian network with probabilistic logic sampling. After that, the structure of BN was learned from complete training set using improved PSO algorithm steps. Finally, the simulation experimental results also demonstrated sthis new approach's efficiency.

Key words: Bayesian network, particle swarm optimization, fitness function, structure learning, velocity

INTRODUCTION

Bayesian Networks (BNs) are quickly becoming the tool of choice of many AI researchers for problems involving reasoning under uncertainty. They have been implemented in applications in areas such as medical diagnostics, classification systems and software agents for personal assistants, multisensor fusion and legal analysis of trials (Heckerman and Geiger *et al.*, 1995). Until recently, the standard approach to constructing belief networks was a labour-intensive process of eliciting knowledge from experts. Methods for capturing available data to construct a Bayesian network or to refine an expert-provided network promise to greatly improve both the efficiency of knowledge engineering and the accuracy of the models. For this reason, learning Bayesian networks from data has become an increasingly active area of research.

Learning a Bayesian network can be decomposed into the problem of learning the graph structure and learning the parameters. An obvious choice to combat the problem of getting stuck on local maxima is to use a stochastic search method (Kennedy 2001; Eberhart and Kennedy, 1995). This study explores the use of Particle Swarm Optimization (PSO) algorithms for learning Bayesian networks. Network structures are especially amenable for PSO algorithms since the substructures of the network behave as building blocks so we can evolve higher fit structures by exchanging substructures of parents with higher fitness.

PROBLEM FORMULATION

Bayesian networks and associated schemes constitute a probabilistic framework for reasoning under uncertainty that in recent years has gained popularity in the community of artificial intelligence (Pearl, 1998; Neapolitan, 1990).

From an informal perspective, Bayesian networks are Directed Acyclic Graphs (DAGs), where the nodes are random variables and the arcs specify the independence assumptions that must be held between the random variables.

To specify the probability distribution of a BN, one must give prior probabilities for all root nodes (nodes with no predecessors) and conditional probabilities for all other nodes, given all possible combinations of their direct predecessors. These numbers in conjunction with the DAG, specify the BN completely. The joint probability of any particular instantiation of all n variables in a BN can be calculated as follows:

$$P(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid \pi_i) \qquad (1)$$

where, x_i represents the instantiation of the variable X_i and π_i represents the instantiation of the parents of X_i.

The most common approach to building Bayesian networks is to elicit knowledge from an expert. This works well for smaller networks, but when the number of variables becomes large, elicitation can become a tedious

Corresponding Author: Xing-Chen Heng, Research Institute of Computer Software of Xi' an Jiaotong University, 710049, Xi' an, Shanxi, People's Republic of China

and time-consuming affair. There may also be situations where the expert is either unwilling or unavailable. Whether or not experts are available, if there are data it makes sense to use it in building a model. The problem of learning a Bayesian network from data can be broken into two components: learning the structure, B_S and learning the parameters, B_P. If the structure is known 1 then the problem reduces to learning the parameters. If the structure is unknown, the learner must first find the structure before learning the parameters (actually in many cases they are induced simultaneously). Generally, it is difficult for experts to give the structure of BN directly and learning it from data is a feasible modeling method. In addition, the structure learning algorithm can itself be decomposed into searching for structures by using search algorithm and evaluating structures by using scoring metric (fitness function), which aims at finding the best structure with highest accuracy of generating training set. So the problem of the structure learning can be formally defined as follows:

Input: A training set D of instances of X, which contains N observation sequences. The length of the kth sequence is l_k and each case $x_k[0]$, $x_k[1]$,..., $x_k[l_k]$ is given.

Output: A BN that best matches D. The notion best matches is defined using a scoring function.

PSBN ALGORITHM

Particle Swarm Optimization (PSO) is an evolutionary computation technique, which is inspired by social behavior of bird flocking and fish schooling (Kennedy, 2001; Eberhart and Kennedy, 1995). Comparing with Genetic Algorithm (GA), PSO' s advantages lies on its easy implementation and few parameters to adjust. It has been found to be extremely effective in solving the continuous optimization problem, but now it has been expanded to discrete domain. Though its strict convergence has not been proved, it can be easily modified for any dis crete/combinatorial problem for which we have no good specialized algorithm. Therefore, as a combinatorial optimization problem, it is possible to learn the structure of BN by using PSO algorithm.

PSBN algorithm can be expressed simply by the following equation, PSBN= (F, X, V, S_{xx}, P_{vv}, M_v, P_{xv}, λ, G_{init}, υ), where, F is a fitness function, X a space of positions of particles, V velocity set of particles, S_{xx} a substraction operation (position, position), P_{vv} a move operation (position plus velocity), M_v a multiplication operation (coefficient times velocity), P_{xv} a addition operation (velocity plus velocity), $\lambda$the swarm size, G_{init} an initial swarm and υ stopping condition.

Fitness function F: The most common fitness function F to evaluating structures is by the posterior probability of the structure given the observations. That is, a structure is good to the extent it is probable given the available information. The posterior probability of a structure can be obtained by applying Bayesian rule:

$$P(B_S \mid D) = \frac{P(D \mid B_S)P(B_S)}{P(D)} \quad (2)$$

where, P (B_S |D) is the posterior distribution of the structure given the data, P(D|B_S) is the likelihood function, P(B_S) is the prior probability of the structure and P(D) is the normalizing constant. Since P (D) is not dependent on the structure, it can be ignored when trying to find the best scoring function. In addition, without prior knowledge of structures, we can as sume they have equal probability. However, if we do have information on structures we can always use the prior information.

The problem is now reduced to finding the structure with the maximum likelihood P (D|B_S). In other words, given a structure, these structures are evalu ated according to how probable it is that the data were generated from the structure.

Cooper and Herskovitz showed that when a Dirichlet prior is used for the parameters in the network, the likelihood P (D|B_S) can be obtained in closed form:

$$F = P(D|B_S) = {}_{i=1}^{n}{}_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N'_{ij} + N_{ij})} {}_{k=1}^{r_i} \frac{\Gamma(N'_{ijk} + N_{ijk})}{\Gamma(N'_{ijk})} \quad (3)$$

where, n is the number of variables in the database, ri is the number of possible states for variable Xi , q_i is the number of possible states for pa (Xi), $\mathbf{N_{ijk}}$ are the sufficient statistics from the database (counts of occurrences of configurations of variables
and their parents), $\mathbf{N_{ijk}}$ are the hyper parameters (prior counts of occurrences of variables and their parents) specified for the parameter prior (assuming an u n informative prior as in the prior for the structure we set the hyper parameters to 1),

and
$$N_{ij} = {}_{K=1}^{r_i} N_{ijk}$$
$$N'_{ij} = {}_{k=1}^{r_i} N'_{ijk}$$

For computational convenience, the number of parents allowed for a particular variable is limited. This scoring metric (3) is commonly referred to as the Bayesian Dirichlet metric. In practice, the logarithm of (3) is usually used to score networks.

When data are complete, the Bayesian Dirichlet metric, a fitness function for BN, exists in closed form. So we may utilize the score decomposition properties, which facilitate the computation of the scoring metric (3) in several ways. Note that the likelihood is expressed as a sum of terms, where each term depends only on the conditional probability of a variable given a particular assignment to its parents. Thus, if we want to find the maximum likelihood parameters, we can maximize within each family independently.

With the Bayesian Dirichlet metric (3), we can now search over possible structures for the one that scores best networks from complete datasets that mean that all of the cases in the data contain values for all of the variables.

Encoding PSO elements for BN: The BN structure can be represented as an adjacency list, (Fig. 1) where each row represents a variable Xi and the members of each row, with the exception of the first member, are the parents of Xi, pa (Xi). The first member of each row, i.e, the first column of the adjacency list, is the variable Xi.

Position of particles and state space: As PSBN algorithm is designed to find the best structure of BN by using PSO, the structure of BN should be encoded into a position of particle.

Although we show the variable Xi in the Fig. 1 for clarity, the internal representation encodes its parents only, with the variable being encoded by sequence. The adjacency list can be thought of as a position where each pa (Xi) represents a local position. For example, the local position of the variable F can be encoded as [D,E]. Because the logarithm of the scoring metric is the summation of scores for each variable, each local position can be scored separately and added to generate the fitness score for the entire structure.

So the search space is enormous. A local position can range from no parents to n-1 parents, where n is the number of variables in the dataset. Thus a local position can take on ${}^{k}_{i=1}\binom{n-1}{i}$ possible values where k is the maximum set of parents a variable can have and n is the number of variables in the dataset. So, the search space can be defined as follow: ${}^{n}_{j=i}\,{}^{k}_{i=j}\binom{n-1}{i}z$

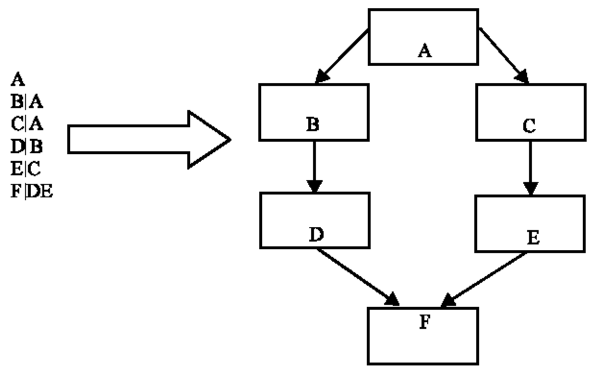


Fig. 1: Encoding the structure of a BN

VELOCITY

Definition 1 switch operator: If a specified BN has n variables, the position of a particle can be expressed as an adjacency list P = $((x_i))$, i =1,…,n. We define a switch operator *SO* which, when applied to a position during one time step, gives an other position. So, here, SO has three types: $+x_i$, $-x_i$ and ϕ. The $+x_i$ denotes adding a variable x_i into original position, $-x_i$ reducing a variable x_i and ϕ null.

Definition 2 switch unit: A sequence composed of one or several switch operators is a switch unit. We denote it by SU.

$$SU = (SO_1, SO_2,., So_k) \quad (4)$$

Where, SO_1, SO_2,., SO_k are k switch operators and the different orderings of them have different signification. The length of the SU is defined by $\|SU\| = k$. Each SU is applied to the change of a local position.

Definition 3 switch list (velocity): A sequence composed of one or several switch units is a switch list. We denote it by SL. Here, actually, the num ber of switch units of each switch list is equal to the number of variables n of BN. The length of the SL is defined by $\| SL \| = {}^{n}_{i=1} \| SU_i \|$
A velocity V is then de fined by:

$$V = SL = (SU_1, SU_2,., SU_n) \quad (5)$$

Where, SU_1, SU_2, ., Su_n are n switch units and each SL or V is applied to the change of a global position.

Definition 4 equivalent set of switch list: If different switch lists are equivalent (same result when applied to any position), the set of them is called equivalent set of switch list.

DESIGNING OPERATIONS FOR PSO

Opposite of a velocity: It means to do the same switch as in original SL, but with reverse operator. For example, - ((-A), (+B-C)) = ((A), (-B+C)). It is easy to verify that we have - (- SL) = SL (and $SL \oplus -SL \cong \phi$, Addition velocity plus velocity).

Addition (P_{xv}) position plus velocity: Let P be a position and V a velocity. The position P'=P±V is found by applying the first switch of V to P, then the second one to the result etc.
Example

$$P = (\phi, A, A, B, C, DE) \qquad (6)$$
$$V = ((+B), (-A), (-A+B), (-B+C), (\phi), (-D-E+A))$$

Applying V to P, we obtain successively

$$P' = (B, \phi, B, C, C, A) \qquad (7)$$

Substraction (S_{xx}) position minus position: Let P_1 and P_2 be two positions. The difference $P_2 - P_1$ is defined as the velocity V, found by a given algorithm, so that applying V to P_1 gives P_2. The condition "found by a given algorithm" is necessary, for, as we have seen, two velocities can be equivalent, even when they have the same size. In particular, the algorithm is chosen so that we have $P_1 = P2 \Rightarrow V = P_2 - P_1 = \phi$

Addition (P_{vv}) velocity plus velocity: Let V_1 and V_2 be two velocities. In order to compute $V_1 \oplus V_2$ we consider the switch list which contains the first switch unit of V_1, followed by the first switch unit of V2, then the second switch unit of V_1, followed by the second switch unit of V2 etc. For example, ((-A), (-B+C)) ⊕ ((-B+A), (-B+D)) = ((-A-B+A), (-B+C-B+D)). In general, we "contract" it to obtain a smaller equivalent velocity. For example, ((-A-B+A), (-B+C-B+D)) = ((-B), (+C+D)). In particular, this operation is defined so that $V \oplus - V. = \phi$. So, we can have the following definition:

Definition 5 basic switch list: The switch list of equivalent set of switch list, which contains the least switch operators, is defined as basic switch list. Each velocity is a basic switch list.

Multiplication (M_v) coefficient times velocity: Let α be a real coefficient and V be a velocity. There are different cases, depending on the value of α.
Case $\alpha = 0$
We have $\alpha \; V = \phi$
Case $\alpha \in [0,1]$
We just "shrink" V. Let $\|\alpha V\|$ be the greatest integer smaller than or equal to$\alpha \|V\|$. So we define $\alpha V = ((SO_1, .., SO_k)_1, \ldots, (SO_1, .., SO_k)_n), k \; {}_1^{(\|\alpha V\|/n)}$
Case $\alpha > 1$
It means we have $\alpha = d + \alpha'$, d is an integer ($d \neq 0$), $\alpha' \in [0,1]$. So we define αV

$$= \sum_{i=1}^{d} (\;V) \;\; \alpha' V$$

Case $\alpha < 0$
As $\alpha V = (-\alpha) * (-V.)$., we only need to consider one of the previous cases.

Control Parameters: The initial swarm G_{init}, as well as the velocities, can be generated either randomly or by a Sobol sequence generator (Henrion, 1988; Press et al., 1992) which ensures that the D-dimensional vectors will be uniformly distributed within the search space.

The swarm size λ should be not kept too big because of the computation time required scoring the fitness function; On the other hand, λ should be not kept too small for improving the diversity of particles of swarm to avoid premature convergence. Hence, we choose λ within [30,100].

The stopping criterionυ for the algorithm is set in term of that when either g_1 generations have been run or when in g_2 successive generations, the value of the fitness function of the best structure corresponds with the average value of the fitness function.

PSBN Algorithm for BN: We can now rewrite the formula from the basic PSO algorithm:

$$V_{id}^{k+1} = w * V_{id}^{k} \quad c_1 * rand() * (P_{id} - X_{id}^{k})$$
$$c_1 * Rand() * (P_{gd} - K_{id}^{k}) \qquad (8)$$

$$X_{id}^{k+1} = X_{id}^{k} + V_{id}^{k+1} \qquad (9)$$

Where, i = 1,2,..., N; N is the swarm's size; d represents the d-dimensional search space; w is the inertia weight factor; c_1 and c_2 are two positive constants, called the cognitive and social parameter respectively; rand () and Rand () are two random numbers uniformly distributed within the range [0,1]; V_{id}^{k} is the velocity of particle i at iteration k; X_{id}^{k} is the current position of particle i at iteration k; P_{id} is the best previous position of particle i at iteration k; P_{gd} is the best neighbour's best previous position at iteration k.

The PSBN algorithm can be described as follows:
Step 1: Initialize the particle swarm (each particle is given a stochastic initial solution/position and switch list/velocity).
Step 2: If stopping criterion is satisfied, turn to Step 5.
Step 3: Calculate the next position X'_{id} (the new solution) according to the current position X_{id} of the particle I.

- Calculate the difference α by $\alpha = P_{id} - X_{id}$ where α is a basic switch list and is applied to X_{id} to obtain P_{id}.
- Calculate the difference β by $\beta = P_{gd} - X_{id}$ where β is also a basic switch list.
- Calculate the velocity V'_{id} in term of the Eq. 8 and transform V'_{id} into a basic switch list.
- Calculate the new solution X'_{id} in term of the Eq. 9.
- If a better solution is found, update P_{id}.

Step 4: If a better solution is found for the whole s warm, update $P_{id}$and turn to Step 2.
Step 5: Show the optimal solution.

EXPERIMENT

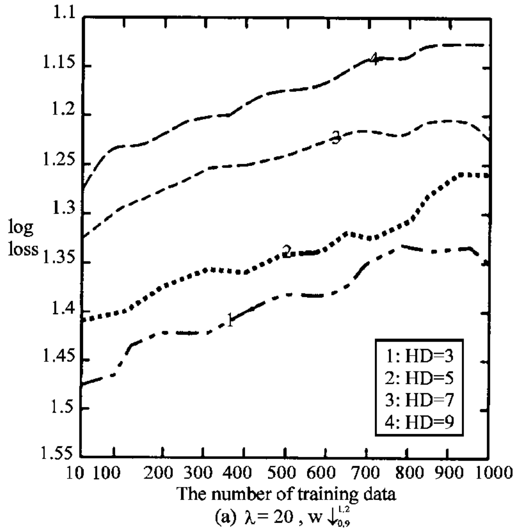
For evaluating the behavior of PSBN algorithm pro posed, we perform the different experimental steps as follows:

Step 1: Begin with a BN (structure + conditional probabilities) and simulate it, generating randomly 1000 samples for the training set D and another 1000 for the test set.
Step 2: Using the approach based on PSBN algorithm try to obtain BN structure B_s^* from D, which maximize the probability P (D|B_S).
Step 3: Evaluate the performance of PSBN algorithm by evaluating the accuracy of B_s^* predicting objective probability distribution.

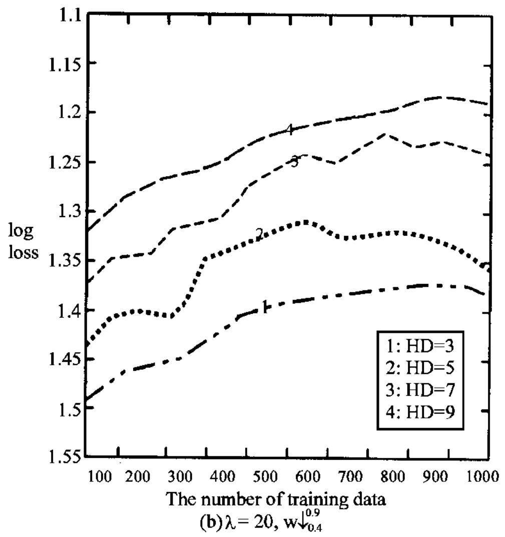
For this experiment we use a Bayesian network known as ASIA. The ASIA network was initially presented by Lauritzen and Spiegelhalter (Lauritzen and Spiegelhalter, 1988). It is a small (nine variables) fictitious model of medical knowledge concerning the relationships between visits to Asia, tuberculosis, smoking, lung cancer, lung cancer or tuberculosis, Positive X-ray, Dyspnoea and bronchitis.

We use probabilistic logic sampling (Henrion, 1988), with which we generate 1000 samples each from the original network for training and test. The algorithm is run with 3, 5, 7 and 9 hidden variables for 5, 10, 15 and 20% missing data, respectively. The experiments are run 20 times for each level of hidden variables and then the best structure of BN of 20 times learning is selected as the final learning result of each level of hidden variables.

For the purpose of comparison, all simulations deploy the same parameter settings for the PSBN except the swarm size λ and inertia weight w. The positive constants c_1 and c_2 are both set as 2; the $\|V_{max}\|$ is set as 9. We try to observe the influence of λ, w and hidden variables on the PSBN performance and different w and λ are chosen for simulations.



(a) $\lambda = 20$, $w \downarrow_{0.9}^{1.2}$



(b) $\lambda = 20$, $w \downarrow_{0.4}^{0.9}$

Fig. 2: Comparision of log loss for different number of hidden variables: 3, 5, 7, 9.

We decide to stop the algorithm when either 1000 generations are reached or when in 200 successive generations, the value of the fitness function of the best structure corresponds with the average value of the fitness function. Finally, We calculate the log loss for the test set using the "best" network from each run, which can be seen in Fig. 2. The log loss is a commonly used metric appropriate for probabilistic learning algorithms. It is a member of the family of proper scoring rules. Proper scoring rules have the characteristic that they are maximized when the learned probility distribution corresponds to the empirically observed probilities.

As can be seen from Fig. 2, the more the hidden variables and the number of training data are introduced, the higher predictive accuracy becomes proper scoring rules. Proper scoring rules have the characteristic that they are maximized when the learned probability distribution corresponds to the empirically observed probabilities.

CONCLUSIONS

In this study we describe a novel approach for learning Bayesian networks. This problem is extremely difficult for deterministic algorithms and is characterized by a large, multi-dimensional, multi-modal search space. Our approach is based on particle swarm optimization algorithm, which is called PSBN algorithm. It is simple and reliable and it can converge rapidly.

Using simulations of the ASIA network, we carry out a performance analysis on the PSBN algorithm proposed. The obtained experimental results also prove its efficiency and good performance. Mean
while, it is confirmed in this paper again that PSO can be applied to solve any discrete/combinatorial problem as same as other evolutionary algorithms.

The future important step forwards would be to extend the proposed structure learning approach based on PSO for trying to find out the optimal ordering of these variables in BN, which is expected to improve the convergence of learning the structure of BN further. Then, we also plan to adapt the described structure learning approach to dynamic Bns (Kjaerulff, 1992).

ACKNOWLEDGEMENTS

The research reported in this paper is funded by the National 973 Key Basic Research Development Planning Project (2004CB719401) and also supported by the National Defence 10th–Five-Year Preliminary Research fund (2002040202).

REFERENCES

Eberhart, R.C. and J. Kennedy, 1995. A new optimizer using particle swarm theory. Proceeding of the sixth International Symposium on Micro Machine and Human Science. IEEE service center, Piscataway, NJ, Nagoya, Japan, pp: 39-43.

Henrion, M., 1988. Propagating uncertainly in bayesian networks by probabilities logic sampling, Uncertainty in Artificial Intelligence, 2: 149-163.

Heckerman, D. and D. Geiger *et al.*, 1995. Learning bayesian networks: The combination of knowledge and statistical data. Machine Learning., 20: 197-243.

Kjaerulff, U., 1992. A computational scheme for reasoning in dynamic probabilistic networks. Proceeding of 8th Conference on Uncertainty in Artificial Intelligence, pp: 121-129.

Kennedy, J., 2001. Minds and cultures: Particle swarm implications. Socially Intelligent Agents: Papers from the 1997 AAAI Fall Symposium pp: 67-72. AAAI Press, Menlo Park, CA, 1997. USA, pp: 289-294.

Lauritzen, S.L. and D.J. Spiegelhalter, 1988. Local computations with probabilities on graphical structures and their application on expert systems, J. Royal Statistical Soc. B, 50: 157-224.

Neapolitan, R.E., 1990. Probabilistic Reasoning in Expert Systems. Theory and Algorithms. John Wiley and Sons.

Pearl, J., 1998. Probabilistic Reasoning in Intelligence Systems: Network of Plausible Inference. San Mateo, Calif.: Morgan Kaufmann.

Press, W.H., W.T. Vetterling, S.A. Teukolsky and B.P. Flannery, 1992. Numerical Recipes in Fortran 77, Cambridge University Press: Cambridge.