A LEARNING AUTOMATA-BASED TECHNIQUE FOR TRAINING BAYESIAN NETWORKS

NABI ALLAH REZVANI
Soft Computing Laboratory
Computer Engineering and
Information Technology Department
Amirkabir University of Technology
Tehran Iran
nrezvani@aut.ac.ir

MOHAMMAD REZA MEYBODI
Soft Computing Laboratory
Computer Engineering and
Information Technology Department
Amirkabir University of Technology
Tehran Iran
mmeybodi@aut.ac.ir

ABSTRACT
One of the most important challenges of Bayesian networks is training an optimal network based on existing training samples. We propose two Learning Automata-based methods for training parameters and structure of the network. Parameter training method is an incremental method which performs training and testing simultaneously and has lower computational cost than enumerative or search based parameter training methods. The structure training method uses a guided search scheme and avoids getting stuck in local maxima. This outputs a network that improves classification accuracy. We could also use both these methods together to train the network. Results indicated that this combinational method further improved classification accuracy while still kept computational cost rational.

KEY WORDS
Bayesian Networks, Training, Learning Automata

1 INTRODUCTION

Bayesian networks are extensively used as classifiers [1][8][15][16][17]. One of the most important challenges of these networks is training an optimal network based on existing training samples. Training problem consists of two sub-problems: training parameters and training structure of the network.

Different researches have been done to improve training methods of Bayesian networks. On parameter training, methods of parameter training have been revised to approach the real goal of classification. Maximum Conditional Likelihood estimation is an example such methods [15]. This sort of estimation involves a search process to find optimal estimations for the parameters. Research works have also been done to improve structure training methods. In these researches network evaluation function is modified and search methods have been proposed and redefined to find optimal network based on the evaluation function [16][17][18]. These methods perform a local search with algorithms like gradient descent or genetic search. These search schemes are blind and heuristic methods and are likely to get stuck in local maxima.

Using discriminative methods for both parameter and structure training involves heavy and impractical computational cost, since both methods should run a search process and therefore total cost of training will be the multiplication of these two search processes. This problem is also addressed in this paper.

In this paper we propose methods for training both parameters and structure of Bayesian networks. First, we propose an incremental method for training parameters of the network. The method has lower computational cost in comparison with rival methods. Besides it can perform classification with much less number of training samples. Afterwards a method is proposed for training structure of Bayesian networks. This method performs structure search using learning automata. This guided search scheme, finds a Bayesian network that improves classification accuracy in comparison with rival search methods like hill climbing and genetic search. In the end, two proposed training methods are applied together in order to find the most optimal network possible while the computational cost of this combinational method is still rational.

The reminder of this paper is organized as follows. In section 2 Bayesian Networks and their training is discussed. Section 3 is a brief introduction to Learning Automata. In sections 4 and 5 proposed methods for training parameters and structure are explained. Section 6 is the experiments and section 7 is the conclusion.

2 BAYESIAN NETWORKS

A Bayesian belief network, describes the joint probability distribution over a set of random variables, with defining a series of probability independence and a series of conditional independences [4]. In other words a Bayesian belief network defines a joint probability distribution over a set of random variables [2]. To describe a Bayesian network we need to provide two items: topology or structure of the graph, and parameters or the conditional probability tables of all variables. Having these two items, one can demonstrate the joint probability distribution implied among variables. Simply we can say that this joint probability distribution is equal to the product of conditional probability of each variable having its immediate parents. The equation is described as below.

$$P(y_1,\ldots,y_n) = \prod_{i=1}^{n} P(y_i \mid Parents(Y_i)) \tag{1}$$

To create a Bayesian networks, one has to learn both of these items based on existing training samples. The learning problem falls into different categories depending on whether the network structure is known or not.

2.1 BAYESIAN NETWORK CLASSIFIERS

Suppose that each training sample is s vector of attributes $(X_1, X_2, \ldots X_{v-1}, C)$. The goal of classification is predicting the right value of class variable $c = x_v$ having $(x_1, x_2, \ldots x_{v-1})$. If performance measure is classification accuracy (percentage of correct predictions on test samples), correct prediction for $(x_1, x_2, \ldots x_{v-1})$ is the class that maximizes $P(c \mid x_1, x_2, \ldots x_{v-1})$. If we have a Bayesian network over $(x_1, x_2, \ldots x_{v-1}, C)$, we could compute these probabilities by inference on it.

After Structure of Bayesian network is specified, it is important to estimate parameters in a way that the network can provide the best prediction for value of class variable in test samples.

2.2 TRAINING BAYESIAN NETWORKS

2.2.1 PARAMETER TRAINING

The simplest scheme for parameter training is Maximum Likelihood Estimation computed for training set $D = \{D_1, D_2, \dots D_M\}$, which is a sum over all nodes as below.

$$LL(B \mid D) = \sum_{d=1}^{n} \log P_B(X_d) = \sum_{d=1}^{n} \sum_{i=1}^{v} \log P_B(x_{d,i} \mid \pi_{d,i}) \qquad (2)$$

Or the conditional likelihood of parameters which is the probability of class variable having other variables. This is closer to the real goal of classification:

$$CLL(B \mid D) = \sum_{d=1}^{n} \log P_B(y_d \mid x_{d,1}, \dots, x_{d,v-1}) \qquad (3)$$

The maximum likelihood is decomposable on network nodes, while maximum conditional likelihood is not. A gradient descent search method called ELR is proposed to solve the problem [15].

In both methods, training and querying the network will be two isolated processes. Also for achieving acceptable results, there should exist relatively large amount of training samples. Learning Automata-based method which will be proposed in this paper, tries to give a solution for these defects.

2.2.2 STRUCTURE TRAINING

For structure training problem we will need to provide two items. A scoring function that evaluates any possible network structure and a search method to find the structure that has the maximum value of that scoring function. Finding the optimal network is a NP-Hard problem. Therefore local search methods have been proposed to find a relatively optimal structure.

Likelihood and conditional likelihood equations discussed earlier are used as scoring functions, one other efficient scoring function called classification rate is proposed in [16]:

$$CR = \frac{1}{|S|} \sum_{m \in S}^{|S|} \delta(B_S(x_{1:N}^{m}), c^{m}) \qquad (4)$$

The equation simply means rate of samples that are classified correctly by the network.

The most popular search method that has been used, is hill climbing. The algorithm starts with an initial structure, in each iteration finds all candidate structures by adding, removing or reversing an arc, and selects the candidate with best score. The procedure continues until score of current structure is not better than the score of previous iteration. The other search method which is used in variety of research works is genetic search. The algorithm uses an adjacency matrix to encode network structure. In order to have closed crossover and mutation operators on adjacency matrix, an ordering is defined on variables. A variable can not be parent of another variable coming earlier in order.

These local search methods are very likely to get stuck in local maxima. Besides, there are confining assumptions in their definitions. For example the genetic algorithm uses the explained variable ordering and hill climbing usually does not remove any arc during search process. In this paper we propose an LA-based method for structure searching which tries to perform a wider search which is also guided by learning scheme of LAs.

3 LEARNING AUTOMATA

Learning automaton is a machine that can perform a finite set of actions. Each selected action is evaluated by a random environment. The result of evaluation is given to automaton in the form of a positive or negative signal and the automaton is affected by this signal for choosing the next action. Final goal is that the automaton learns to choose the best action among its action set. Best action is the action that maximizes probability of getting reward from the environment [6].

3.2 GENERALIZED LEARNING AUTOMATA (GLA)

To solve associative reinforcement learning problems, we need a learning automaton with different definition. In Such problems the goal is to solve a pattern recognition problem. In each working step of LA, a vector of attribute values from corresponding distribution, or a sample, is also input to LA. This vector is also called context vector.

In [6] a model called generalized learning automaton is proposed. In GLA, LA structure is modified to allow for context vector input. A GLA is defined by the following tuple.

$$GLA \equiv <X, \alpha, \beta, U, g, T> \quad (5)$$

Where X is the set of all context vectors that can be in input to GLA, a is a finite set of outputs or LA actions, β is set of values that reinforcement signal can take and is usually taken from [0,1] interval, g is probability generating function, and U is the internal state which is a vector of real numbers. T is the learning algorithm that updates U. if set of actions of GLA is $\alpha=\{\alpha_1, ..., \alpha_r\}$, action probabilities of GLA are generated by

$$\text{Prob}[a(k) = \alpha_i \mid \mathbf{u}, \mathbf{x}] = g(\mathbf{x}, \alpha_i, \mathbf{u}) \quad (6)$$

Note that instead of probability vector, function g is used to generate action probabilities. This function satisfies two conditions.

$$g(\mathbf{x}, \alpha_i, \mathbf{u}) >= 0 \quad \forall\ \alpha_i, \mathbf{u}, \mathbf{x}\ , \sum_{i=1}^{r} g(\mathbf{x}, \alpha_i, \mathbf{u}) = 1 \quad \forall\ \mathbf{u}, \mathbf{x} \quad (7)$$

In step k, the learning algorithm T updates $\mathbf{u}(k)$ based on current values of $\mathbf{x}(k)$ and $\mathbf{u}(k)$, reinforcement signal $\beta(k)$, and the action chosen by GLA $a(k)$. Dependence of the updating on context vector $\mathbf{x}(k)$ is the main characteristic of GLA.

The motivation for defining GLA is to be able to tackle associative reinforcement learning problems directly. The probability generating function of the GLA is dependent on both the state $\mathbf{u}$ and the context vector $\mathbf{x}$. As stated earlier, this is the most important characteristic of GLA, that its next action, in addition to internal state of LA is dependent on input context vector.

4 PROPOSED METHOD FOR PARAMETER TRAINING

In this section an iterative learning automata-based method is proposed, for training conditional probability tables of Bayesian networks. This method initializes parameter values from a monotonous distribution, and on visiting each training sample, modifies the distribution of the parameters to approach the real distribution of training samples. To do this, a GLA is defined with the following features.

Action set (*α*): in the LA for each value of class variable an action is considered.

Feedback computation (*β*): feedback received from the environment depends on the processing result of the sample just visited. First an action (or a class value) is randomly selected, and then the real class value is taken from the training sample. If these two values match, the selected action is rewarded, otherwise it is penalized.

Probability generating function (*g*): as mentioned in section 3, in GLA action probabilities are not held in a vector statically. Instead, for each training sample and internal state of Bayesian network, probability generating function will compute all action probabilities of LA. For a sample S_i, generated probability for each action c_j is equal to $P(c_j|a_{1i},a_{2i},...,a_{ni})$ which could be computed from the network. a_{ni} is the value of *n*th variable in sample *i*.

Choosing actions: against general VSLAs that the next action is chosen only based on values in action probability vector, in this GLA, choosing next action also depends on the training sample which is being processed. As mentioned earlier this dependence is dynamically implied in computing action probabilities.

Reward and penalty: as stated earlier, internal state vector of this GLA (*U*) is identical to values in conditional probability tables of variables. Context vector *X* is set of training samples which are available. For rewarding and penalizing, a learning function *T* is needed to update *U*. To do this, we have used a method similar to L_{R-P} algorithm. As mentioned earlier, selected class value for a sample by the network, will be rewarded in case of equality with its correct value in sample, and penalized otherwise. Since computed probability for each class (or each action) is the product of relevant values in conditional probability vector of variables, all these values take part in getting reward or penalty. Each row in conditional probability table of a node is a probability vector and we could update its entries using equations used in linear learning automata. Reward and penalty equations are as below.

$$p_{ik}(n+1) = p_{ik}(n) + a \cdot (1 - p_{ik}(n))$$
$$p_{jk}(n+1) = (1-a) \cdot p_{jk}(n) \quad \forall j\ j \neq i \qquad \text{(reward)}$$
$$p_{ik}(n+1) = (1-b) \cdot p_{ik}(n)$$
$$p_{jk}(n+1) = b/(r_k - 1) + (1-b) \cdot p_{jk}(n) \quad \forall j\ j \neq i \qquad \text{(penalty)}$$

(8)

Row *k,* is the relevant row in conditional probability table of each node. Column *i* is the column from which probability value is taken. Other columns are denoted by *j* subscript. Number of columns of each table is denoted by r_k. *a* and *b* are learning parameters that could be tuned between zero and one. The greater these parameters are, the faster convergence rate will be but the less precision will be obtained. These values are chosen near one in experiments. Fig.1 shows the running algorithm of proposed method.

```
1. Assign each entry of each table, the value $1/r_{n,k}$ which $n$ is the index for node and $k$ is the index for the corresponding row in that node.
2. For each training sample $S_d$ do the following steps
   2.1. For value $c$ of class variable
        2.1.1. For each node $X_n$ in network
               2.1.1.1. Find row k and column i of each variable's table from which probability $P_{v(n)}=P(v(n)|v(Parents(n))$ must be extracted. $v(n)=x_n$ is the value of variable $X_n$ in training sample $S_d$.
        2.1.2. Multiply probabilities extracted from all nodes and consider it as the probability $P(c|x_1 ,\dots, x_{v-1})$.
   2.2. Provide the class with maximum probability as the prediction of classification.
   2.3. Compare predicted value with the real value in training sample.
        2.3.1. If the two values match, update probabilities in corresponding rows by reward equation, otherwise update them by penalty equation.
```

Fig.1 Algorithm of proposed parameter training method

5 PROPOSED METHOD FOR STRUCTURE TRAINING

The proposed method uses a set of LAs to perform search. In this method, for each possible connection in the network an LA is considered. These possible connections can be any arc in the full graph of the variables. Each LA has the following characteristics:

Set of actions (α): on each connection in the graph we can have three states; having an arc in two directions or having no arc. These three states are considered as three actions of each LA.

Output (β): The output or feedback is computed based on the value of scoring function. If the score in current iteration is greater than the score in previous iteration, all LAs will be rewarded and otherwise will be penalized.

Learning function (T): Linear reward penalty algorithms presented in section 3 will be used to learn each LA's best action.

In each iteration every LA chooses an action. Actions chosen by all LAs make up a graph. The directed graph will then be assessed to be a valid Bayesian network. For this, we need to check that the class variable has no parents and the graph has no cycles. This validity check involves $O(n^2)$ computational cost where *n* is the number of network variables. If the graph is invalid all LAs will be penalized. If the graph is valid, based on comparison of score in current and previous iterations LAs will be rewarded or penalized. Search process continues until no improvement in structure score is observed. The algorithm is shown in Fig.2.

```
1.  Do the following steps until the current score is not more that the score
    of previous step
    1.1. Start by an initial structure (like NB)
    1.2. In each step, each LA chooses an action based on its action
         probability vector.
    1.3. Let actions chosen by all LAs shape the network graph.
    1.4. If produced graph is not a valid graph (class variable has no
         parents and graph has no directed cycles) penalize chosen actions
         of all LAs.
    1.5. Otherwise, if current network has greater score than score of
         previous step reward LAs and otherwise penalize them.
```

Fig.2 Algorithm of proposed structure training method

6. RESULT OF EXPERIMENTS

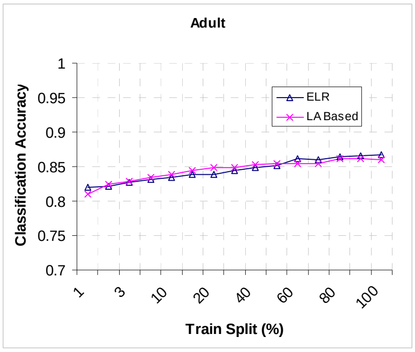
6.1. PARAMETER TRAINING

To evaluate the proposed algorithm, we have used five datasets from UCI Machine Learning Repository [12]. Tab.1 gives a brief description of these datasets.

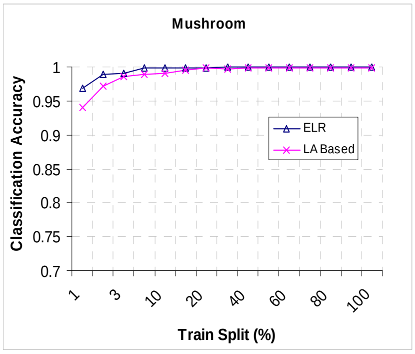
Tab.1 Datasets and their characteristics

Dataset	number of attributes	number of samples	number of classes
Adult	15	32561	2
Iris	4	150	3
mushroom	22	8124	2
Wine	13	178	3
Cancer	32	569	2

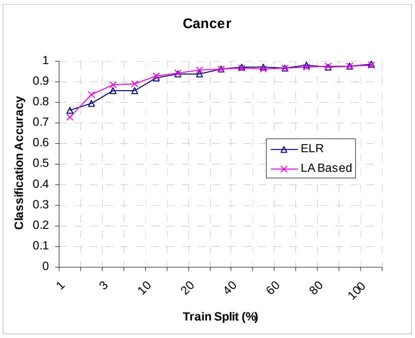
At the beginning, datasets that contained numerical values are discretized. Network structure is prepared with TAN algorithm, and having this structure, results of the proposed algorithm is compared to ELR algorithm. Learning algorithms is L_{R-P} in the GLA. Learning curve for ELR and the proposed algorithm (LA-Based) is shown in Fig.3.



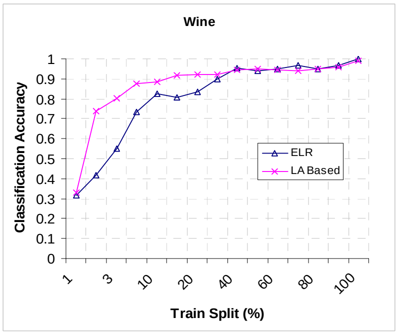
(a)



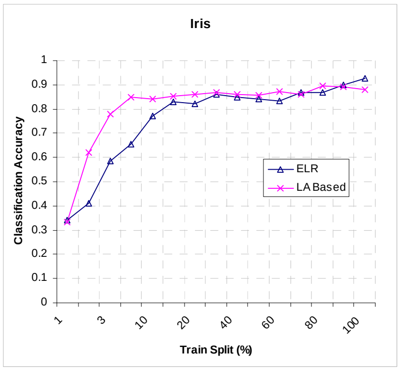
(b)



(c)

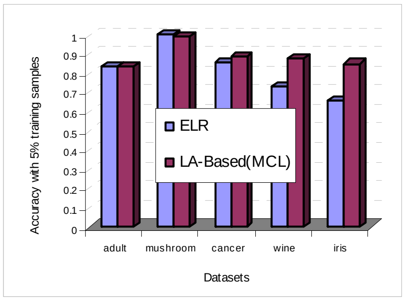


(d)



(e)

Fig.3. Learning curve on different datasets, a through e.



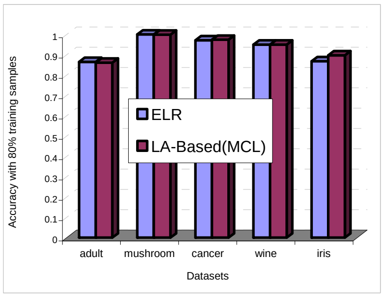


Fig.4 Accuracy with 5% training samples

Fig.5 Accuracy with 80% training samples

As it is clear, the proposed algorithm did not provide good results for some datasets, but the results are acceptable for others. On datasets that have high noise rate, like Adult, although there are many training samples available, it is observed that neither LA-Based nor ELR algorithm have good classification results. For this dataset, LA-Based algorithm does not show acceptable classification result, with less number of training samples. According to definition, having noise in a sample means that the values of attributes in the sample do not comply with the function that implicitly exists over all samples [2]. In pattern recognition problems, basically the value of class variable is a function of the values of other variables in training and test samples. A noisy sample does not comply with this rule.

The Mushroom dataset, because of having very little noise rate, shows good accuracy for ELR algorithm, and LA-Based algorithm does not show its superiority on reducing the number of training samples. But for other three datasets, we can clearly see the faster convergence of LA-Based algorithm to final classification accuracy, with less number of training samples. For Cancer dataset, although the convergence is not that sensible, finally with having all training samples, LA-Based algorithm has performed slightly better than ELR algorithm. In the end, we can say that when training samples are scarce and there is a moderate amount of noise in samples, the proposed algorithm can reach the same classification accuracy as ELR algorithm, with much less number of training samples. This is further illustrated in Fig.4 and Fig.5. In Fig.4 propose algorithm outperforms classification accuracy of ELR algorithm when only 5 percent samples are used for training. Fig.5 shows that the proposed algorithm has nearly the same classification accuracy of ELR algorithm when 80 percent of samples are used to train the network.

6.2 STRUCTURE TRAINING

For evaluation of proposed structure training algorithm, 25 datasets are used, which consist of 21 datasets from UCI [12] and three others from [19]. Tab.2 shows a brief description of these datasets. The proposed structure training algorithm is implemented in three versions, where they only differs in the evaluation function they use. These three algorithms are compared to seven other algorithms. Below is a description of all implemented algorithms:

Tab.2 Datasets and their characteristics

Dataset	number of classes	number of attributes	number of samples
australian	2	14	690
breast	2	10	683
chess	2	36	2130
cleve	2	13	296
corral	2	6	128
crx	2	15	653
diabetes	2	8	768
flare	2	10	1066
german	2	20	1000
glass	7	9	214
glass2	2	9	163
heart	2	13	270
hepatitis	2	19	80
iris	3	4	150
letter	26	16	15000
lymphography	4	18	148
mofn-3-7-10	2	10	300
pima	2	8	768
shuttle-small	7	9	3866
vote	2	16	435
satimage	6	36	4435
segment	7	19	1540
soybean-large	19	35	562
vehicle	4	18	846
waveform-21	3	21	300

- LA-based (ML): in this algorithm a Learning Automata-based technique is used to train network structure. The evaluation function is Maximum Likelihood. All three versions of proposed structure training methods use Maximum Likelihood as their parameter estimation method.

- LA-based (MCL): uses Maximum Conditional Likelihood as its evaluation function. In fact this algorithm is similar to BNC algorithm proposed in [17]. The only difference is that it uses LA-based method for searching.
- LA-based (CR): evaluation function is classification rate (CR). This function was proposed in [16].
- NB (ML): This is the Naïve Bayes which uses ML as its parameter estimator.
- TAN (ML): The structure is trained using TAN algorithm. This also uses ML for parameter estimation.
- HC(ML): The algorithm uses hill climbing search method with ML as scoring function.
- HC(CL): This is exactly BNC structure training algorithm. Using hill climbing it search for the best value of MCL function. Parameter estimator is ML.
- HC(CR): Just like the previous algorithm except for the evaluation function which is CR here.
- GA(CL): Structure searching method is a genetic algorithm. Best ordering of variables is searched before starting the search process [18].
- GA(CR): search algorithm is the same as above but the scoring function is classification rate.

Fig.6 compares average classification error of all algorithms on different datasets.

As it is clear in Fig.6 our algorithm, Except for LA-based(ML), has outperformed simple structures like NB and TAN. On comparison with hill climbing search, again LA-based search is indicating better results. This could be seen when comparing each LA-based algorithm with its rival hill climbing algorithm which uses the same scoring function. Two genetic algorithms are also compared to their rival LA-based algorithms. When the scoring function is MCL, genetic search performs a bit better than its LA-based counterpart. But for CR scoring function, LA-based search outperforms its rival genetic algorithm either. The best result among all algorithms is gained when search method is LA-based and scoring function is CR.

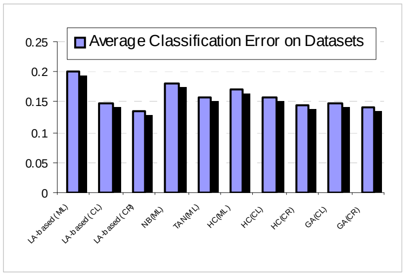


Fig.6 Average classification error for structure training algorithms

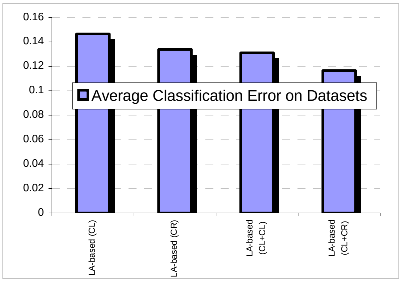


Fig.7 Average classification error for combinational algorithms

6.3. COMBINATIONAL TRAINING

In this section a combinational algorithm is tested on datasets. In previous experiment, LA-based structure training was reported as the best search method among other tested algorithms. Here we are going to test that structure training method with the company of our proposed parameter training method. The superiority of this combinational training is that we can perform efficient discriminative parameter training methods with an efficient search based structure training methods. If the two training problems are high-cost search algorithms, the problem will remarkably become heavy since the order of two training methods will be multiplied. Our parameter training method is a low-cost algorithm and it would be practical to use it with a search based structure training method.

Four algorithms are tested. Two former algorithms are just the ones in previous experiment. The latter two ones use LA-based parameter training method with MCL probability estimation function. Fig.7 shows the average classification error for all algorithms.

It is clear that the combinational algorithm can improve performance compared to the case where only structure training is a discriminative method. LA-based(CL+CL) outperforms its rival LA-based(CL) and the same result is achieved on comparing LA-based(CL+CR) and LA-based(CR) algorithms. The best result is observed when structure scoring function is CR and parameter estimation is performed by MCL function. Here we can see that we used discriminative methods for both parameter and structure training, while we still held computational cost of parameter training low.

7. CONCLUSION

In this paper two Learning Automata-based methods were proposed for training both parameters and structures of Bayesian networks. For proposed parameter training method results indicated that when there is moderate amount of noise in training samples of the datasets, the proposed algorithm can reach the same accuracy of rival algorithms with much less number of training samples. The proposed algorithm has less computation cost and training and test phases for the network can be performed simultaneously.

The proposed structure training algorithm tries to perform a guided search on space of network structures. Result showed that the algorithm can outperform its rivals such as genetic and gradient descent algorithms. Finally, two proposed algorithms were used in a combinational way. This further improved classification accuracy since we used two discriminative methods for training parameters and structure of the network. Also computational cost of combinational method was still rational.

REFERENCES

[1] J. Cheng and R. Greiner, "Comparing Bayesian Network Classifiers", Morgan Kaufmann, In Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence, Pages 101—107, 1998.

[2] Tom M. Mitchell, "Machine Learning", McGraw-Hill, 1997.

[3] S. Sarkar and S. Chavali, "Modeling Parameter Space Behavior of Vision Systems Using Bayesian Networks", Elsevier Science Inc., Computer Vision and Image Understanding, Volume 79, Pages 183 – 223, 2000.

[4] K. P. Murphy, "An Introduction to Graphical Models", Technical Report, Intel Research Technical Report, 2001.

[5] J. Cheng and R. Greiner, "Learning Bayesian Belief Network Classifiers: Algorithms and System", Proceedings of 14 Biennial Conference of the Canadian Society for Computational Studies of Intelligence, Volume 2056, Pages 141-151, 1998.

[6] M. A. L. Thathachar and P. S. Sastry, "Varieties of Learning Automata: An Overview", IEEE, Transaction on Systems, Man, and Cybernetics, Volume 32, Pages 711- 722, 2002.

[7] S. H. Zahiri, "Learning Automata Based Classifier", Elsevier Science, Pattern Recognition Letters, Volume 29, Pages 40–48, 2008.

[8] McCallum and K. Nigam, "A Comparison of Event Models for Naive Bayes Text Classification", In Proceedings of AAAI/ICML-98, Workshop on Learning for Text Categorization, 1998.

[9] G.F. Cooper and E. Herskovits, "A Bayesian Method for the Induction of Probabilistic Networks from data", Machine Learning, Volume 9, Pages 309-347, 1992.

[10] C. Haung and A. Darviche, "Inference in Belief Networks: A Procedural Guide", Elsevier Science, International Journal of Approximate Reasoning, Volume 15, Pages 225-263, 1994.

[11] H. Beygi and M.R. Meybodi, "A New Action-Set Learning Automaton for Function Optimization", Int. J. Franklin Inst. Volume 343, Pages 27–47, 2006.

[12] P. M. Murphy and D. W. Aha, UCI Repository of Machine Learning Databases, 1995, Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

[13] J. H. Friedman, "On bias, variance, 0/1 - loss, and the curse-of-dimensionality". Data Mining and Knowledge Discovery, Volume 1, Pages 55-77, 1997.

[14] C. K. Chow and C. N. Liu, "Approximating Discrete Probability Distributions with Dependence Trees". IEEE Transactions on Information Theory, Volume 14,Pages 462-467, 1968.

[15] Grossman, D., & Domingos, P., "Learning Bayesian Network Classifiers by Maximizing Conditional Likelihood", ACM, 21st International Conference of Machine Learning (ICML), 69, 361–368, 2004.

[16] Pernkopf, F., "Bayesian Network Classifiers versus Selective k-NN Classifier", Elsevier, Pattern Recognition, 38, 1–10, 2005.

[17] Greiner, R., & Zhou, W., "Structural Extension to Logistic Regression: Discriminative Parameter Learning of Belief Net Classifiers", AAAI, 18th National Conference on Artificial Intelligence, 167-173, 2002.

[18] Larranaga, P. & Kuijpers, C.M.H & Murga, R.H. & Yurramendi Y., "Learning Bayesian Network Structures by Searching for the Best Ordering with Genetic Algorithms", IEEE Transactions on, Systems, Man and Cybernetics, 26, 487-493, 1996.

[19] Kohavi, R., & John, G., "Wrappers for Feature Subset Selection", Elsevier Science, Artificial Intelligence, 97, 273-324, 1997.

[20] Pernkopf, F. & Bilmes, J., "Discriminative versus Generative Parameter and Structure Learning of Bayesian Network Classifiers", ACM, 22nd International Conference on Machine learning, 119, 657-664, 2005.

[21] Chow, C. K., & Liu, C. N., "Approximating Discrete Probability Distributions with Dependence Trees", IEEE Transactions on, Information Theory, 14, 462-467, 1968.

[22] Pedro, M. & Yosu, R.H. & Cindy, M.H., "Learning of Bayesian Networks by Genetic Algorithms: A Performance Analysis of Control Parameters", IEEE Transactions on, Pattern Analysis and Machine Intelligence, 18, 912-926, 1996.