

## Evolutionary Learning of Dynamic Naive Bayesian Classifiers \*

**Miguel A. Palacios-Alonso**

Department of Computer Sciences - CICESE Research Center  
Km. 107 carretera Tijuana-Ensenada, Ensenada 22860, B.C., México  
mpalacio@cicese.mx

**Carlos A. Brizuela** †

Facultad Politécnica - Universidad Nacional de Asunción  
Campus Universitario, San Lorenzo, Paraguay  
cbrizuel@cicese.mx

**L. Enrique Sucar**

Department of Computer Science - Instituto Nacional de Astrofísica, Óptica y Electrónica,  
Luis Enrique Erro N. 1, Santa Maria Tonantzintla, Puebla 72840, México  
esucar@inaoep.mx

### Abstract

Naive Bayesian classifiers work well in data sets with independent attributes. However, they perform poorly when the attributes are dependent or when there are one or more irrelevant attributes which are dependent of some relevant ones. Therefore, to increase this classifier accuracy we need a method to design network structures that can capture the dependencies and get rid of irrelevant attributes. Furthermore, when we deal with dynamical processes there are temporal relations that should be considered in the network design. We propose an evolutionary optimization algorithm to solve this design problem. We introduce a new encoding scheme and new genetic operators which are natural extensions of previously proposed encoding and operators for grouping problems. The design methodology is applied to solve the recognition problem for nine hand gestures. Experimental results show that the evolved network has higher average classification accuracy than the basic dynamic naive Bayesian classifier.

### Introduction

Many problems such as voice recognition, speech recognition, images processing and many other tasks have been tackled with Hidden Markov Models (HMM) (Rabiner 1989). These problems can also be dealt with an extension of the Naive Bayesian Classifier (NBC) known as Dynamic NBC (DNBC) (Aviles-Arriaga *et al.* 2003; Aviles-Arriaga, Sucar, & Mendoza 2006). The DNBC has shown better performance than the HMM when the number of training samples is small. The NBC is a very powerful method to deal with data where the attributes are independent given the class. However, it is known that when the attributes are dependent, or when one or more irrelevant attributes have some degree of dependency with the

relevant ones, then the performance of this simple classifier decreases considerably (Pazzani 1996). This fact has motivated the study of methodologies that can help having better network designs to cope with this performance deterioration problem. We want a method capable of grouping together attributes that are dependent and get rid of irrelevant ones. If we apply this classifier to model dynamical processes, then temporal relations should also be taken into account. The DNBC as opposed to the NBC needs to find the optimal number of states in the class node. That is, the design problem requires us to provide the number of states in the hidden node and, most importantly, the association of variables corresponding to the children nodes. The exponential nature of the exact computation of this design problem has motivated the development of alternative non exhaustive procedures.

Two schemes can be identified to solve the general structural learning problem: model selection by search and score (Friedman 1998; Friedman, Murphy, & Russell 1998), and dependency tests (Sucar, Gillies, & Gillies 1994; Pazzani 1996). A recently proposed method that falls in the second category was proposed by (Martinez-Arroyo & Sucar 2006), the method learns an optimal NBC, at the same time that performs discretization. This method contributes interesting ideas to our approach since it proposes the grouping and elimination of attributes. Our approach, that falls in the first category, proposes to use an evolutionary algorithm to determine near optimal solutions for the number of states and association of attributes. Evolutionary computation has widely been applied to the design of network structures (Larrañaga & Poza 1996; Wong, Lee, & Leung 2002; Myers, Laskey, & DeJong 1999; Ross & Zuviria 2007). These approaches belong to model selection by search and score method. Most of these works concentrate on the structure evolution of static BNs (Ross & Zuviria 2007). It is worth noting that these approaches can be easily extended to deal with DBNs, however none of them try to evolve nor the groups, neither the number of states of the hidden node. Our main contribution here is the proposal of a new approach to

\*The authors would like to thank Hector Avilés for his valuable help in many aspects of the work. This work was supported by CONACyT under grant C01-45811.

†On his sabbatical leave from CICESE  
Copyright © 2008, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

evolve DNBCs in order to group dependent attributes and eliminate irrelevant ones. To achieve this we propose a special coding scheme, new corresponding genetic operators, and a novel fitness function evaluation that considers the classification accuracy on a partial data set.

### The Problem

The problem we are dealing with has to do with the design of DBNCs. In order to define this design problem we need to introduce first some preliminary concepts.

#### Dynamic naive Bayesian classifiers

This model is composed of the set  $A = \{A_n^1, A_n^2, \dots, A_n^T\}$ , where each  $A_n^t$  for  $t = 1, \dots, T$  is a set of  $n$  instantiated attributes or observation variables generated by some dynamic process, and  $C = \{C_1, C_2, \dots, C_T\}$  the set of  $T$  classes random variables  $C_t$ , generated by the same process at each time  $t$ .

A DNBC  $\lambda = (S, \theta)$ , where  $S$  is the structure and  $\theta$  the parameters, has the following general probability distribution function:

$$P(A, C) = P(C_1) \prod_{t=1}^T \prod_{j=1}^n P(A_j^t | C_t) \prod_{t=2}^T P(C_t | C_{t-1}) \quad (1)$$

where:

- $P(C_1)$  is the initial probability distribution for the class variable  $C_1$ ,
- $P(A_j^t | C_t)$  is the probability distribution of an attribute given the class variables over time.
- $P(C_t | C_{t-1})$  is the class transition probability distribution among class variables over time.

The product  $\prod_{j=1}^n P(A_j^t | C_t)$  stands for the naive assumptions of conditional independence among attributes given the class. To represent the model, we rely on two standard assumptions: i) the process is Markovian, which establishes independence of the future respect to the past given the present, and ii) the process is stationary, i.e., the transition probabilities among states are not time dependent.

Following the graphical representation of probabilistic independence (Pearl 1988), a DNBC model unrolled two times can be depicted as shown in figure 1. Although it is possible to describe these models using an analytical form, it is simpler and clearer to describe them under a graph representation. This representation allows us to consider well-known techniques for probability propagation in Bayesian networks (Pearl 1988) and the EM algorithm for training with missing data (Rabiner 1989).

In order to avoid the loss of temporal information, we can consider all the information generated by the dynamic process as attributes in a sequence, without the need of discretizing activity observations on a constant number of samples. Then, the class that best explains the observations at each time  $t$  can be found. The effects of previous classes on the recognition of the current class is described in terms of the transition probability distribution  $P(C_t | C_{t-1}, \lambda)$ .

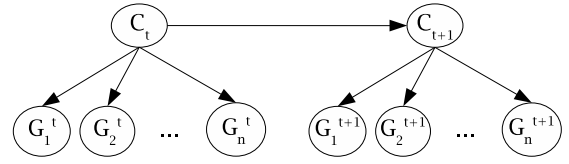


Figure 1: Dynamic Naive Bayes Classifier

After this brief introduction of DNBC we need to motivate our formulation of the problem. As we mentioned before NBCs performs poorly on data sets with dependent attributes or when there exist irrelevant attributes that have a degree of dependency with the relevant ones (Pazzani 1996). Therefore, in order to design a more accurate classifier we need to discover which attributes are dependent and which are irrelevant.

With this in mind we can define our problem. To do so let us first restrict our universe of network structures  $S$  to the NBC type of structures  $S'$ , i.e. graph having a single parent node, representing the class node, and its children nodes that represent the grouping  $G$  of variables or attributes. Once a grouping  $G$  of the variables is given the above mentioned techniques can be used to compute near optimal parameters for that grouping. Then, **the problem is** to decide which grouping  $G$  to use and the number of states in the hidden node, since the optimum will be the one grouping together dependent attributes and leaving aside the irrelevant ones. If we use brute force to determine this optimal structure (grouping) and its corresponding optimal number of states, in a problem with  $n$  variables (attributes), then we need to search in a solution space of size given by the following equation:

$$|S'_n| = \left( \sum_{i=1}^n \binom{n}{n-i} B_i \right) * n_s, \quad (2)$$

where  $B_i$  is the Bell number of  $i$  elements (Cameron 1994) and  $n_s$  is the number of states in the hidden class. It is not hard to see that  $S'_n$  grows exponentially with  $n$ . Therefore, we cannot exhaustively explore the solution space even for a small number of variables and we need an alternative to the brute force to find the optimal or near optimal grouping  $G$ .

### The Proposed Evolutionary Learning Approach

Our DNBC has a structure like the one depicted in figure 1. There is a single hidden node, i.e. the classification node  $C_t$ , with a given number of states. Then the children of this node,  $G_1, G_2, \dots, G_r$ , represent the groups or associations of variables with  $r$  ( $0 < r \leq n$ ) the number of groups or associations, and  $n$  the number of variables. Each group  $G_i$  is composed of a number of variables ranging from  $1 \leq |G_i| \leq n$ .

We can see that a solution is given by a specific grouping of the random variables and by the number of states in the classification node. Therefore, a candidate representation

for this solution will be a group based codification which is explained next.

## The Representation

We use a variant of the group representation proposed by Falkenauer, (1994). The chromosome consists of two parts: the object part (in our case the random variable part) and the group part. In the object part each locus is an identifier for each random variable and its corresponding allele the group it belongs to. The group part has the identifier for each group. We know that in a grouping problem each object must belong to a group. However, in our structure optimization problem, a random variable may not be assigned to any group. Therefore, we assign a special identifier to an object (a random variable) that does not belong to any group. We also need to encode the number of states for the hidden node, we propose to use a binary string to encode this part. It is important to consider that this basic unit in the representation is repeated a number of times equal to the number of models we are dealing with. Figure 2 shows a part of individual  $j$ , representing the encoding for one of the models (model  $i$ ), the hidden node has seven states (0111), variable  $\Delta_x$  is associated to group C,  $\Delta_y$  belongs to group D, variable F belongs to group F, variables A and R are together in group B. The group part indicates that we have four groups. Variables  $\Delta_a$  (locus 3) and T (locus 7) are not assigned to any group.

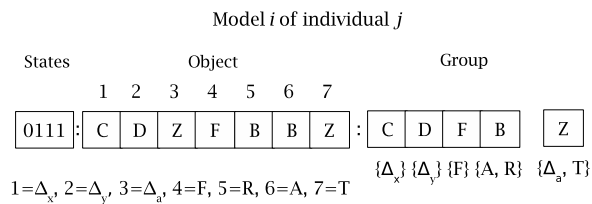


Figure 2: Representation of model  $i$  that belong to individual  $j$

Once the representation is given the pseudocode for the main algorithm is described in Algorithm 1. The input to the algorithm are the training data set and the user defined parameters needed by the algorithm. The output is a DNBC,  $\lambda$ , for each of the models. In step 1 a set of groups are randomly formed and their parameters computed. Step 2 computes the fitness for each of the generated individuals based on the partial test data set. In step 3 a loop is initiated and it finishes when a maximum number of iterations without changes in the best individual fitness is achieved or a maximum number of iterations is reached. We dedicate the next sections to the explanations of each component in the loop.

### Algorithm 1. EvoDNBC

**Input:** Data ( $D$ ), number of models ( $n\_models$ ), training data ( $P\_train$ ), partial test data ( $P\_test$ ), weighting factor ( $\alpha$ ), maximum number of iterations ( $Max\_Iter$ ), mutation rate ( $P_m$ ), Population Size ( $PopSize$ ),

tournament size ( $J$ ), the maximum number of iterations without changes in the fitness of the best individual ( $Max\_Iter\_No\_Change$ ).

**Output:** A DNBC and its corresponding score.

- 1 Initialize  $PopSize$  individuals with random  $n\_models$  models.
- 2 Evaluate the fitness of the  $PopSize$  individuals.
- 3 While the number of generations without changes in the fitness is less than  $Max\_Iter\_No\_Change$  and less than  $Max\_Iter$ , Do:
  - 4 For  $i=1$  to  $\text{ceil } PopSize/2$
  - 5 Choose  $J$  individuals to participate in the tournament and the winner will be Parent1
  - 6 Choose  $J$  individuals to participate in the tournament and the winner will be Parent2
  - 7 Perform crossover between Parent1 and Parent2 to obtain two new individuals.
  - 8 Apply mutation to each new individual with probability  $P_m$ .
  - 9 Evaluate the fitness of the new population.
  - 10 Replace the actual population with the best  $PopSize$  among the parent and children populations.
  - 11 Evaluate the classification rate of the last generation
  - 12 End While

## The Genetic Operators

We use an adapted version of the standard crossover and mutation operators proposed for groups (Falkenauer 1994). Each of these operators are described in the following.

**Crossover** The crossover operator (Step 7) can be explained using the specific example illustrated in figure 3. Only the group part of each parent is considered. Two randomly selected positions are defined on each parent. In the figure we can see these positions, for Parent 1 (P1) between groups A and D (the first point), and between groups F and B (second point). The same selection is performed for Parent 2 (P2), but this time the first point is between groups A and C, and the second point between groups D and E. We can also observe that Parent 1 has two eliminated variables  $\Delta_a$  and T, while Parent 2 has none. In the second step we can see that groups D and F of P1 are inserted at the beginning of the first crossover point in P2. Then elements in Z of P1 are added to elements in Z in P2. In the third step we start eliminating all repeated variables, in this example, they are  $\Delta_y$  and F. Also variables in the left side that appear in the Z group are eliminated, in our example, we see that variable T in group A also appears in group Z, therefore it has to be

eliminated. Finally, at the fourth step we merge the variables that are not part of any existing group. If a single variable is left then with a probability of two thirds it is inserted as a new group and with probability of one third it is inserted as a part of an existing group, where each group has the same probability of being selected. If more than one variable are left we can take one of the following three actions with the same probability: all variables are inserted as members of a new group, all variables are inserted as a part of an existing group, or each variable is separately inserted as a new group. In our example a single variable R was left, then the chosen option was to insert it into group B. The crossover operator for the number of states in the hidden node follows the standard one point crossover operator for binary strings (Eiben & Smith 2003).

Notice that the whole procedure is repeated for all models that are selected to undergo crossover.

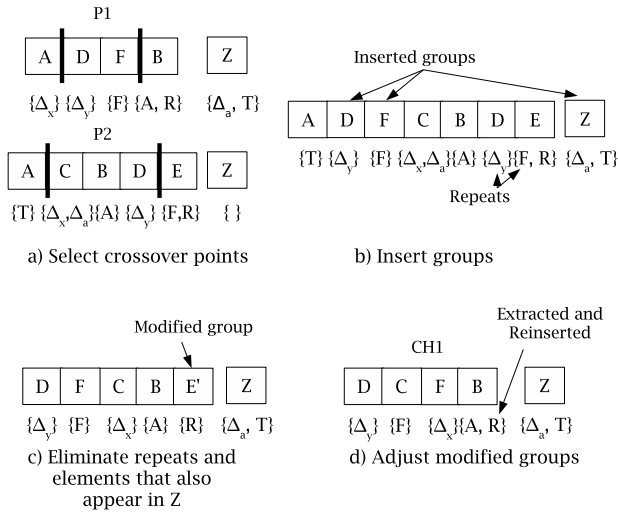


Figure 3: Crossover Operator

**Mutation** In this case (Step 8) we also choose an example to illustrate how the operator works. We have two options that are equally likely to be performed: Insertion or Deletion. In Insertion we can select to insert a variable or to insert a group. If we insert a variable this is taken from Z and it is inserted in any of the existing groups with the same probability. If we insert a group then this has to come from Z, and its size and composition are also randomly selected. In case of Deletion we also have two options, to delete a variable, randomly selected from a group, or to delete a group. In both cases the deleted elements are inserted into Z. We can see in figure 4 the case where a variable is deleted, variable A from group B is deleted. Group B is deleted in the proposed example for group deletion. For the Insertion option we also have two choices, we can see how variable T is inserted as a part of group A, and group C, made of variable T, is inserted as a new group. Notice that if Z is empty then the only valid option is Deletion, i.e. Insertions are not allowed.

The mutation operator, for the number of states, is a single

bit mutation which happens with probability  $P_m$ .

**Selection and Replacement** The parent selection (steps 5 and 6) is performed by tournament (Goldberg 1989), with a tournament of size  $J$ . The replacement strategy (Step 10) is the  $(\mu + \lambda)$ , commonly used in evolutionary strategies (Rechenberg 1973).

**The Objective Function** The fitness function for individual  $i$  is given by the following expression:

$$fitness(i) = \alpha Acc(i) + (1 - \alpha)(1 - comp(i)), \quad (3)$$

where  $\alpha$  is the factor to weight the classification accuracy ( $Acc$ ) and the resulting network complexity. The normalized complexity (by the maximum number of parameters) measure  $comp$  is given by the sum of the number of parameters of each model in the classifier. The number of parameters of one model is obtained as follows:

$$\#parameters = \sum_{i=1}^g ||Pa(N_i)|| * (||N_i|| - 1) \quad (4)$$

where  $g$  is the number of nodes, including the class node,  $||Pa(N_i)||$  is the number of parameters of parents of node  $N_i$ , which is composed by a group of variables.  $||N_i||$  is the number of parameters of node  $N_i$ . This value is defined as follows:

$$||N_i|| = \prod_{R_j \in N_i} |R_j|$$

where  $|R_j|$  is the number of values that variable  $R_j$ , a member of  $N_i$ , can take. Notice that if node  $N_i$  has no parents then  $||Pa(N_i)|| = 1$ . In equation 4  $\alpha$  defines a specific compromise between accuracy and complexity. Since these criteria are in conflict with each other the problem can be actually modeled as a multi-objective optimization problem.

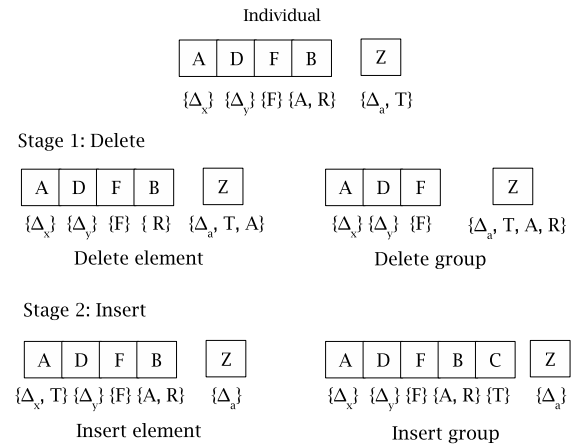


Figure 4: Mutation Operator

## Experimental Setup and Results

The proposed algorithm was evaluated in the visual recognition of 9 hand gestures: come, attention, right, left, stop, turn-right, turn-left, pointing and waving-hand; used for commanding mobile robots (Aviles-Arriaga, Sucar, & Mendoza 2006). Each gesture is modeled using a DNBC considering 7 attributes: 3 motion features and 4 posture features. These motion and posture features were obtained from a sequence of images. The motion features are:  $\Delta_a$ , or changes in the hand area,  $\Delta_x$  and  $\Delta_y$  indicate changes in hand position of the  $XY$  - *axis* of the image. Each of these features takes only one of three possible values: (+), (-) or (0) that indicate increment, decrement or no change between two consecutive images, depending on changes in the area and hand position of two images, respectively. The posture features are: *Form*, that indicates the form of the hand ((+) if the hand is vertical, (-) if the hand is horizontal, or (0) if the hand is lean to the left), *right*, indicates that the hand is at the right side of the head, *above*, if the hand is above the head, and *torso*, if the hand is over the user's torso, these last three features take binary values. For comparison purposes, we considered for each gesture a basic model with all the attributes (separated) and two states.

We conducted four experiments to evaluate the classification accuracy of the evolutionary learned classifiers. The gesture data set is composed of 50 samples for each of the nine gestures, taken from a single user, this data set is provided by (Aviles-Arriaga, Sucar, & Mendoza 2006). We select  $D_{training}$  samples per gesture to construct the complete training set, a partial testing data set  $D_{test_{partial}}$  is necessary to evaluate (compute the fitness) each one of the individuals in the evolutionary process. Finally, we evaluate the classification accuracy of the best individual with the  $D_{test_{final}}$  remaining samples.

In all experiments the crossover and mutation rates are set to  $P_c = 1.0$  and  $P_m = 0.35$ , respectively.  $PopSize = 12$ ,  $Max\_Iter\_No\_Change = 4$ , and  $Max\_Iter = 20$ . These values were obtained after a non exhaustive trial and error procedure. An statistical analysis is required to determine the best set of parameters. In the experiments we change  $\alpha$  and the size of the evaluation set as follows:

- Experiment 1.  $D_{training} = 10$ ,  $D_{test_{partial}} = 10$ ,  $D_{test_{final}} = 30$ ,  $\alpha = 0.8$ .
- Experiment 2.  $D_{training} = 10$ ,  $D_{test_{partial}} = 15$ ,  $D_{test_{final}} = 25$ ,  $\alpha = 0.8$ .
- Experiment 3.  $D_{training} = 10$ ,  $D_{test_{partial}} = 10$ ,  $D_{test_{final}} = 30$ ,  $\alpha = 0.7$ .
- Experiment 4.  $D_{training} = 10$ ,  $D_{test_{partial}} = 15$ ,  $D_{test_{final}} = 25$ ,  $\alpha = 0.7$ .

The EM algorithm with the same convergence criterion was used to estimate every instance of the DNBCs. Transition and observation probabilities for all the models in the population were initialized with discrete uniform distributions. The probability of each gesture sequence  $A$ ,  $P(A|\cdot)$ , was computed using the Forward algorithm (Rabiner 1989). All the experiments were carried out on a PC with AMD

Athlon 1.8Ghz, 3Gb of RAM, the algorithm was implemented in Matlab release 7.0.

Table 1 shows the mean and standard deviation of the accuracy and fitness of the best individual produced by the evolutionary learning process. The means are computed over 10 samples, i.e. the algorithm is run 10 times for each experiment.

Table 1: Mean classification accuracy and standard deviation computed for ten runs

	Accuracy (mean)	Fitness (mean)	Std. dev. (accuracy)	Std. dev. (fitness)
Exp1	0.957	0.993	0.020	0.006
Exp2	0.966	0.989	0.011	0.003
Exp3	0.967	0.994	0.013	0.003
Exp4	0.970	0.986	0.014	0.004

Figure 5 shows the nine models that belong to the evolved classifier obtained in the tenth run of Experiment 4. We can see that the proposed algorithm is able to learn an specific setting (variables association, variables elimination and specific number of states) for each model of the classifier.

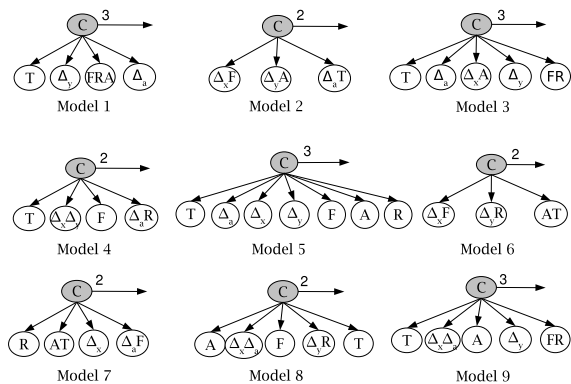


Figure 5: Evolved Dynamic Naive Bayes Classifier for the 9 gestures (come, attention, right, left, stop, turn-right, turn-left, pointing, and waving-hand)

The evolutionary process introduces the elimination and combination of variables at the same time that evaluates different number of states until the simplest classifier with a high accuracy is obtained. The recognition rates presented in Table 2 concern to the classifier learned by the evolutionary process presented in figure 5. As we described above the basic classifier has basic models, these models can contain redundant or dependent variables. We can see that the evolved classifier is better than the basic classifier in the average accuracy criterion, moreover each one of the models better describes the associated gesture. This is because the relations among variables and the number of states of the class variable is defined by the gesture in the evolutionary process. For example, Model 4 (left gesture) in figure 5 considers the variables  $\Delta_x$  and  $\Delta_y$  to be dependent of each other, T and F independents of all others, given the variable

class,  $\Delta_a$  and R are dependent of each other, while A is irrelevant to this particular gesture, and the number of states of the hidden node is 2.

Table 2: Gesture recognition rates using the dynamic naive Bayesian classifier: the basic model vs. the evolved model.

Gesture	Accuracy of the basic classifier	Accuracy of the evolved classifier
Come	96%	100%
Attention	100%	88%
Right	100%	100%
Left	96%	100%
Stop	100%	100%
Turn-right	100%	100%
Turn-left	100%	100%
Pointing	88%	88%
Waving-hand	72%	100%
Average	94.67%	97.33%

## Conclusions and Future Work

An evolutionary approach to solve the structural learning problem to design a DNBC has been proposed. The design of the best network structure is modeled as an optimization problem that measures the classification accuracy weighted by the resulting network complexity. To design the algorithm we propose a variant of the group based representation and its corresponding adapted operators. We test the resulting network using data generated from nine hand gestures. The experimental evaluation shows that the models obtained using our evolutionary approach improve in a significant way the recognition rates, and at the same time produce simpler and more intuitive structures.

Future work is aimed at reducing the computation time by computing the parameters of similar models only once. Another line of research has to do with the proposal of an evolutionary incremental learning approach in such a way that we do not need to run the algorithm from scratch when a new gesture is introduced to the system. Additional experiments are planned to analyze the robustness of the evolved classifier when noise and different users are considered.

## References

Aviles-Arriaga, H. H.; Sucar, L. E.; Mendoza, C. E.; and Vargas, B. 2003. Visual recognition of gestures using dynamic naive bayesian classifiers. In *The 12th IEEE International Workshop on Robot and Human Interactive Communication, 2003*, 133–138. Washington, DC, USA: IEEE Computer Society.

Aviles-Arriaga, H. H.; Sucar, L. E.; and Mendoza, C. E. 2006. Visual recognition of similar gestures. In *ICPR '06: Proceedings of the 18th International Conference on Pattern Recognition*, 1100–1103. Washington, DC, USA: IEEE Computer Society.

Cameron, P. J. 1994. *Combinatorics: topics, techniques, algorithms*. Cambridge.

Eiben, A. E., and Smith, J. E. 2003. *Introduction to evolutionary computing*. Germany: Springer-Verlag.

Falkenauer, E. 1994. A new representation and operators for genetic algorithms applied to grouping problems. *Evolutionary Computation* 2(2):123–144.

Friedman, N.; Murphy, K.; and Russell, S. 1998. Learning the structure of dynamic probabilistic networks. In *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence (UAI-98)*, 139–147. San Francisco, CA: Morgan Kaufmann.

Friedman, N. 1998. The bayesian structural EM algorithm. In *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence (UAI-98)*, 129–138. San Francisco, CA: Morgan Kaufmann.

Goldberg, D. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Massachusetts: Addison-Wesley Publishing Company.

Larrañaga, P., and Poza, M. 1996. Structure learning of bayesian networks by genetic algorithms: A performance analysis of control parameters. *IEEE transactions on pattern analysis and machine intelligence* 18(9):912–926.

Martinez-Arroyo, M., and Sucar, L. E. 2006. Learning an optimal naive bayes classifier. In *ICPR '06: Proceedings of the 18th International Conference on Pattern Recognition*, 1236–1239. Washington, DC, USA: IEEE Computer Society.

Myers, J. W.; Laskey, K. B.; and DeJong, K. A. 1999. Learning bayesian networks from incomplete data using evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, 458–465. Orlando, Florida, USA: Morgan Kaufmann.

Pazzani, M. 1996. Searching for dependencies in bayesian classifiers. In *Learning from Data: Artificial Intelligence and Statistics V*, 239–248. Springer-Verlag.

Pearl, J. 1988. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Rabiner, L. R. 1989. A tutorial on hidden markov models and selected applications in speech recognition. In *Readings in Speech Recognition, IEEE Proceedings*, 257–284. Morgan Kaufmann Publishers.

Rechenberg, I. 1973. *Evolutionstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution*. Stuttgart, Germany: Frommann-Holzboog.

Ross, B. J., and Zuviria, E. 2007. Evolving dynamic bayesian networks with multi-objective genetic algorithms. *Applied Intelligence* 26(1):13–23.

Sucar, L. E.; Gillies, D. F.; and Gillies, D. A. 1994. Probabilistic reasoning in high-level vision. *Image and vision computing* 12(1):42–60.

Wong, M. L.; Lee, S. Y.; and Leung, K. S. 2002. A hybrid approach to learn bayesian networks using evolutionary programming. In *CEC '02: Proceedings of the Evolutionary Computation on 2002. CEC '02. Proceedings of the 2002 Congress*, 1314–1319. Washington, DC, USA: IEEE Computer Society.