J Autom Reasoning (2010) 45:21–37
DOI 10.1007/s10817-009-9130-0

Evolutionary Learning of Dynamic Naive Bayesian Classifiers

Miguel A. Palacios-Alonso · Carlos A. Brizuela · L. Enrique Sucar

Received: 26 July 2008 / Accepted: 30 March 2009 / Published online: 16 April 2009
© Springer Science + Business Media B.V. 2009

Abstract Many problems such as voice recognition, speech recognition and many other tasks have been tackled with Hidden Markov Models (HMMs). These problems can also be dealt with an extension of the Naive Bayesian Classifier (NBC) known as Dynamic NBC (DNBC). From a dynamic Bayesian network (DBN) perspective, in a DNBC at each time there is a NBC. NBCs 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. In order to learn automatically these models from data and increase the classifier accuracy 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 DNBC and a HMM.

M. A. Palacios-Alonso · C. A. Brizuela (✉)
Department of Computer Science, CICESE Research Center,
Km. 107 carretera, Tijuana-Ensenada, Ensenada 22860, Baja California, México
e-mail: cbrizuel@cicese.mx

M. A. Palacios-Alonso
e-mail: mpalacio@cicese.mx

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

Keywords Naive Bayes classifier · Dynamic Bayesian networks · Genetic algorithms · Gesture recognition

1 Introduction

Many problems such as voice recognition, speech recognition, images processing and many other tasks have been tackled with Hidden Markov Models (HMMs) [18]. These problems can also be dealt with an extension of the Naive Bayesian Classifier (NBC) known as Dynamic NBC (DNBC) [1, 2]. 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 relevant ones, then the performance of this simple classifier decreases considerably [16]. This fact has motivated the study of methodologies that can help having better network designs to cope with this performance deterioration problem. Tree Augmented Networks (TANs) [8] are structures that introduce directed arcs between attributes, the class variable has no parents and each attribute has as parents the class variable and at most one other attribute. The disadvantage is that the structure becomes more complex as the number of attributes grows as it also happens with the inference algorithms used in the applications of this model. The basic TAN classifier does not consider temporal relations and attribute selection. To maintain a NBC as the basic structure in the dynamic model, it is required a method capable of grouping together attributes that are dependent and getting rid of irrelevant ones. If we apply this classifier to model dynamical processes, then temporal relations should also be taken into account. From this perspective, a DNBC is a dynamic Bayesian network whose base structure is a NBC. It is like a HMM in which the observation node has been decomposed in a number of attributes. The DNBC as opposed to NBC needs to find the optimal number of states in the hidden state variable as for HMMs. That is, the design problem requires us to provide the number of states in the hidden state variable 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 [7, 9], and by dependency tests [16, 21]. A recently proposed method that falls in the second category was proposed by [13], 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. Recently Martinez-Arroyo and Sucar [12] extend their work to learn DNBC and compare its performance with the one obtained by HMMs. 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. The DNBC combine the advantage of a NBC—simple structure and algorithms, with those of a HMM—the capacity to model complex dynamic processes. We learn the base structure (NBC) and the transition model (number of states of the hidden state variable and transition parameters) at the same time.

Evolutionary computation has widely been applied to the design of network structures [11, 14, 20, 22]. These approaches belong to model selection by search and score method. Most of them concentrate on the structure evolution of static BNs [20]. 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 state variable. Our main contribution here is the proposal of a new approach to evolve DNBCs in order to group dependent attributes and eliminate irrelevant ones.[1] 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. This methodology has been applied to learn DNBCs for gesture recognition, showing a significant improvement over the basic models.

The rest of the paper is organized as follows. Section 2 formalizes the problem and summarizes relevant related work. Section 3 describes the proposed evolutionary learning approach. In Section 4 the experimental results are presented, and we conclude with a summary and directions for future work in Section 5.

2 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.

2.1 Dynamic Naive Bayesian Classifiers

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

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

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

where:

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

The product $\prod_{j=1}^{n} P(A_j^t|C_t, M)$ stands for the naive assumptions of conditional independence among attributes given the hidden variable state. 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)

[1] An earlier version of this paper was presented as a conference paper [15].

the process is stationary, i.e., the transition probabilities among states are not time dependent.

A DNBC can be seen as a DBN, in particular as an extension of HMMs. It is like a HMM in which the observation node has been decomposed in a number of attributes and there is one hidden node (hidden state variable). From the DBN perspective, it is a particular class of DBN that has as base structure a naive Bayes classifier. Following the graphical representation of probabilistic independence [17], a DNBC model unrolled two times can be depicted as shown in Fig. 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 [17] and the EM algorithm for training with missing data [18].

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 state that best explains the observations at each time t can be found. The effects of previous states on the recognition of the current state is described in terms of the transition probability distribution $P(C_t|C_{t-1}, M)$.

After this brief introduction of DNBC we need to motivate our formulation of the problem. As we mentioned before, NBCs perform poorly on data sets with dependent attributes or when there exist irrelevant attributes that have a degree of dependency with the relevant ones [16]. 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 hidden state variable, 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 state variable, with the optimum being 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 (that process the temporal information), in a problem with n variables (attributes), then we need to search in a solution space of size given by the following equation:

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

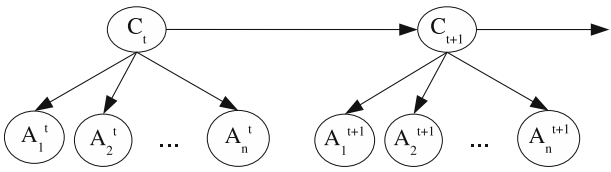

Fig. 1 A Dynamic Naive Bayes Classifier unrolled two times. C represents the hidden state variable, and A_1 ... A_n the attributes that are assumed independent given the state

where B_i is the Bell number of i elements [3] and n_s is the number of states in the hidden state variable. 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 and number of hidden states, n_s.

2.2 Related Work

Larrañaga and Posa [11] used a solution's representation based on a particular variable ordering, this approach evolved the variable ordering that is the input for the K2 algorithm to learn the network's structure. The approach proposed by Wong et al. [22] uses evolutionary programming to learn BNs and the metric to evaluate the networks is the minimum description length principle. This approach does not need to have a complete ordering as input and can learn multiple-connected network structures. Another interesting approach is the one presented by Myers et al. [14], they evolved the structure and the missing data. The missing values are represented by a string of data and the structure is represented by an adjacency list, the Bayesian Dirichlet score is used to evaluate the individuals. Wong and colleagues [22] present an improvement to their previous algorithm, they present a hybrid approach, use dependency analysis approach for checking the validity of a conditional independence assertion, which reduces the search space by excluding networks that contain invalid edges. Ross and Zuviria [20] present another approach to deal with the learning of DBNC's, given a sample sequence of multivariate data. They use a genetic algorithm to synthesize a network structure that models the causal relationships that explain the sequence, this approach uses a multi-objective evaluation strategy. The multi-objective criteria is established by the Bayesian information criterion score, they separate the log-likelihood and the complexity. They compare the performances of the single-objective genetic algorithm with their multi-objective algorithm to learn biological networks. General methods for learning DBNs do not consider attribute selection and defining the number of states in the hidden state variable, so they are not directly applicable to this problem. Recently Martinez-Arroyo and Sucar [12] propose a new approach to learn DNBC's and compared it with HMMs. The method determines: the number of hidden states, the relevant attributes, the best discretization, and the structure of the model. The resulting models improve in recognition rates compared to HMMs, and at the same time they are simpler. However, the method is based on a simple search strategy that tests the different design parameters in a predefined order; we explore an alternative approach based on evolutionary computation that varies all aspects at the same time, so it has a higher probability of obtaining a better configuration.

3 The Proposed Evolutionary Learning Approach

Our DNBC has a structure like the one depicted in Fig. 2. There is a single hidden node, i.e. the hidden state variable C_t, with a given number of states. Then the children of this node, $G_1, G_2, \cdots, G_r$, represent the groups or associations of variables with r ($0 < r \leq n$), the number of groups or associations, and n the number

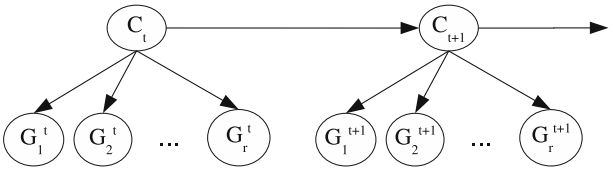

Fig. 2 A Dynamic Naive Bayes Classifier unrolled two times. C represents the hidden state variable, and G_1 ... G_r the grouping variables that are assumed independent given the state

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 hidden state variable. Therefore, a candidate representation for this solution will be a group based codification which is explained next.

3.1 The Representation

We use a variant of the group representation proposed by Falkenauer [6]. 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 state variable, 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 3 shows a part of individual j, representing the encoding for one of the models (model i), the hidden state variable has seven states (0111), variable Δ_x is associated to group C, Δ_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

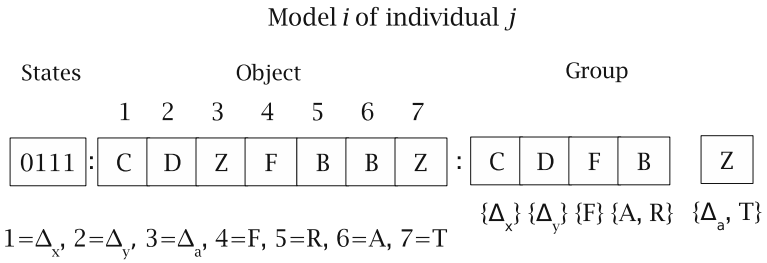

Fig. 3 Representation of model i that belong to individual j. The representation of an individual is conformed by: (a) the number of states represented as a binary string, (b) the object part representing the attributes and the group to which they belong, (c) the group part indicating the number of existing groups, and (d) the auxiliary group Z indicating the eliminated attributes

four groups. Variables Δ_a (locus 3) and T (locus 7) are not assigned to any group. Once the representation is given the pseudocode for the main algorithm is described in the Fig. 4. The input to the algorithm are the training data set and the user defined parameters needed by the algorithm. The output is a DNBC, M, for each of the models, as the algorithm can obtain several DNBCs simultaneously. 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.

3.2 The Genetic Operators

We use an adapted version of the standard crossover and mutation operators proposed for groups [6]. Each of these operators are described in the following sections.

3.2.1 Crossover

The crossover operator (Step 7) can be explained using the specific example illustrated in Fig. 5. 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,

EvoDNBC
Input: Data (D), number of models (n_models), training data (P_train), partial test data (P_test), weighting factor (α), maximum number of iterations (Max_Iter), mutation rate (Pm), Population size (PopSize), tournament size (J), the maximun 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 individual with random n_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 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 Pm.
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 rateof the last generation
12 End While

Fig. 4 The proposed evolutionary algorithm. The input to the algorithm is the training data set and the user defined parameters needed by the algorithm. The output is a DNBC, M, for each of the models to be learned

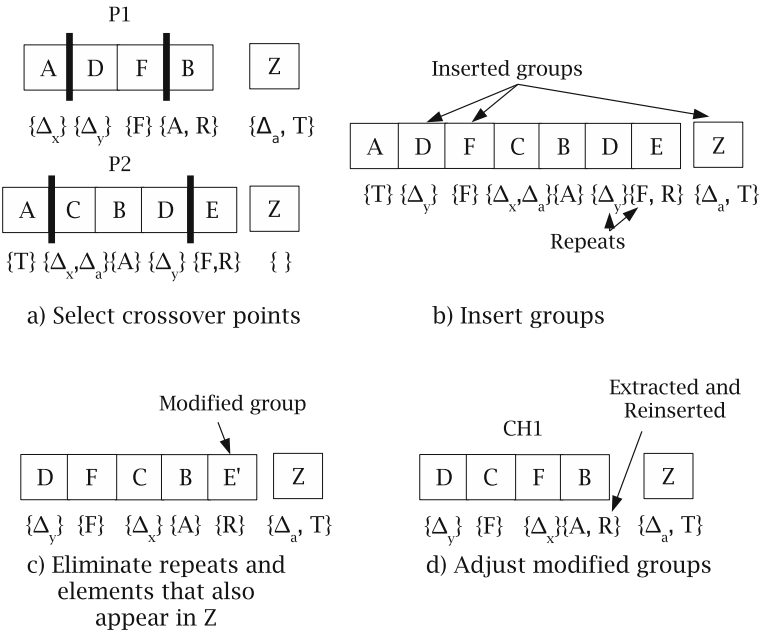

Fig. 5 Crossover operator. **a** Randomly select two crossover sections, **b** the section defined in the first parent is inserted in the second parent, **c** and **d** Adjustment of the resulting individual

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 Δ_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 Δ_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 state variable follows the standard one point crossover operator for binary strings [5].

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

3.2.2 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 Fig. 6 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. Please 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.

3.2.3 Selection and Replacement

The parent selection (steps 5 and 6) is performed by tournament [10], with a tournament of size J. The replacement strategy (Step 10) is the $(\mu + \lambda)$, commonly used in evolutionary strategies [19]; where μ and λ are the number of parents and the number of offspring, respectively. After creating λ offsprings and calculating their fitness, the best μ of the union of parents and offspring are chosen deterministically based on rank.

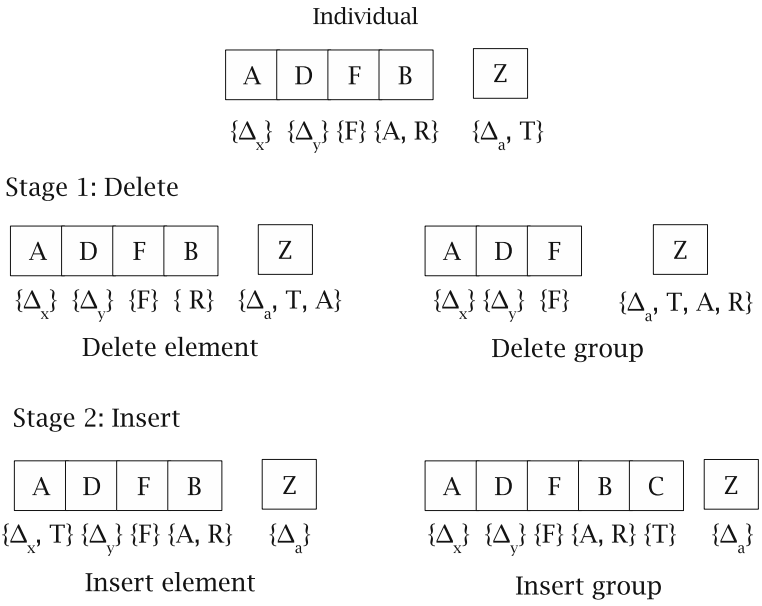

Fig. 6 Mutation Operator. There are two stages with two options. *Stage 1*: delete one group or one attribute. *Stage 2*, insert an attribute or a group

3.2.4 The Objective Function

The fitness function for individual i establish a compromise between the accuracy and complexity of the model, and it is given by the following expression:

$$\text{fitness}(i) = \alpha \text{Acc}(i) + (1 - \alpha)(1 - \text{comp}(i)), \tag{3}$$

where α is the factor to weight the classification accuracy (Acc) and the resulting network complexity (comp). The accuracy is the normalized rate obtained by testing the models on an evaluation data set, different from the training and test sets.

Let G' be an observation sequence, $P(G'|M)$ is calculated by means of the Forward algorithm [18]. By means of the maximum likelihood criterion, the instance that maximizes the probability of the observation sequence corresponds to the class (Fig. 7). The results obtained (incorrect and correct) on having repeated the previous process for all the observation sequences, are registered in a confusion matrix. Then the normalized rate (or accuracy) is given by:

$$\text{acc}(i) = \frac{\text{Number of observation sequences correctly classified}}{\text{Number of total observation sequences}} \tag{4}$$

The normalized complexity (by the maximum number of parameters) measure comp is given by the sum of the number of parameters of the model. The number of parameters of one model is obtained as follows:

$$\#\text{parameters} = \sum_{i=1}^{g} ||\text{Pa}(N_i)|| * (||N_i|| - 1) \tag{5}$$

where g is the number of nodes, including the hidden state node, $||\text{Pa}(N_i)||$ 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 $||\text{Pa}(N_i)|| = 1$. In (3) α defines a specific compromise between accuracy and complexity. Since these criteria are in

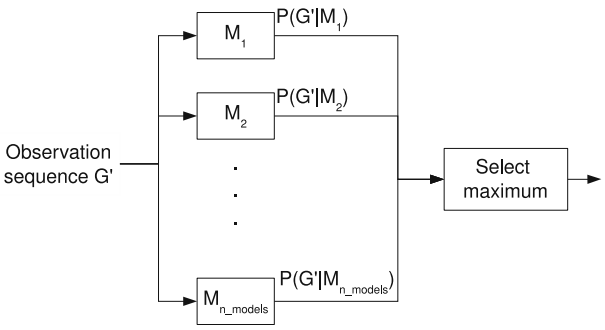

Fig. 7 Classification process. A observation sequence G' is presented to every model, followed by calculation of model likelihoods for all possible models, the instance that maximizes this likelihood corresponds to the class

conflict with each other the problem can be actually modeled as a multi-objective optimization problem.

4 Experimental Setup and Results

The proposed algorithm was evaluated in the visual recognition of nine hand gestures (Fig. 8): come, attention, right, left, stop, turn-right, turn-left, pointing and waving-hand; used for commanding mobile robots [2]. Each gesture is modeled using a DNBC considering seven attributes: three motion features and four posture features. These motion and posture features were obtained from a sequence of images. The motion features are: Δ_a, or changes in the hand area, Δx and Δy 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 leant 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 gestures in 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 [2]. We select $D_{training}$ samples per gesture to construct the complete training set, a partial testing data set $D\_\text{test}_{\text{partial}}$ is necessary to evaluate (compute the fitness) each one

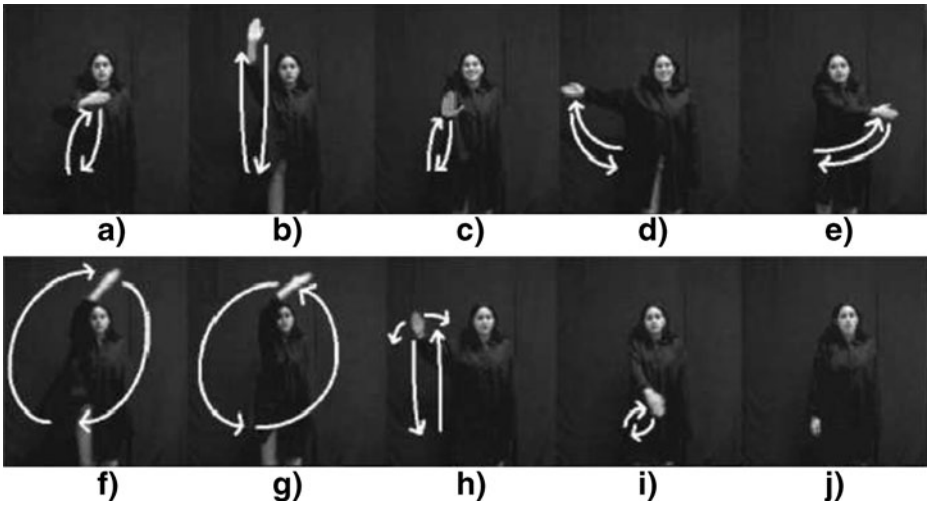

Fig. 8 Hand gestures considered **a** come, **b** attention, **c** right, **d** left, **e** stop, **f** turn-right, **g** turn-left, **h** waving-hand, **i** pointing and **j** initial position

of the individuals in the evolutionary process. Finally, we evaluate the classification accuracy of the best individual with the $D\_\text{test}_{\text{final}}$ 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 exhaustive statistical analysis is required to determine the best set of parameters. The differences among the experiments are as follows:

- Experiment 1. $D\_\text{training} = 10$, $D\_\text{test}_{\text{partial}} = 10$, $D\_test_{final} = 30$, $\alpha = 0.8$.
- Experiment 2. $D\_\text{training} = 10$, $D\_\text{test}_{\text{partial}} = 15$, $D\_test_{final} = 25$, $\alpha = 0.8$.
- Experiment 3. $D\_\text{training} = 10$, $D\_\text{test}_{\text{partial}} = 10$, $D\_test_{final} = 30$, $\alpha = 0.7$.
- Experiment 4. $D\_\text{training} = 10$, $D\_\text{test}_{\text{partial}} = 15$, $D\_test_{final} = 25$, $\alpha = 0.7$.

The EM algorithm with the same convergence criterion was used to estimate every instance of the DNBCs. All the models were set to follow a standard linear transition topology. Transition and observation probabilities for all the models in the population were initialized with discrete uniform distributions. The probability of each gesture sequence A, $P(A|.)$, was computed using the Forward algorithm [18]. All the experiments were carried out on a PC with AMD Athlon 1.8 GHz, 3 Gb of RAM, we used the Matlab software 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 ten samples, i.e. the algorithm is run ten times for each experiment. Table 2 shows the mean and standard deviation for the computation time.

Figure 9 shows the nine models that belong to the evolved classifier obtained in Experiment 2. The first model (come gesture) have five children nodes where variables F, Δ_x, A and T are independents given the hidden state variable C, variables Δ_y and Δ_a are associated in a single node, the hidden state variable has six states and R was eliminated. The second model (attention gesture) has four children nodes with variables Δ_x and Δ_a in the first group, Δ_y, F and A, in the second, R in the third and T in the last group, respectively, the hidden state variable has three states. In the *basic model* all the variables are considered in the model and are supposed to be independent of each other given the hidden state variable, the hidden state variable has two states. Then the basic classifier have nine models of this type. We can see that the proposed algorithm is able to learn a specific setting (variables association, variables elimination and specific number of states) for each model of the classifier.

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. Table 3 compares the recognition rates of the evolved models against a basic DNBC for each gesture. The recognition rates presented in Table 3 correspond to the classifier learned by the evolutionary process

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

Table 2 Mean computation time ($\bar{T}$) and standard deviation (STD) for ten runs

	$\bar{T}$ (min)	STD(T) (min)
Exp1	186.0742	43.5421
Exp2	215.7301	40.4538
Exp3	195.61	46.322
Exp4	206.3951	43.6830

presented in Fig. 9. The basic DNBC models consider two hidden states, and include all the attributes without any grouping, so they 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 hidden state variable is defined by the gesture in the evolutionary process.

The proposed algorithm gets a DNBC better than the basic classifier, regarding the average accuracy. To validate if this difference is statistically significant we carried out 30 training and test processes. We randomly selected ten samples per gesture to construct the complete training data set for computing the models parameters, and the remaining 40 samples for testing. We carried out this process with both classifiers (basic and evolved). Since the recognition rates do not follow a normal distribution, we used the Wilcoxon [4] rank significance test. The obtained p-value in the test was $p = 1.6031e^{-10}$. This result indicates that, with 99% of confidence, the

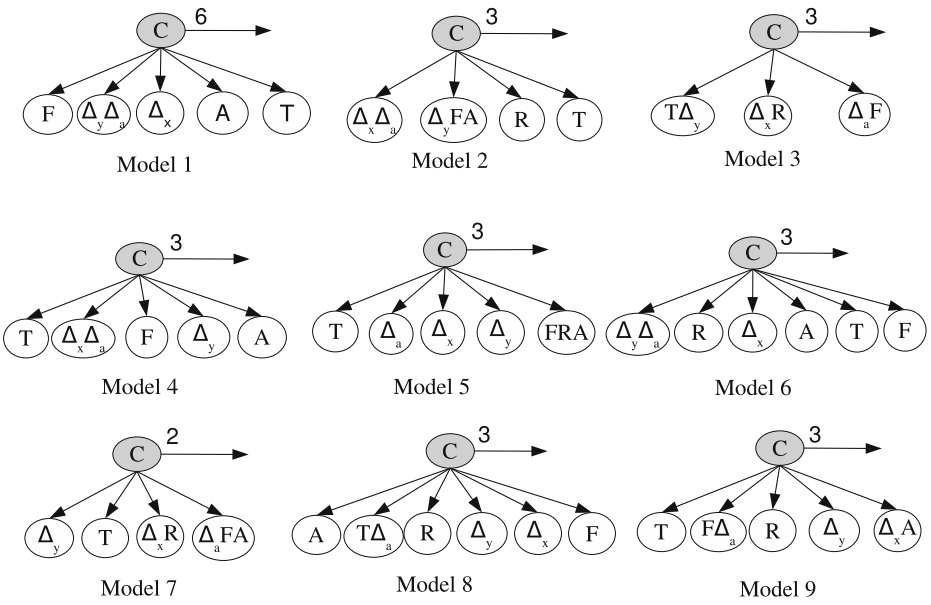

Fig. 9 Evolved Dynamic Naive Bayes Classifier for the nine gestures (come, attention, right, left, stop, turn-right, turn-left, pointing, and waving-hand). For each model the number of states, the selected attributes and their grouping are shown

Table 3 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	100
Right	100	100
Left	96	84
Stop	100	96
Turn-right	100	100
Turn-left	100	100
Pointing	88	96
Waving-hand	72	100
Average	94.67	97.33

mean classification rates obtained by the evolved DNBC and the basic classifier are different.

We adapted the algorithm to HMMs learning. In the representation the group part of the chromosome has only one group. The crossover operator (Step 7) only insert variables from one parent to the other. In mutation we have two options that are equally likely to be performed: Insertion or Deletion of one variable. Deletion of groups are not allowed. The mutation operator, for the number of states, parent selection, replacement and objective function are the same as those in DNBC learning. For comparison purposes, we considered for each gesture a HMM with all the attributes (grouped) and two states. Table 4 compares the preliminary recognition rates of the evolved HMMs against a HMM for each gesture and the same training and test samples used in Table 3. Figure 10 shows the nine HMMs that belong to the evolved classifier obtained. For example, the fifth model (left gesture) considered that Δ_y, Δ_a, F, A and T are associated in the observation node, the hidden state variable has three states, R and Δ_x was eliminated as opposed to the basic HMM that considered all variables grouped and two states.

We can see that the evolved HMMs is better than the clasic HMM, regarding the average accuracy. However, DNBC is better than the evolved HMMs. Further experiments are required to statistically assess the difference.

Given that the evolutionary learning process obtains not just a single model per gesture, but a population of models, we carried out an experiment in which we

Table 4 Gesture recognition rates using HMMs: the basic model vs. the evolved model

Gesture	Accuracy of the basic HMM (%)	Accuracy of the evolved HMM (%)
Come	100	97
Attention	100	100
Right	100	100
Left	80	97
Stop	80	100
Turn-right	100	100
Turn-left	100	100
Pointing	93	83
Waving-hand	100	93
Average	94.81	96.67

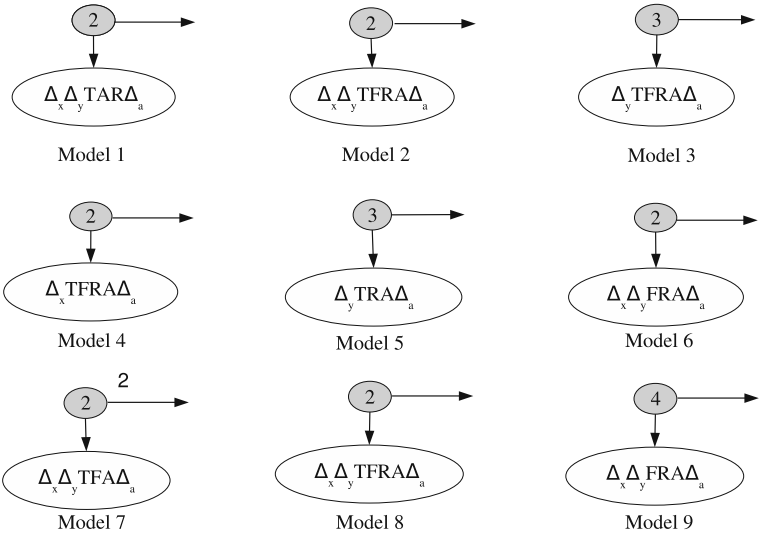

Fig. 10 Evolved HMM for the nine gestures (come, attention, right, left, stop, turn-right, turn-left, pointing, and waving-hand). For each model the number of states, the selected attributes and their grouping are shown

combine the top models per gesture. For recognition, all the models are combined via a majority vote to select the gesture. However, the results were no better than using a single model. We think that this is because all the models in the population at the end of the evolutionary process are very similar. As future work we plan to explore alternative evolutionary strategies; in particular it will be interesting to analyze this approach in a multi-objective setting since in that case the solutions in the non-dominated front will be diverse.

5 Conclusions and Future Work

An evolutionary approach to solve the structural learning problem to design a DNBC has been proposed considering as optimization criterion a compromise between accuracy and complexity. 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. The proposed method has been adapted to design HMMs, and could also have been modified to learn dynamic models that have a TAN as base structure.

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 as well as to analyze this approach in a multiobjective setting.

Acknowledgements The authors thank the anonymous referees for their comments and suggestions which help to improve the paper. The authors would also like to thank Héctor Avilés for his helpful advise in many stages of this research. The first and second authors' work were supported by CONACyT under grant C01-45811.

References

1. Aviles-Arriaga, H.H., Sucar, L.E., Mendoza, C.E., Vargas, B.: Visual recognition of gestures using dynamic naive bayesian classifiers. In: The 12th IEEE International Workshop on Robot and Human Interactive Communication, 2003, pp. 133–138. IEEE Computer Society, Washington, DC, USA (2003)
2. Aviles-Arriaga, H.H., Sucar, L.E., Mendoza, C.E.: Visual recognition of similar gestures. In: ICPR '06: Proceedings of the 18th International Conference on Pattern Recognition, pp. 1100–1103. IEEE Computer Society, Washington, DC, USA (2006)
3. Cameron, P.J.: Combinatorics: Topics, Techniques, Algorithms. Cambridge University Press, Cambridge (1994)
4. Devore, J.L.: Probability and Statistics for Engineering and the Sciences. Wadsworth, Belmont (1995)
5. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer, Germany (2003)
6. Falkenauer, E.: A new representation and operators for genetic algorithms applied to grouping problems. Evol. Comput. **2**(2), 123–144 (1994)
7. Friedman, N.: The bayesian structural em algorithm. In: Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence (UAI-98), pp. 129–138. Morgan Kaufmann, San Francisco, CA (1998)
8. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. Mach. Learn. **29**(2–3), 131–163 (1997)
9. Friedman, N., Murphy, K., Russell, S.: Learning the structure of dynamic probabilistic networks. In: Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence (UAI-98), pp. 139–147. Morgan Kaufmann, San Francisco, CA (1998)
10. Goldberg, D.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Massachusetts (1989)
11. Larrañaga, P., Poza, M.: Structure learning of bayesian networks by genetic algorithms: A. performance analysis of control parameters. IEEE J. Pattern Anal. Mach. Intell. **18**(9), 912–926 (1996) citeseer.ist.psu.edu/larranaga94structure.html
12. Martínez, M., Sucar, L.E.: Learning dynamic naive bayesian classifier. In: Florida Artificial Intelligence Research Symposium (FLAIRS–21), pp. 655–659. AAAI, Menlo Park, California (2008)
13. Martinez-Arroyo, M., Sucar, L.E.: Learning an optimal naive bayes classifier. In: ICPR '06: Proceedings of the 18th International Conference on Pattern Recognition, pp. 1236–1239. IEEE Computer Society, Washington, DC, USA (2006)
14. Myers, J.W., Laskey, K.B., DeJong, K.A.: Learning bayesian networks from incomplete data using evolutionary algorithms. In: Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., Smith, R.E. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference, vol. 1, pp. 458–465. Morgan Kaufmann, Orlando, Florida, USA (1999)
15. Palacios-Alonso, M.A., Brizuela, C.A., Sucar, L.: Evolutionary learning of dynamic naive bayesian classifiers. In: Florida Artificial Intelligence Research Symposium (FLAIRS–21), pp. 660–665. AAAI, Menlo Park, California (2008)
16. Pazzani, M.: Searching for dependencies in bayesian classifiers. In: Learning from Data: Artificial Intelligence and Statistics V., pp. 239–248. Springer, New York (1996)
17. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Francisco, CA, USA (1988)

18. Rabiner, L.R.: A tutorial on hidden markov models and selected applications in speech recognition. In: Readings in Speech Recognition, IEEE Proceedings, pp. 257–284. Morgan Kaufmann Publishers, San Francisco (1989)
19. Rechenberg, I.: Evolutionstrategie: Optimieriung Technisher Systeme Nach Prinzipien der Biologisten Evolution. Frommann-Holzboog, Stuggart, Germany (1973)
20. Ross, B.J., Zuviria, E.: Evolving dynamic bayesian networks with multi-objective genetic algorithms. Appl. Intell. **26**(1), 13–23 (2007)
21. Sucar, L.E., Gillies, D.F., Gillies, D.A.: Probabilistic reasonig in higth-level vision. Image Vis. Comput. **12**(1), 42–60 (1994)
22. Wong, M.L., Lee, S.Y., Leung, K.S.: 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, pp. 1314–1319. IEEE Computer Society, Washington, DC, USA (2002)