2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)
July 6-11, 2014, Beijing, China

Learning Fuzzy Rules through Ant Optimization, LASSO and Dirichlet Mixtures

A. Garcia-Garcia
CINVESTAV GDL
Zapopan, Jalisco, México
Email: aggarcia@gdl.cinvestav.mx

A. Mendez-Vazquez
CINVESTAV GDL
Zapopan, Jalisco, México
Email: amendez@gdl.cinvestav.mx

***Abstract*—In the area of fuzzy systems, one of the main problems is finding the set of rules that can give us the best results in specific problems. Further, the finding of this set is a combinatorial problem. There are several techniques for building these sets, but it is possible to group them in two main classes: The bottom-up approaches and the top-down approaches. This work proposes a new top-down approach to the fuzzy systems learning based in clustering and optimization techniques. The algorithm is split in two stages: First, it determines the fuzzy sets of each input and output linguistic variable, and second, it calculates the fuzzy rules from the obtained fuzzy sets. For the first part, a Dirichlet Mixture (DM) is used to cluster data to assign a fuzzy sets to each new cluster, since a fuzzy set can be seen as a generalized probability function, and hence the fuzzy sets of a given linguistic variable can be seen as a mixture of probabilities (a Gaussian Mixture). Then, an optimization problem is solved by using Ant Colony Optimization (ACO) to generate the minimum set of possible rules for classification by using a version of the Least Absolute Shrinkage and Selection Operator(LASSO) for the fitness function. This ACO was implemented in a CUDA GPU to deal with the combinatorial problem of rule generation. Finally, this new algorithm is used to attack the problem of color image segmentation.**

I. Introduction

THE theory of fuzzy logic, developed by Zadeh [1] in 1965, has been used in a lot of different areas as computer vision, decision making, control, among others. The main use of fuzzy logic in those areas is the development of fuzzy rule systems in order to take decisions about learning, selection and control. However, the main problem on these methods is the explosion of rules when experts try to generate these fuzzy rule systems. For this reason, the need of generating fuzzy set systems by minimizing the help of experts, and with a minimum size is a must. A lot of works has been developed to deal with this problem in different areas. They mostly use a series of interpolation methods, optimization techniques, clustering and bio-inspired methods in order to overcome the lack of an expert.

In order to understand this problem, the fuzzy sets in a linguistic variable can be seen as a partition of the linguistic variable in a series of classes. Based on this, it is possible to use an initial approximation using a Mixture Model [2] [3] to obtain the fuzzy sets of each linguistic variable. Here is where the Dirichlet Process [4] method can help to find the fuzzy sets. A Dirichlet Process [5] is a stochastic process using Bayesian non-parametric models of data. The Dirichlet process as a stochastic clustering method can be used to obtain the mixture distribution without the need of knowing something beforehand about the structure of the data [6] [7] [8]. This property allows to construct mixtures of data with certain properties. In the case of this work, these mixtures represent the possible fuzzy sets for a fuzzy system to do color image segmentation.

Xu et al. [9] propose two models to solve the evolutionary clustering problem based on the Dirichlet Process (DP). In this work, the authors try to solve the clustering problem where the data items evolves over time. Their Dirichlet Process Mixture Chain [6] (DPChain) model is based in a Dirichlet Process Mixture [8] (DPM) model and Markov Chain Monte Carlo Model [10] (MCMC) method. Additionally, they propose a second method called Hierarchical Dirichlet Process Evolutionary Clustering Model [6] (HDP-EVO), which is inspired in the Hierarchical Dirichlet Process [11] (HDP) with the capacity of evolve over time. In another example Zare et al. [12] use the DP in hyper-spectral images to find the number of endmembers. Their method is a bottom-up algorithm, where at initialization, there is only one component, and new components are added when needed. It assumes that each pixel of the image is a combination of the endmembers of the image, i.e. each pixel has an influence on every endmember. It is based on the particular DPM case named Chinese Restaurant Process (CRP). A limitation on this work is that does not estimate some of parameters that need to be used in the algorithm. In other example Bouguila et al. [7] propose a no supervised algorithm to learn a finite mixture model of multivariate data. Their hypothesis is that the vector $\vec{x} = (x_1, \cdots, x_d)$ follows a Dirichlet distribution, and the samples comes from a Dirichlet Mixture Model (DMM). With this in mind, they use a Maximum Likelihood (ML) algorithm [13] [14] to estimate the number of components M and the parameters of each Dirichlet component. This method is top-down because it needs an overestimated number of components. Then, the algorithm decreases the number of components until the parameters of each component do not vary.

All these works propose a solution for the clustering problem using a DPM. This can help to make a first approach for the generation of fuzzy sets of each linguistic variable as mixture models. Next, it is necessary to have a way to generate the set of fuzzy rules for the system. This generation of rules is constrained by the need to have as few rules as possible, which is a desirable characteristic to minimize noise by the system. For example, Casillas et al. [15] introduce a learning system based in the ACO to learn the fuzzy rules.

978-1-4799-2072-3/14/$31.00 ©2014 IEEE

In order to learn these fuzzy rules, the authors map the problem to a Quadratic Assignment Problem (QAP). The main weakness of this method is that it requires the fuzzy sets before the execution of the algorithm. This can be overcome by using the previous ideas about linguistic variables. Another method is proposed by Chia-Feng et al. [16]. Their method is an auto-generated fuzzy system with learning abilities. For this, the authors used an On-Line Self- Aligning Clustering (OSAC) to create the fuzzy sets, and an Ant and Particle Swarm Cooperative Optimization (APSCO) to obtain the fuzzy rules. In their work, they try to learn a control equation to make a circle. Bellaaj et al. [17] propose a method for fuzzy rule base reduction using similarity concepts and interpolation techniques. The method begins with an initial rule base, then it computes the similarity between rules. Next, it deletes rules with a high similarity to the accepted rule base. Further, when new data is presented, the system is able to produce new rules using a simple method: If there is a minimum number of fire rules, then add a new one using an interpolation method. A drawback of the method is its sensitivity to outliers.

Important observations to be made are that all these techniques, for fuzzy rule generation, are very specific for each of the studied problems. In addition, most of them use fuzzy sets provided in a prior way by experts and their data set inputs are quite small. These are reasons why this work proposes the use of Dirichlet Mixture for automatic fuzzy set generation and the ACO optimization under LASSO [18] for rule set minimization as a way to try to automatize the generation of the fuzzy rule system.

Thus, this work is divided in the following sections: In section II, a basic review of the theory behind the proposed algorithm is given, from Dirichlet Mixtures to Mamdani fuzzy inference. In section III, an exposition of the proposed algorithm is done, from the generation of the fuzzy sets to the minimization of the rule base. Section IV, shows the experimental results for the generated Segmentation Fuzzy System by the proposed algorithm. Finally, in section V some conclusion of the work are given.

II. Basic Theory

A. Dirichlet Mixture for Clustering

The use of the Dirichlet distribution is an excellent option to model data [7] when not underlaying model exists. In our specific case, it is important for analyzing data clustering for automatic fuzzy set generation.

The Dirichlet Mixture consists of M Dirichlet components. Each of the j-th components has a parameters vector $\vec{\alpha}_j$ that defines the Dirichlet distribution, and an associated mixture proportion $P(j)$ $(0 < P(j) < 1)$. The sum of all the mixture proportion is equal to one $(\sum_{j=1}^{M} P(j) = 1)$. Finally, a Dirichlet mixture with M components is defined in (Eq. 1):

$$p(\vec{x}|\Theta) = \sum_{j=1}^{M} p(\vec{x}|j,\Theta_j)P(j), \tag{1}$$

where $p(\vec{x}|j,\Theta_j)$ is the Dirichlet distribution, which is shown in (Eq. 2), and $\Theta_j = (\vec{\alpha}_j)$.

$$\begin{aligned} p(\vec{x}|j,\Theta_j) &= p(x_1,\cdots,x_d|\alpha_{j,1},\cdots,\alpha_{j,d+1}) \\ &= \frac{\Gamma(|\vec{\alpha}_j|)}{\prod_{i=1}^{d+1}\Gamma(\alpha_{j,i})}\prod_{i=1}^{d+1} {x_i}^{\alpha_i-1} \end{aligned} \tag{2}$$

where

$$\sum_{i=1}^{d} x_i < 1 \tag{3}$$

$$|\vec{x}| = \sum_{i=1}^{d} x_i, \quad 0 < x_i < 1 \quad \forall i = 1\cdots d \tag{4}$$

$$x_{d+1} = 1 - |\vec{x}| \tag{5}$$

$$|\vec{\alpha}_j| = \sum_{i=1}^{d+1} \alpha_{j,i},\ \alpha_{j,i} > 0\ \forall j = 1\cdots M, \forall i = 1\cdots d+1 \tag{6}$$

$$\Theta = (\vec{\alpha}_1,\cdots,\vec{\alpha}_M, P(1),\cdots,P(M)). \tag{7}$$

The problem is to determine Θ. This is done by using Maximum Likelihood (ML) estimation (Eq. 8).

$$\widehat{\Theta} = \max_{\Theta} p(\vec{X}|\Theta) \tag{8}$$

Thus, using the Lagrange Multipliers the function becomes (Eq. 9):

$$\begin{aligned} \Phi(\vec{\mathbf{x}},\Theta,\Lambda) = &\sum_{i=1}^{N} \ln\big(\sum_{j=1}^{M} p(\vec{x_i}|j,\Theta_j)P(i)\big) + \\ &\Lambda\big(1 - \sum_{j=1}^{M} P(j)\big) + \\ &\mu\sum_{j=1}^{M} P(j)\ln(P(j)), \end{aligned} \tag{9}$$

where Λ is the Lagrange multiplier, $\sum_{j=1}^{M} P(j)\ln(P(j)$ is an entropy criteria used in Gaussian Mixture cases [2] [19], and μ is the compensation between data likelihood and the number of components.

The estimation of the $\vec{\alpha}$ parameters is done by the Fisher Scoring Method (FSM). We use $\alpha_{j,\ell} = e^{\beta_{j,\ell}}$ to hold the restriction over α. The iterative scheme is given by (Eq. 10):

$$\widehat{\boldsymbol{\beta}}^{t} = \widehat{\boldsymbol{\beta}}^{t-1} + \mathbf{C} \times \boldsymbol{\Upsilon}^{t-1}, \tag{10}$$

where j is the class number, $\widehat{\boldsymbol{\beta}}^{t-1}$ is the vector $[\hat{\beta}_{j,1}^{t-1},\cdots,\hat{\beta}_{j,k+1}^{t-1}]^T$, $\mathbf{C}$ is the variance-covariance matrix, and $\boldsymbol{\Upsilon}^{t-1}$ is the vector $\Big[\frac{\partial}{\partial\hat{\beta}_{j,1}}\Phi^{t-1},\cdots,\frac{\partial}{\partial\hat{\beta}_{j,k+1}}\Phi^{t-1}\Big]^T$. The variance-covariance matrix is obtained as the inverse of the Fisher information matrix $\mathbf{F}$ defined as:

$$\mathbf{F} = I_{\ell_1,\ell_2} = -E\big[\frac{\partial^2}{\partial\beta_{j,\ell_1}\partial\beta_{j,\ell_2}}\Phi(\vec{\mathbf{x}},\Theta,\Lambda)\big]. \tag{11}$$

Now, this work proposes a variation of the Dirichlet Mixture Estimation (DME) algorithm [7] used in the estimation of the

mixture values. This variation consists on the update of the P_j values by (Eq. 12):

$$P(j)^t = \frac{N_j^{t-1}}{N}, \tag{12}$$

where N_j^{t-1} is the number of elements belonging to the j-th cluster, N is the total number of data. This idea comes from the Chinese restaurant process [20] [21]. This updating allows to control the convergence of the information matrix used in the DME under the case studies considered for this work.

B. Ant Colony Optimization

The ACO is an algorithm inspired in nature [22] [23], and it takes its inspiration on the behavior of the ants when they are searching for food. For example, each ant leaves a pheromone trail over the path used by that ant to reach the food. Thus, other ants may choose the same path, or make a new path to the food. At the end, the path with a highest quantity of pheromones is a good or optimal path to the food [23]. In graph theory, this is related with the problem of finding the shortest path between two nodes.

An ant is a tabu list which records the vertices visited and leaves a quantity of pheromone over the edges. When the goal vertex is in the tabu list the algorithm evaluate the path of the ant using a fitness function defined for each particular problem. An intensity of trail matrix τ is used to keep the pheromones left by the ant. In addition, a evaporation coefficient ρ is defined to determine which amount of pheromones is removed from each path.

For the evaluation of the path of each ant is necessary to know the cost between two nodes, and this is kept in the cost matrix δ. Thus, it is necessary to have a visibility matrix η that help to define the transition probability which is used by the ant to select to which vertex it should go. This is shown in (Eq. 13):

$$p_{i,j}(t) = \frac{(\tau_{ij}(t))^{\alpha}(\eta_{ij})^{\beta}}{\sum_{k=0}^{N}(\tau_{ik}(t))^{\alpha}(\eta_{ik})^{\beta}} \tag{13}$$

where $p_{i,j}(t)$ is the transition probability in the t-th iteration to go form vertex i to a vertex j, α and β are parameters that allow a user control on the relative importance of trail versus visibility.

The fitness function is generally a function to be maximized or minimized, and its definition is entirely dependent of the problem. For this, an ant keeps a path which is a possible optimal solution for the problem, thus the need of a high number of ants to have a complete exploration of the solution space. In order to accomplish this, each ant walks the graph independently of the others ants. This makes the ACO a highly parallelizable algorithm. This parallel feature of the ACO allows for a clean implementation of the algorithm on GPUs through the use of CUDA [24] [25]. Finally, in (Algorithm 1), it is possible to see the basic ACO algorithm.

Algorithm 1 ACO algorithm

1: $\rho \leftarrow initEvapCoefficinet()$ // Initialize the evaporation coefficient
2: $t \leftarrow 0$
3: $\tau^t \leftarrow initTrailIntensity()$
4: $b^t \leftarrow initAntPlace()$
5: **for** $r, s \in V$ **do**
6: $\Delta\tau^t(r,s) \leftarrow 0$
7: Repeat until tabu list is full:
8: // r is the actual vertex and s is the possible next vertex to move on.
9: **for** $i \leftarrow 1$ to n **do**
10: // for every vertex
11: **for** $k \leftarrow 1$ to $b^t(i)$ **do**
12: Choose the vertex to move to with $aggTransProb\_rs(r,s)$
13: Move the k-th ant to the chosen vertex (in $b^{t+1}(s)$)
14: $\Delta\tau_k^{t,t+1}(r,s) \leftarrow aggDTau\_rs(r,s,k,M\_T)$
15: $\Delta\tau(r,s)^{t,t+1} \leftarrow \Delta\tau_k^{t,t+1}(r,s) + \Delta\tau(r,s)^{t,t+1}$
16: **end for**
17: **end for**
18: **end for**
19: **for** $r, s \in V$ **do**
20: $\tau^{t+1}(r,s) \leftarrow \rho\tau^t(r,s) + \Delta\tau^{t,t+1}(r,s)$
21: $P^{t+1}(r,s) \leftarrow aggTransPob_rs(r,s)$
22: **end for**
23: Memorize the shortest path (or the path with the lowest cost) and empty all tabu list
24: **if** not end test **then**
25: $t \leftarrow t+1$
26: $\forall r, s \in V$ do $\Delta\tau^t(r,s) \leftarrow 0$
27: go to line 5
28: **else**
29: return the shortest path
30: **end if**

C. Mamdani Fuzzy Inference

The Mamdani Fuzzy Inference was proposed by Mamdani et al. [26], in 1975, to control a steam engine and a boiler combination by a set of linguistic control rules obtained from experienced human operators. A Mamdani fuzzy rule of two input variables and two output variables can be described by (Eq. 14):

$$\text{IF } x_1 \text{ is } C_1 \text{ AND } x_2 \text{ is } C_2 \text{ THEN } y_1 \text{ is } B_1,\ y_2 \text{ is } B_2 \tag{14}$$

where x_1 and x_2 are the input variables, and y_1 and y_2 are the output variables. In addition, C_1 and C_2 are input fuzzy sets, and B_1 and B_2 are the output fuzzy sets. In order to operate over these fuzzy sets AND/$\min$ (T-norm) and OR/$\max$ (T-conorm) operators are used. Thus, "IF x_1 is C_1 AND x_2 is C_2" is called the rule antecedent, whereas the remaining part is named th rule consequent.

In (Fig. 1), it is possible to see the operations used by the Mamdani Fuzzy Inference. For example, $\mu(x_i)$ is the membership value given by x_i when is evaluated in one of the membership function for the input fuzzy variable $\mathbf{x}_i$. Thus, $\mu(y_1)$ is the membership value for the output fuzzy variable $\mathbf{y}_1$. B is the fuzzy set obtained by the inference process. The defuzzification is the way a crisp value is obtained from the fuzzy set B. The centroid method [27] is one of the most used methods for defuzzification.

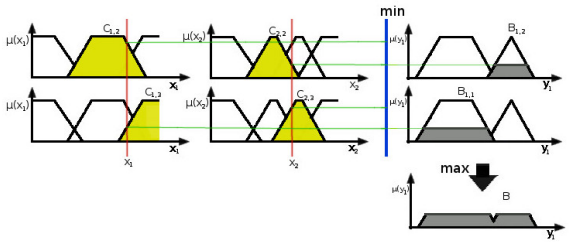

Fig. 1: Mamdani Fuzzy Inference

Finally, two methods of defuzzification are used in this work, the centroid method, and the index with the maximum value (IMV) (Eq. 15):

$$y_i = \arg\max_k(\vec{y}), \tag{15}$$

where $\vec{y} = (y'_1, \cdots, y'_p)$ is the vector of the fuzzy values for the p output fuzzy set given by the inference engine, and y_i is the output of the system.

III. Learning Fuzzy Systems

This section shows the basics of the proposed algorithm. For this, the fuzzy system algorithm is divided in two stages:

1) In the first stage, the fuzzy sets are obtained for each input and output linguistic variables through the use of the Dirichlet Mixtures.
2) In the second stage, the proposed algorithm builds a first approximation of the Rule Base. Then, it minimizes the set of rules by using an ACO algorithm.

A. Creating the Fuzzy Sets

A fuzzy linguistic variable can be seen as a Mixture of probabilities. Thus, it is possible to use Dirichlet Mixture estimation to automatically obtain the mixture model. This can be done by transforming the Dirichlet Distributions into Gaussian distributions using (Eq.16):

$$Beta(\alpha_1, \alpha_2) \approx N\left(\frac{\alpha_1}{\alpha_1+\alpha_2}, \sqrt{\frac{\alpha_1\alpha_2}{(\alpha_1+\alpha_2)^2(\alpha_1+\alpha_2+1)}}\right). \tag{16}$$

In addition, a Dirichlet distribution of two variables is a beta distribution, making possible to approximate the Gaussian if it satisfies $\frac{\alpha_1+1}{\alpha_1-1} \approx 1$ and $\frac{\alpha_2+1}{\alpha_2-1} \approx 1$.

For this, the algorithms 2 and 3 are used to obtain the Dirichlet Mixture for each color component and for the image I. This is done in the following way: First, the Algorithm 2 initialize a first approximation for the Dirichlet mixture for each component and for the complete image. Second, the Algorithm 3 is launched for each component and for the complete image to obtain the Dirichlet mixture. Finally, the number of fuzzy sets and the parameter of each fuzzy set of each linguistic variable is taken from the Dirichlet mixture.

In order to obtain the parameters of each fuzzy set of each component of the input from the Dirichlet mixture, it is necessary to assign a mean and a variance to get each Gaussian membership function. Thus, the mean is simply the number of the cluster (i.e. $\mu_i = i$) generated by the DME. Finally, for each fuzzy set in the input linguistic variables, and the variance is obtained using an approximation by (Eq. 16).

Algorithm 2 Initialization Algorithm

1: $\vec{X}$
2: $M \leftarrow$ number of components
3: Apply the fuzzy C-means to obtain the elements, covariance matrix and mean of each component
4: Apply the MM (Method of Moments) for each component j to obtain the vector of parameters $\vec{\alpha_j}$
5: Assign the data clusters, assuming that the current model is correct
6: Update the $P(j)$ using the following:
7: $P(j) = \frac{\#OfElementsInClassj}{N}$
8: If the current model and the new model are sufficiently close to each other, terminate, else go to 4
9: Return: Vector of parameters $\vec{\alpha}$

Algorithm 3 DME Algorithm

1: $X_i \leftarrow$ Dimensional data, $i = 1, \cdots, N$
2: $M \leftarrow$ number of components
3: INITIALIZATION$(\vec{X}, M)$
4: Update the $\vec{\alpha_j}$ using(10)
5: Update the $P(j)$ using (12)
6: **if** $P(j) < \epsilon$ **then**
7: discard component j, and go to 4
8: **end if**
9: **if** Convergence test is passed **then**
10: terminate, else go to 4
11: **end if**
12: Vector of parameters $\vec{\alpha_j}$ of the cluster j and the number of clusters M

The lack of information about the output fuzzy sets makes difficult to select parameter for these outputs. The fuzzy sets from a n-dimensional data given by a n-dimensional Dirichlet mixture data, with $n \geq 2$, are actually really hard to map. To obtain the i-th fuzzy set of an output variable, it is necessary the conversion of the i-th Dirichlet component of the i-th cluster to a one dimensional Gaussian membership function. In order to do this this work proposes two ways: In the first one, the mean is the number of the cluster (i.e. $\mu_i = i$), and the variance is simply a constant value a ($\sigma_i = a$). In the second, the mean is the number of the cluster (i.e. $\mu_i = i$) and a variable variance given by (Eq. 17):

$$\sigma_i = \frac{\sum_{j=1}^{N_m} \sigma_{i,j}}{N_m}, \tag{17}$$

where $\sigma_{i,j}$ is the mean of the j-th marginal of the i-th cluster, and N_m is the number of marginals.

It is more, the Beta distribution can be approximated to a Gaussian distribution (Eq. 16), and the Dirichlet marginals are

Beta distributions (Eq. 18):

$$X_i \sim Beta(\alpha_i, \sum_{j=1}^{d} \alpha_j - \alpha_i), \tag{18}$$

where α_i are the α parameters of the Dirichlet distribution. Thus, to obtain the means and the variances, equations (Eq. 16) and (Eq. 18) are used.

B. Creating the Base Rules

The initial combinatorial part of the algorithm is used to create the first rule base approximation. Thus, for each training data do the following:

- First build the fuzzification of the input variables and obtain the antecedent fuzzy sets by looking at which sets were activated through the use of a threshold.
- Then, makes the fuzzification of the output variables by the Dirichlet Cluster algorithm to obtain a consequent by the same procedure.

This allows to generate the minimal possible activation antecedent and consequent sets. Finally, combinations of these two groups are done to obtain a preliminary rule set. This first approximation is denoted like RB_0.

A modified ACO is used to minimize this first approximation, RB_0, to the rule base [15]. In the proposed algorithm, the ACO uses a new version of the fitness function, (Eq. 20), using a LASSO [18] regularization to obtain a minimal set of rules by the use of a sparsity component. This fitness function is a modified Means Square Error (MSE) with a L1-Norm regularization term:

$$\arg\min_{\beta} \sum_{i=1}^{N} ||y_i - X\beta|| + \lambda||\beta||_1 \tag{19}$$

For this, a modified version of the LASSO is proposed (Eq. 19) in this work:

$$MSE(RB_i) = \frac{1}{2 \cdot |E|} \sum_{e_k \in E} (y^k - F_i(\vec{x}^k))^2 + \lambda \sum_{i=1}^{N_R} |\delta(i)|\psi(i), \tag{20}$$

where λ is the tuning parameter of the LASSO, $\psi(i)$ indicates if the i-th rule was activated, $\delta(i)$ is the weight of the activated rule, e_k is an element of the training data ($e_k = (\vec{x}^k, y^k)$) and E is the training data set. Basically, this new version uses the weight matrix of the ACO as the β parameters to be found, and the $X\beta$ as the output of the Fuzzy System for the given input X. Finally, the $F_i(\vec{x}^k)$ is the output obtained by the Fuzzy Rules Based System (FRBS), which is used by the Rule Base (RB) generated by the ants using the input $\vec{x}^k$.

Next, the rules are formed by an antecedent and a consequent. Thus, the rules on the rule base can be seen as a bipartite graph. In which one node set represents the set of antecedents, and the other the set of the consequents (Fig. 2).

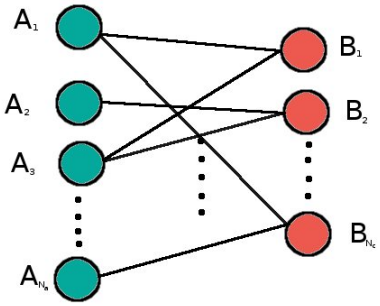

Fig. 2: Bipartite graph for the rule base

Therefore, the tabu list is a set of rules that have an edge in the bipartite graph. The tabu list of each ant (RB_i, i-th ant) has a subset of the first rule base (RB_0), i.e. $RB_i \subset RB_0$. The initial pheromone value is given by (Eq. 21) and (Eq. 22):

$$\eta_i = \max_{j=1,\cdots,N_c} \eta_{ij} \tag{21}$$

$$\tau_0 = \frac{\sum_{i=1}^{N_a} \eta_i}{N_a}, \tag{22}$$

where:

$$\eta_{ij} = \max_{e_k \in E} \Big(\min \left(\mu_{C_{i,1}}(x_1^k), \cdots, \mu_{C_{i,d}}(x_d^k), \mu_{B_j}(y^k)\right)\Big) \tag{23}$$

with $\mu_{C_{i,j}}$ as the membership function of the j-th input variable of the i-th antecedent (with $j = 1, \cdots, d$), μ_{B_j} as the membership function of the j-th consequent, and $(x_1^k, \cdots, x_d^k, y^k)$ as the k-th data training vector.

C. Proposed Algorithm

The final proposed algorithm is shown in (Algorithm 4). First, the algorithm takes an image and decomposes it in its color components (HSV) normalizing both the components and the image. Then, for each color component X_c a Dirichlet Mixture cluster is obtained to generate the antecedent fuzzy sets. Next, using the segmentation of entire Image I by DME, it is possible to obtain the consequent fuzzy sets. Thus, the calculation of the input fuzzy sets and output fuzzy sets for each linguistic variable is done by the conversion of the Dirichlet distribution to an a Gaussian approximation given by (Eq. 16), and the two ways proposed in the Section III-A. Further, the first approximation of the knowledge Base is done by a combinatorial approach explained in the Section III-B. Finally, the ACO algorithm is run to minimize the number of rules.

The ACO algorithm was proposed because its high degree of parallelization when generating the rule system. Moreover, the restriction on dynamic memory at the CUDA kernels limits other types of algorithms when creating new elements during execution time.

A parallel version of the proposed ACO algorithm was implemented using NVIDIA [28] cards and CUDA. The algorithm of the ant is splitted in four main sequential processes:

Algorithm 4 Proposed Algorithm

$I \leftarrow Image$
$X_{c_{h,s,v}} \leftarrow I_{c_{h,s,v}}$
Normalize $I, X_{c_{h,s,v}}$
for c in $\{c_h, c_s, c_v\}$ **do**
  Obtain the DME of X_c
  $DME_c \leftarrow CalculateDME(X_c)$
end for
Obtain the DME of I by DME
$DME_I \leftarrow CalculateDME(I)$
Obtain the Fuzzy Sets
$FZ_{IN} \leftarrow CalculateFuzzySets(DME_{r,g,b})$
$FZ_{OUT} \leftarrow CalculateFuzzySets(DME_I)$
$X_T \leftarrow DataTraining(I, X_{r,g,b}, DME_I, DME_{r,g,b})$
$KB_0 \leftarrow Combinatorial(X_T, FZ_{IN}, FZ_{OUT})$
Minimize the number of rules in KB_0 by ACO algorithm
$KB_{BEST} \leftarrow ACO(X_T, FZ_{IN}, FZ_{OUT}, KB_0)$
Output Fuzzy System: $FS(FZ_{IN}, FZ_{OUT}, KM_{BEST})$

First, the launching the ants to make a path; second, the evaluation of the paths found by the ants; third, the updating of the pheromones' path which were found by the ants; finally, the update process of the principal matrix (τ matrix and the transition probability matrix). Each of these processes is done in parallel.

In the CUDA implementation a kernel [24] [25] represents an ant traveling through the bipartite antecedent-consequent graph, and its execution finishes when the goal is reached. This goal is basically to store a certain given number of rules kept in the tabu list. In the evaluation process, each kernel evaluates the rule base stored by the ant with the data training by using the (Eq. 20). In the updating pheromone process and in the matrix update process, each kernel updates a part of the pheromones τ, and the transition probability matrices if the number of rows in the matrices are greater than a maximum fixed number of kernels.

IV. Experimental Results

In this section, the proposed algorithm is used to generate a Fuzzy System capable of segmenting an image as the DME does. The algorithm is tested over a data set of images from the Berkeley Segmentation Dataset and Benchmark [29] (BSDB), and the best results are shown in this work.

Because of the consequent problem, which was discussed early, this work takes in account two different ways of generating these consequents. The first one uses the index of the cluster given by the DME as a mean of the fuzzy set and it uses a fixed $\sigma = 0.7$. The second one uses the index of the cluster given by the DME, and a mean of the marginals (Eq. 26).

Taking in account that the marginals of a d-dimensional Dirichlet distribution (Eq. 18), and the approximation of the Beta distribution to a Gaussian distribution [30] (Eq. 16). The variance of a fuzzy set output is given by (Eq. 24):

$$\sigma' = var[Y] = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)} \tag{24}$$

$$Y \sim Beta(\alpha, \beta) \tag{25}$$

$$\sigma_k = \rho\frac{\sigma'_1 + \sigma'_2 + \sigma'_3}{3}, \tag{26}$$

ρ is a normalization factor.

The number of ants used was 256 per iteration in each experiment. Fort this, the ants were divided into eight groups of 32 ants. Each group have a different size for the tabu list depending of the initial number of rules. For the four experiments in this works, the sizes of the tabu list are shown in (Table I).

In this work, the ant selects his elements in the tabu list in two different ways: The first one is called All in One Iteration ($A1I$). For each of the iteration, each ant selects from all the rules (in RB_0) a subset in which the output fuzzy set was activated at least once. The second is called All Iterations for One ($AI1$): For each of the consequents, in each of the iteration, each ant select a subset from the initial rule set with the restriction that each rule have the same consequent, i.e. in each iteration the algorithm try to learn the minimum number of rules for a given consequent. Finally, (Table II) shows the methods used for the defuzzification, for the selection of the elements in the tabu list, and for the methods used to obtain the variance σ over the tests. Thus, after running the experiments, it is possible to see in (Table III) some of the results. The table is explained in the following way: The first column corresponds to a test number; the second corresponds to the initial number of rules in the rule base RB_0; the third one is the the initial MSE; the fourth one is the execution time to learn the new fuzzy system; the fifth one is the the number of misclassified pixels. Finally, from the sixth to the ninth column, the same metrics after execution of the ACO algorithm.

	No. Group							
Test	1	2	3	4	5	6	7	8
01	8	16	20	32	40	64	80	128
03	8	12	16	25	32	45	64	90
08	2	3	4	6	8	13	16	20
12	2	3	4	6	8	13	16	20
18	2	3	4	6	8	13	16	20

TABLE I: Number of ants used by group.

Test	Defuzzification method	Selection method	σ selection
01	Centroid	$A1I$	Variable
03	Centroid	$A1I$	Variable
08	Centroid	$AI1$	Fixed
12	IMV	$AI1$	Variable
18	Centroid	$AI1$	Variable

TABLE II: Number of ants used by group.

Test	Initial #Rules	MSE	Time (s) Time (s)	No Classified	Final #Rules	MSE	Time (s)	No Classified
01	216	1.6387	8.2100	0	8	3.1682	0.3300	14257
03	360	2.3451	11.590	0	12	4.6914	0.5700	18774
08	353	1.2773	11.850	0	47	2.4780	1.7200	2001
12	182	2.6500	12.380	0	27	2.3875	1.7300	45
18	178	1.2240	13.340	0	22	0.5887	1.7600	0

TABLE III: Number of rules on the KB before and after the ACO algorithm.

The qualitative results of (Table III) can be seen at the following figures. First, in (Fig. 3), there are the results for the test 01: The upper image of the left is the image input; the upper image on the right is the image in HSV color space; the lower image on the left is the segmented image using DME; the lower image on the middle gives the segmentation given by the fuzzy system with the initial rule base (RB_0); finally, the lower image on the right is the segmented image by the fuzzy system learned (with the minimized rule base by the ACO algorithm). In the same way in (Fig. 4), (5), (6), and (7), there are the results of the experiments 03, 08, 12, and 18, respectively.

It is noticeable how the results show a qualitative improvement, when a comparison is made between the segmentation done by the Fuzzy System with a huge rule base, and the segmentation done for the Fuzzy System with the rule base is minimized by the ACO under LASSO.

The noise added to the fuzzy system by the initial rule base (RB_0) was reduced for the fuzzy system with the minimized base rule. The use of the selection method $A1I$ gives a less rule reduction, and a high execution time compared to the use of the selection method $AI1$. However, the number of pixels not classified decreases drastically using $AI1$. In the same way, it is possible to observe a decrement on the error values for the tests using the selection method $AI1$.

V. Conclusion

This work presents a novel algorithm for Fuzzy System Learning, using a Dirichlet Mixture estimation to determine the Fuzzy Sets of the input linguistic variables and the number of sets of the output linguistic variable. Due to the structure of proposed algorithm, several parts of it can be easily implemented under parallelism. Specifically, the ACO was implemented using the massive computational power of the NVIDIA GPU to speed up its execution. In addition, the use of the LASSO in the fitness function allows for an efficient pruning of the rule base in order to improve classification.

Still, there is a lot of work left to be done in order to to improve the proposed algorithm. For example, the study of the type of defuzzification method is an important work that could allow to have better approximations of the desired fuzzy system. Additionally, the restrictions imposed on the possible structures of the rules by the Mamdani fuzzy system, for the base rule, tend to limit the results. Thus, it is necessary to implement a better method to generate new rules that are not in the initial rule base. For example, the way the fuzzy sets are obtained implies a limitation in the granularity of these sets in a given linguistic variable. Thus, it is necessary new methods that allow the generation of new rules by taking into account an initial rule base.

VI. Acknowledgment

The authors would like to thank to the "Consejo Nacional de Ciencia y Tecnología" (CONACyT) for their financial support in the development of this work (number of scholarship 243193).

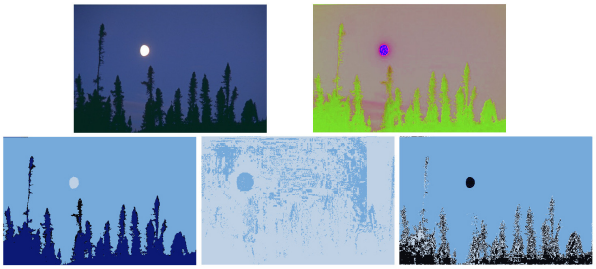

Fig. 3: Results of the expriment 01. The original image is in the upper left corner.

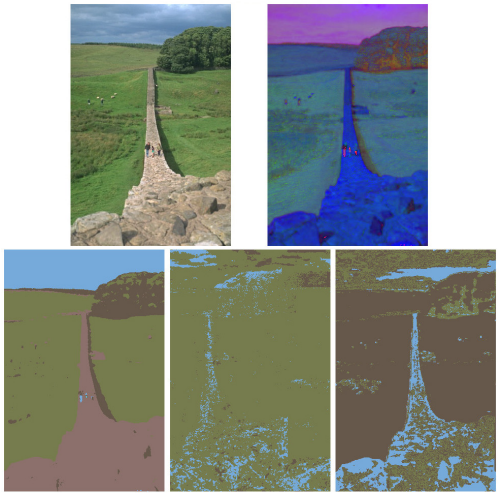

Fig. 4: Results of the experiment 03.

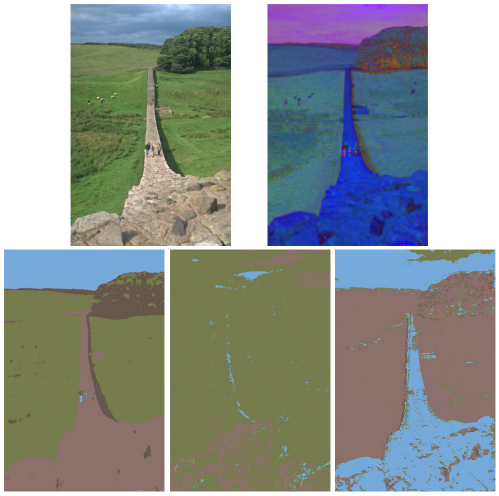

Fig. 5: Results of the experiment 08.

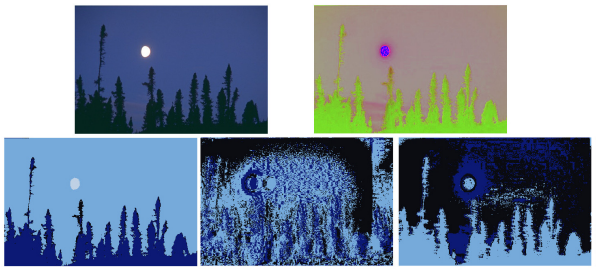

Fig. 6: Results of the experiment 12.

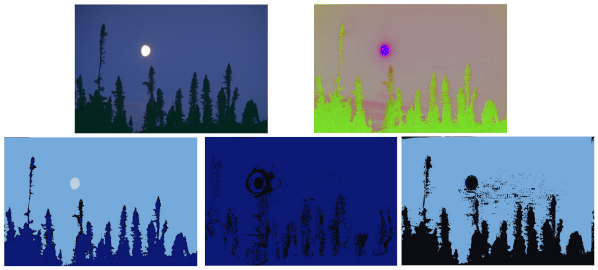

Fig. 7: Results of the experiment 18.

REFERENCES

[1] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, no. 3, pp. 338–353, 1965.

[2] Z. R. Yang and M. Zwolinski, "Mutual information theory for adaptive mixture models," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, no. 4, pp. 396–403, 2001.

[3] M. A. T. Figueiredo and A. Jain, "Unsupervised learning of finite mixture models," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 3, pp. 381–396, 2002.

[4] T. S. Ferguson, "A Bayesian Analysis of Some Nonparametric Problems," *The Annals of Statistics*, vol. 1, no. 2, pp. 209–230, 1973. [Online]. Available: http://dx.doi.org/10.2307/2958008

[5] Y. W. Teh, "A tutorial on dirichlet processes and hierarchical dirichlet processes." Gatsby Computational Neuroscience Unit University College London, mar 2007.

[6] T. Xu, Z. Zhang, P. Yu, and B. Long, "Dirichlet process based evolutionary clustering," in *Data Mining, 2008. ICDM '08. Eighth IEEE International Conference on*, 2008, pp. 648–657.

[7] N. Bouguila, D. Ziou, and J. Vaillancourt, "Unsupervised learning of a finite mixture model based on the dirichlet distribution and its application," *Image Processing, IEEE Transactions on*, vol. 13, no. 11, pp. 1533 –1543, nov. 2004.

[8] J. Paisley and L. Carin, "Dirichlet process mixture models with multiple modalities," in *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, 2009, pp. 1613–1616.

[9] T. Xu, Z. M. Zhang, P. S. Yu, and B. Long, "Evolutionary clustering by hierarchical dirichlet process with hidden markov state," in *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ser. ICDM '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 658–667. [Online]. Available: http://dx.doi.org/10.1109/ICDM.2008.24

[10] R. M. Neal, "Probabilistic inference using markov chain monte carlo methods," 1993.

[11] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei, "Hierarchical Dirichlet processes," *Journal of the American Statistical Association*, vol. 101, no. 476, pp. 1566–1581, 2006.

[12] A. Zare and P. Gader, "Endmember detection using the dirichlet process," in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, dec. 2008, pp. 1 –4.

[13] C. M. Bishop, *Neural Networks for Pattern Recognition*. New York, NY, USA: Oxford University Press, Inc., 1995.

[14] P. Rao, Ed., *Asymtotic theory of statistical inference*. New York: Wiley Series in Probability and Mathematical Statistics.

[15] J. Casillas, O. Cordn, and F. Herrera, "Learning fuzzy rules using ant colony optimization algorithms," in *Abstract proceedings of ANTS2000 From Ant Colonies to Arti Ants: A Series of International Workshops on Ant Algorithms*, 2000, pp. 13–21.

[16] C.-F. Juang and C.-Y. Wang, "A self-generating fuzzy system with ant and particle swarm cooperative optimization," *Expert Syst. Appl.*, vol. 36, pp. 5362–5370, April 2009. [Online]. Available: http://portal.acm.org/citation.cfm?id=1497653.1498521

[17] H. Bellaaj, R. Ketata, and M. Chtourou, "A new method for fuzzy rule base reduction," *Journal of Intelligent and Fuzzy Systems*, vol. 25, no. 3, pp. 605–613, 2013.

[18] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society, Series B*, vol. 58, pp. 267–288, 1994.

[19] S. Medasani and R. Krishnapuram, "Categorization of image databases for efficient retrieval using robust mixture decomposition," *Comput. Vis. Image Understanding*, vol. 83, 2001.

[20] S. J. Gershman and D. M. Blei, "A tutorial on bayesian nonparametric models," *Journal of Mathematical Psychology*, vol. 56, no. 1, pp. 1 – 12, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S002224961100071X

[21] D. Aldous, "Exchangeability and related topics," in *cole d't de Probabilits de Saint-Flour XIII 1983*, ser. Lecture Notes in Mathematics, P. Hennequin, Ed. Springer Berlin Heidelberg, 1985, vol. 1117, pp. 1–198. [Online]. Available: http://dx.doi.org/10.1007/BFb0099421

[22] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed optimization by ant colonies," in *European Conference on Artificial Life*, 1991, pp. 134–142.

[23] M. Dorigo and L. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *Evolutionary Computation, IEEE Transactions on*, vol. 1, no. 1, pp. 53–66, 1997.

[24] J. Sanders and E. Kandrot, *CUDA by Example: An Introduction to General-Purpose GPU Programming*, 1st ed. Addison-Wesley Professional, 2010.

[25] D. B. Kirk and W.-m. W. Hwu, *Programming Massively Parallel Processors: A Hands-on Approach*, 1st ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2010.

[26] E. H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of Man-Machine Studies*, vol. 7, no. 1, pp. 1–13, 1975.

[27] R. Gupta, B. Pant, P. Sinha, R. Mehta, and V. Agarwala, "Study on ductility of ti aluminides using mamdani fuzzy inference system," in *Proceedings of the International Conference on Soft Computing for Problem Solving (SocProS 2011) December 20-22, 2011*, ser. Advances in Intelligent and Soft Computing, K. Deep, A. Nagar, M. Pant, and J. C. Bansal, Eds. Springer India, 2012, vol. 130, pp. 11–22. [Online]. Available: http://dx.doi.org/10.1007/978-81-322-0487-9_2

[28] N. Corporation. (2014) Welcome to NVIDIA - world leader in visual computing technologies. [Online]. Available: http://www.nvidia.com/

[29] B. U. of California. (2007) The berkeley segmentation dataset and benchmark. Website. [Online; Accessed: 17/04/2012],http://www.eecs.berkeley.edu/Research/Projects/.

[30] D. B. Peizer and J. W. Pratt, "A normal approximation for binomial, f, beta, and other common, related tail probabilities, i," *Journal of the American Statistical Association*, vol. 63, no. 324, pp. pp. 1416–1456, 1968. [Online]. Available: http://www.jstor.org/stable/2285895