ELSEVIER

Pattern Recognition Letters 21 (2000) 759–769

Pattern Recognition Letters

www.elsevier.nl/locate/patrec

Learning mixture models using a genetic version of the EM algorithm

Aleix M. Martínez [a,b,*], Jordi Vitrià [c]

[a] *Robot Vision Lab – School of Electrical and Computer Engineering, Purdue University, 1285 EE Building, West Lafayette, IN 47907-1285, USA*

[b] *Sony Computer Science Laboratory, 6, rue Amyot, 75005 Paris, France*

[c] *Centre de Visió per Computador and Dept. Informàtica, Universitat Autònoma de Barcelona – Edifici O, 08193 Bellaterra, Spain*

Received 29 April 1999; received in revised form 14 March 2000

Abstract

The need to find new pattern recognition techniques that correctly classify complex structures has risen as an important field of research. A well-known solution to this problem, which has proven to be very powerful, is the use of mixture models. Mixture models are typically fitted using the expectation-maximization (EM) algorithm. Unfortunately, optimal results are not always achieved because the EM algorithm, iterative in nature, is only guaranteed to produce a local maximum. In this paper, a solution to this problem is proposed and tested in a complex structure where the classical EM algorithm normally fails. This, we will do by means of a genetic algorithm (GA) which will allow the system to combine different solutions in a stochastic search so as to produce better results. The reported results show the usefulness of this approach, and suggest how it can be successfully implemented. Two new algorithms are proposed. The first one is useful when a priori information of the observed data is not available. The second solution is useful for those cases where some knowledge of the structure of the data-set is known. This second solution has proven to converge faster than the first one, although the final results reached are very similar to each other. © 2000 Published by Elsevier Science B.V. All rights reserved.

Keywords: EM algorithm; Genetic algorithms; Mixture models; Computer vision

1. Introduction

A typical problem of machine learning (Mitchell, 1997) and pattern recognition (Fukunaga, 1990) is the search for algorithms which correctly classify data structures corresponding to non-linear manifolds. A classic example of this is the classification of spiral-shaped data. Consider the case of Fig. 1, where the data are embedded in a simple two-dimensional space. The objective of any inductive learning method is to produce a classifier that accurately represents the underlying data structure. In this case, this means that the low-dimensionality of the sample data must be reflected in the resulting classifier. In other words, the best classifier will be the one which describes the data most precisely (McLachlan and Basford, 1988).

A possible solution to this and other difficult classification problems is the use of mixture

* Corresponding author. Fax: +1 765-494-0880.
E-mail address: aleix@ecn.purdue.edu (A.M. Martínez).

0167-8655/00/$ - see front matter © 2000 Published by Elsevier Science B.V. All rights reserved.
PII: S0167-8655(00)00031-3

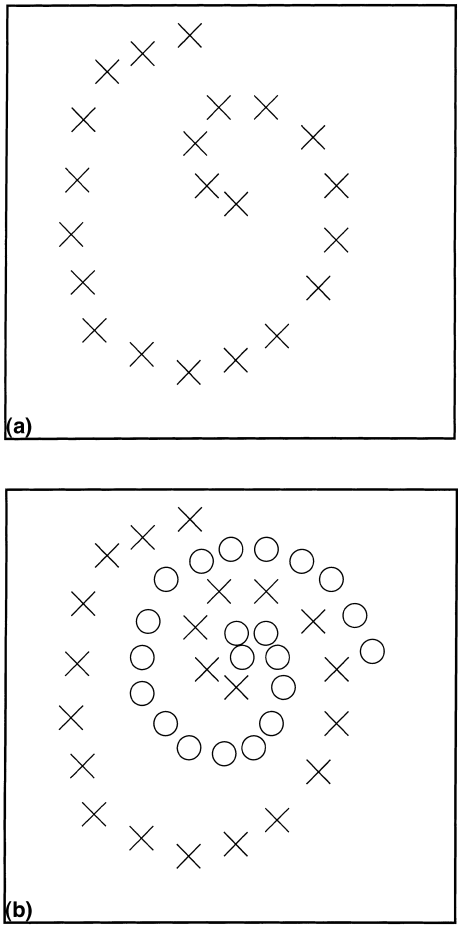

Fig. 1. Two examples of spiral-like structures. In (a) only one class is to be classified; in (b) two classes are to be classified.

models (e.g., mixture of Gaussians) (Redner and Walker, 1984; McLachlan and Basford, 1988). Learning in this framework is an estimation problem requiring an explicit probabilistic model and an algorithm for estimating the parameters of the model. An important advantage of this method is that models can be fitted by methods like singular value decomposition (SVD) and expectation-maximization (EM) which are more efficient than gradient descent techniques (Hinton et al., 1997).

We can use the EM algorithm to find maximum likelihood estimators for mixture models in these kinds of problems. Unfortunately, the solutions obtained by this method are not always good enough. The reason for this is that the EM algorithm is an iterative algorithm that can only guarantee a local maximum of the a posteriori probability distribution (McLachlan and Krishnan, 1997). At each step, the algorithm increases the a posteriori probability, $p(\theta|\boldsymbol{x})$ ($\boldsymbol{x} \in \mathbb{R}^p$ represents the p-dimensional observed data-set, and θ the unknown parameters of the mixture), but does not guarantee convergence to the global maximum when there are multiple maxima. In this case, it totally depends on the selected initial values of the estimates, which are usually initialized at random.

A possible improvement for this problem would be to calculate n solutions by randomly initializing the initial parameters, θ, of the mixture models at each algorithm execution, and then selecting that with the highest likelihood. That would just be a brute force approach. We propose another method, based on evolutionary strategies. In this paper, we show how the EM algorithm can be extended with the help of genetic algorithms (GAs) for fitting a mixture of Gaussian models to spiral-shaped data.

As an extension to this genetic approach, we propose the use of some type of *knowledge* of the data which can favor the best current solutions and thus reach better final results (or, at least, reach them faster). Even though the shape of the data is not known, in many applications it is worth to favor those solutions that are composed by connected Gaussian models. Precisely, we are looking for concatenations of Gaussian distributions as shown in Fig. 2(a). One way to impose that, is by selecting those families which have interconnections along the largest eigenvector of each model. This new idea can be readily incorporated into the selection function and it is similar to incorporate a priori information into the EM algorithm. This new method has proven to slightly improve the previous results (where the GA only maximized the maximum-likelihood (ML) hypothesis). More important, however, is that this new method allows faster convergence to the solution, which is a critical point in many of the learning systems built to date. This is consistent with previously reported results on learning, where the use of knowledge does not yield much improvement, but does make the system to converge faster.

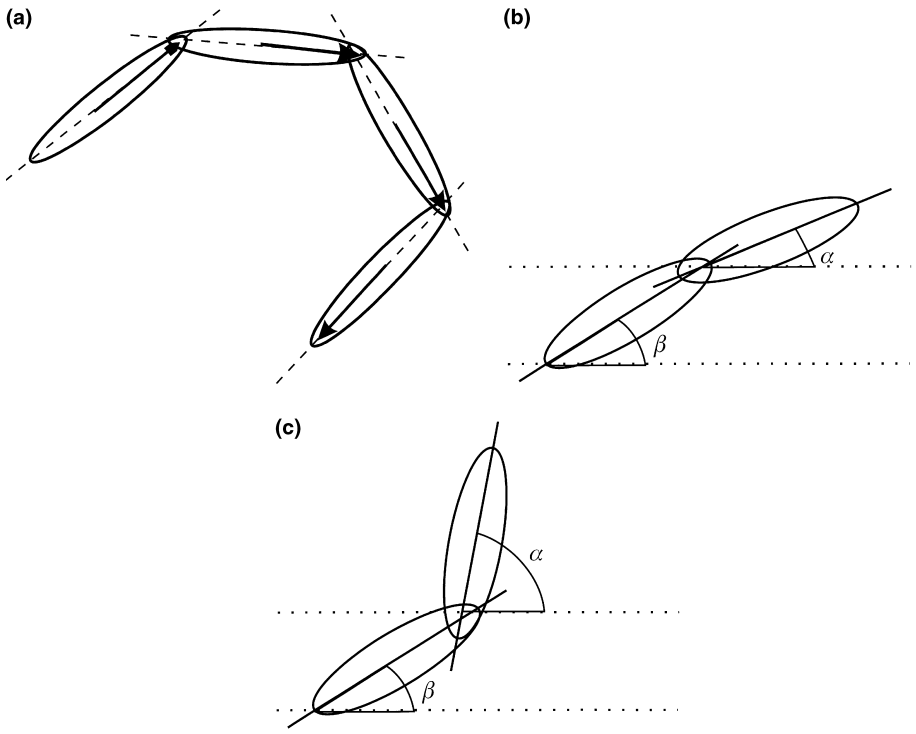

Fig. 2. (a) A mixture of Gaussians linked through their first eigenvectors. The case of figure (b) is much more desirable than the case of figure (c), since the approximation done is smoother.

The importance of these new genetic versions of the EM algorithm lies in the fact that the EM algorithm has a widespread application in the areas of computational molecular biology (e.g., Krogh et al., 1994), speech recognition (e.g., Rabiner, 1989), and image processing and computer vision (e.g., Vitrià and Llancer, 1996; Martínez, 2000), among others.

2. Learning mixture models using the EM algorithm and GA

2.1. Mixture models

The need for cluster analysis has risen in a natural way in many fields of study. These are methods to divide a set of n observations into g groups so that members of the same group are more alike than members of different groups. This objective becomes difficult when clusters have complex structures and their number is not known. The mixture modeling framework for clustering is an alternative that has the potential to handle complex structured data because it is model-based. An advantage of mixture models is that they combine much of the flexibility of non-parametric methods with certain of the analytical advantages of parametric methods (McLachlan and Basford, 1988).

Formally, let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ be a multivariate observation, where each $\boldsymbol{x}_i$ belongs to a p-dimensional space. Under the finite mixture model, each $\boldsymbol{x}_i$ can be viewed as rising from a superpopulation $G_1, \ldots, G_t$ of probabilistic models with their corresponding associated probabilities $\pi_1, \ldots, \pi_t$ (such that, $\sum_{i=1}^{t} \pi_i = 1$). Then, a mixture model is defined by its probability density function (p.d.f.) $f(\boldsymbol{x}; \phi)$, where $\phi = \{\pi, \theta\}$ represents the unknown parameters that describe the p.d.f. We can decompose this p.d.f. as a group of particular p.d.f.s for each model, i.e., $f(\boldsymbol{x}; \phi) = \sum_{i=1}^{t} \pi_i f_i(\boldsymbol{x}; \theta_i)$; where $\theta = \{\theta_1, \ldots, \theta_t\}$.

Normal mixture model estimation is a common form of *soft* clustering. In such a way the data $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ are assumed to be given by $\boldsymbol{x}_i \sim N(\mu_j, \Sigma_j)$ (where each N_j belongs to the model superpopulation with probability π_j). The means $\mu_1, \ldots, \mu_t$ and the covariance matrices $\Sigma_1, \ldots, \Sigma_t$, constitute the parameters, θ, associated with the density functions.

One of the most appealing features of mixture models is that they can deal with very complex p.d.f.s.

The problem of estimating mixture densities can itself be viewed as a missing data problem. The typical algorithm used to estimate the missing values is the EM algorithm (Dempster et al., 1977).

2.2. The EM algorithm

The EM algorithm (Tanner, 1990; Moon, 1996; McLachlan and Krishnan, 1997 are recommended for a review) is a well-known probabilistic technique that provides a quite general approach to learn in the presence of unobservable or missing variables.

Mathematically, let $\boldsymbol{y} \in \mathbb{R}^c$ be the set of complete variables (we assume that all $\boldsymbol{y}$ outcome from an underlying space ψ in $\mathbb{R}^c$) and $\boldsymbol{x} \in \mathbb{R}^o$ the set of observable variables ($o < c$). The complete dataset, $\boldsymbol{y}$, is not directly observable, but only by means of $\boldsymbol{x}$ as $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{y})$. This expression describes a mapping from many-to-one that makes the problem difficult to be solved. We can assume, however, that this mapping can be modeled by means of a p.d.f., which depends on a parameter θ, $f(\boldsymbol{y}|\theta)$ (where $\theta \in \boldsymbol{\Theta} \subset \mathbb{R}^r$ is the set of parameters of the density). If we assume this function to be continuous through θ and appropriate differentiable, the ML estimate of θ can be assumed to lie within the region $\boldsymbol{\Theta}$, and we can write the p.d.f. of the incomplete data as

$$g(\boldsymbol{x}|\theta) = \int_{\psi(x)} f(\boldsymbol{y}|\theta)\, \mathrm{d}\boldsymbol{x}. \tag{1}$$

Let $L_x(\theta) = \log g(\boldsymbol{x}|\theta)$ denote the log-likelihood function. Under these circumstances we want to find the θ that maximizes $f(\boldsymbol{y}|\theta)$. Unfortunately, we do not have the data, $\boldsymbol{y}$, to compute the log-likelihood. The EM algorithm solves this problem by first estimating the $\log f(\boldsymbol{y}|\theta)$ given $\boldsymbol{x}$ and the current estimation on θ, i.e., $Q(\theta|\theta^{[k]}) = E[\log f(\boldsymbol{y}|\theta)|\boldsymbol{x}, \theta^{[k]}]$, and then, calculating the argument θ that maximizes the previous estimation, i.e., $\theta^{[k+1]} = \arg\max_\theta Q(\theta|\theta^{[k]})$. These two steps are called the *expectation* and the *maximization* step, respectively. An initial value (normally at random) is given to θ, and then, both steps are iterated until convergence. Convergence is reached when the ML hypothesis does not change or is smaller than a threshold, i.e., $|p(\boldsymbol{x}|\theta^{[k]}) - p(\boldsymbol{x}|\theta^{[k-1]})| < \varepsilon$.

When using normal mixture models, the E-step corresponds to the expectation that an observable value $\boldsymbol{x}_i$ belongs to a normal model θ_j, denoted as $E[z_{ij}|\boldsymbol{x}_i, \theta_j]$. We temporally assume that the current estimations of each model, θ_j, is correctly known. This can be written as

$$z_{ij}^{[k+1]} = \frac{\left|\Sigma_j^{[k]}\right|^{-1/2} \exp\left\{ -\frac{\left(\boldsymbol{x}_i - \mu_j^{[k]}\right)^{\mathrm{T}} \Sigma_j^{[k]-1} \left(\boldsymbol{x}_i - \mu_j^{[k]}\right)}{2} \right\}}{\sum_{l=1}^{t} \left|\Sigma_l^{[k]}\right|^{-1/2} \exp\left\{ -\frac{\left(\boldsymbol{x}_i - \mu_l^{[k]}\right)^{\mathrm{T}} \Sigma_l^{[k]-1} \left(\boldsymbol{x}_i - \mu_l^{[k]}\right)}{2} \right\}}. \tag{2}$$

The M-step will, then, re-estimate the means and the covariances of the Gaussian models assuming that the estimated z_{ij} is correct:

$$\mu_j^{[k+1]} = \frac{\sum_{i=1}^{s} z_{ij}^{[k]} \boldsymbol{x}_i}{\sum_{i=1}^{s} z_{ij}^{[k]}}, \tag{3}$$

$$\Sigma_j^{[k+1]} = \frac{\sum_{i=1}^{s} z_{ij}^{[k]} \left(\boldsymbol{x}_i - \mu_j^{[k+1]}\right)\left(\boldsymbol{x}_i - \mu_j^{[k+1]}\right)^{\mathrm{T}}}{\sum_{i=1}^{s} z_{ij}^{[k]}}, \tag{4}$$

where s is the number of samples, and t is the number of models used in the mixture. Moreover, we are assuming that all models are equi-probable (i.e., $\pi_i = \pi_j \forall\ i, j$). In any other case, the equations will slightly differ from above (see McLachlan and Basford, 1988 for further details). The algorithms reported in this communication can be easily generalized to this case. The initial values (i.e., $\mu_j^{[0]}$ and $\Sigma_j^{[0]}\ \forall j$) are typically taken at random. Only in some very specific cases a better initialization might be found.

2.3. A genetic version of the EM algorithm

Nature has a robust way of evolving successful individuals. Individuals who are ill-posed for an environment die off, whereas the fit ones live to reproduce. Somehow, similarly to how nature works, GAs generate a group of offsprings by means of recombination and mutation of their parents. A fitness function is then needed to select those offsprings that will survive, and thus, pass to be the parents of the next generation. (See Michalewicz, 1994, for an introduction on GA.)

While the straightforward way to resolve the local maxima problem would be to correspond to computing the EM algorithm several times and select the one with highest probability outcome, the GA version makes use of the "knowledge" of the past iterations in an attempt to improve future results.

Formally, let us define a normal mixture model as

$$h_i = \{N(\mu_{i1}, \Sigma_{i1}), \ldots, N(\mu_{it}, \Sigma_{it})\}, \tag{5}$$

where μ_{ij} and Σ_{ij} are the mean and covariance matrix of the jth normal model of the ith mixture, and h_i the mixture model, i, that involves all t normal models described. When referring to any of the specific models of each mixture, in the following, we will use the notation $\boldsymbol{O}_{ij}$ (i.e., $\boldsymbol{O}_{ij} = N(\mu_{ij}, \Sigma_{ij})$).

In a GA, we are given an initial population $H = \{h_1, \ldots, h_n\}$ of n different normal mixture models. Each individual of the population is initialized at random within a normal distribution with zero mean and identity matrix as covariance matrix, i.e., $N(\boldsymbol{0}, \boldsymbol{I})$.

Then, a (n, m)-strategy (Bäck and Schwefel, 1996) is used to select those individuals which best represent our data (that is to say, those ones which are best adapted). The notation (n, m) indicates that n parents create m offsprings by means of recombination and mutation, and the best n offspring individuals are deterministically selected to replace the parents. Note that the method is not *elitistic*, in the sense that the best member of the population at generation $t + 1$ can perform worse than the best individual at generation t. Thus, we facilitate the system to accept temporary deteriorations that might help to leave the region of attraction of a local maximum and reach a better final state. This strategy has been found to be more efficient than the $(n + m)$-strategy, where n survivors are selected from the union of parents and offsprings (Bäck and Schwefel, 1996).

The mutation process is done at a rate of 30% and, even then, only 25% of the parameters of each model are mutated. The mutation operator uses a normally distributed random vector $N(0, 1)$. Moreover, we impose the standard deviation (σ) to be identical for all object variables. To achieve that, all object variables are mutated as follows:

$$\begin{aligned} \sigma' &= \sigma \cdot \exp\left\{\sqrt{p}^{-1} \cdot N(0,1)\right\}, \\ y'_{k,j,i} &= y_{k,j,i} + \sigma' \cdot N(0,1), \end{aligned} \tag{6}$$

where $y_{k,j,i}$ represents the kth component of the $\boldsymbol{O}_{ij}$ model of the mutated mixture h_i, $'$ means new assigned value and p is the dimensionality of the data.

The rest of the offspring individuals (i.e., 70% of the m offsprings to be computed) are obtained by recombination. To do that, we use the typical recombination operator, crossover. We first select two individuals at random, e.g., $B = (b_1, \ldots, b_t)$ and individual $C = (c_1, \ldots, c_t)$. After that, we select a random integer number *pos* from the range $[1, \ldots, t]$ (recall that t is the length of each individual). Then the two individuals are replaced by the pair of their offsprings $(b_1, \ldots, b_{pos}, c_{pos+1}, \ldots, c_t)$ and $(c_1, \ldots, c_{pos}, b_{pos+1}, \ldots, b_t)$.

The evaluation process consists of applying the EM algorithm for normal mixture models described above (in Eqs. (2)–(4)).

After obtaining the m individuals, a selection operator based on the ML hypothesis is used. To do that, we calculate the a posteriori probability of all the mixture models, $p(h_i|\boldsymbol{x})$ for all $i = 1, \ldots, s$ (where $p(h_i|\boldsymbol{x}) = \sum_{j=1}^{t} p(\theta_{ij}|\boldsymbol{x})$, and $\theta_{ij} = \{\mu_{ij}, \Sigma_{ij}\}$). The n individuals associated with the highest a posteriori probabilities are selected to pass to the next step.

Further a *lucky-factor* was added to the system. In 20% of the iterations, q parents also survive, where $q \in [1, n/4]$ (obviously, when this is the case, only $n - q$ individuals can survive the selection process).

Algorithm I (EM + GA).

```
Initialize (H);
Evaluate_EM (H);
While (not convergence)
begin
  H' = Mutate (H);
  H' = H' ∪ Recombine (H);
  Evaluate_EM (H');
  H = Lucky_individuals (H);
  H = H ∪ Select (H');
end;
```

Recall that in 80% of the cases, `Lucky_individuals (·)` will return an empty set. An example of the outcome of this strategy in the problem of the *spiral* can be appreciated in Fig. 3(c).

When using this genetic approach, there are many different ways of considering convergence. To just number a few:

(i) when consecutive generations do not represent much improvement;
(ii) when a pre-specified number of generations is reached or
(iii) when a pre-determined minimum value of the a posteriori probability is obtained.

2.4. Using knowledge to learn

A better way of selecting the *best* individuals (mixtures) in each generation can be found if some knowledge of the underlying structure (space) of the data is given. Fortunately, this is the case in many real applications. In the case of the spiral-shaped data, for example, we know that all clusters must be aligned to one another in some way. That is, all the models of our mixture should be interconnected in such a way that they together describe the *imaginary* line drawn by the spiral. When using Gaussian models this can be easily implemented by imposing on all axes (which are described by the largest eigenvector of each covariance matrix) to be aligned to its predecessor and successor ones (see Fig. 2(a)). Since the system is not cyclic, the first and the last models do not have such a restriction and must only be interconnected from one of their extremes.

If we think of those largest eigenvectors of each model as local linear approximations of the imaginary non-linear line described by the spiral, we can readily see that when good approximations are searched, a smoothing factor on the angle described between consecutive eigenvectors must be imposed (Bregler and Omohundro, 1994) (see Fig. 2(b)–(c) for clarity). Obviously, as more models are added to the mixture, better linear approximations might be found. However, in general and under the same conditions (i.e., same number of models), those that describe the data most precisely are those that best approximate the curve; this is the same as those that have the largest eigenvector more similar to its predecessor and successor ones.

In such a case, the first problem to be attempted is that of finding the closest models to each current model $\boldsymbol{O}_{ij}$. The optimal way of doing that would consist of computing the geodesic distance between the mean (i.e., center) of the current model to all other models. The two smallest distances will deterministically select the two closest models to $\boldsymbol{O}_{ij}$. Unfortunately, this method would be too computationally intensive. Further studies should be conducted in this direction before good, low-cost methods for calculating this geodesic distance can be found (an attempt to this problem is (Saul and Jordan, 1997)). A simpler way of doing it (but also a good approximation), consists of using the Mahalanobis distance between each pair of models. If the distance between two models needs to be calculated, we can use both the Mahalanobis distance between model $\boldsymbol{O}_{ij}$ and model $\boldsymbol{O}_{ik}$ and the Mahalanobis distance between $\boldsymbol{O}_{ik}$ and $\boldsymbol{O}_{ij}$. While the first measurement takes only into account the space deformation done by model $\boldsymbol{O}_{ij}$, the second one will only consider the deformation of model $\boldsymbol{O}_{ik}$. Finally, the average between both distances will be taken as the final measure of proximity between both models. Mathematically speaking, we define this as

$$D_{ij,ik} = \frac{d_{ij,ik} + d_{ik,ij}}{2}, \tag{7}$$

where $d_{ij,ik}$ is the Mahalanobis distance from $\boldsymbol{O}_{ij}$ to $\boldsymbol{O}_{ik}$, and $d_{ik,ij}$ is the Mahalanobis distance from $\boldsymbol{O}_{ik}$ to $\boldsymbol{O}_{ij}$. The Mahalanobis distance is expressed as

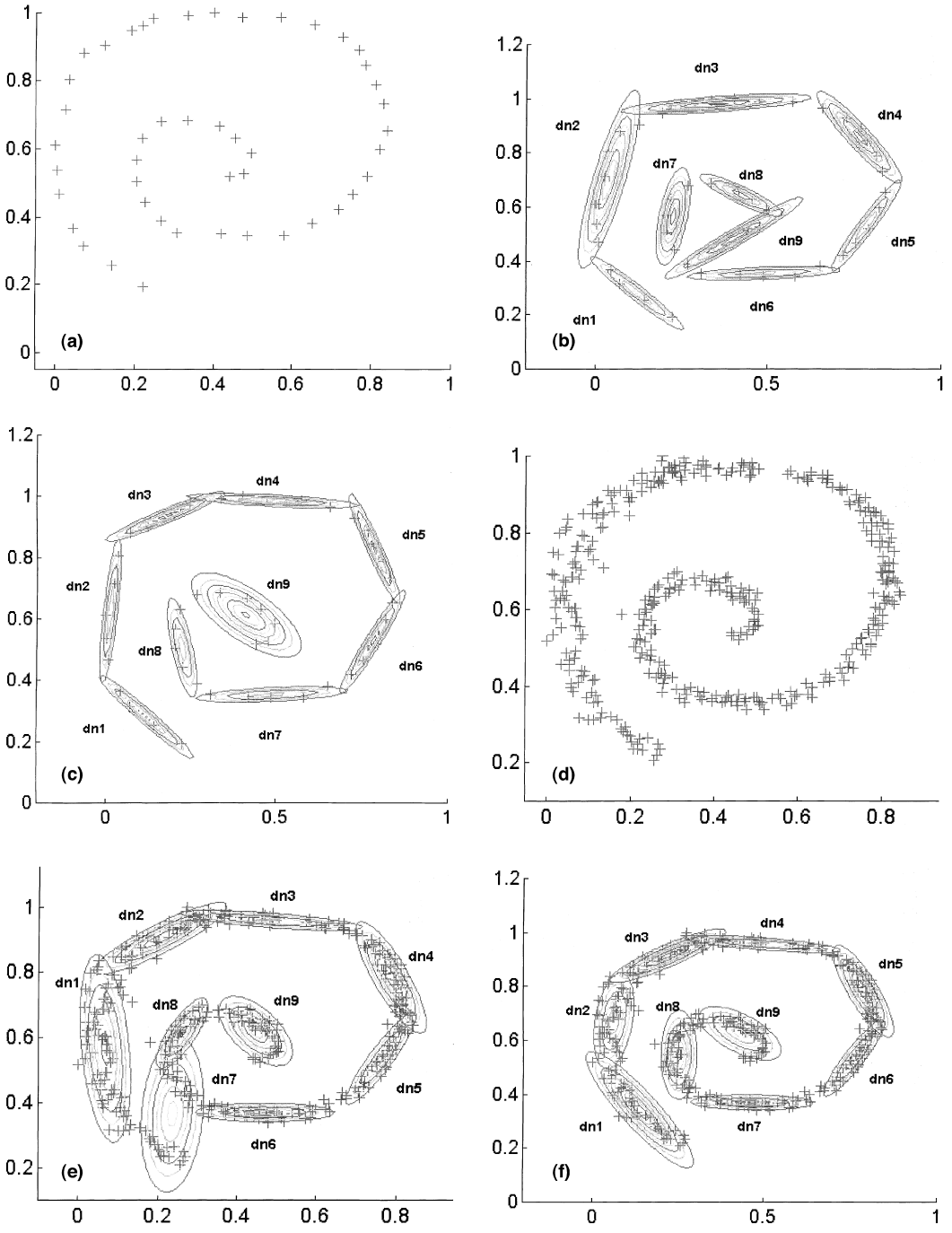

Fig. 3. (a) and (d): The two spiral-shaped data-sets used in the experiments. (b) An output of the brute force algorithm applied to fit nine Gaussian models to the data-set (a) – using 10 random initializations. (c) An output of an execution of Algorithm I applied to the data-set (a) – using a population of five individuals and two generations. (e) Same as (b) but using data-set (d). (f) Same as (c) but using data-set (d).

$$d^2_{ik,ij} = (\mu_{ik} - \mu_{ij})^{\mathrm{T}} \Sigma^{-1}_{ij} (\mu_{ik} - \mu_{ij}). \quad (8)$$

Once the predecessor and successor models of each model $\boldsymbol{O}_{ij}$ have been found (as pointed out before, the first and the last models do not have such a restriction and only the closest model has to be found), we can calculate the smoothness factor among their angles. That is, assuming that the data are well described by the local linear models, the more different the angles are, the worse the approximation is (Fig. 2(b) and (c)). This can be formally expressed as

$$q_{ij} = \frac{1}{|\alpha - \beta| + |\alpha - \gamma| + 1}, \quad (9)$$

where α is the clockwise angle from the axis described by the largest eigenvector of the current model $\boldsymbol{O}_{ij}$ to the horizontal, β the clockwise angle from the predecessor to the horizontal, and γ is the clockwise angle from the successor to the horizontal. This final factor, q_{ij}, can be seen as a measurement of quality of mixture i. The higher the quality, the better.

Finally, we can use this quality measurement together with the *log-likelihood* to select those mixtures that best describe the data.

The selecting function should, thus, be proportional to both the quality of the model (i.e., $q_{i,j}$) and the a posteriori probability (i.e., $p(h_i|\boldsymbol{x})$). We can write the probability of selecting (each) mixture h_i as

$$p(\text{select}_i) \propto \tau \left(\sum_{j=1}^{t} q_{i,j} \right) + (\tau - 1) p(h_i|\boldsymbol{x}), \quad (10)$$

where τ is the value that determines which of the two factors have more relevance, the smoothing factor or the probability.

The new algorithm is then described as follows:

Algorithm II (EM + GA + Knowledge).

```
Initialize (H);
Evaluate_EM (H);
While (not convergence)
begin
  H' = Mutate (H);
  H' = H' ∪ Recombine (H);
  Evaluate_EM (H');
  ξ = KNOWLEDGE (H');
  H = Lucky_individuals (H);
  H = H ∪ Select (H', ξ);
end;
```

where KNOWLEDGE represents the evaluation of the offsprings by means of the knowledge we are given (e.g., the interconnection of the first eigenvector of each model), and ξ denotes the quality value of each model (i.e., $\xi = \{q_{11}, \ldots, q_{mt}\}$).

The importance of this new algorithm is obvious, since many practical applications of signal and image processing and computer vision have similar shaped data-sets. Although many of them are not as complicated as the spiral, almost all of them are continuous and need an interconnection of the same type (which, as we have seen, can be easily implemented).

An extension of this second algorithm might correspond to introduce this knowledge into the mutation and recombination operators.

3. Experimental results

In order to test the accuracy of the two algorithms described in this paper, we have built two different spiral-shaped data, the first one using a small sample data-set and the second one using a density sample data-set (Fig. 3(a) and (d) – these data-sets are publicly available from http://rvll.ecn.purdue.edu/~aleix/gaem.html). In each of the cases, three algorithms have been tested under equivalent computational conditions (same hardware, same computational cost): a brute force algorithm and the two genetic versions described in this contribution. The results show that both genetic versions of the EM algorithm can achieve better results than the random version.

3.1. Three different algorithms

In order to make all implementations equivalent in computational cost, the following three algorithms are described:

1. Random algorithm: I different random initializations are given to the EM algorithm.

The best result (among all I), that is to say, the one with highest a posteriori probability, is deterministically selected as the output (solution) of the algorithm. Recall that the random initialization refers to the parameters $\Sigma_j^{[0]}$ and $\mu_j^{[0]}$.

2. `Algorithm I (EM + GA)`: A population of P individuals is generated. This population is then iterated through F different generations in such a way that $P * F = I$. In doing so, we guarantee that both algorithms have an equivalent computational cost.
3. `Algorithm II (EM + GA + knowledge)`: The latest algorithm is initialized as before (in Algorithm I). However, the computational cost will be higher, since we need to compute the `KNOWLEDGE (·)` operator in each iteration. Strictly speaking, it involves the order of $p^2 + p$ multiplications and additions, and $2p$ subtractions (for the calculation of the Mahalanobis distance), where p is the dimensionality of the data. While the number of subtractions remains small, the number of multiplications and additions increases quadratically. That is to say, when the number of dimensions is very small (as is in our case of the spiral, where $p = 2$), that cost is very low and can be ignored. However, when the dimensionality of the data is high, the cost should be taken into consideration.

3.2. Tests

Two different spirals were used to conduct the experiments, Fig. 3(a) and (d). In each spiral, two different values were given to I, P and F: (i) $I = 100$, $P = 5$ and $F = 20$; and (ii) $I = 100$, $P = 10$ and $F = 10$.

As one can appreciate in Fig. 3 the genetic versions of the EM algorithm came up with better results than the random version.

To better analyze this result, we can visualize the *log-likelihood* probability of each outcome. Fig. 4(a) and (b) shows an example of applying all three algorithms using the values described in (i) and (ii). (Note that in these cases, the plot of the EM algorithm reflects the best choice of five different initializations; the dotted line represents the output of the brute force algorithm along time.)

Then, we execute these two cases, (i) and (ii), 1000 times, and compute the means of all outcomes. Fig. 4(c)–(f) shows the results of all three algorithms. (In these cases, EM refers to the outcome of the brute force algorithm.) These results clearly show that both genetic versions of the EM algorithm obtained better results than the random (brute force) algorithm.

The results of Algorithm II were slightly better than those obtained in Algorithm I. However, the improvement is not as much as one could have expected. Nevertheless, when comparing the convergence velocity of all three algorithms, we can see that Algorithm II converges faster than the other ones. This is consistent with previous results reported by the EM algorithm, where the MAP version does not result in much better solutions than the ML version, but does converge faster (McLachlan and Basford, 1988).

3.3. Discussion

Our tests have shown that both genetic versions of the EM algorithm to fit mixture models end up with better results than a brute force (random initialization) version would. Fig. 4(c) and (d) shows that different parameterizations of the genetic versions of the EM algorithm obtain very similar results. Note that the random algorithm (brute force) does never approach the performance of the GAs; the difference of performance is (approximately) the same after one generation than after 20 (10) generations.

However, the EM algorithm applied to fit a mixture of Gaussian models is best used when high-density data-sets are available. Fig. 4(e) and (f) reflects this. In this second case, the classical EM algorithm "constantly" approaches the genetic. Nevertheless, both genetic version of the EM reach the final desirable solution much faster. In the first generation, the result is already very close to the result one gets in the last. Around generation 7 in Fig. 4(e) and in generation 6 in Fig. 4(f) Algorithm I has already reached its best outcome. Algorithm II is slightly faster; which is an important point of our results, because the high computational cost of the learning methods has been one of the major problems of machine learning

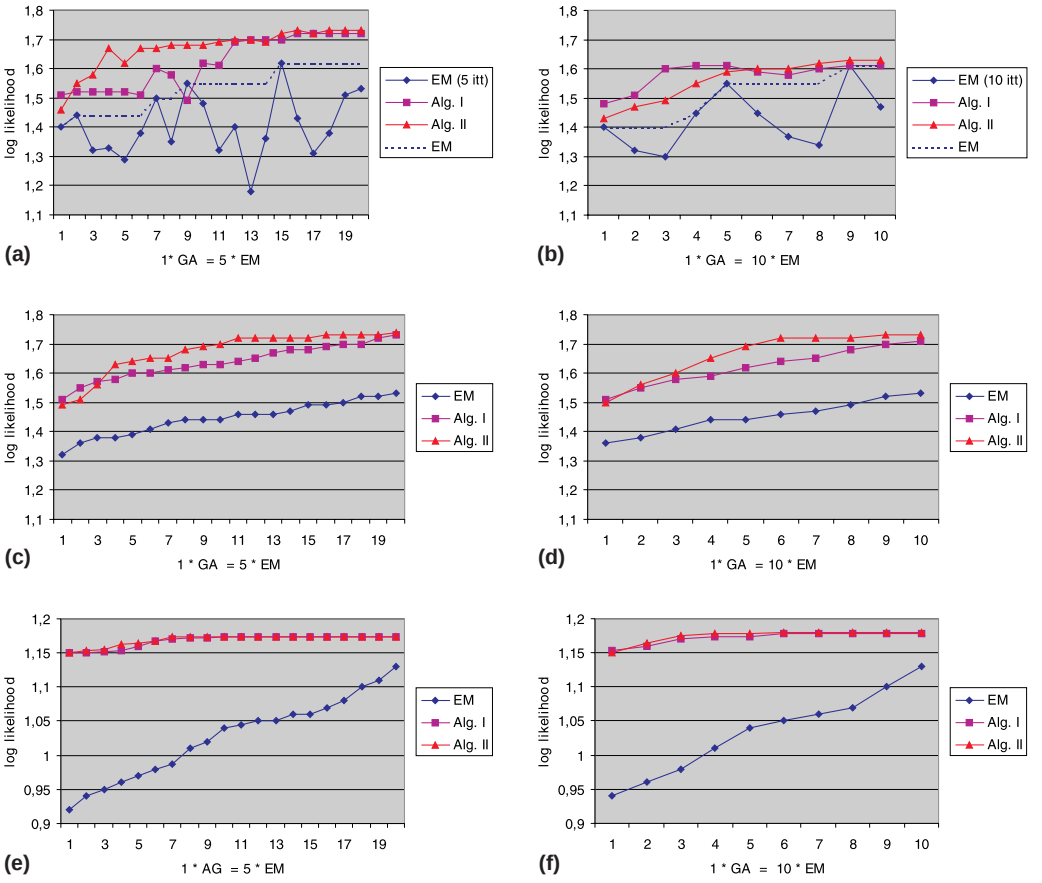

Fig. 4. Comparison between the results of all three algorithms (see text). (a) A result example of an execution of all three algorithms using the data-set of Fig. 3(a); and with parameters $I = 100$, $P = 5$ and $F = 20$. (Notation: $1 * GA = 5 * \text{EM}$ means that for each generation of the genetic version(s) of the EM algorithm, the classical EM algorithm is iterated using five different random initializations and the best outcome selected as the solution.) (b) Same as (a) but using parameters $I = 100$, $P = 10$ and $F = 10$ (note that in this second case, each generation equals to ten random initializations). (c) The mean of 1000 different executions of all three algorithms as in (a). (d) The mean of 1000 different executions of all three algorithms as in (b). (e) Same as (c) but using the data-set of Fig. 3(d). (f) Same as (d) but using the data-set of Fig. 3(d).

and pattern recognition strategies reported in the past (nowadays, we are dealing not only with better classification or learning methods, but also with methods of ever increasing speed).

Both algorithms have been successfully used in an application of vision-guided mobile robotics (more precisely, in a robot navigation problem) within a computer vision appearance-based approach (Martínez and Vitrià, 1998). Both algorithms have proven to be more efficient than other previously reported appearance-based approaches (such as principal component analysis, linear discriminant analysis and normal mixture models learned using the classical EM algorithm).

4. Conclusions

The need to find pattern recognition techniques that correctly classify and represent complex data has risen as an important field of research with many interesting applications in both signal processing and image processing communities.

Mixture models are a very useful technique to the learning of complex structures. In this paper, we have proposed two new algorithms to fit mixture models by means of the EM algorithm and GA. The first proposed algorithm uses a population of mixture models that evolves in time in an attempt to overcome the local maxima problem. The second algorithm with similar ideas, which additionally adds knowledge of the underlying structure to be learned, has also been reported. Both algorithms have proven to be superior to the typical EM implementation (with random initializations) for the spiral-shape data case; under the same computational conditions (that is to say, same hardware, same conditions and same cost). Algorithm II (which uses knowledge to learn) does not result in much better solutions than those reached by Algorithm I (which only uses GA), but it converges faster to the final solution. This is consistent with other previous results of the EM algorithm, where the MAP hypothesis does not yield much better results than the ML hypothesis, but does converge faster too.

The application of these new algorithms to the classification of complex spaces (especially in computer vision) is our major interest for future research.

Acknowledgements

This research was supported in part by grant TAP94-0401 from the CICYT. We thank the referees for so valuable comments. The algorithms (written in Matlab) and the data-sets reported in this contribution are publicly available at http://RVL.www.ecn.purdue.edu/~aleix/gaem.html. Soledad Garcia has implemented a more user-friendly version of Algorithm I in Visual C++ (also available). We thank Gert Westermann for proofreading. The first author is also indebted to Mar and Hobbes.

References

Bäck, T., Schwefel, H., 1996. Evolutionary computation: An overview. In: Proc. IEEE Conf. Evolutionary Computation (ICEC '96), Nagoya, Japan.

Bregler, C., Omohundro, S.M., 1994. Surface learning with applications to lipreading. In: Cowan, J.D., Tesauro, G., Alspector, J. (Eds.), Advances in Neural Information Processing Systems, Vol. 6. Morgan Kaufmann, Los Altos, CA, pp. 43–50.

Dempster, A.P., Laird, N.M., Rubin, D.B., 1977. Maximum likelihood from incomplete data via the EM algorithm. J. Roy. Statist. Soc. 30 (1), 1–38.

Fukunaga, K., 1990. Introduction to Statistical Pattern Recognition, second ed. Academic Press, New York.

Hinton, G.E., Dayan, P., Revow, M., 1997. Modeling the manifolds of images of handwritten digits. IEEE Trans. Neural Networks 8 (1), 65–74.

Krogh, A., Brown, M., Mian, I.S., Sjölander, K., Haussler, D., 1994. Hidden Markov models in computational biology: applications to protein modeling. J. Mol. Biol. 2354, 1501–1531.

Martínez, A.M., 2000. Recognition of partially occluded and/or imprecisely localized faces. In: Proc. IEEE Comput. Vision and Pattern Recognition (CVPR).

Martínez, A.M., Vitrià, J., 1998. Visual annotations for mobile robot navigation. CVC Technical Report #25.

McLachlan, G., Basford, K., 1988. Mixture Models: Inference and applications to clustering. Marcel Dekker, New York.

McLachlan, G., Krishnan, T., 1997. The EM Algorithm and Extensions. Wiley, New York.

Michalewicz, Z. (Ed.), 1994. Genetic Algorithms + Data Structures = Evolution Programs. Springer, Berlin.

Mitchell, T.M., 1997. Machine Learning. McGraw-Hill, New York.

Moon, T.K., 1996. The expectation-maximization algorithm. IEEE Signal Processing Magazine 13 (6), 47–60.

Rabiner, L.R., 1989. A tutorial on hidden Markov models and selected applications in speech recognition. Proc. IEEE 77 (2), 257–285.

Redner, R., Walker, H., 1984. Mixture densities, maximum likelihood and the EM algorithm. SIAM Rev. 26 (2), 195–239.

Saul, L.K., Jordan, M.L., 1997. A variational principle for model-based morphing. In: Mozer, M.C., Jordan, M.I., Petsche, T. (Eds.), Advances in Neural Information Processing Systems 9, pp. 267–273.

Tanner, T.M., 1990. Tools for Statistical Inference. Lecture Notes in Statistics, Springer, Berlin.

Vitrià, J., Llancer, J., 1996. Reconstructing 3D light microscopic images using the EM Algorithm. Pattern Recognition Lett. 17 (14), 1491–1498.