Sequentially Greedy Unsupervised Learning of Gaussian Mixture Models by Means of A Binary Tree Structure

Nicola Greggio[a,b] [1], Alexandre Bernardino [a] and José Santos-Victor [a]
[a] *Instituto de Sistemas e Robótica, Instituto Superior Técnico*
1049-001 Lisboa, Portugal
[b] *ARTS Lab - Scuola Superiore S.Anna, Polo S.Anna Valdera*
Viale R. Piaggio, 34 - 56025 Pontedera, Italy

Abstract. *We propose an unsupervised learning algorithm for the estimation of the number of components and the parameters of a mixture model. It starts from a single mixture component covering the whole data set (therefore avoiding the ill-posed problem of the components' initialization, saving also computational burden). Then, it incrementally splits that component during expectation maximization steps, thus exploiting the full space of solutions following a binary tree structure. After each component insertion it evaluates whether accepting this new solution or discarding it according with the chosen information criterion. We show that the method is faster that state-of-the-art alternatives, is insensitive to initialization (deterministic initialization strategy), and has better data fits in average. This is illustrated through a series of experiments, both with synthetic and real images.*

Keywords. Machine Learning, Unsupervised Clustering, Self-Adapting Expectation Maximization

1. Introduction

Several techniques have been proposed in the literature for unsupervised learning, from Kohonen maps [1], Growing Neural gas [2], k-means [3], to Independent component analysis [4], [5], etc. Particularly successful is the Expectation Maximization algorithm applied to finite mixture models. Fitting a mixture model to the distribution of the data is equivalent, in some applications, to the identification of the clusters with the mixture components [6].

One of the most widely used distributions is the normal, or Gaussian, distribution. The normal distribution can be used to describe, at least approximately, any variable that tends to cluster around the mean. If data is generated by a mixture of Gaussians, the clustering problem will reduce to the estimation of the number of Gaussian components and their paramenters. Expectation-Maximization (EM) algorithm is well known and attractive approach for learning the parameters of mixture models [7], [6]. It always converges to a local optimum [8], especially for the case of Normal mixtures [6], [9]. How-

[1] Corresponding Author: Nicola Greggio; E-mail: ngreggio@ist.ist.utl.pt.

ever, it also presents some drawbacks. For instance, if requires the *a-priori* specification of the model order, namely, the number of components and its results are sensitive to initialization.

The selection of the right number of components is a critical issue. The more components there are within the mixture, the better the data fit will be. Unfortunately, increasing the number of components will lead to data overfitting and to increase in the computational burden. Therefore, finding the best compromise between precision, generalization and speed is an essential concern. A common approach to address this compromise is to try different hypothesis for the number of components, and then selecting the best model according to some appropriate model selection criteria.

1.1. Related Work

Different approaches can be used to select the best number of components in a mixture distribution. These can be divided into two main classes: *off-line* and *on-line* techniques.

The first ones evaluate the best model by executing independent runs of the EM algorithm for many different initializations, and evaluating each estimate with criteria that penalize complex models (e.g. the Akaike Information Criterion (AIC) [10], the Schwarz's Bayesian Information Criterion [11], the Rissanen Minimum Description Length (MDL) [12], and Wallace and Freeman Minimum Message Lenght (MML) [13]). All of these criteria, in order to be effective, have to be evaluated for every possible number of models under comparison. Therefore, it is obvious that, for having a sufficient search range the complexity goes with the number of tested models as well as the model parameters.

The second ones start with a fixed set of models and sequentially adjust their configuration (including the number of components) based on different evaluation criteria. Pernkopf and Bouchaffra proposed a Genetic-Based EM Algorithm capable of learning gaussians mixture models [14]. They first selected the number of components by means of the minimum description length (MDL) criterion. A combination of genetic algorithms with the EM has been explored.

A greedy algorithm is characterized by making the locally optimal choice at each stage with the hope of finding the global optimum. Applied to the EM algorithm, they usually start with a single component (therefore side-stepping the EM initialization problem), and then increase their number during the computation. However, the big issue in these kind of algorithm is the insertion selection criterion: Deciding when inserting a new component and how can determine the success or failure of the subsequent computation. At the time, no precise solution has been posted to address this drawback. In 2002 Vlassis and Likas introduced a greedy algorithm for learning Gaussian mixtures [15]. They start with a single component covering all the data. Then they split an element and perform the EM locally for optimizing only the two modified components. Nevertheless, the total complexity for the global search of the element to be splitted $O(n^2)$. Subsequently, Verbeek et al. developed a greedy method to learn the gaussians mixture model configuration [16]. Their search for the new components is claimed to take $O(n)$. Greedy algorithms mostly (but not always) fail to find the globally optimal solution, because they usually do not operate exhaustively on all the data. Our technique try to overcome this limitation by using a binary tree for deciding which component has to be replicated in an exhaustive way. Besides, our recursive search by means of the binary tree costs only $O(\log n)$.

1.2. Outline

In sec. 1.1 we analyze the state of the art of unsupervised learning of mixture models. In sec. 3 we introduce the proposed algorithm. Specifically, we describe its initialization (sec. 3.1), the replication process (sec. 3.2), the stopping criteria (sec. 3.3).Then, in sec. 4 we study the particular case of Gaussian mixtures, and the tuning of replication process within this specific context (sec. 4.1). Then, in sec. 5 we describe our experimental set-up for testing the validity of our new technique and in sec. 6 we compare our results against some alternatives, either classical or cutting-edge. Finally, in sec. 7 we conclude and propose directions for future work.

2. Model Selection Criterion: Minimum message length (MML)

The application of the EM algorithm for mixtures relies significantly on the *a-priori* knowledge of the number of components k and their initialization. If values too far from the optimal ones are chosen the algorithm may not be able to reach the optimal solution. In [17] there is a comprehensive survey on the most well-known and used criteria. We adopted the minimum message length (MML) criterion developed in [18], which formulation is:

$$\bar{\vartheta}_{opt} = arg\min_{\bar{\vartheta}} \left\{ -L\left(X|\bar{\vartheta}\right) + \frac{N}{2}\sum_{i=1}^{c} \ln\left(\frac{n\cdot w_i}{12}\right) + \frac{c}{2}\left(N+1-\ln 12n\right) \right\} \quad (1)$$

which meaning with respect to the original MML and MDL has been discussed in [18].

3. Mixture Learning Algorithm

Our algorithm starts with a single component, with the relative mixture initialization (sec. 3.1). Then, by following a binary tree structure (sec. 3.2.1) new classes are added by means of replicating existing ones (sec. 3.2.2), once a time. Furthermore, the cost function (1) is evaluated in order to decide whether keeping or discarding the current mixture (sec. 3.3). In the first case, the binary tree will be updated with the new solution. When a new mixture element is added, it will become a child together with the original one. Therefore, within our representation, its father dies, and only the two children survive. Otherwise, in the second case (i.e. when the new mixture configuration is discarded), the previous mixture will be restored as a starting point for a new component replication, and that node will never be proposed to have children anymore. Finally, when there will no node eligible to have children (i.e. when all the combinations have been tried), the algorithm terminates.

3.1. ***Parameters' initialization***

Before starting any computation, the first component (the only mixture class) will be automatically initialized to the whole data set parameters. This means, e.g. in case of normal mixture, the mean of the covariance relative to the whole data set, as follows:

$$\begin{aligned} \mu_{data,d} &= \frac{1}{N}\sum_{i}^{N} x_d^N \\ \Sigma_{data,i} &= \langle \bar{x}_i - \bar{\mu}_{data} \rangle \langle \bar{x}_i - \bar{\mu}_{data} \rangle^T \end{aligned} \tag{2}$$

where N is the number of input data vectors $\bar{x}$, and D their dimensionality.

3.2. ***Component Replication***

When a new class is introduced, the key points are:

- *When* a component is inserted (*decision process*);
- *How* the considered component is added (*replication process*).

Instead of deciding whether replicating a component or another, we sequentially replicate all the components following binary tree structure (sec. 3.2.1). Moreover, in this work we adopt a replication procedure (sec. 3.2.2) for adding a component rather than the splitting concept, due to its ill-posedness. This will result in a unique solution, side-stepping the original problem. Then the EM is run for optimizing the mixture and finally we decide if keeping the current solution of discarding it based on the cost function in evaluated by (1).

3.2.1. ***Binary tree decision structure***

We adopted a full binary tree - called also sometimes proper binary tree or 2-tree - which is a tree in which every node other than the leaves has two children. The binary tree is used for deciding which particular component will be replicated. The structure we use has the particularity that only the leaves contain a mixture component. The data structure organization is as follows:

- The initial tree starts with the root, only;
- Each node has no children (so that it is a leaf) or two children;
- Only the leaves can contain the mixture components; when a class is inserted by a leaf replication, the latter was the father and now it becomes a child together with the new inserted, creating a new parent without mixture components.
- The node eligible for being replicated are those of the last level only.

The binary tree serves only for the decision process, so that all the replicating possibilities are exploited.

3.2.2. ***Replication Process***

Rather than the splitting operation, the replication procedure admits a unique solution. This will side-step the ill-posedness of the original problem. A component $\bar{\vartheta}_{OLD}$ will be replicated exactly with its parameters $\bar{\vartheta}_{OLD} = \vartheta_{OLD-1}, \vartheta_{OLD-2}, \vartheta_{OLD-n}$, which

are the mean and the covariance matrix in the case of normal mixtures, into the new $\bar{\vartheta}_A$ and $\bar{\vartheta}_B$, with half of the original prior probability. Then, the EM steps will adapt the new components to best cover the input data.

However, it is clear that even though the EM algorithm is capable of modeling the new component, if two of them are exactly superimposed with the same mean and covariance matrix the EM will not be able to evolve separating them, and then converging to another (albeit local) minimum, different from the previous one. Therefore, here we introduce a small variation on the replication procedure. All the class parts (i.e. mean and covariance matrix in case of gaussian mixture) will be exactly cloned, except for the class location within the data, i.e. its mean. This will be replicated apart from an $\epsilon > 0$, with the assumption:

$$\begin{aligned} &\bar{\epsilon} = [\epsilon, \epsilon, \ldots, \epsilon] \\ &\bar{\mu}_A = \bar{\mu}_{OLD} + \bar{\epsilon}; \quad \bar{\mu}_B = \bar{\mu}_{OLD} - \bar{\epsilon} \\ &\lim_{\epsilon \to 0} \bar{\mu}_A = \lim_{\epsilon \to 0} \bar{\mu}_B = \bar{\mu}_{OLD} \end{aligned} \tag{3}$$

while the new *a-priori* probabilities will be:

$$w_A = \frac{1}{2} w_{OLD} \qquad w_B = \frac{1}{2} w_{OLD} \tag{4}$$

This small variation will make the EM to escape from the situation in which $\bar{\vartheta}_A$ and $\bar{\vartheta}_B$ are exactly superimposed, which corresponds to have the only $\bar{\vartheta}_{OLD} \equiv \bar{\vartheta}_A \equiv \bar{\vartheta}_B$ component.

3.3. ***Updating Mixture: Decision Procedure Iteration Step***

In our approach, at each new mixture configuration (addition of a component by means of the replicating operation) the original EM is performed in order to reach the local best optimization of that distribution configuration. Once the re-estimation of the vector parameter $\bar{\vartheta}$ has been computed in the EM step, our algorithm evaluates the current likelihood of each single component c as:

$$\Lambda_{curr(c)}(\vartheta) = \sum_{i=1}^{N} \ln(w_c \cdot p_c(\bar{x}_i)) \tag{5}$$

During each iteration, the algorithm keeps memory of the previous likelihood of each mixture component $\Lambda_{last(c)}(\vartheta)$.

Then, we defined our stopping criterion for the EM algorithm not only when the total distribution log-likelihood ceases increasing, but also when all each single component no longer modifies, i.e.:

$$\begin{aligned} &\sum_{i=1}^{c} |\Lambda_{incr(i)}(\vartheta)| \leqslant c \cdot \delta \\ &\Lambda_{incr(c)}(\vartheta) = \left| \frac{\Lambda_{curr(c)}(\vartheta) - \Lambda_{last(c)}(\vartheta)}{\Lambda_{curr(c)}(\vartheta)} \right| \cdot 100 \end{aligned} \tag{6}$$

where here $\Lambda_{incr\,(c)}(\vartheta)$ denotes the percentage increment in log-likelihood of the component c, $|\cdot|$ is the module, or absolute value of $(\cdot)$, and $\delta \ll 1$, e.g. $\delta = 0.098$. In our experiments (see sec. 5) we considered a log-likelihood not growing when the increment is below 0.098%. Besides, we used a percentage threshold rather than an absolute value for a better generalization.

This ensures that the before stopping the EM optimization no component is being updated, therefore a local optimum is really reached.

4. Application to a Gaussian mixture density

4.1. ***Replication Process***

It is worth noticing that the way the component is replicated greatly affects further computations. For instance, consider a 2-dimensional case, in which an *elongated* gaussian is present. This component may be approximating two components with diverse configurations: Either covering two smaller data distribution sets, placed along the longer axis, or two overlapped sets of data with different covariances, etc. So, replicating a component is a ill posed problem and the best way to solve it depends on the problem at hand. In this paper we make a choice suited to applications in color image segmentation whose purpose is to obtain components with lower overlap. For this case a reasonable way of replicating is to put the new means at the two major semi-axis' end points. Doing so, the new components will promote non overlapping components and, if the actual data set reflects this assumption, it will result in faster convergence.

To implement this replication operation we make use of the singular value decomposition. A rectangular n x p matrix A can be decomposed as $A = USV^T$, where the columns of U are the left singular vectors, S (which has the same dimension as A) is a diagonal matrix with the singular values arranged in descending order, and V^T has rows that are the right singular vectors.

More precisely, the original gaussians with parameters $\bar{\vartheta}_{OLD}$ when replicated will generate the two new components A and B, with means:

$$\begin{aligned} &\bar{\Sigma}_{OLD} = USV^T \\ &\bar{u}_d = U_{*,d} \quad s_d = S_{d,d} \\ &\bar{\mu}_A = \bar{\mu}_{OLD} + \epsilon s_d \bar{u}_d \quad \bar{\mu}_B = \bar{\mu}_{OLD} - \epsilon s_d \bar{u}_d \end{aligned} \tag{7}$$

where d is the selected dimension.

The covariance matrixes will then be updated as:

$$S_{d,d} = s_d \quad \Sigma_A = \Sigma_B = USV^T \tag{8}$$

where $\bar{u}_{max}$ is the first column of U, and s_d the d-dimensional element of S.

The new *a-priori* probabilities will be:

$$w_A = \frac{1}{2} w_{OLD} \quad w_B = \frac{1}{2} w_{OLD} \tag{9}$$

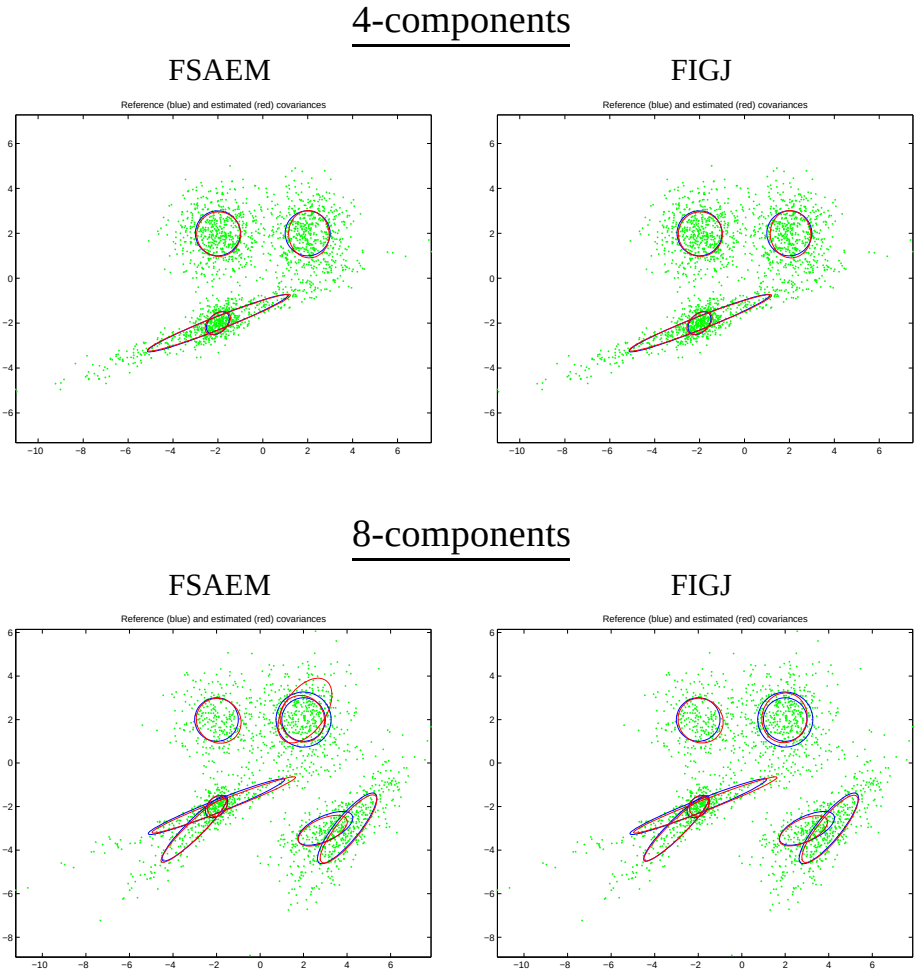


Figure 1. For each plot set: Generation mixture (blue) and the evaluated one (red) for FSAEM and FIGJ on the same input sets. Moreover, the 3D histograms the bottom in each subfigure represent: The generated mixture, our algorithm's estimated one, that estimated by [18], respectively.

5. Experimental Validation

We will compare our procedure versus three well-known *off-line* (see sec. 1.1) model selection criteria, AIC, BIC, and MML (in the version described by (1)), and [18], both on synthetic data (2D and 3D) and image segmentation. We evaluated our technique's performances by comparing it against these algorithms by applying them to different multi-dimensional input data, both synthetic data (artificially generated with a known mixture), and real images (taken by a webcam or by our robotic platform iCub's cameras). Although our proposed algorithm can be used for any kind of mixture, we mainly focussed on Gaussian mixture, due their generality and wide usage.

It is worth noticing that although that algorithm has been demonstrated to be robust and accurate for describing synthetic data, in [18] it has not been performed any experiment on image segmentation. Contrariwise, we want to focus more on image processing, due to its relevant importance in several different scientific fields, like robotics and medicine. Therefore, testing these algorithms on image segmentation is an important part of our study.

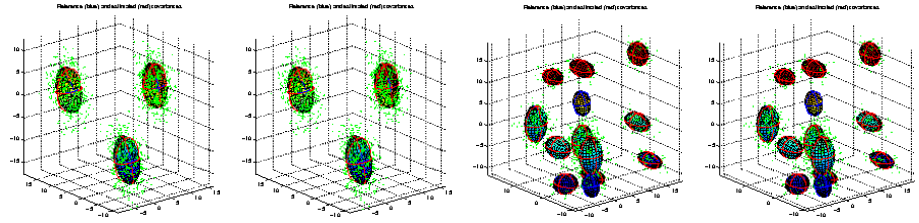
5.1. ***Initialization***

The initialization is a main issue. We overcome this by a unique setting, comprehending a single mixture component. It is worth noticing that our algorithm is deterministic: To the same input always corresponds the same output.

However, AIC, BIC, MML, and [18] rely on their initialization. Therefore, in order to reduce this artifact, we adopted a common used standard approach, i.e. we averaged the previous algorithms' outputs 100 times, starting with different initial random conditions.

2D Synthetic data											
Input	Algorithm	# inital components	# detected components	actual number of components	# iter	# time	# % our vs FIGJ time	log-likelihood	% log-likelihood our vs FIGJ	Normalized L2 Distance	% Norm L2 dist our vs FIGJ
4-components	AIC	from 1 to 11	4	4	66	3.811764	78.73612496	-7403.656573	15.17315842	6.595441	-16.27158247
	BIC	from 1 to 11	4		64	3.810045		-7405.021887		6.382962	
	MML	from 1 to 11	6		56	1.022003		-7453.510519		12.755668	
	FIGJ	11	4		142	3.312242		-8729.761818		0.073613	
	FSAEM	1	4		94	0.704311		-7405.181228		0.085591	
8-components	AIC	from 1 to 23	9	8	96	17.074475	74.59609099	-8400.626025	14.19557292	34.796101	99.94639739
	BIC	from 1 to 23	7		88	3.141896		-8428.323612		18.092732	
	MML	from 1 to 23	9		64	2.691401		-8535.472681		23.390192	
	FIGJ	23	7		231	17.74719		-9798.154848		13.866861	
	FSAEM	1	8		529	4.50848		-8407.250632		0.007433	

Table 1. Experimental results on synthetic data.



(a) 3-components FSAEM 3D estimated mixture (b) 3-components FIGJ 3D estimated mixture (c) 11-components FSAEM 3D estimated mixture (d) 11-components FIGJ 3D estimated mixture

Figure 2. 3D Gaussian mixtures

In our experiments we used $\epsilon = 2.7$ for 2D data, $\epsilon = 0.6$ for the 3 and 11 components 3D Gaussian mixture data, $\epsilon = 0.1$ for the 3D cylinder, the generic 3D figure and the color images, and $\epsilon = 0.007$ for the 3D shrinking spiral.

3D Synthetic data											
Input	Algorithm	# inital components	# detected components	actual number of components	# iter	# time	# % our vs FIGJ time	log-likelihood	% log-likelihood our vs FIGJ	Normalized L2 Distance	% Norm L2 dist our vs FIGJ
3D_3-components	AIC	from 1 to 8	5	3	47	0.566862	85.80533562	-13798.93271	25.22383455	578.789426	0.004692929
	BIC	from 1 to 8	4		54	0.5563		-13800.00784		486.775163	
	MML	from 1 to 8	5		52	0.558664		-13799.41579		693.330131	
	FIGJ	8	3		77	1.27183		-18444.52515		48.09363	
	FSAEM	1	3		29	0.180532		-13792.10864		48.091373	
3D_11-components	AIC	from 1 to 32	23	11	134	3.210384	66.96033444	-10679.3581	22.68386415	6248.514308	4.993862989
	BIC	from 1 to 32	16		176	3.141896		-10771.3229		5413.790139	
	MML	from 1 to 32	18		144	3.168279		-10741.69632		5865.453957	
	FIGJ	32	10		229	10.38861		-13818.98315		2086.57765	
	FSAEM	1	11		264	3.432362		-10684.30379		1982.376821	

Table 2. Experimental results on synthetic data.

5.2. ***Experiment 1: Synthetic data***

We divided these data in 2D and 3D input data set. Then, 2D sample are shared into *overlapping* and *non-overlapping* mixtures. Each input set is composed by 2000 points.

- *2D input data:* These comprehend both overlapping and non overlapping generation components. The description results are showed together with the original generation mixtures in Fig. 4.1.

- *3D input data:* We used two mixture components, one with 3 and the other with 11 generation classes. These are shown in Fig. 2.

5.3. ***Experiment 2: Colored real images***

To our knowledge this is the first time [18] is applied to real image segmentation. We segmented the images as 5-dimensional input in the (R,G,B) space and (x,y), i.e. a generic input point was of kind: $p \in (R, G, B, x, y)$. The color image segmentation results are shown in Fig. 5.3. The set of images is divided into two groups: Some general images, on the left (from (1) to (4)), and some images taken by the iCub's cameras, on the right (from (5) to (8)). For each group we show the original images, those obtained with [18], and those obtained with our algorithm on the left, in the middle, and on the right, respectively.

6. Discussions

Since now we use the following notation:

- FSAEM: Our algorithm;
- FIGJ: The approach proposed in [18].

We graphically showed the output results of our algorithm and that of [18] only, as exponents of the *on-line* techniques, in Fig. 4.1, and in Fig. 5.3, while AIC, BIC, and MML are illustrated in tab. 1. The plots in Fig. 4.1 are divided into non overlapping mixtures on the left, and overlapping ones on the right. Each subplot set is composed by the graphical output representation for the 2-D point distribution (top) and the 3-D estimation mixture histogram (bottom), composed by the generated mixture, our algorithm's estimated one, that estimated by [18], respectively. Here we show the representation for different mixtures of 4, 5, 8, and 16 gaussian components. The data plots show the generation mixture (blue) and the evaluated one (red). On the left the data result from our approach is shown, while on the right those of [18], relative to the same input data set.

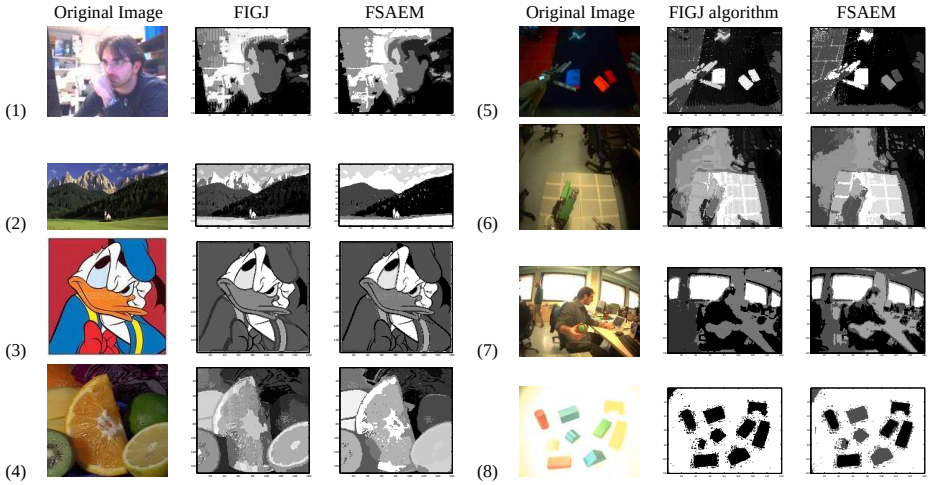


Figure 3. Color images segmentation. From image (1) to (4) we tested the algorithms on well-known images, or synthetic ones, and from (5) to (8) we exploit the algorithms possibilities on real images captured by our robotic platform iCub's cameras.

Real Images								
Input	Algorithm	# inital components	# detected components	# iter	# time	# % our vs FIGJ time	log-likelihood	% log-likelihood our vs FIGJ incr
1	FIGJ	18	18	392	263.307626	65.46098213	-468089.7361	10.72963553
	FSAEM	1	10	673	90.943868		-417865.4134	
2	FIGJ	20	20	330	391.247999	96.64602221	-527746.5813	13.74918089
	FSAEM	1	6	153	13.122371		-455185.7492	
3	FIGJ	23	23	335	533.479837	57.32574706	-685872.0772	21.06491496
	FSAEM	1	14	313	227.658535		-541393.7074	
4	FIGJ	34	34	466	1923.274434	88.74887061	-657578.044	14.00731491
	FSAEM	1	15	975	216.390095		-565469.0166	
5	FIGJ	34	34	514	2069.988404	95.65038327	-591302.6719	10.04804378
	FSAEM	1	13	598	90.036562		-531888.3205	
6	FIGJ	24	24	304	308.476958	97.31314486	-455378.5028	13.0785465
	FSAEM	1	7	104	8.288329		-395821.6135	
7	FIGJ	16	16	402	242.379629	93.25498638	-261622.9384	-65.16305039
	FSAEM	1	5	152	16.348539		-432104.4256	
8	FIGJ	2	2	18	2.603419	-26.88771957	-457033.9839	23.47007425
	FSAEM	1	3	58	3.303419		-349767.7685	

Table 3. Experimental results on real images segmentation.

In tab. 1 we report:

- The number of initial mixture components;
- The number of detected components;
- The actual number of components, i.e. that of the generation mixture;
- The number of total iterations;
- The elapsed time;
- The percentage difference in time for our algorithm $(Time_{FSAEM})$ with respect to [18] $(Time_{FIGJ})$, evaluated as $\frac{Time_{FIGJ}-Time_{FSAEM}}{Time_{FIGJ}} \cdot 100$;
- The final log-likelihood;
- The percentage difference in final log-likelihood for our algorithm (LL_{FSAEM}) with respect to [18] (LL_{FSAEM}), evaluated as $\frac{LL_{FIGJ}-LL_{FSAEM}}{LL_{FIGJ}} \cdot 100$;

- The normalized L2 distance to the generation mixture with respect to [18].

6.0.1. *Mixture precision estimation*

A deterministic approach for comparing the difference between the generation mixture and the evaluated one is to adopt a unique distance measure. In [19] Jensen *et Al.* exposed three different strategies for computing such distance: The Kullback-Leibler, the Earh Mover, and the Normalized L2 distance. The first one is not symmetric, even though a symmetrized version is usually adopted in music retrival. However, this measure can be evaluated in a close form only with mono-dimensional gaussians. The second one also suffers analog problems of the latter. The third choice, finally is symmetric, obeys to the triangle inequality and it is easy to compute, with a comparable precision with the other two. We then used the last one. Its expression states [20]:

$$
\begin{aligned}
z_c N_x(\bar{\mu}_c, \bar{\Sigma}_c) &= N_x(\bar{\mu}_a, \bar{\Sigma}_a) \cdot N_x(\bar{\mu}_b, \bar{\Sigma}_b) \\
&\quad where \\
\bar{\Sigma}_c &= (\bar{\Sigma}_a^{-1} + \bar{\Sigma}_b^{-1})^{-1} \quad and \quad \bar{\mu}_c = \bar{\Sigma}_c(\bar{\Sigma}_a^{-1}\bar{\mu}_a + \bar{\Sigma}_b^{-1}\bar{\mu}_b) \\
z_c &= |2\pi\bar{\Sigma}_a\bar{\Sigma}_b\bar{\Sigma}_c^{-1}|^{\frac{1}{2}} exp\left\{-\frac{1}{2}(\bar{\mu}_a - \bar{\mu}_b)^T\bar{\Sigma}_a^{-1}\bar{\Sigma}_c\bar{\Sigma}_b^{-1}(\bar{\mu}_a - \bar{\mu}_b)\right\} \\
&= |2\pi(\bar{\Sigma}_a + \bar{\Sigma}_b)|^{\frac{1}{2}} exp\left\{-\frac{1}{2}(\bar{\mu}_a - \bar{\mu}_b)^T(\bar{\Sigma}_a + \bar{\Sigma}_b)^{-1}(\bar{\mu}_a - \bar{\mu}_b)\right\}
\end{aligned}
\tag{10}
$$

6.0.2. *Discussion*

Our algorithm is better for image segmentation, because it is fast, it does not need any initialization, and gives rise to better results in terms of the compromise between computational complexity and sufficient segmentation. One can argue that the AIC, BIC, and MML criterion together with FIGJ lead to more accurate results, while our approach underestimates the right number of components. Our approach demonstrates to be able to produce images enough segmented to be realistically reproduced without considerable loss of details (see Fig. 5.3). We think this rely on the requirements of the application that uses the algorithm. On one hand, if one desires the best segmentation can simply use one of the four tested approaches, with a high number of classes. On the other hand, if the best compromise between simplicity of representation (and therefore lower computational burden for a likely further processing) and accuracy together with a reasonable elapsed time is a must, our approach demonstrates to be the best one.

7. Conclusion

In this paper we proposed a unsupervised algorithm that learns a finite mixture model from multivariate data on-line. The algorithm can be applied to any data mixture where the EM can be used. We approached the problem from the opposite way of [18], i.e. by starting from only one mixture component instead of several ones and progressively adapting the mixture by adding new components when necessary. Our algorithm starts from a single mixture component and sequentially both increases the number of components and adapting their means and covariances. Therefore, due to its unique initializa-

tion it is not affected by different possible starting points like the original EM formulation. Moreover, by starting with a single component the computational burden is low at the beginning, increasing only whether more components are required. Finally, we presented the effectivity of our technique in a series of simulated experiments with synthetic data and real images, and we compared the results against the approach proposed in [18].

Acknowledgements

This work was supported by the European Commission, Project IST-004370 RobotCub and FP7-231640 Handle, and by the Portuguese Government - Fundação para a Ciência e Tecnologia (ISR/IST pluriannual funding) through the PIDDAC program funds and through project BIO-LOOK, PTDC / EEA-ACR / 71032 / 2006.

References

[1] T. Kohonen, "Analysis of a simple self-organizing process." *Biological Cybernetics*, vol. 44, no. 2, pp. 135–140, 1982.

[2] B. Fritzke, "A growing neural gas network learns topologies." *Adv ances in Neural Inform ation Processing Systems 7 (NIPS'94), MIT Press, Cambridge MA*, pp. 625–632, 1995.

[3] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations." *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability.*, pp. 281–297, 1967.

[4] P. Comon, "Independent component analysis: a new concept?" *Signal Processing, Elsevier*, vol. 36, no. 3, pp. 287–314, 1994.

[5] A. Hyvärinen, J. Karhunen, and E. Oja, "Independent component analysis," *New York: John Wiley and Sons*, vol. ISBN 978-0-471-40540-5, 2001.

[6] G. McLachlan and D. Peel, "Finite mixture models." *John Wiley and Sons*, 2000.

[7] G. McLachlan and K. T., "The em algorithm and extensions," *New York: John Wiley and Sons*, 1997.

[8] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood estimation from incomplete data via the em algorithm," *J. Royal Statistic Soc.*, vol. 30, no. B, pp. 1–38, 1977.

[9] L. Xu and J. M., "On convergence properties of the em algorithm for gaussian mixtures," *Neural Computation*, vol. 8, pp. 129–151, 1996.

[10] Y. Sakimoto, M. Iahiguro, and G. Kitagawa, "Akaike information criterion statistics," *KTK Scientific Publisher, Tokio*, 1986.

[11] G. Schwarz, "Estimating the dimension of a model," *Ann. Statist.*, vol. 6, no. 2, pp. 461–464, 1978.

[12] J. Rissanen, "Stochastic complexity in statistical inquiry." *Wold Scientific Publishing Co. USA*, 1989.

[13] C. Wallace and P. Freeman, "Estimation and inference by compact coding," *J. Royal Statistic Soc. B*, vol. 49, no. 3, pp. 241–252, 1987.

[14] F. Pernkopf and D. Bouchaffra, "Genetic-based em algorithm for learning gaussian mixture models," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 27, no. 8, pp. 1344–1348, 2005.

[15] N. Vlassis and A. Likas, "A greedy em algorithm for gaussian mixture learning," *Neural Processing Letters*, vol. 15, pp. 77–87, 2002.

[16] J. Verbeek, N. Vlassis, , and B. Krose, "Efficient greedy learning of gaussian mixture models," *Neural Computation*, vol. 15, no. 2, pp. 469–485, 2003.

[17] A. Lanterman, "Schwarz, wallace and rissanen: Intertwining themes in theories of model order estimation," *Int'l Statistical Rev.*, vol. 69, pp. 185–212, 2001.

[18] A. Figueiredo and A. Jain, "Unsupervised learning of finite mixture models," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 24, no. 3, 2002.

[19] J. H. Jensen, D. Ellis, M. G. Christensen, and S. H. Jensen, "Evaluation distance measures between gaussian mixture models of mfccs," *Proc. Int. Conf. on Music Info. Retrieval ISMIR-07 Vienna, Austria*, pp. 107–108, October, 2007.

[20] P. Ahrendt, "The multivariate gaussian probability distribution," http://www2.imm.dtu.dk/pubdb/p.php?3312, Tech. Rep., January 2005.