

# A Particular Gaussian Mixture Model for Clustering

Hichem Sahbi

Machine Intelligence Laboratory and Certis Laboratory  
Department of Engineering      Ecole Nationale des Ponts et Chaussees  
Cambridge University, UK      France  
hs385@cam.ac.uk,      hichem.sahbi@enpc.fr

**Abstract.** We introduce a new approach for data clustering based on Gaussian Mixture Models (GMMs). Each cluster of data, modeled as a GMM in an input space, is interpreted as a hyperplane in a high dimensional mapping space where the underlying coefficients are found by solving a quadratic programming problem (QP). The main contribution is an approach for GMM estimation which requires only finding the mixture parameters in contrast to other density estimation methods which are very sensitive to initialization. Furthermore, the introduced QP is solved easily using a new decomposition algorithm which involves trivial linear programming sub-problems. The validity of the method is demonstrated for clustering  $2D$  toy examples as well as image databases.

## 1 Introduction

Consider a training set  $\mathcal{S}_N = \{x_1, \dots, x_N\}$  generated i.i.d. according to an unknown but a fixed probability distribution  $P(X)$ . Here  $X$  is a random variable standing for training data in an *input space* denoted  $\mathcal{X}$ . Our goal is to define a partition of this training set, i.e., assign each training example to its actual cluster  $y_k$ ,  $k = 1, \dots, C$ . Existing clustering methods can be categorized depending on the “crispness” of data memberships, i.e., whether each training example belongs to only one cluster or not. The first category includes hierarchical approaches [1, 2], EM (Expectation Maximization) [3],  $C$ -means [4], self organizing maps (SOMs) [5] and recently original approaches based on kernel methods [6, 7]. In the second family the decision as whether a training example belongs to one or another cluster is fuzzy and a family of algorithms dealing with fuzziness exist in the literature for instance [8, 9]. All these methods have been used in different domains including Gene expression [10], image segmentation [11] and database categorization [12].

One of the main issues in the existing clustering methods remains setting the appropriate number of classes for a given problem. Many methods for instance hierarchical agglomeration [13, 14, 2] has proven to perform well when the application allows us to know a priori the number of clusters or when the user sets it manually. Of course, the estimation of this number is application-dependent, for

instance in image segmentation it can be set a priori to the number of targeted regions. Unfortunately, for some applications such as database categorization, it is not always possible to predict automatically and even manually the appropriate number of classes.

Given a training set  $\mathcal{S}_N$  in the input space  $\mathcal{X}$ , in our method each cluster in  $\mathcal{X}$  is modeled as a GMM and interpreted as a hyperplane in a high dimensional space referred to as the mapping space, denoted  $\mathcal{H}$ . Clustering consists in maximizing a likelihood function of data memberships and is equivalent to finding the parameters of the cluster hyperplanes by solving a QP problem. We will show that this QP can be tackled efficiently by solving trivial linear programming sub-problems. Notice that when solving this QP, the number of clusters (denoted  $C$ ), fixed initially, might be overestimated and this leads to several overlapping clusters; therefore the actual clusters are found as constellations of highly correlated hyperplanes in the mapping space  $\mathcal{H}$ .

In the remainder of this paper, we refer to a cluster as a set of data gathered using an algorithm while a class (or a category) is the actual membership of this data according to a well defined ground truth. Among notations,  $i$  and  $j$  stand for data indices while  $k$ ,  $c$  and  $l$  stand for cluster indices. Other notations will be introduced as we go along through different sections of this paper which is organized as following: In §2 we provide a short reminder on GMMs followed by our formulation in §3. In §4 we reformulate the clustering minimization problem as a succession of simple linear programming subproblems which are solved efficiently. We present in §5 experiments on simple as well as challenging problems in content based image retrieval. Finally, we conclude in §6 and we provide some directions for a future work.

## 2 A Short Reminder on GMMs

Given a training set  $\mathcal{S}_N$  of size  $N$ , we are interested in a particular Gaussian Mixture Model (denoted  $\mathcal{M}_k$ ) [15] where the number of its components is equal to the size of the training set. The parameters of  $\mathcal{M}_k$  are denoted  $\Theta_k = \{\Theta_{i|k}, i = 1, \dots, N\}$ . Here  $\Theta_{i|k}$  stands for the  $i^{th}$  component parameter of  $\mathcal{M}_k$ , i.e., the mean and the covariance matrices of a Gaussian density function. In this case, the output of the likelihood GMM function related to a cluster  $y_k$  is a weighted sum of  $N$  component densities:

$$p(x|y_k) = \sum_i^N P(\Theta_{i|k}|y_k) p(x|y_k, \Theta_{i|k}) \quad (1)$$

here  $P(\Theta_{i|k}|y_k)$  (also denoted  $\mu_{ik}$ ) is the prior probability of the  $i^{th}$  component parameter  $\Theta_{i|k}$  given the cluster  $y_k$  and  $p(x|y_k, \Theta_{i|k}) = \mathcal{N}_k(x; \Theta_{i|k})$  is the normal density function of the  $i^{th}$  component. In order to guarantee that  $p(x|y_k)$  is a density function, the mixture parameters  $\{\mu_{ik}\}$  are chosen such that  $\sum_i \mu_{ik} = 1$ .

Usually the training of GMMs can be formulated as a maximum likelihood problem where the parameters  $\Theta_k$  and the mixture parameters  $\{\mu_{ik}\}$  are estimated using expectation maximization [15].

We consider in our work,  $\mathcal{N}_k(x; \Theta_{i|k})$  with a *fixed* mean  $x_i \in \mathcal{S}_N$  and covariance matrix  $\sigma \Sigma_i$  ( $\Sigma_i \in \mathbb{R}^{p \times p}$ ,  $\sigma \in \mathbb{R}$ ). Now, each cluster  $y_k$  is modeled as a GMM where the *only* free parameters are the mixture coefficients  $\{\mu_{ik}\}$ ; the means and the covariances are assumed constant but of course dependent on a priori knowledge of the training set (cf. §5).

### 3 Clustering

A “good” clustering algorithm should make clusters, containing data from different classes, as different as possible while keeping data from the same classes as close as possible to their *actual* clusters. Usually this implies the optimization of an objective function (see for instance [16]) involving a fidelity term which measures the distance of each training example to its model and a regularizer which reduces the number of clusters, i.e., the complexity.

#### 3.1 Our Formulation

Following notations and definitions in §2, a given  $x$  is assigned to a cluster  $y_k$  if:

$$y_k = \arg \max_{y_l} p(x|y_l) \quad (2)$$

here the weights  $\mu = \{\mu_{il}\}$ , of the GMM functions  $p(x|y_l)$   $l = 1, \dots, C$ , are found by solving the following constrained minimization problem (cf. below):

$$\begin{aligned} \min_{\mu} \sum_{k,i} \mu_{ik} \left( \sum_{l \neq k} p(x_i|y_l) \right) \\ \text{s.t. } \sum_i \mu_{ic} = 1, \quad \mu_{ic} \in [0, 1], \quad c = 1, \dots, C \end{aligned} \quad (3)$$

In the above objective function, the training data belonging to a cluster  $y_l$  are assumed drawn from a GMM with a likelihood function  $p(x|y_l)$ . Each mixture parameter  $\mu_{il}$  stands for the degree of membership (or the contribution) of the training example  $x_i$  to the cluster  $y_l$ . The overall objective is to maximize the membership of each training example to its actual cluster while keeping the memberships to the remaining clusters relatively low. Usually, existing clustering methods (see for instance [8]) find the parameters  $\{\mu_{il}\}$  as those which maximize the membership of each training example to its actual cluster. In contrast to these methods, our formulation proceeds using a dual principle; *the purpose is to minimize the memberships of training examples to their **non-actual** clusters.*

Using (1), we can expand the objective function (3) as:

$$\min_{\mu} \sum_{k \neq l} \sum_{i,j} \mu_{ik} \mu_{jl} \mathcal{N}_l(x_i; \Theta_{j|l}) \quad (4)$$

here  $\mathcal{N}_l(x_i; \Theta_{j|l})$  is the response of a Gaussian density function (also referred to as kernel), with fixed parameters  $\Theta_{j|k} = \{x_j, \sigma, \Sigma_j\}$ ;  $x_j$  is the mean,  $\Sigma_j$  is the covariance and  $\sigma$  is the scale. We will denote this kernel simply as  $K_{\sigma}(\|x_i - x_j\|)$ . It is known that the Gaussian kernel is positive definite [17] so this function corresponds to a scalar product in the mapping space  $\mathcal{H}$ , i.e., there is a mapping  $\Phi_{\sigma}$  from the input space into an infinite dimensional space such that  $K_{\sigma}(\|x_i - x_j\|) = \langle \Phi_{\sigma}(x_i), \Phi_{\sigma}(x_j) \rangle$  where  $\langle \rangle$  stands for the inner product in  $\mathcal{H}$ . At this stage, the response of a GMM function  $p(x|y_k)$  is equal to the scalar product  $\langle \omega_k, \Phi_{\sigma}(x) \rangle$  where  $\omega_k = \sum_i \mu_{ik} \Phi_{\sigma}(x_i)$  is the normal of a hyperplane in  $\mathcal{H}$ . Now, the objective function (4) can be rewritten:

$$\min_{\mu} \sum_{k \neq l} \langle \omega_k, \omega_l \rangle \quad (5)$$

The above objective function minimizes the sum of hyperplane *correlations* taken pairwise among all different clusters. Now, we can derive the new form of the constrained minimization problem (3):

$$\begin{aligned} \min_{\mu} \sum_{k,i} \sum_{l \neq k,j} \mu_{ik} \mu_{jl} K_{\sigma}(\|x_i - x_j\|) \\ \text{s.t. } \sum_i \mu_{ic} = 1, \quad \mu_{ic} \in [0, 1], \quad c = 1, \dots, C \end{aligned} \quad (6)$$

This defines a constrained QP which can be solved using standard QP libraries (see for instance [18]). When solving this problem, training examples  $\{x_i\}$  for which the mixing parameter  $\{\mu_{ik}\}$  are positive will be referred to as the *GMM vectors* of the cluster  $y_k$ .

## 4 Training

The number of parameters intervening in (6) is  $N \times C$ , so for clustering problems of reasonable size, for instance  $N = 1.000$  and  $C = 20$ , solving this QP, using standard packages, can quickly get out of hand. Chunking methods have been successfully used to solve QP for large scale training problems such as SVM [19]. The idea consists in solving a QP problem using an active subset of parameters, referred to as a *chunk*, which is updated iteratively. When the QP is convex, the process is guaranteed to converge to the global optimum after a sufficient number of iterations [19].

Using the same principle as [19, 20], we will show in this section that for a particular choice of the active chunk, the QP (6) can be decomposed into linear programming subproblems each one can be solved trivially.

## 4.1 Decomposition

Let us fix one cluster index  $p \in \{1, \dots, C\}$  and rewrite the objective function (6) as:

$$\min_{\mu} 2 \sum_i \mu_{i\mathbf{p}} c_{ip} + \sum_{i,k \neq p} \sum_{j,l \neq k,p} \mu_{ik} \mu_{jl} K_{\sigma}(\|x_i - x_j\|) \quad (7)$$

here:

$$c_{ip} = \sum_{j,l \neq p} \mu_{jl} K_{\sigma}(\|x_i - x_j\|) \quad (8)$$

Consider a chunk  $\{\mu_{ip}\}_{i=1}^N$  and fix  $\{\mu_{jk}, k \neq p\}_{j=1}^N$ , the objective function (7) is linear in terms of  $\{\mu_{ip}\}$  and can be solved, with respect to this chunk, using linear programming. Only one equality constraint  $\sum_i \mu_{ip} = 1$  is taken into account in the linear programming problem as the other equality constraints in (6) are independent from the chunk  $\{\mu_{ip}\}$ .

We can further reduce the size of the chunk  $\{\mu_{ip}\}_{i=1}^N$  to only two parameters. Given  $i_1, i_2 \in \{1, \dots, N\}$ , we can rewrite (7) as:

$$\begin{aligned} & \min_{\mu_{i_1 p}, \mu_{i_2 p}} 2(\mu_{i_1 \mathbf{p}} c_{i_1 p} + \mu_{i_2 \mathbf{p}} c_{i_2 p}) + b \\ \mathbf{s.t} \quad & \mu_{i_1 p} + \mu_{i_2 p} = 1 - \sum_{i \neq i_1, i_2} \mu_{ip} \\ & 0 \leq \mu_{i_1 p} \leq 1 \\ & 0 \leq \mu_{i_2 p} \leq 1 \end{aligned} \quad (9)$$

here  $b$  is a constant independent from the chunk  $\{\mu_{i_1 p}, \mu_{i_2 p}\}$  (see; 7). When  $c_{i_1 p} = c_{i_2 p}$  the above objective function and the equality constraints are linearly dependent, so we have an infinite set of optimal solutions; any *one* which satisfies the constraints in (9) can be considered. Let us assume  $c_{i_1 p} \neq c_{i_2 p}$  and denote  $d = 1 - \sum_{i \neq i_1, i_2} \mu_{ip}$ , since  $\mu_{i_2 p} = d - \mu_{i_1 p}$ , the above linear programming problem can be written as:

$$\begin{aligned} & \min_{\mu_{i_1 p}} \mu_{i_1 \mathbf{p}} (c_{i_1 p} - c_{i_2 p}) \\ \mathbf{s.t} \quad & \max(d - 1, 0) \leq \mu_{i_1 p} \leq \min(d, 1) \end{aligned} \quad (10)$$

Depending on the sign of  $c_{i_1 p} - c_{i_2 p}$ , the solution of (10) is simply taken as one of the bounds of the inequality constraint in (10). At this stage, the parameters  $\{\mu_{1p}, \dots, \mu_{Np}\}$  which solve (7) are found by solving iteratively the trivial maximization problem (10) for different chunks, as shown in **Algorithm1**, and the whole QP (6) is solved by looping several times through different cluster indices as shown in **Algorithm2**.

---

**Algorithm1**( $p, \mu$ )

---

**Do** the following steps **ITERMAX1** iterations  
Select randomly  $(i_1, i_2) \in \{1, \dots, N\}^2$ .  
 $(c_{i_1 p}, c_{i_2 p}) \leftarrow (8)$ .  
**if**  $(c_{i_1 p} - c_{i_2 p} < 0)$   $\mu_{i_1 p} \leftarrow \min(d, 1)$  (see; 10)  
**else**  $\mu_{i_1 p} \leftarrow \max(d - 1, 0)$   
 $\mu_{i_2 p} \leftarrow d - \mu_{i_1 p}$  (see; 9)  
**endDo**  
**return**  $(\mu_{i_1 p}, \dots, \mu_{i_N p})$

---

---

**Algorithm2**

---

Set  $\{\mu\}$  to random values.  
**Do** the following steps **ITERMAX2** iterations  
Fix  $p$  in  $\{1, \dots, C\}$ , update  $\mu_{1p}, \dots, \mu_{Np}$  using:  
 $(\mu_{1p}, \dots, \mu_{Np}) \leftarrow \text{Algorithm1}(p, \mu)$   
**endDo**

---

## 4.2 Agglomeration

Notice that:

$$\frac{\partial \langle \omega_k, \omega_l \rangle}{\partial \sigma} = 2 \sum_{i,j} \mu_{ik} \mu_{jl} \frac{K_\sigma(\|x_i - x_j\|)}{\sigma^3} \geq 0 \quad (11)$$

so the correlation is a convex increasing function of the scale  $\sigma$ . Assuming  $\|w_k\| = \|w_l\| = 1$ , it results that  $\forall \tau \in [0, 1], \exists \sigma$  such that  $\langle \omega_k, \omega_l \rangle \geq \tau$ . Let us define the following adjacency matrix:

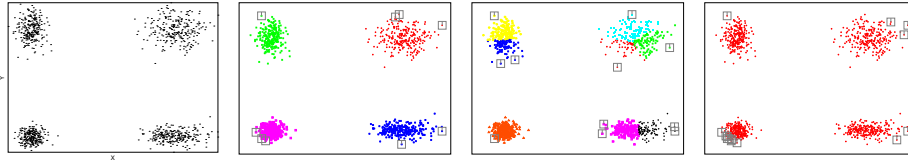
$$A_{k,l} = \begin{cases} 1 & \text{if } \langle \omega_k, \omega_l \rangle \geq \tau \\ 0 & \text{otherwise} \end{cases} \quad (12)$$

This matrix characterizes a graph where a node is a hyperplane and an arc corresponds to two highly correlated hyperplanes (i.e.,  $\langle \omega_k, \omega_l \rangle \geq \tau$ ). Now, the actual number of clusters is found as the number of connected components in this graph. For a fixed threshold  $\tau$ , the scale parameter  $\sigma$  acts as a *regularizer*. When  $\sigma$  is overestimated, the graph  $A$  will be connected resulting into only one big cluster whereas a small  $\sigma$  results into lots of disconnected subgraphs, therefore the total number of clusters will be large (cf. Figure 1).

## 4.3 Toy Examples

These experiments are targeted to show the performance brought by this decomposition algorithm both in term of precision and speed. We consider a two dimensional training set, of size  $N$ , generated using four Gaussian densities centered at four different locations (see; Figure 1, left). We cluster these data using

**Algorithm2**, for different values of  $N$  (40, 80, 100, 500, 1000). For  $N = 100$ , table (1) shows all the pairwise correlations between the hyperplanes found (i) when running **Algorithm2** and (ii) when solving the whole QP (6) using a standard package LOQO [18]. We can see that each hyperplane found using **Algorithm2** is highly correlated with *only* one hyperplane found using LOQO. Table (2) shows a comparison of the underlying runtime performances.



**Fig. 1.** (Left) Data randomly generated from four Gaussian densities centered at four different locations ( $N = 1000$ ). When running *Algorithm2*,  $C$  is fixed to 20, so the total number of parameters in the training problem is  $20 \times 1000$ . (Other Figures) Different clusters are shown with different colors and the underlying GMM vectors are shown in gray. In these experiments  $\sigma$  is set respectively to 10, 4 and 50.

**Table 1.** This table shows the correlations (normalized scalar products) between the hyperplane normals found, when (i) running **Algorithm2** and (ii) solving the QP (6) using the LOQO package, on the 2D toy data of Figure (1). ( $\sigma = 10$ ).

LOQO pack vs. Algorithm2	Cluster 1	Cluster 2	Cluster 3	Cluster 4
Cluster 1	<b>0.994</b>	0.043	0.331	0.187
Cluster 2	0.036	<b>0.999</b>	0.238	0.133
Cluster 3	0.284	0.313	<b>0.985</b>	0.058
Cluster 4	0.152	0.147	0.057	<b>0.998</b>

## 5 Database Categorization

Experiments have been conducted on both the Olivetti and Columbia standard databases in order to show the good performance of our clustering method. The Olivetti database contains 40 persons each one represented by 10 faces. The Columbia set contains 100 categories of objects each one represented by 72 images. Each image from both the Olivetti and Columbia datasets is processed using histogram equalization and encoded using 20 coefficients of projection into a subspace learned using ISOMAP [21]. Notice that ISOMAP embeds a training

**Table 2.** Comparison of the run-time performances on the training samples in Figure (1) for different values of  $N$ . ( $C = 20, \sigma = 10$ ). These experiments were run on a 3 Ghz Pentium III PC.

# $S$ ( $N$ )	40	80	100	500
# of parameters in the QP ( $N \times C$ )	280	480	700	3500
LOQO pack	24.6 (s)	87.9 (s)	317.8(s)	> 1 (H)
Algorithm2	0.09 (s)	0.32 (s)	0.56 (s)	24.99 (s)

database into a subspace such that non-linearly separable classes become more separable, homogeneous in terms of scale, and easier to cluster.

**Table 3.** This table shows the distribution of the categories through the clusters on the Olivetti set using our clustering algorithm. We can see that this distribution is concentrated near the diagonal. The scale parameter  $\sigma$  is set to 24 and  $C = 30$ . Errors in the cardinalities of the clusters are mentioned in red.

categories	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	total	
clusters																						
1:	9	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	9
10:	.	10	.	1	3	.	.	.	1	.	.	.	.	.	.	.	.	.	.	3	.	18
15:	.	.	10	.	.	.	.	.	.	.	.	.	.	.	.	.	.	5	.	.	.	15
3:	.	.	.	10	.	.	.	3	.	.	.	.	.	.	.	.	2	.	.	.	.	15
4:	.	.	.	.	9	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.	10
26:	.	.	.	.	.	5	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	5
24:	.	.	.	.	.	.	5	.	2	.	.	.	.	.	.	.	.	.	.	.	.	7
19:	.	.	.	.	1	.	.	.	9	.	.	.	.	.	.	.	.	.	.	2	.	12
27:	.	.	.	.	.	.	.	.	.	3	.	.	.	.	.	.	.	.	.	.	.	3
25:	1	.	.	.	.	.	.	.	.	.	4	5	.	2	.	.	.	.	.	.	.	12
13:	.	.	.	.	.	.	.	.	.	5	.	.	.	.	.	.	.	.	.	.	.	5
20:	.	.	.	.	.	.	.	.	.	.	6	3	.	.	.	.	.	.	.	.	.	9
11:	.	.	.	.	.	.	.	.	.	.	.	2	10	.	1	.	.	.	.	.	.	13
6:	.	.	.	.	1	.	.	.	.	.	.	.	.	8	.	.	.	.	.	.	.	9
16:	.	.	.	.	1	.	.	.	.	.	.	.	.	.	9	.	.	.	.	.	.	10
9:	.	.	.	.	4	1	2	2	.	.	.	.	.	.	.	8	2	.	.	.	.	19
2:	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	3	.	5	.	.	8
17:	.	.	.	.	.	4	.	.	.	.	.	.	.	.	.	.	.	.	7	.	.	11
7:	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	2	2
total	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10

It might be easier for database categorization to predict the variance of data rather than the number of clusters, mainly for large databases living in high dimensional spaces. As the number of clusters is initially set to an overestimated value (cf. §4.2), our method relies mainly on one parameter, i.e., the scale  $\sigma$ , which makes it possible to find automatically the actual number of clusters. In our experiments, we predict  $\sigma$  by sampling manually few images from some categories, estimating the scales of the underlying classes, then setting  $\sigma$  to the expectation of the scale through these classes. While this setting is not auto-

matic, it has at least the advantage of introducing a priori knowledge on the variance of the data at the expense of few interactions and reasonable effort from the user. Tables 3 and 4 show the confusion (or contingency) matrices on the Olivetti and Columbia datasets where most of the distribution is concentrated in the diagonal and this clearly shows that most of the training data are assigned to their actual clusters.

**Table 4.** Same experiments as above on the Columbia set.  $\sigma = 0.4$  and  $C = 20$ .

categories clusters	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	total
9:	72	.	.	.	.	.	.	.	.	.	.	.	.	.	.	72
1:	.	72	.	.	.	.	.	.	.	.	.	.	.	.	.	72
11:	.	.	72	.	.	.	.	.	.	.	.	.	.	.	.	72
7:	.	.	.	72	.	.	.	.	.	.	.	.	.	.	.	72
6:	.	.	.	.	72	.	.	.	.	.	.	.	.	.	.	72
10:	.	.	.	.	.	37	.	34	.	.	.	.	.	.	.	71
14:	.	.	.	.	.	35	.	38	.	.	.	.	.	.	.	73
5:	.	.	.	.	.	.	72	.	.	.	.	.	.	.	.	72
15:	.	.	.	.	.	.	.	.	72	.	.	.	.	.	.	72
0:	.	.	.	.	.	.	.	.	.	72	.	.	.	.	.	72
2:	.	.	.	.	.	.	.	.	.	.	72	.	.	.	.	72
4:	.	.	.	.	.	.	.	.	.	.	.	72	.	.	.	72
8:	.	.	.	.	.	.	.	.	.	.	.	.	30	.	.	30
13:	.	.	.	.	.	.	.	.	.	.	.	.	42	72	72	186
total	72	72	72	72	72	72	72	72	72	72	72	72	72	72	72	72

## 6 Conclusion and Future Work

We introduced in this work an original approach for clustering based on a particular Gaussian Mixture Models (GMMs). The method considers an objective function which acts as a regularizer and minimizes the overlap between the cluster GMMs. The GMM parameters are found by solving a quadratic programming problem using a new decomposition algorithm which considers trivial linear programming subproblems. The actual number of clusters is found by controlling the scale parameter of these GMMs; in practice, it turns out that predicting this parameter is easier than predicting the actual number of clusters mainly for large databases living in high dimensional spaces.

The concept presented in this paper is different from kernel regression where the latter might be assimilated to density estimation while our approach perform such estimation for each cluster. Obviously, the proposed approach performs clustering and density estimation at the same time and its validity is demonstrated on toy data as well as database categorization problems.

## References

1. A.K. Jain and R. C. Dubes, "Algorithms for clustering data," *Prentice Hall*, 1988.

2. C. Posse, "Hierarchical model-based clustering for large datasets," *Journal of Computational and Graphical Statistics*, vol. 10, no. 3, pp. 464–486, 2001.
3. A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of Royal Statistical Society. Ser. B*, vol. 39, no. 1, pp. 1–38, 1977.
4. J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, 1965.
5. E.K. Tsao, J.C. Bezdek, and N.R. Pal, "Fuzzy kohonen clustering networks," *Pattern Recognition*, vol. 27, no. 5, pp. 757–764, May 1994.
6. A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik, "Support vector clustering," In *Neural Information Processing Systems*, pp. 367–373, 2000.
7. F.R. Bach and M.I. Jordan, "Learning spectral clustering," In *Neural Information Processing Systems*, 2003.
8. J.C. Bezdek, "Pattern recognition with fuzzy objective function algorithms," *Kluwer Academic Publishers, Norwell, MA*, 1981.
9. R.N. Dave, "Characterization and detection of noise in clustering," In *Pattern Recognition*, vol. 12, no. 11, pp. 657–664, 1991.
10. C.A. Orengo, D.T. Jones, and J.M. Thornton, "Bioinformatics - genes, protein and computers. bios," *ISBN: 1-85996-054-5*, 2003.
11. C. Carson, S. Belongie, H. Greenspan, and J. Malik, "Blobworld: Image segmentation using expectation-maximization and its application to image querying," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 8, pp. 1026 – 1038, 2002.
12. B. Le-Saux and N. Boujemaa, "Unsupervised robust clustering for image database categorization," *IEEE-IAPR International Conference on Pattern Recognition*, pp. 259–262, 2002.
13. P.H. Sneath and R.R. Sokal, "Numerical taxonomy- the principles and practice of numerical classification," In *W. H. Freeman, San Francisco*, 1973.
14. C. Fraley, "Algorithms for model-based gaussian hierarchical clustering," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 270–281, 1998.
15. C.M. Bishop, *Neural networks for pattern recognition*, Clarendon Press, Oxford, 1995.
16. H. Frigui and R. Krishnapuram, "Clustering by competitive agglomeration," In *Pattern Recognition*, vol. 30, no. 7, 1997.
17. N. Cristianini and J. Shawe-Taylor, "An introduction to support vector machines," *Cambridge University Press*, 2000.
18. R. J. Vanderbei, "LOQO: An interior point code for quadratic programming," *Optimization Methods and Software*, vol. 11, pp. 451–484, 1999.
19. E. Osuna, R. Freund, and F. Girosi, "Training support vector machines: an application to face detection," In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pp. 130–136, 1997.
20. J. Platt, "Fast training of support vector machines using sequential minimal optimization," In *B. Scholkopf and C. Burges and A. J. Smola editors. Advances in Kernel Methods — Support Vector Learning. Cambridge. MA. MIT Press*, pp. 185–208, 1999.
21. J. Tanenbaum, V de Silva, and J Langford, "A global geometric framework for non-linear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.