

# GAMIXTURE - Matlab toolbox documentation.

## Version 1.1

February 19, 2007

### 1 Introduction

This is a Matlab toolbox for optimizing finite mixture models (FMM) parameters. The algorithm is based on real coded genetic algorithms (GAs). The algorithm was designed to be used in applications for voxel classification in brain imaging, but it is not limited to that application. However, the terminology used in this document refers to the voxel classification application.

The algorithm is described in

J. Tohka, E. Krestyannikov, I.D. Dinov, A. MacKenzie-Graham, D.W. Shattuck, U. Ruotsalainen ja A.W. Toga. Genetic algorithms for finite mixture model based voxel classification in neuroimaging. IEEE Transactions on Medical Imaging, 2007 (In Press).

Please cite the above publication if you use the method.

Please read also the README file contained in the zip-archive.

### 2 Finite Mixture Models

Observed image intensities are denoted by  $x_i \in \mathbb{R}^d, i = 1, \dots, N$ . All of these intensities are drawn from one of the  $K$  classes. It is assumed that  $K$  is a constant selected using prior information about the voxel classification task. Intensities drawn from the class  $k$  follow the pdf  $f_k(x|\theta_k)$ , where  $k = 1, \dots, K$ . The parametric form of the pdf  $f_k$  is known but the value of the parameter vector  $\theta_k$  is unknown. The pdfs  $f_k$  are called *component densities*. Each class has a prior probability  $p_k \in [0, 1]$  expressing the fraction of the intensity values following the density  $f_k$ . These *mixing parameters* satisfy

$$\sum_{k=1}^K p_k = 1 \quad (1)$$

and their values are initially unknown. We denote the set of all parameter values by  $\Theta = \{p_k, \theta_k : k = 1, \dots, K\}$ . Combining the above models, the complete model is

$$f(x|\Theta) = \sum_{k=1}^K p_k f_k(x|\theta_k). \quad (2)$$

The objective is to estimate the parameters  $\Theta$  given the data  $\{x_i : i = 1, \dots, N\}$ . The estimation is based on the maximum likelihood (ML) principle. This leads to a complex optimization problem, which is solved by the genetic algorithm.

Once we have the estimate  $\hat{\Theta}$ , the image voxels can be classified by the Bayes classifier based on their intensity values. That is, the class  $\omega_i$  of the voxel  $i$  is

$$\omega_i = \arg \max_{k \in \{1, \dots, K\}} \hat{p}_k f_k(x_i | \hat{\theta}_k). \quad (3)$$

We offer three possibilities in modeling the component densities:

1. component densities can be 1-dimensional Gaussian densities;
2. component densities can be 1-dimensional Gaussian densities with partial volume classes. See the paper for explanation;
3. component densities can be multidimensional Gaussian densities. In this case, we use a hybrid of genetic algorithm and the EM algorithm to achieve the optimization result in a manageable time.

In future releases, we expect to add support for multidimensional Gaussian with partial volume classes.

### 3 Installation

Unpack the zip-file and copy the m-files onto your favorite directory. This package should work with Matlab versions 5.2 and up. Testing is mostly performed on the Matlab version 7.1. The GAMIXTURE Toolbox is platform independent.

Some examples may require specific Matlab toolboxes. However, the core of the Toolbox does not require any additional Toolboxes.

### 4 Basic Usage

For the most users, the following Matlab functions/scripts form the core of the Toolbox:

1-dimensional FMMs	
gamixture_nopve_example.m	An example script on how to use the algorithm on a toy example
gamixture_example.m	An example script on how to use the tools in the package for tissue classification in MRI
mixturega_simple.m	The main GA function for the 'simple' FMMs: no PVE or PVE classes are mixtures of the pure classes 1 and 2, 2 and 3 and so forth. Suitable for the tissue classification within MRI
mixturega_specs.m	The full version of the GA main function. Requires a specs struct specifying the FMM. One example applicable within FDG PET is provided with this package (fdgspecs.mat), further support for this function is underway but not yet available
parzen.m	A function for computing the Parzen estimates of the whole data pdfs
mixturega_bayesclassifier_simple.m	The Bayes classifier for voxel classification with gamixture_simple.m. This package does not yet contain similar function for mixturega_specs.m . It is expected to the next version.
multidimensional FMMs	
mixturega_hybridga.m	The function for optimising multidimensional FMMs
hybridga_fisherdemo.m	A script demonstrating the use of mixturega_hybridga.m
emclustering_sainit.m	The EM algorithm required by the hybrid method for multidimensional FMM optimization.

With this toolbox, we do not provide any means to read your favorite image format into Matlab. Probably the easiest way to start with the package is to take a look to the example files provided with the package.

## 5 Advanced usage

This section provides some additional information about the usage of the package.

### 5.1 mixturega\_simple

This function is meant for the learning of finite mixture model parameters given the Parzen estimate of the whole data pdf. This is the main GA function and the simple

version of it, i.e. pve classes are assumed to be mixtures of the class 1 and 2, the class 2 and 3, the class 3 and 4 and so forth. There is no background class. If these assumptions are restrictive, then try `mixturega_specs` instead. The Parzen estimate can be obtained by using the function `parzen.m`.

### 5.1.1 Inputs

The function requires the following inputs:

<code>x</code>	x-coordinates of the whole data pdf as given by <code>parzen.m</code>
<code>y</code>	y-coordinates of the whole data pdf as given by <code>parzen.m</code>
<code>params</code>	struct containing parameters for GA

The `params` struct is explained in the file `mixturega_simple.m`. The only field that it has to contain is `'params.nof_gaussians'`. This gives the number of pure tissue classes in the FMM. If `'params.pve = 0'`, this is the same as the number of classes. If `'params.pve = 1'`, there are, in addition, `params.nof_gaussians - 1` partial volume classes.

The most interesting parameters to play with are `'params.uplimit_p'` and `'params.lowlimit_p'`. These can be used to constrain the values of the mixing parameters for each class. For example, in tissue classification of the T1 weighted MRI with partial volume classes, giving

```
params.nof_gaussians = 3;
params.pve = 1;
params.uplimit = [0.2 1 1 0.2 0.2];
```

states that there cannot be more than 20% of CSF in the brain, the CSF/GM and GM/WM partial volume classes cannot contain more than 20 % of the voxels.

See the file `mixturega_simple.m` for further clarification.

### 5.1.2 Outputs

The function outputs a struct `stats`. The most important field of that struct is `'stats.result'`, which contains the optimised FMM parameters. If `'params.pve = 0'` then, the 1st component of this vector is the mean of 1st class, the 2nd component is the variance of the 1st class, the 3rd component is the mixing parameter for the 1st class, the 4th component is the mean of the 2nd component, and so on. If there are partial volume classes, then mixing parameters for these are given after the parameters (mean, variance, mixing parameter) for the pure classes are given.

See the examples and demos for clarification.

## 5.2 mixturega\_specs

This function allows for the full functionality when specifying the partial volume classes. For example, if you want to have 4 pure classes and a partial volume class between the pure classes 1 and 4, this the function to use. If you don't need partial volume classes, then all the functionality is contained in the `mixturega_simple.m`. **There is not yet full support available for this function in the version 1.1 of the toolbox.**

### 5.2.1 Inputs

The function requires the following inputs:

x	x-coordinates of the pdf
y	y-coordinates of the pdf
params	struct containing parameters for GA
specs	the specifications of the FMM

See the file `mixturega_specs.m` for further info.

### 5.2.2 Outputs

The function outputs a `stats` struct. The struct is as with `mixturega_simple`. The important thing is the field `'stats.result'`. See the file `mixturega_specs.m` for further info.

## 5.3 mixturega\_hybridga

The function for learning finite mixture model parameters for the  $d$ -dimensional mixtures. This is a hybrid algorithm of the GA and the EM algorithm rooted in the ability of the GA to solve 1-dimensional FMM optimization problems very accurately. The 'direct GA' described also in the paper is not included because the hybrid alternative seems to be much better.

At the moment, many parameters which could be tuned, are hard coded into the algorithm.

### 5.3.1 Inputs

The function requires the following inputs:

data	the data to be classified (in a matrix format). Each column of the matrix 'data' gives 1 data-point to be classified.
nof_classes	number of classes. Note that this version of the Toolbox does not yet support the partial volume classes in the multidimensional case.
class_ordering	Explained below.

**class\_ordering:** A matrix specifying the expected ordering of the classes based on all elements of the data points. Assume that the mean vector of the class  $i$  is  $\mu_i = [\mu_{i1}, \dots, \mu_{id}]^T$ . Now the  $j$ th column of the `class_ordering` matrix gives the order of classes based on the  $j$ th element of the mean vector, i.e. if

```
class_ordering(:, j) = [2 1 3]
```

then it is assumed that

$$\mu_{2j} \leq \mu_{1j} \leq \mu_{3j}.$$

You can also give a column of zeros, i.e

```
class_ordering(:,j) = [0 0 0]
```

This means that the ordering with respect to  $j$ th element of the mean vector is not known. In this case, the algorithm tries to guess the correct ordering based on the ordering with respect to other elements. Naturally, this may cause label switching problems.

### 5.3.2 Outputs

The outputs of the algorithm are:

bestcls	The resulting classification
bestfmm	The resulting FMM parameters
ene	The log-likelihood of each EM run
stats	FMM parameters from each EM run
cls	Classifications resulting from each EM run
statsga	The results of the GA part.

## 6 Known issues

1) The default lower limit for the component variances is too low and this may lead to spurious maximizers. Theoretically speaking, constraining the component variances to be larger than a very small positive constant should be enough to avoid spurious maximizers. However, in practice when the number of data is finite, this is not that easy. See e.g. Chapter 10 in Duda, Hart, Stork, Pattern Classification, 2nd edition, Wiley 2001.

2) In the multidimensional case, checking for label switching is a bit too primitive. This may cause some label switching problems. Also, the mechanism for the re-ordering the classes has not been tested fully and it may be buggy.

3) The algorithm is meant for handling large amounts of data. The default parameters for the Parzen-estimation function are selected based on large sample size. If the sample size is small, you may want to tune those parameters.