# Lab 4 - Gaussian Mixture Models

Thursday, July 15, 2010
4:58 PM

## PURPOSE

Goal: By the end of this lab, you will understand the how to use GMM models - a probabilistic clustering and "soft classification" technique.

You will pursue using GMMs for a variety of tasks of YOUR CHOOSING.

## TODAY'S PROJECT

Create one of the following:

A simple set of artist classification GMMs.
A simple set of genre classification GMMs.
A simple set of < other audio-based >classification GMMs.
Instrument Recognition
Speaker Recognition
<Other>

Note that I provide functions traingmm and evalgmm in the Toolboxes folder.
If you are a LAPTOP (non-network) user, please redownload the latest code from /usr/ccrma/course/mir2010/code

## FEEATURE DATA

1. For your ease of use, I have pre-extracted a large feature collection on numerous audio recordings that are currently posted in the following folders:
 /usr/ccrma/courses/mir2010/audio/RWC
 /usr/ccrma/courses/mir2010/audio/Some Popular Music

2. The features are stored in ".mat" files which correspond to each file in a given folder. You can import the features by using Matlab's **load** comment.

3. If you input the .mat file for a file, they are stored in a structure **features**

```
features =
   frames: [4647x57 double]
     mean: [1x57 double]
      std: [1x57 double]
```

The **raw audio features** **per frame** (100ms frames, hop size of 50%) are stored in **features.frames**
The mean of all features **per song** are stored in mean.
The stddev of all features **per song** are stored in std.

The 57-dimension feature vector:

| Feature #: 1 | 2 | 3 | 4 | 5 | 6:18 | 19:31 | 32:44 | 45:57 |
|---|---|---|---|---|---|---|---|---|
| zcr | centroid | bandwidth | skew | kurtosis | MFCC_mean | MFCC_std | MFCC_delta_mean | MFCC_delta_std] |

Unlike the previous labs, this lab is more free form and allows you greater room to explore these new techniques.

Experiment with what FEATURES you try, various parameters, and classifiers.

Special thanks and credit given to Dan Ellis at LabROSA / Columbia University for allowing modification and use of his labs and practicals. This lab includes commands and comments leveraged from his "Music Content Analysis" analysis.

## CREATING a GMM

To create GMMs in netlab (the Matlab Toolbox for GMMs) :

```
help traingmm
        [M0, LR0, acc, time] = traingmm(DATA, NGAUSS)
          Train one Gaussian mixture model.
         Input:
          DATA is rows of training feature vectors
          NGAUSS components
         Output
          M0 - the model GMM model created
          LR0 - data likelihood computed with training data
        Return total elapsed time in time.
        2003-06-30 dpwe@ee.columbia.edu muscontent practical
        2010 - Slightly update to support only 1 GMM training at a time by Jay  LeBoeuf
```

**Example:**

```
[DreamTheater.gmm,LR0, acc, time]=traingmm(DreamTheaterFeatureData.frames,5);   % Builds GMM with 5 components
```

And

```
[DaveMathews.gmm,LR0]=traingmm(DaveMathewsFeatures.frames,11);
```

% The resulting GMM model is stored in **DreamTheater.gmm**, and can be used for future evaluation.

## EVALUTING a GMM

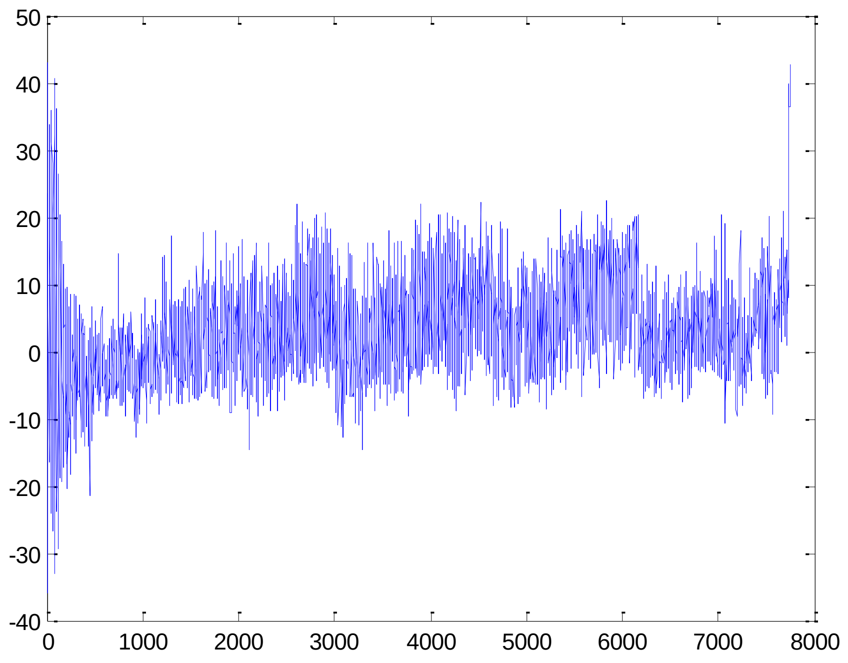
Now, take your test material (say, a test song) pass the **testing_features** into the GMMs, querying the likelihood of it agreeing with each GMM.

```
[loglikelihood,meanOutput,time] = evalgmms(DreamTheater_TestSong.frames,DaveMathews.gmm,DreamTheater.gmm);
% if > 0, it's more likely in the first GMM , if < 0, it's more likely in the second GMM

meanOutput =
  -5.3517
```

Since it's negative, we find that "DreamTheater_TestSong" is more likely to be DreamTheater than DaveMathews.

```
[loglikelihood,meanOutput,time] = evalgmms(DaveMathews_TestSong.frames,DaveMathews.gmm,DreamTheater.gmm);
% if > 0, it's more likely in the first GMM , if < 0, it's more likely in
% the second GMM

meanOutput =
   3.1336
```

The **loglikelihood** is the per-frame loglikehood of the frames being in each GMM - and shows you roughly which frames are likely to be in each GMM.



## HOW DO WE CALCULATE "SIMILARITY"?

Here are a few examples:

- We could build many GMMs which are modeled after audio samples. (say, "kick" , "snare", and "hi hat") Then you could compare any arbitrary sample (i.e. it's feature vector) against each of these classifiers. The GMM with the highest log-loglikelihood is the most probable output.

- We could classify **each frame** in a piece of audio separately against the GMMs, obtaining loglikelihoods for each frame. A challenge with this approach is that the frame-by-frame classifications vary very rapidly. (For example, try it and plot the result of the loglikelihood over time.) One way around this is to take the mean-value value of the loglikelihoods over all frames, and call this the description of the audio file.

- **ADVANCED AND NOT TESTED - USE AT YOUR OWN PERIL:**
    - We could compare **one distribution against other GMMs** (distributions) using a distance measure like EMD (Earth Mover's Distance), KL-divergence, centroid distance, etc. You would do something like this if you were comparing a GMM of a given song (all of the frames of a song) against the GMM

      The simplest to understanding is the “centroid distance”, which is the Euclidean distance between the means of the Gaussian models.

      To do this, you would use the function **dist2** to get the distance between **gm1.centres and gm2.centres.**

      Alternatively, we can use one the many statistical methods for computing distances between probability distributions. Since many of these methods are computationally intensive, one of the best (and fastest) is provided for you. (It's closely related to KL divergence.)

      distance = ppk ( gm1, gm2)

      where gm1 and gm2 are netlab GMMs.

      Note that the distance scores returned are relative, so it only makes sense to look at comparing the distances of multiple mixtures against each other and determining the best match. (such as 1 song worth of frames vs. 10 artist mixtures)