Lab 6 - Gaussian Mixture Models

Wednesday, July 16, 2008
2:20 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.

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

DEMO

To view a demo of the EM algorithm, type: **demgmm1**

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.

...using the audio files posted in your \scratch folder.

You might need this script Reading MP3 Files posted by Jason.

To create GMMs in Matlab's Netlab, you perform a few steps.

Below is a trivial example of creating 2 GMMs - one to represent kick drum samples and one to represent snare drum samples. Obviously, you'll need to modify the code to your own uses.

1. **Create default GMMs** specifying the number of dimensions (features) and number of Gaussian components in our mixture

   For example, for 2 features and 2 components:

```
% Initialize diagonal-covariance models for PDFs to be estimated
gm1 = gmm(2,2,'diag');  % GMM # 1
gm2 = gmm(2,2,'diag');  % GMM # 2
options = foptions;       % default optimization options
options(14) = 5;          % 5 iterations of k-means in initialization
gm1 = gmminit(gm1, featuresKick, options);  % Initialize from feature data for data type 1
 gm2 = gmminit(gm2, featuresSnare, options);  % Initialize from feature data for data type 2
```

2. **Run EM procedure** to estimate mixture parameters.

```
options = zeros(1, 18);
options(14) = 20;           % Number of iterations
gm1 = gmmem(gm1, featuresKick, options);
gm2 = gmmem(gm2, featuresSnare, options);
```

3. Determine how these PDF estimates perform as classifiers by calculating the log of the ratio of the likelihoods.

```
% Likelihoods of new feature data fitting under model
likelihoodKick = gmmprob(gm1,testing_features)
likelihoodSnare = gmmprob(gm2,testing_features)

% Log likelihood ratio
loglikelihood  = log(likelihoodKick ./likelihoodSnare )
```

```
% Emperically, if likelihoodKick > likelihoodSnare , then their ratio will be > 1, hence the log will be > 0
%                 If likelihoodKick < likelihoodSnare , then their ratio will be < 1, hence the log will be < 0

% Try to classify based on a likelihood threshold  (snares are labeled "1")
mean (  (loglikelihood  > 0) == 1 )
```

When do we run these likelihood calculations?

Here are a few examples:

- We could classify entire audio files (samples, in this case)
- We could classify **each frame** in a piece of audio separately - and you would notice that frame-by-frame classifications vary very rapidly. (For example, try it and plot the result of the loglikelihood over time.)
- We could compare one distribution against other GMMs (distributions) using a distance measure like EMD (Earth Mover's Distance), KL-divergence, centroid distance, etc. (For this lab, if you want to compare distributions to each other, you'll need to compute your own centroid distance until I can find some suitable distance measurement code for us to use. The "centroid distance" is the Euclidean distance between the overall means.)

Some commands you want like:

GMM	Creates a Gaussian mixture model with specified architecture.
GMMINIT	Initialises Gaussian mixture model from data
GMMEM	EM algorithm for Gaussian mixture model.

GMMPROB	Computes the data probability for a Gaussian mixture model.
GMMPOST	Computes the class posterior probabilities of a Gaussian mixture model.

Additional information

Lookup : gaussian mixture model in Matlab.
The statistics toolbox contains additional functions and information on Matlab's own implementation. Note this is not the same implementation as the Netlab version that we are using above!

\ EMD
run "emd_test" to compile the mex files.

Distance measures between clusters
>mahal

kl = kldivergence(x,y) % x and y are your feature data for each cluster?

KLDIV Kullback-Leibler or Jensen-Shannon divergence between two distributions.
http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=13089

---
http://www.ofai.at/~elias.pampalk/ma/documentation.html

* Investigate the classifier accuracy as the number of Gaussian mixture components and/or the number of feature dimensions is varied. You can even vary the amount of training data used by subselecting rows in ftrs and labs. Can you

make a plot of accuracy versus system complexity? How about versus training time?

Music Content Analysis:
A Practical Investigation of Singing Detection

Our task will be to build a detector to find the portions of a piece of pop music where there is singing. We will start off with some training data that has been hand-marked to indicate where the singing starts and stops, then use that to build statistical models of the signal properties that differentiate segments music with and without singing. We will then test the models on another example for which we have hand-marked ground truth, to get a quantitative measure of how well our system works, such as the proportion of time frames correctly labelled.
The practical is broken up in to several sections:

1. **Data and features**: Looking at the labelled music examples that we use to train our classifier, and calculating the cepstral features we will use for the classification.
2. **Gaussian mixture models (GMMs)**: We will attempt to capture the distribution of feature values for each of our two classes by fitting a set of multidimensional Gaussian blobs to their scatter plots. Using these continuous approximations, we can then label a new feature vector with the appropriate class by seeing which class model has a greater likelihood at that point.
3. **Neural network classification**: As an alternative to modeling each distribution, we can try to estimate the posterior probabilities of each class directly. One way to do this is to train a neural network, which has some interesting differences from a GMM approach.
4. **Evaluation**: Rather than measuring performance on the test data, a better test of system quality is to test on separate, unseen test data - sometimes more complex systems actually perform worse on unseen cases because they 'overfit' the training data. By varying the number of model parameters, we can experiment to find where overfitting begins.
5. **Temporal smoothing**: So far, we have been classifying each time frame individually, but in fact there is a lot of temporal structure: if you know the frame at time N has label A, then time N+1 is most likely to have the same label. We can try to exploit this local correlation either by simple temporal smoothing of the porbabilities, or with Markov models. And finally, you will have the chance to build your own classifier using the choice of features, model types, and parameters that you think will work best. We can then compare the performance of all the systems on another separate test example to see how well we can do!

You can download all the files associated with the practical here: muscontent-practical.tar.gz (74 MB, 76031689 bytes), or as a zip file: muscontent-practical.zip (74 MB, 76045455 bytes).

Practical's from Dan Ellis:
file:///C:/Documents%20and%20Settings/Jay/Desktop/Introductory%20MIR/Courses,%20Lectures,%20Tutorials/Music%20Content%20Course/Music%20Content%20Analysis%20%20Practical.htm