ECE 417 MP3 Walkthrough

Daniel Soberal, ECE 417 TA
Spring 2014

Introduction

The Basic Idea

1. At this point, we have worked with speaker/speech recognition and image (face) recognition
2. So, the question to ask now is: "is it possible to do BOTH?"
3. The answer is: YES!
4. We can create a *multi-modal* system that uses both image features and audio features to create a recognizer

The Data

1. Audio Data
   - 1.1 TRAINING SET: 15 samples from 4 speakers for a total of 60 samples
   - 1.2 TEST SET: 10 samples from 4 speakers for a total of 40 samples
2. Image Data
   - 2.1 TRAINING SET: 15 sample views from 4 subjects for a total of 60 samples
   - 2.2 TEST SET: 10 sample views from 4 subjects for a total of 40 samples

Overview

1. Generate Features
   1.1 Cepstral Matrix for each sample for audio
   1.2 PCA projections for image
2. For Audio, train a GMM over the cepstral features
3. For Image, use k-NN to compute the probability of each class occuring

Feature Space

Working With Audio Data

1. Compute cepstral features of all audio samples: use 10% overlap, 12 coefficients, Hamming Window, Window Size $= 500$
2. Do NOT squeeze the resulting matrix of vectors into a single vector: keep it in matrix form, because we will need to have individual observation vectors to send to the GMM training algorithm
3. Thus, we will end up with a matrix $\mathbf{X}_i$ for each speaker sample $1 \leq i \leq N_{test/train}$ in our database, such that $dim(\mathbf{X}_i) = 12 \times T$ where T is the number of frames.

Cepstrum Code Excerpt

If you did this wrong in MP1...

```
X = zeros(window_size,NumFrames);
lo_lim = 1;
up_lim = window_size;
for k = 1:(NumFrames)
    if(up_lim>length(x))
        s = [s; zeros(up_lim-length(x),1)];
    end
    X(:,k) = ifft(log10(abs(fft(s(lo_lim:up_lim).*Window))));
    lo_lim = up_lim - ceil(window_size*overlap) + 1;
    up_lim = lo_lim + window_size-1;
end
```

Working With Image Data

1. Compute the PCA projection onto the subspace
2. Keep the L eigenvectors required to get 95% of the total energy
3. Use $e_{net} = \sum_{k=1}^{K} |\lambda_k|^2$

Gaussian Mixture Model: Theoretical Overview

What *IS* a GMM Anyway?

A mixture model can be generally defined as an overall probabilistic model that takes into account the presence of sub-models. The Gaussian Mixture Model (GMM) is a weighted combination of M conditional Gaussian probability density functions. Each of these Gaussian pdf's has a set of parameters λ_m that describes it, as shown in Equation (1).

$$\lambda_m = \{w_m,\ \mathbf{\Sigma}_m,\ \boldsymbol{\mu}_m\} \qquad 0 \leq m \leq (M-1) \tag{1}$$

In the (1), the scalar w_m is the weight that the invidual Gaussian has in the overall mixture, the vector $\boldsymbol{\mu}_m$ is holds the mean of the Gaussian, and the matrix $\mathbf{\Sigma_m}$ is the covariance matrix. The vector $\boldsymbol{\mu}_m$ is D-dimensional, and the matrix $\mathbf{\Sigma_m}$ is DxD-dimensional. A subset of λ_m called θ_m will also be introduced, as shown in Equation (2).

$$\theta_m = \{\boldsymbol{\mu}_m,\ \mathbf{\Sigma}_m\} \tag{2}$$

Multivariate Normal Probability Density Function

Assume that you have a set of N observation vectors in a set X_{train}, each of which is denoted $\mathbf{x}_n$. This is show in Equation (3).

$$\mathbf{x}_n \in X_{train}, \qquad 0 \leq n \leq (N-1) \tag{3}$$

Each vector $\mathbf{x}_n$ is assumed to be D-dimensional. We further assume that these data vectors are independent and identically distributed (IID) with distribution p.
The probability density function that models the likelihood that a vector $\mathbf{x_n} \in X_{train}$ belongs to a particular Gaussian with parameter set θ_m is given by (4)[1].

$$p(\mathbf{x}_n|\theta_m) = \frac{1}{\sqrt{(2\pi)^D|\mathbf{\Sigma}_m|}} \exp\left[-\frac{1}{2}(\mathbf{x}_n - \boldsymbol{\mu}_m)^T \mathbf{\Sigma}_m^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_m)\right] \tag{4}$$

[1] In this equation, $|\mathbf{X}| = det(\mathbf{X})$, ie, the determinant of matrix $\mathbf{X}$.

Overall Mixture

Let the overall set of Gaussian mixture model parameters (including all M components) be given by (5).

$$\Lambda = \{\lambda_0,\ \lambda_1,\ ..., \lambda_{M-1}\} \tag{5}$$

The liklihood that a particular vector $\mathbf{x}_n$ belongs to the overall mixture is given by (6).

$$p(\mathbf{x}_n|\Lambda) = \sum_{m=0}^{M-1} w_m p(\mathbf{x}_n|\theta_m) \tag{6}$$

Note that now the set Λ is used in place of individual sets θ_m because the weight of the individual component Gaussians is used in the definition.

Likelihood

Since the training vectors are IID, the resulting density for the set of samples is given by (7). This will also be referred to as the likelihood function $L(\Lambda|X_{train})$ which is the likelihood of the parameter Λ given the training data.

$$\mathcal{L}(\Lambda; X_{train}) = p(X_{train}|\Lambda) = \prod_{n=0}^{N-1} p(\mathbf{x}_n|\Lambda) \tag{7}$$

The log-likelihood is applied by taking the logarithm of both sides of this equation. The final result is shown in (8).

$$\begin{aligned} L(\Lambda; X_{train}) &= \ln(\mathcal{L}(\Lambda; X_{train})) \\ &= \ln\left(\prod_{n=0}^{N-1} p(\mathbf{x}_n|\Lambda)\right) \\ &= \sum_{n=0}^{N-1} \ln\left(\sum_{m=0}^{M-1} w_m p(\mathbf{x}_n|\lambda_m)\right) \end{aligned} \tag{8}$$

Training a GMM

To train a GMM, we attempt to generate a maximum likelihood (ML) estimate of the GMM parameters (Λ). Specifically, we train a model parameters in such a way that we maximize the likelihood of the parameters given the observations (see Equation (8)). Direct optimization of this equation is not possible, therefore, an iterative solution must be used. In this case, the Expectation-Maximization (EM) algorithm is used.

EM: The Basic Idea

The general idea of the EM algorithm is to start with some initial model model Λ with some likelihood and then to iteratively update the model such that the likelihood of the model given the observations increases in each iteration. It is also worth noting that the EM algorithm is not globally convergent. However, it can still converge to a local optimum. It is up to the user of the algorithm to impose conditions that lead the algorithm to converge to a "good" optimum.

EM for a GMM: Expectation Step

In the EM algorithm, the a posteriori likelihood for component m in the mixture from the n-th observation at algorithm iteration t is:

$$\gamma_{m,n}(t) = p\left(m|\mathbf{x}_n, \lambda_m\right) = \frac{w_m p(\mathbf{x}_n|\theta_m)}{p(\mathbf{x}_n|\Lambda)} \tag{9}$$

This can be interpreted as the contribution of each component to each mixture. The calculation of these likelihoods across the observation space and mixture space comprises the Expectation step of the EM algorithm.

EM for a GMM: Maximization Step

The Maximization step involves using the probabilities to update the parameters for the next iteration, as well as evaluating the model likelihood.

$$\eta_m^{(t)} = \sum_{n=0}^{(N-1)} \gamma_{(n,m)}^{(t)} \tag{10}$$

$$w_m^{(t+1)} = \frac{\eta_m^{(t)}}{N} \tag{11}$$

$$\boldsymbol{\mu}_m^{(t+1)} = \frac{1}{\eta_m^{(t)}} \sum_{n=0}^{N-1} \gamma_{(n,m)}^{(t)} \mathbf{x}_n \tag{12}$$

$$\boldsymbol{\Sigma}_m^{(t+1)} = \frac{1}{\eta_m^{(t)}} \sum_{n=0}^{(N-1)} \gamma_{(n,m)}^{(t)} (\mathbf{x}_n - \boldsymbol{\mu}_m^{(t+1)})(\mathbf{x}_n - \boldsymbol{\mu}_m^{(t+1)})^T \tag{13}$$

The algorithm can be said to have converged to a local optimum if the difference between the likelihoods of subsequent iterations is sufficiently small (where it is up to the user to select this threshold).

$$|L^{t+1} - L^t| < \epsilon \tag{14}$$

Pitfalls

1. Floating point issues abound!
2. exp(-746) = 0 in Matlab (or, rather, in IEEE double precision FP)
3. Can your probabilities be that low during training? Absolutely.
4. There are work-arounds for this, including normalization and log-conversion

Generating Your Probabilities

Audio

1. We *give* you the code for this on the website...
2. Loop through each audio sample from the training set and concatenate the matrices for samples corresponding to the same speaker (have a matrix for speaker A, B, C, D). Use intelligent indexing to do this, don't hard-code.
3. Train a GMM for each speaker. Use $M = 2$
4. Loop through each audio sample from the test set (don't concatenate for each speaker this time).
5. Evaluate the likelihood of each test matrix $\mathbf{X}_{test}$ given each speaker GMM. In other words, evaluate $p(\mathbf{X}_{test,i}|\Lambda_{speaker,j})$.
6. Don't need to thow away data this time, so things ought to be simpler

Image

1. Use "k-NN" (except not really)
2. Compare each test projection to against the training projections
3. k-NN is the same as before. However, when you get to the stage where you vote, compute the histogram of the returned classes instead.
4. In other words, your histogram will have x-axis categorical variables A,B,C,D, and your y-axis will have the percentage that each class occurs (number of occurences of A through D, all divided by K)

Multimodal Fusion: Putting It All Together

We have the likelihoods: $p(x_{i,audio}|C_{j,audio})$ and $p(x_{i,image}|C_{j|image})$. We need to find a way to put things together and create a holistic likelihood $p(x_i|C_j)$. To do this, we will weight the audio and image likelihoods and multiply them together.

$$p(x_i|C_j) = p(x_{i,audio}|C_{j,audio})^{\mu} p(x_{i,image}|C_{j,image})^{1-\mu} \tag{15}$$

In log-form, this is:

`Fusion` `Audio GMM` `Fake KNN PCA`

$$\ln(p(x_i|C_j)) = \mu \ln(p(x_{i,audio}|C_{j,audio})) + (1-\mu) \ln(p(x_{i,image}|C_{j,image})) \tag{16}$$

You should get results for various $\mu = \{0.0, 0.1, 0.2, \ldots, 1.0\}$.

Deliverables

Code

Email me your code. Have as many functions as you want, but there should be one main wrapper that I run to do the entire experiment that requires *NO* feedback on my part. Include a README.txt files that tells me how to run your code.

Writeup

1. Email me **one pdf file** that contains your report.
2. It should contain a discussion on the idea behind the three methods, the advantages and disadvantages of each one, and an explanation of why weighting is needed for fusion. Which weight gives you the greatest performance? How does it compare to just using audio and image alone? How do they compare with MP1, MP2?
3. For the best weighting scheme, return the recognition accuracy per person.
4. In other words, I want an intelligent discussion in paragraph form about the experiment and your results (think along the lines of an informal lab report or formal technical memo)
5. Your discussion should include the following graphs and tables:
6. Plot of the overall recognition accuracy as a function of the weighting
7. plot of the recognition accuracy per person for the best weighting
8. table of the overall recognition accuracy as a function of the weighting
9. table of the recognition accuracy per person for the best weighting

Some Friendly Advice

1. DON'T HARD-CODE
2. Use Matlab's logical indexing functionality to it's fullest extent
3. Use meaningful variable names
4. COMMENT YOUR CODE
5. When doing arithmatic calculations, use Matlab's built in functions, and do operations as matrix vector calculations (MATLAB = MATrix LABoratory)
6. If you need to do a lot of nested, iterative calculations, it might be easier to do in C...look into MEX. That being said, you should not have to do this in this MP or any of the others we have done thus far.

Some MATLAB functions you might find useful

1. doc, help
2. tic, toc
3. log10, log, abs, exp, mvnpdf, rand, randn, randi
4. find, unique, hist, strcmp, strfind
5. bsxfun, cellfun, cell, zeros, ones, struct
6. mean, median, mode, max, min
7. eig, transpose, inv, repmat, reshape
8. cumsum, sum, prod