

# MP4: Audio-Visual Person Identification

ECE 417 – Multimedia Signal Processing  
Fall 2017

# The Basic Idea

- At this point we have worked separately with speaker/speech recognition and image (face) recognition.
- Is it possible to do both at once? YES!
- We can create a *Audio-Visual (multi-modal)* system that uses both image features and audio features to create a recognizer. We'll call this as the *AV system*.
- The result is a *fusion* of the audio and video approaches.

# The Data

- **4 Persons = {A, B, C, D}**
- **Audio Data**
  - TRAINING SET: 15 audio files/person
  - TEST SET: 10 audio files/person (  $A^a[1-10]$ ,  $B^a[1-10]$ ,  $C^a[1-10]$ ,  $D^a[1-10]$  )
- **Visual Data**
  - TRAINING SET: 10 images/person
  - TEST SET: 10 images/person (  $A^v[1-10]$ ,  $B^v[1-10]$ ,  $C^v[1-10]$ ,  $D^v[1-10]$  )
- **Audio-Visual (AV) Data**
  - TEST SET: 100 AV/person. (Every valid combination of audio and video test data)
  - E.g. Person A = {  $A^a[1]$   $A^v[1-10]$ ,  $A^a[2]$   $A^v[1-10]$ , ... ,  $A^a[10]$   $A^v[1-10]$  }.
  - E.g. Person B = {  $B^a[1]$   $B^v[1-10]$ ,  $B^a[2]$   $B^v[1-10]$ , ... ,  $B^a[10]$   $B^v[1-10]$  }.

# Overview: 4 Steps

- **Step 1: Generate Features**

- Audio: Generate **Cepstral features** for each audio sample.
- Images: Generate **PCA features** for each image.

- **Step 2: Audio (In-Set Speaker Id)**

- Train: **4 GMMs (1 per person)**. Thus, A: GMM(1), ..., D: GMM(4).
- Test: Compute 4 **GMM Likelihoods** (1 per GMM) for each test datum (audio file).
- 4 likelihoods/test file x 40 test files.
- Determine the id of the person and report accuracy.

# Overview: 4 Steps

- **Step 3: Vision** (In-Set Image ID)
  - Use **kNN** to compute 4 *posterior probabilities* (1 per person) for each test datum (image file).
  - 4 posteriors/test file x 40 test files.
  - Determine the id of the person and report accuracy.
- **Step 4: AV**
  - Use the GMM likelihoods (Step 2) and the kNN posteriors (Step 3) to compute 4 **AV probabilities** (1 per person) for each AV test datum.
  - 4 AV probabilities/test datum x 100 test data.
  - Determine the id of the person and report accuracy.

## Step 2: Audio

- Compute **cepstral features** of all audio samples: use 10% overlap, 12 coefficients, Hamming Window, Window Size=500.
- You can use your **cepstrum.m** from MP3 to generate a matrix **X** per audio file. Size =  $[12 \times N_f]$ ,  $N_f$  = # frames in the audio file. Do **NOT** unroll **X** into a single vector.
- Use matrix **X** to train GMMs with  $M = 2$  (a.k.a Gaussian mixtures or component densities). Use **gmm\_train.m**.

## Step 2: Audio

- **gmm\_train.m**: To train a GMM for person A, construct a big matrix by stacking the matrices of all (training) audio files from person A.
  - $\text{cepsA} = [\mathbf{X1} \ \mathbf{X2} \ \dots \ \mathbf{X15}]; \ \% \ [D \times (Nf1 + \dots Nf15)], D = N_{cc} = 12 \ (\# \text{ ceps coeff})$
  - $\text{GMM}(1) = \text{gmm\_train}(\text{cepsA}', M);$
  - $\text{GMM}(1).\mu = [D \times M], \text{GMM}(1).\sigma = [D \times M], \text{GMM}(1).\text{weight} = [1 \times M]$
  - Denote  $\text{GMM}(i)$  as  $\Lambda_i$ , where  $i = 1$  (A),  $2$  (B),  $3$  (C),  $4$  (D)
- **gmm\_eval.m**: Evaluates the avg log-likelihood (LL)  $P(\mathbf{X}_{\text{test}}; \Lambda_i)$  of the test observations in matrix  $\mathbf{X}_{\text{test}}$  w.r.t GMM model  $\Lambda_i$ .
  - For a test audio file, extract feature matrix  $\mathbf{X}_{\text{test}}$ .
  - $\text{LL}(i) = \text{gmm\_eval}(\mathbf{X}'_{\text{test}}, \text{GMM}(i)); i = 1, \dots, 4$
  - $\text{person id of } \mathbf{X}_{\text{test}} = \arg\max \{ \text{LL}(1), \text{LL}(2), \text{LL}(3), \text{LL}(4) \}$

## Step 2: Audio (Notations in gmm\_\*.m)

- GMM Likelihood (LL) for feature vector  $x_i$  w.r.t model  $\Lambda$ :

$$p(x_i; \Lambda) = \sum_{k=1}^K p(k|\Lambda)p(x_i|k; \Lambda) = \sum_{k=1}^K w_k p(x_i|k; \Lambda)$$

- Gamma probabilities:  $\gamma_{i,k} = p(k|x_i; \Lambda)$   
$$= \frac{w_k p(x_i|k; \Lambda)}{p(x_i; \Lambda)}, i = 1, \dots, N$$

- Rho (Normalized Gamma):  $\rho_{i,k} = \frac{\gamma_{i,k}}{\sum_{i=1}^N \gamma_{i,k}} = \frac{\gamma_{i,k}}{N_k}$

- Mean

$$\mu_k = \sum_{i=1}^N \rho_{i,k} x_i$$

- Covariance

$$\Sigma_k = \sum_{i=1}^N \rho_{i,k} (x_i - \mu_k)(x_i - \mu_k)^T$$



## Step 3: Image

- Compute the PCA Projection onto the subspace
- Keep the L eigenvectors required to get 95% of the total energy
- Use  $e_{net} = \sum_{k=1}^K |\lambda_k|^2$
- Use your **pca.m** from MP2.
- Modify your **knn.m** from MP2 to implement 10-NN (k=10). Additionally compute the posterior probability for each person.  
E.g.  $P(B | \text{test image}) = (\# \text{ NNs belonging to class B})/10$ .

## Step 4: AV

- GMM likelihood =  $p(X_{\text{test}, a} | C_i)$  from Step 2.
- k-NN posterior =  $p(C_i | X_{\text{test}, v})$  from Step 3.
- AV fusion metric: Combines both the likelihood and posterior

$$p(X_{\text{test}, a} | C_i) p(C_i | X_{\text{test}, v}) = p(X_{\text{test}, a}, C_i | X_{\text{test}, v})$$

Audio

Vision

AV Fusion

- In fact, the joint density is directly proportional to the AV fusion metric:

$$p(X_{\text{test}, a}, X_{\text{test}, v}, C_i) \propto p(X_{\text{test}, a}, C_i | X_{\text{test}, v}) \xrightarrow{\text{because of}} \begin{array}{c} \text{C} \\ \swarrow \quad \searrow \\ \text{V} \quad \text{A} \end{array} \text{ DAG (directed acyclic graph)}$$

- But we'll use a scaled version for our experiments:

$$p(X_{\text{test}, a}, C_i | X_{\text{test}, v}) = p(X_{\text{test}, a} | C_i)^w p(C_i | X_{\text{test}, v})^{1-w}$$

Try  $w = \{0.1, 0.2, \dots, 0.9\}$ .


# Results (Matlab Output)

On Matlab,

```
> datadir = '/ws/ece417/hw4/data'
```

```
> run(datadir)
```

- Keep all your data files under **one top level datadir** directory when you write your code.
- Do **NOT** change the names of the subdirectories (testing, training, testspeech, trainspeech) or the file names within the top level datadir.
- I'll use this **datadir** to test your code. If I **can't run your code** w/o your help or if I have to **enter additional parameters** to execute your code (e.g. `run(datadir, w = 0.5, M = 2, D = 12)`), then you are likely to **lose points**.

- 
- /ws/ece417/hw4/data
    - testing/
    - testspeech/
    - training/
    - trainspeech/

## Results (Matlab Output)

```
> datadir = '/ws/ece417/hw4/data'
```

```
> run(datadir)
```

----- Person ID Accuracy: Visual Only -----

Acc

A 100

B 80

C 10

D 80

Avg 67.5

----- Person ID Accuracy: Audio Only -----

Acc

A 50

B 80

C 60

D 90

Avg 70

----- Person ID Accuracy: AV -----

w\_01 w\_02 w\_03 w\_04 w\_05 w\_06 w\_07 w\_08 w\_09

A 97 \* \* \* \* \*

B 76 \* \* \* \* \*

C 22 \* \* \* \* \*

D 97 \* \* \* \* \*

Avg 73      \*      \*      \*      \*      \*      \*      \*

# On gmm\_train.m, gmm\_eval.m

- There are about **15 lines** in gmm\_train.m and gmm\_eval.m with incomplete code. You are required to fill in those lines.
- Each line can be completed with **one line of Matlab code**. However, you are **welcome to create your own functions** within gmm\_[train|eval].m if you are unable to accomplish this in one line of code.
  - For e.g, this would be ideal: `Z = bsxfun(@minus, X, Mu(:,k));`
  - But this is also OK: `Z = Center_Data(X, Mu(:,k));` % and you have a function Center\_Data().
- To get full credit, you must complete each line of incomplete code. If you **ignore/delete/comment those lines and somehow directly arrive at the answers** (Mu, Sigma etc.), then you will lose points.

# Turn In

- Your writeup. All of your experimental results in table form (A, V, AV).
- Your Matlab code
  - run.m: function run(datadir).
  - gmm\_train.m, gmm\_eval.m, sig2frames.m, cepstrum.m, knn.m, pca.m
  - Other \*.m dependencies (no restriction on i/p or o/p arguments)
- Upload to Compass:
  - MP4\_<NetID>.zip (Matlab code, members.txt. But no audio/image/video files.)
  - MP4\_<NetID>.pdf (write up)

# What's a Gaussian Mixture Model (GMM) Anyway (1)?

A mixture model can be generally defined as an **overall probabilistic model** that takes into account the presence of **sub-models**. The GMM is a ***weighted combination of  $M$  conditional Gaussian probability density functions***. Each of these pdf's has a set of parameters that describes it.

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

# What's a Gaussian Mixture Model (GMM) Anyway (2)?

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

$w_m$  : the (scalar) **weight** that an individual Gaussian has in the overall mixture

$\boldsymbol{\mu}_m$  : the vector that holds the **mean** of the Gaussian, D-dimensional

$\mathbf{\Sigma}_m$  : the **covariance matrix** for the Gaussian, DxD-dimensional

$$\theta_m = \{\boldsymbol{\mu}_m, \mathbf{\Sigma}_m\} \quad (2)$$



# Multivariate Normal Probability Density Function (1)

Assume that you have a set of **N observation vectors** in a set  $X_{\text{train}}$ , each of which is denoted  $x_n$ .

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

Each vector  **$X_n$  is D-dimensional**. We further assume that these data vectors are **independent**, and **identically distributed (IID)** with ***distribution  $p$*** .

# Multivariate Normal Probability Density Function (2)

The probability density function that models the **likelihood that a vector  $\mathbf{x}_n \in \mathbf{X}_{\text{train}}$  belongs to a particular Gaussian with a parameter set  $\theta_m$  is given by:**

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

Where:  $|\boldsymbol{\Sigma}| = \det(\boldsymbol{\Sigma})$ , ie, the determinant of matrix  $\boldsymbol{\Sigma}$

# The Overall Mixture

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

$$\Lambda = \{\lambda_0, \lambda_1, \dots, \lambda_{M-1}\} \quad (5)$$

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

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

Note that now the set  $\Lambda$  is used in place of individual sets  $\theta_m$  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.

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

The log-likelihood is applied by taking the log of both sides of the equation, resulting in (8).

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

# Training a GMM

To train a GMM, we attempt to generate a maximum likelihood (ML) estimate of the GMM parameters ( $\Lambda$ ).

Specifically, we train the model parameters in such a way that we maximize the likelihood of the parameters given the observations (see eqn. (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

Start with some **initial model parameters  $\Lambda$** , with some likelihood. Then, **iteratively update the model** such that the likelihood of the model, given the observations, increases at each iteration.

Note that it can converge at local optimum; it doesn't converge globally, necessarily. It's up to the designer/developer to impose constraints which encourage global convergence.

# EM for a GMM: Expectation Step (update likelihoods)

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

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

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 is the Expectation step of the EM algorithm.

*Use the parameter estimates from the last iteration to estimate the likelihoods.*

# EM for a GMM: Maximization Step (update parameters)

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

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

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

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

$$\Sigma_m^{(t+1)} = \frac{1}{\eta_m^{(t)}} \sum_{n=0}^{(N-1)} \gamma_{(n,m)}^{(t)} (\mathbf{x}_n - \mu_m^{(t+1)})(\mathbf{x}_n - \mu_m^{(t+1)})^T \quad (13)$$

The algorithm **converges to a local optimum** if the difference between likelihoods of subsequent iterations is small (you select threshold).

$$|L^{t+1} - L^t| < \epsilon \quad (14)$$

*Use the probability estimates from the last iteration to estimate the parameters.*