A Comparative Study of Resource Usage for Speaker Recognition Techniques

Pulkit Verma and Pradip K. Das

{*v.pulkit, pkdas*}*@iitg.ernet.com*
Department of Computer Science and Engineering
Indian Institute of Technology Guwahati

2016 International Conference on
Signal Processing and Communication

Overview

Outline

Introduction

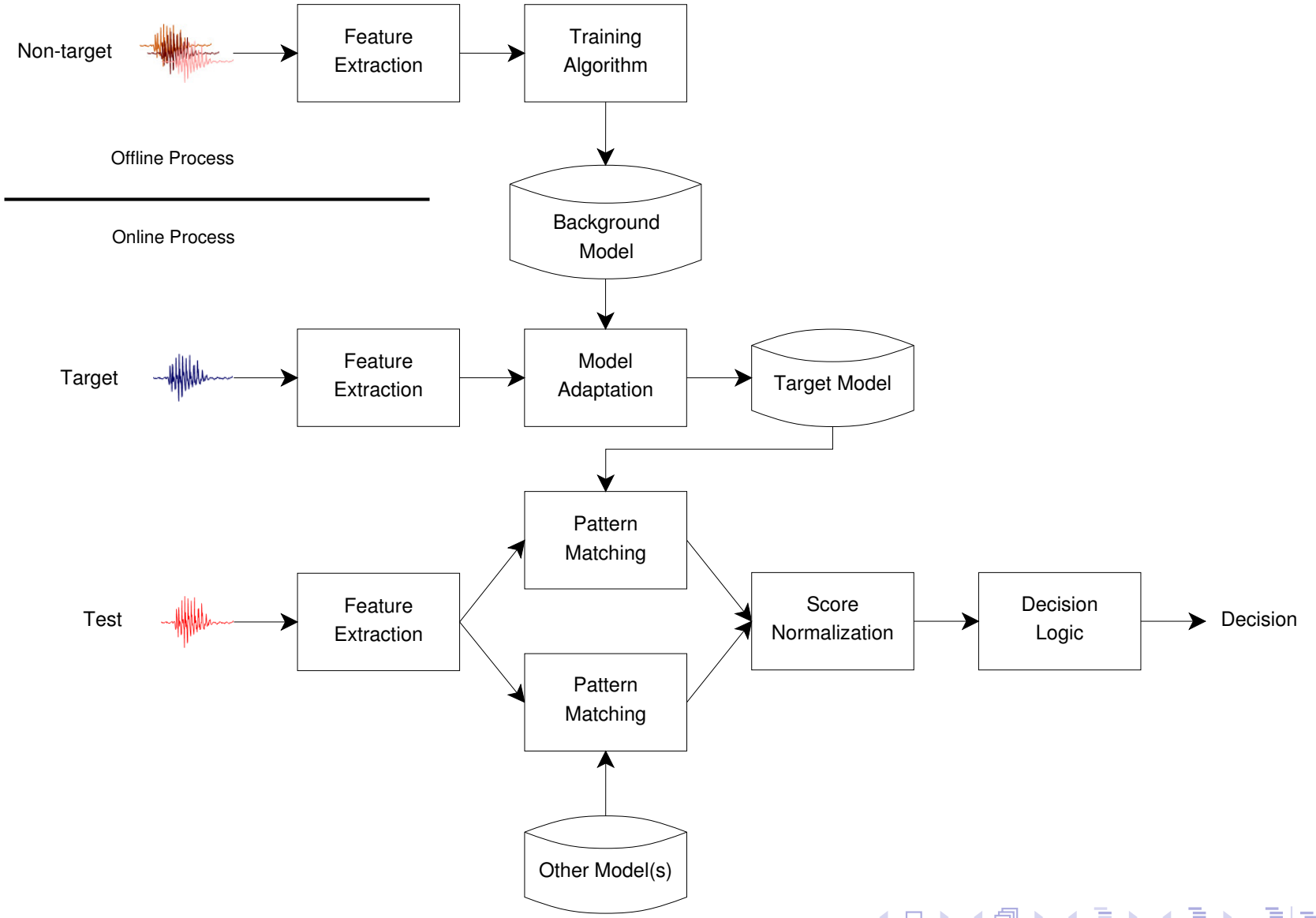
Speech Recognition

- What is Speech Recognition?
- Problems with Accuracy

Resource Usage

- Why Resource Usage?
- Importance of Power Usage
- Other parameters

Generic System



Feature Extraction

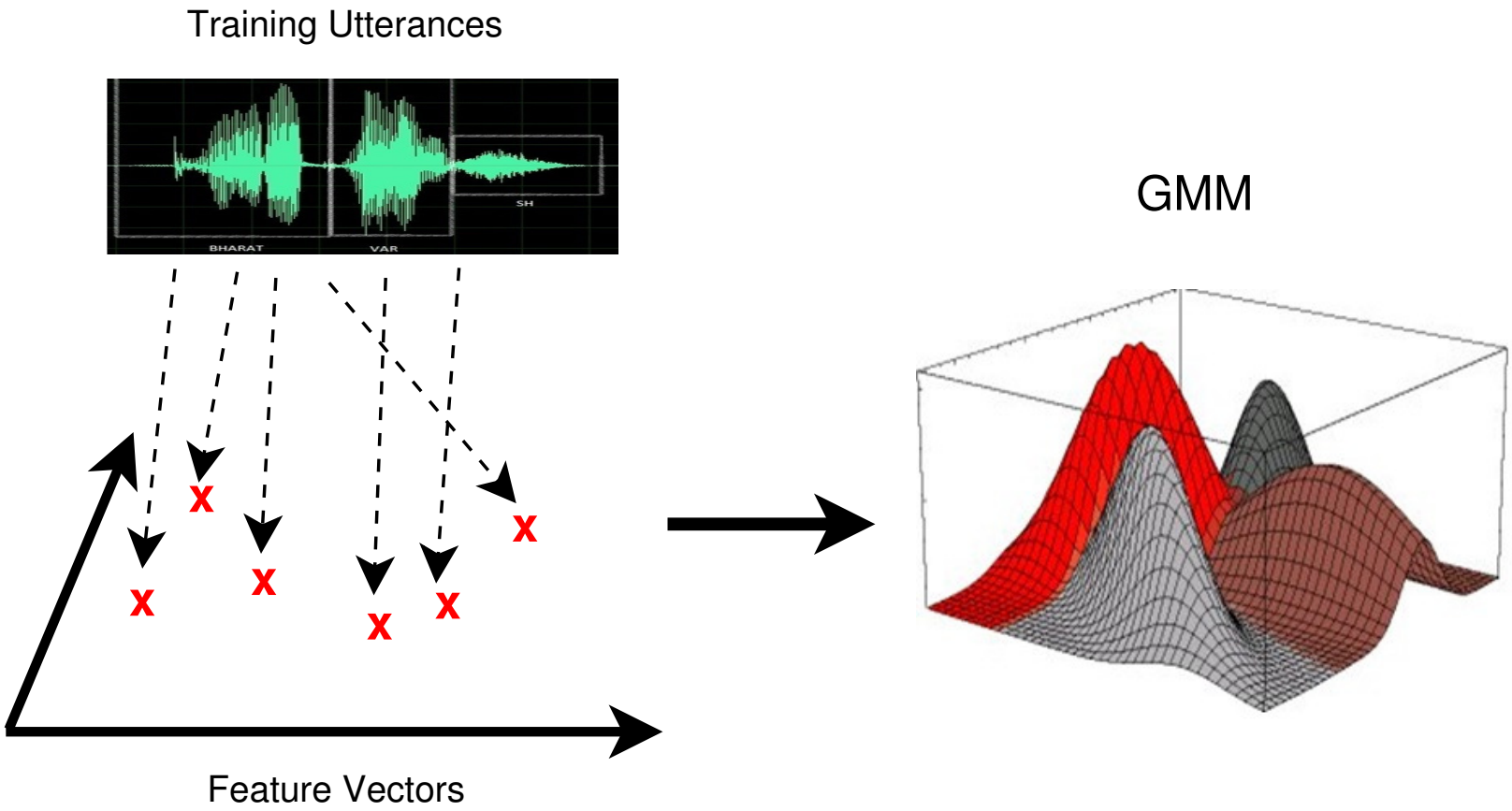
Desired characteristics of features

They should:

- be less susceptible to environmental noise
- occur frequently and naturally in normal speech
- not be affected by health of the speaker
- be easily measurable
- be difficult to copy by impostors

Outline

Gaussian Mixture Models Info



Why GMMs?

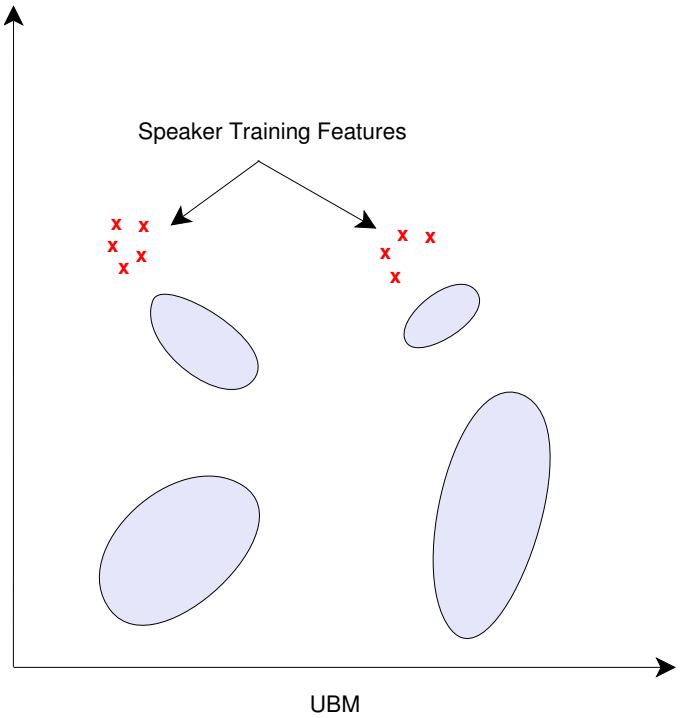
- They can model a large class of speech characteristics.
- Even for arbitrary component densities the model gives a good approximation.
- Capable of modeling the hidden characteristics of speech data (vocal cords, voice tract information, etc.)

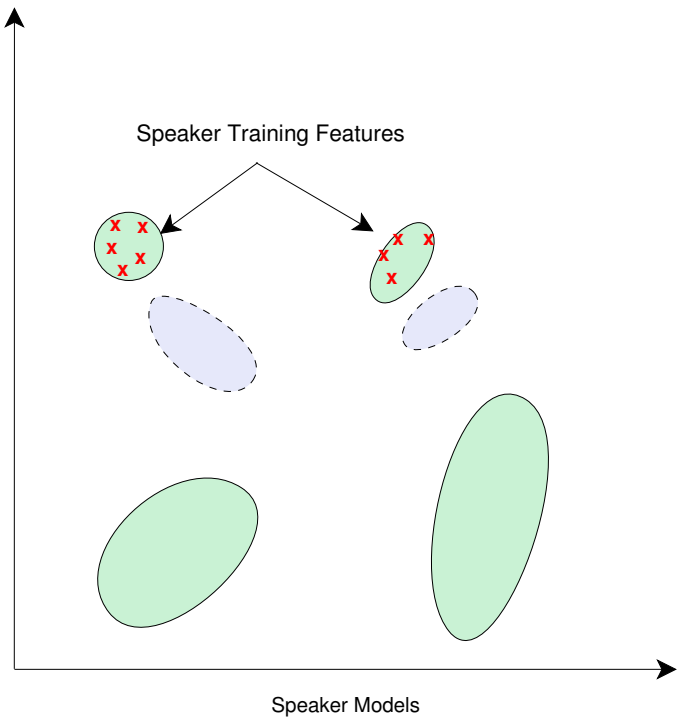
Universal Background Models

- Trained using speech from many speakers.
- Represents a general speaker independent model.
- Expected to contain the features of any utterance on which system is tested.
- Background model for GMM-UBM Framework.

The GMM-UBM Framework - MAP Adaptation

- Maximum A-Posteriori (MAP) training.
- Target model is trained by adapting UBM.





Joint Factor Analysis

Assume speaker and channel supervectors Info are distributed normally and are statistically independent.

$$M = s + c \tag{1}$$

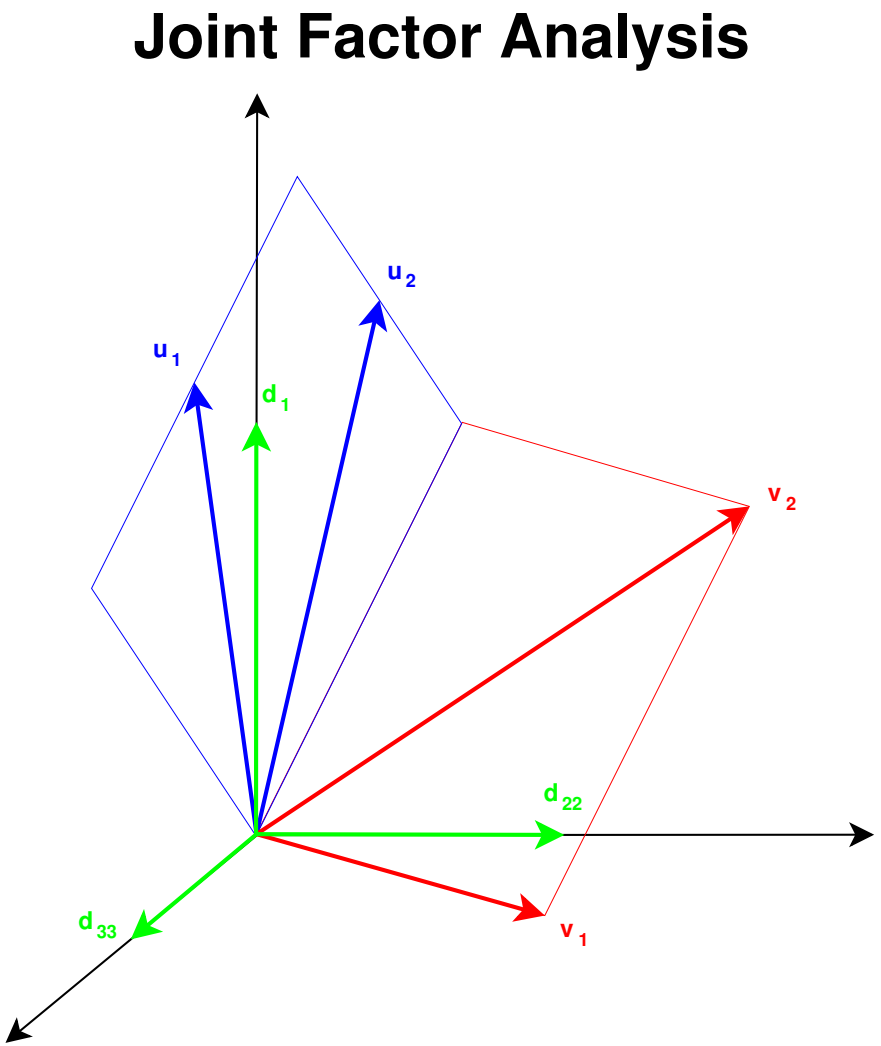
where,

M is speaker and channel dependent supervector;

s is speaker dependent supervector; Info

c is channel dependent supervector. Info

Joint Factor Analysis



$$M = m + Vy + Dz + Ux$$

Joint Factor Analysis

- Assumed that the channel factors will only model the channel effects.
- Dehak [1] observed that the channel dependent supervector also models the speaker features.
- Front End Factor Analysis proposed to address this issue.

Front End Factor Analysis

- New low dimensional was introduced to account for both the variabilities.
- Called Total Variability Space T.
- M is given by:

$$M = m + Tx \tag{2}$$

where,

m is the UBM supervector (speaker and channel independent supervector);

x is a normally distributed random vector in this space; and

T is a low ranked rectangular matrix.

Front End Factor Analysis - Why Preferred?

Low Computation Cost

- Channel compensation is done in total factor space
- The new vectors here are known as identity vectors or i-vectors.
- the dimensionality of i-vectors is lower as compared to GMM supervectors
- Computation cost is much less as compared to JFA.

Unsupervised Learning

- Supervised training is not needed in this model unlike JFA and GMM-UBM.
- Creates clusters on its own.

Outline

Resources Managed

- Power or Energy
- Memory
- Secondary Space

Power Measurement

- Used Energy Measurement Library (EML) [2] to measure the power requirements.
- Acts as a middleware between code for measuring power consumption and hardware based measurement tools.
- Currently available for a limited number of hardware.

EML - Device Support

The power measurement support provided by EML is for three kinds of devices currently:

1. Intel CPUs:
   1. Intel CPUs Sandy Bridge and later: Read via Intel RAPL MSR interface for the Intel Xeon family.
   2. Intel Xeon Phi: Read via Intel MPSS 3.x (Many Platform Software Stack)
2. Nvidia Fermi and Kepler cards: Read via NVML (Nvidia Measurement Library).
3. Metered Power Distribution Units (PDU)

EML - Methodology

- Supports four basic steps:
  - Broadcasting data.
  - Division of work.
  - Actual Computation.
  - Collecting the results.
- Power calculated for each step separately.

Memory and Space Measurements

Memory

- Calculated the resident set size (RSS) portion of the memory. Info
- Used `pmap` utility available in UNIX based OSs. Info
- Extract RSS value from all the maps in memory map of process.
- Sum them to get the total RSS at that instant.
- Total RSS value is calculated at fixed intervals.

Secondary Space

- Calculated the amount of space occupied by the executables on the secondary disk.
- `ls` command was used for this purpose.

Outline

The MIT Mobile Device Speaker Verification Corpus [3]

- Recordings were done using microphones and internal headsets
- Contains inter-session variability
- Text: Names and Ice cream flavors
- 5,184 recordings for enrolled users over 2 sessions
- 54 recordings per session per speaker

Recordings

- Enrollment: 48 speakers - 26 male, 22 female
- Impostor : 40 speakers - 23 male, 17 female

Platform Used

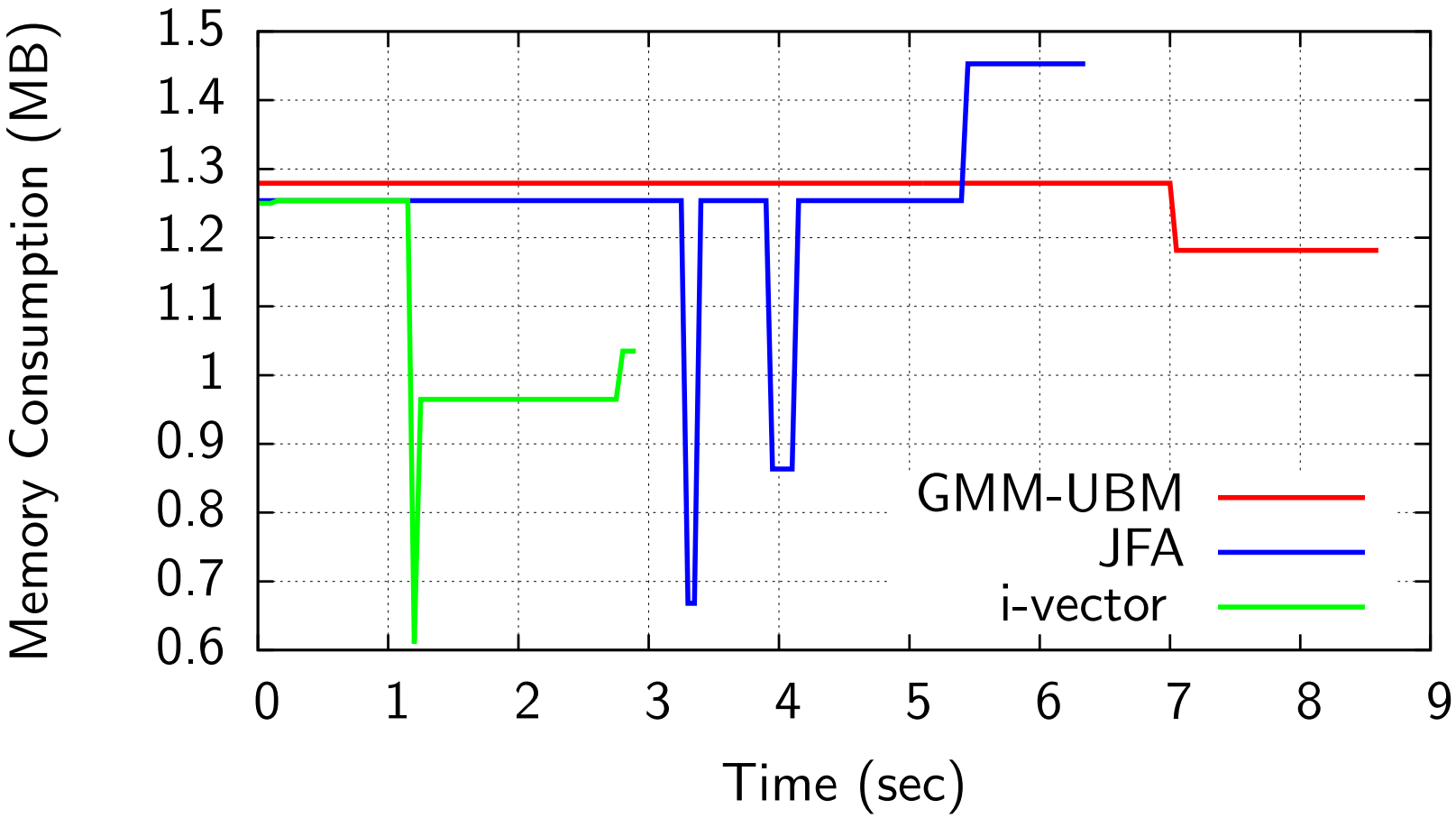
- 1 x Intel® Xeon® CPU E5530 @ 2.40GHz
- 16 Cores
- 48 MB L3 Cache
- 32 GB RAM
- gcc 4.1.2
- Intel® MSR RAPL interface
- Intel® MPSS 3.4.2

Results

Table: Resource consumption for GMM-UBM, JFA and i-vector approaches for speaker recognition

Approach	Memory Consumption (MB)		Energy (W.h)	Space (KB)	Time (sec)
	Average	Maximum			
GMM-UBM	1291.5	1310.0	0.3648	1013	8.65
JFA	1292.0	1488.0	0.3641	1034	6.45
i-vectors	1106.0	1284.0	0.3472	3369	2.95

Results


Memory Consumption (MB)
1.5
1.4
1.3
1.2
1.1
1
0.9
0.8
0.7
0.6
0
1
2
3
4
5
6
7
8
9
Time (sec)
GMM-UBM
JFA
i-vector

Results Analysis

- i-vector approach performs significantly better in terms of speed
- Power consumption of all 3 approaches is comparable
- i-vectors are slightly optimal in terms of memory consumption
- Large code-base of i-vectors uses the maximum space

Outline

Conclusion

- Comparison of the current state of the art techniques in terms of their resource utilization targeted towards speaker recognition is conducted.
- Experiments were performed on a standard corpus collected using mobile devices in a noisy environment.
- i-vector approach is more efficient in terms of memory usage and power consumption.

Future Work

- Current work only analyzes the resource usage on laptops and desktops.
- Performance on real time data.
- Support for mobile devices like like Android™, Windows® Phone, etc.

References I

[1] N. Dehak, "Discriminative and Generative Approaches for Long- and Short-term Speaker Characteristics Modeling: Application to Speaker Verification," Ph.D. dissertation, Ecole de Technologie Superieure (Canada), 2009, aAINR50490.

[2] A. Cabrera, F. Almeida, J. Arteaga, and V. Blanco, "Measuring energy consumption using EML (Energy Measurement Library)," *Computer Science - Research and Development*, pp. 1–9, 2014.

[3] R. Woo, A. Park, and T. Hazen, "The MIT Mobile Device Speaker Verification Corpus: Data Collection and Preliminary Experiments," in *Speaker and Language Recognition Workshop, 2006. IEEE Odyssey 2006: The*, June 2006, pp. 1–6.

References II

[4] D. Reynolds, "Gaussian Mixture Models," in *Encyclopedia of Biometrics*, S. Li and A. Jain, Eds. Springer US, 2009, pp. 659–663. [Online]. Available: http://dx.doi.org/10.1007/978-0-387-73003-5_196

Questions?


भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Technology Guwahati
Guwahati - 781039, INDIA

Thank You

Outline

Gaussian Mixture Models

Back

A Gaussian mixture model [4] is a weighted sum of M component Gaussian densities as given by the equation,

$$p(x|\lambda) = \sum_{i=1}^{M} w_i \; g(x|\mu_i, \Sigma_i) \tag{3}$$

where,

x is a D-dimensional continuous-valued data vector (i.e. measurement or features);

w_i, $i = 1, ..., M$, are the mixture weights; and

$g(x|\mu_i, \Sigma_i)$, $i = 1, ..., M$, are the component Gaussian densities.

Gaussian Mixture Models

Back

The complete Gaussian mixture model is parameterized by three parameters:

- Mean vectors,
- Covariance matrices, and
- Mixture weights from all component densities.

These parameters are collectively represented by the notation,

$$\lambda = \{w_i, \mu_i, \Sigma_i\}, i = 1, ..., M \tag{4}$$

How are GMMs used? Back

- GMMs are used to grab the distribution of MFCCs in the feature.
- Assumed that given these feature vectors $u = y_1, y_2, y_3, ...y_L$ are independent of each other.
- Probability of a given utterance u given the model λ is simply the product of each L frames.
- Since L could be large and probabilities are less than or equal to 1, such a large product may lead to underflow.

GMMs - Log likelihood

Back

The complete Gaussian mixture model is parameterized by three parameters:

- To use a GMM, we need to do two things
  - Compute the likelihood of a sequence of features given a GMM
  - Estimate the parameters of a GMM given a set of feature vectors
- If we assume independence between feature vectors in a sequence, then we can compute the likelihood as

$$p(x_1, ..., x_N|\lambda) = \prod_{n=1}^{N} p(x_n|\lambda) \tag{5}$$

- In the form of log likelihood,

$$log\ p(x_1, ..., x_N|\lambda) = \sum_{n=1}^{N} log\ p(x_n|\lambda) = \prod_{n=1}^{N} \left(\sum_{i=1}^{M} w_i\ g(x|\mu_i, \Sigma_i) \right) \tag{6}$$

GMMs - Log likelihood

Back

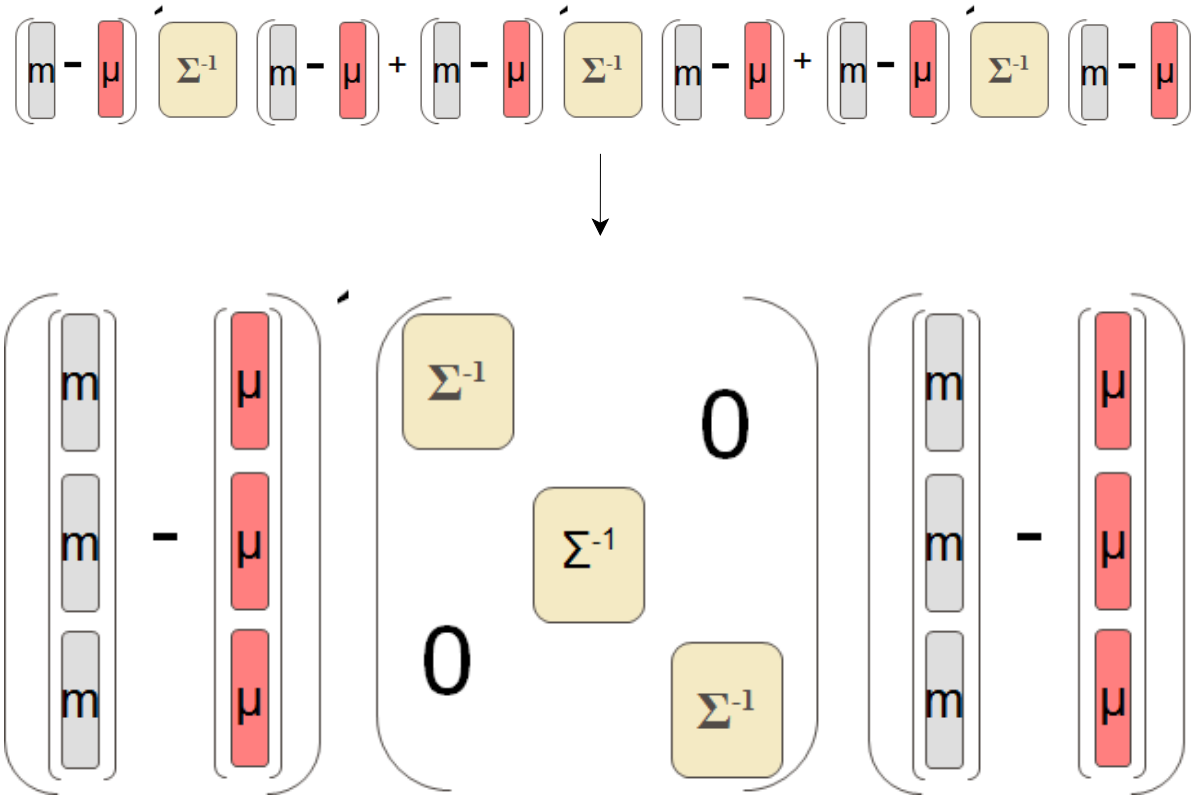
GMM parameters are estimated by maximizing the likelihood of on a set of training vectors

$$\lambda^* = arg \max_{\lambda} \sum_{n=1}^{N} log\ p(x_n|\lambda) \tag{7}$$

Supervectors

Back

By stacking vectors and matrices, we can work directly with vector-matrix manipulations



JFA- Speaker Factors

Back

$$s = m + Vy + Dz \tag{8}$$

where,

s is speaker dependent supervector;

m is $CF \times 1$ speaker and channel independent supervector;

V is eigenvoice matrix (rectangular matrix of low rank);

D is $CF \times CF$ residual diagonal matrix;

y is a vector representing speaker factors;

z is a normally distributed CF dimensional random vector representing speaker specific residual factors.

JFA- Channel Factors

Back

$$c = Ux \tag{9}$$

where,

c is channel dependent supervector;

U is eigenchannel matrix (matrix of low rank);

x is a normally distributed random vector representing channel factors.

Resident Set Size

Back

Resident set size (RSS) is the portion of memory occupied by a process that is held in main memory (RAM). The rest of the occupied memory exists in the swap space or file system, either because some parts of the occupied memory were paged out, or because some parts of the executable were never loaded.

pmap

Back

`pmap` gives the following memory information related to each map of memory map of a process:

1. Start address of the map.
2. Map size in kilobytes.
3. Resident Set Size in kilobytes.
4. Dirty pages (both shared and private) in kilobytes.
5. Read, write, execute, shared and private (copy on write) permission on map
6. Mapping type. It can be either (1) a file name; (2) `[ anon ]` for allocated memory; or (3) `[ stack ]` for program stack.
7. Offset of map into the file
8. Device name