Low development cost, high quality speech recognition for new languages and domains

"Cheap ASR"

Participants:

- "Senior members": Lukas Burget, Nagendra Kumar Goel, Daniel Povey, Richard Rose
- Graduate students: Arnab Ghoshal, Petr Schwarz, Samuel Thomas
- Undergraduates: Mohit Agarwal, Pinar Akyazi
- Special thanks to (in alphabetical order): Tony Feng, Mark Gales, Ondrej Glembek, Martin Karafiat, Tomas Kasparek, Patrick Nguyen, Ariya Rastrow, Shou-Chun Yin

What is this project about

- "Cheap ASR" is a unifying theme for two sub-projects:
  - "Subspace Gaussian Mixture Models" (SGMM), an acoustic modeling technique.
  - Automatic learning of lexicons, which is a topic that Nagendra and Samuel have been working on.
- Reason for theme:
  - SGMMs are more compact, so need less training data to train equally good models.
  - Automatic learning of lexicons means less need for human input.

Schedule for this afternoon

1.30 – 3:00	**Presentations**
3:00 – 3:15	**Break**
3:15 – 4:05	**Presentations**
4:05 – 4:30	**Questions and Discussion**

First block

1.35 – 1:50	Rick Rose	Introduction to Subspace GMMs
1:50 – 2:15	Lukas Burget	More details on SGMMs
2:15 – 2:35	Petr Schwarz	Experimental setup with CallHome and main results
2:35 – 2:40	Mohit Agarwal	Clustering for SGMM initialization
2:40 – 3:00	Arnab Ghoshal	Adaptation with SGMM system

Second block

3:15-3:25	Rick Rose	Experiments with Wall Street Journal
3:25-3:45	Nagendra Goel, Samuel Thomas, Pinar Akyazi	Lexicon learning
3:45-4:00	Daniel Povey	Technical overview of SGMM training/
4:00-4:05		Summary
4:05-4:30	Questions and Discussion	

JHU CLSP Workshop 2009

Motivating Sub-Space Modeling for Automatic Speech Recognition

July 29, 2009

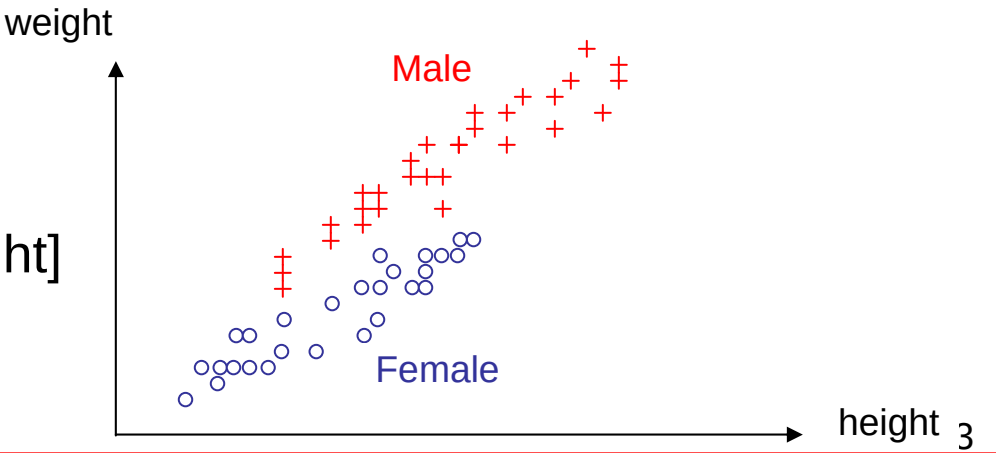
OUTLINE

- Introduce the Notion of Identifying Low Dimensional Subspaces over Model Parameters
- Subspace Based Speaker Adaptation
- Generalization to a Generalized Joint Subspace Model of Acoustic Variability in ASR

Identifying Low Dimensional Feature Space – Dimensionality Reduction

- Problems with high dimensional ***feature spaces***:
  - High computational complexity
  - Poor generalization to unseen data
  - "Curse of dimensionality"
- Starting with high dimensional ***data***, identify low dimensional feature space
  - Principal components analysis (PCA) – Capture maximum variance
  - Linear Discriminant Analysis (LDA) – Maximum class separability

Example:
2-D Feature Space: [height,weight]
2 Classes: male,female

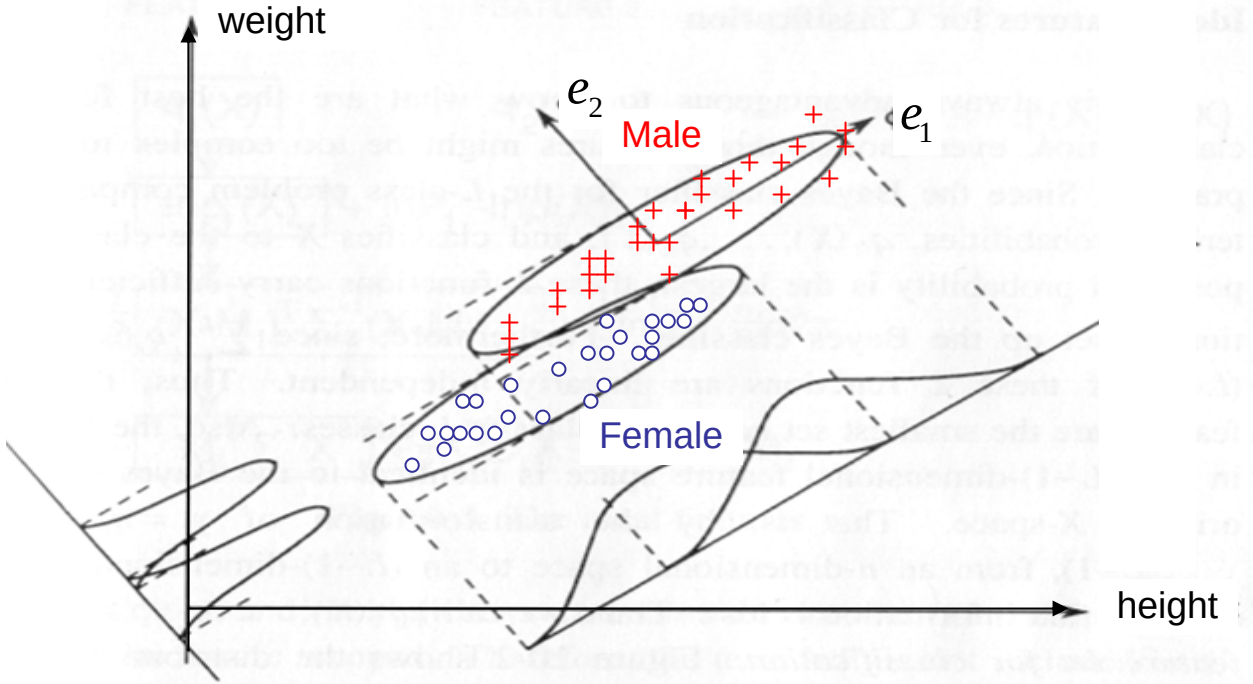


Identifying Low Dimensional Feature Space

- Low dimensional feature, y, obtained from high dimensional feature, x, by a linear transformation:

Maximum Separability: $y = e_2^T x$ (Linear Discriminant Analysis)

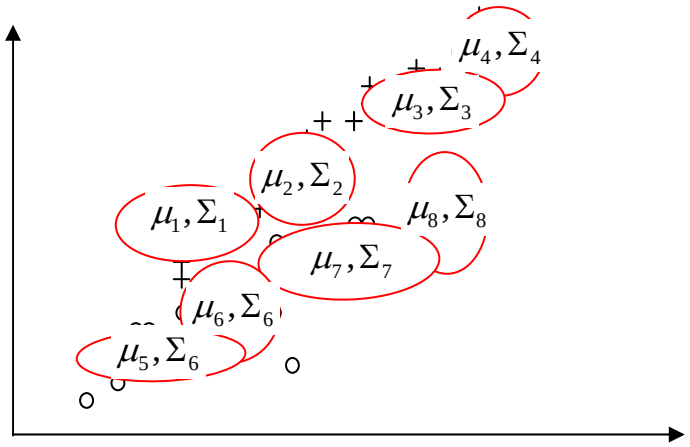
Maximum Variance: $y = e_1^T x$ (Principal Components Analysis)



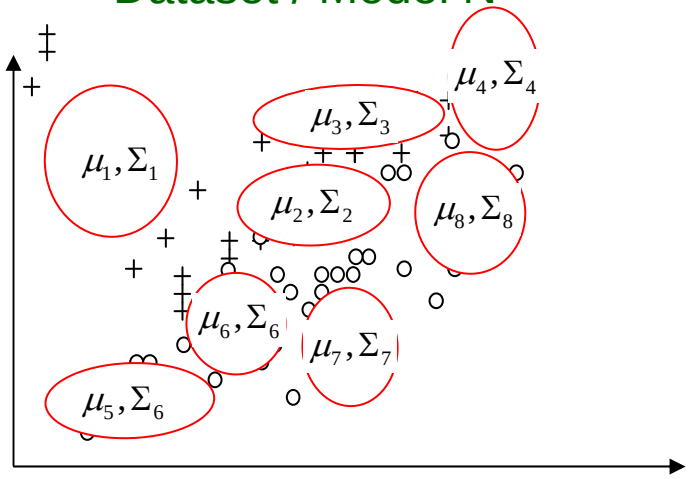
Identifying Low Dimensional Model Space

- Suppose there are multiple data sets describing similar populations and models are to fit each data set:

Dataset / Model 1



Dataset / Model N



- A low dimensional subspace can be identified that describes variation of the parameters

Form Super Vectors from Models:

$$\boldsymbol{\mu}^1 = \begin{bmatrix} \mu_1^1 \\ \vdots \\ \mu_8^1 \end{bmatrix}, \cdots, \boldsymbol{\mu}^N = \begin{bmatrix} \mu_1^N \\ \vdots \\ \mu_8^N \end{bmatrix}$$

Estimate Sub-space Projection:

$$\boldsymbol{\mu} = \mathbf{E}\boldsymbol{\nu}$$

… or:

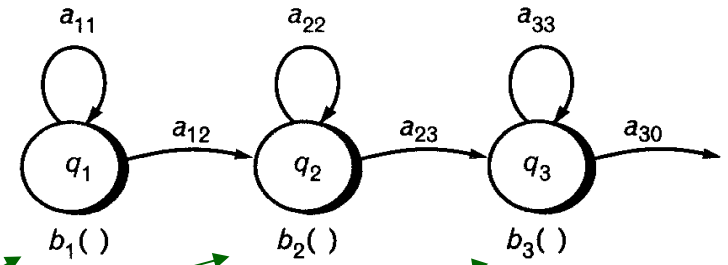
$$\boldsymbol{\mu} = \mathbf{m}_0 + \mathbf{E}\boldsymbol{\nu}$$

Electrical & Computer Engineering

Speaker Space Adaptation – Super-vector

- Continuous Gaussian Mixture Observation Density HMMs

State dependent Gaussian Mixtures



$$p(\vec{x} \mid s_j) = \sum_{m=1}^{C_j} w_{m,j} f_{m,j}(\vec{x})$$, with $C = \sum_m C_m$ mixture components, and $f_{m,j}(\vec{x}) : N\left[\vec{x}; \vec{\mu}_{m,j}, \Sigma_{m,j}\right]$

- A speaker, *S*, is generally defined over a "super-vector" of the concatenated means of component Gaussians:

$$\vec{\mu}^s = \begin{bmatrix} \mu_1^s \\ \mu_2^s \\ \vdots \\ \mu_C^s \end{bmatrix}$$ Dimension: $M = CF$

- Example: Wall Street Journal HMM
  - Component Gaussians $C \approx 100,000$
  - Feature Vector Dimension $F \approx 40$
  - Super Vector Dimension $CF \approx 4,000,000$
- Super-vector dimension can be very large

Speaker Space Based Adaptation

- Adapt Super-vector in Low Dimensional Subspace
- Training (Off-Line): Identify basis vectors of low dimensional speaker subspace from speaker dependent super-vectors:

$$\vec{\boldsymbol{\mu}}^1,\ldots,\vec{\boldsymbol{\mu}}^S \longrightarrow \mathbf{E}=\vec{\mathbf{e}}^1,\ldots,\vec{\mathbf{e}}^K$$

where $\vec{\boldsymbol{\mu}}^s$ is dimension, *M*, and $\mathbf{E}$ is dimension $M \times K$ where $K << M$

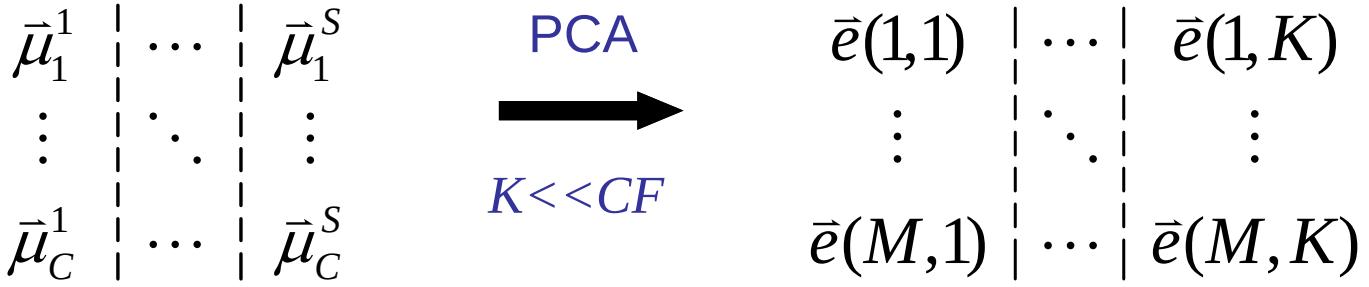
- Adaptation: Estimate weights $w_k^s, k=1,...,K$ from adaptation data to obtain adapted super-vector:

$$\hat{\vec{\boldsymbol{\mu}}}^s = \vec{\boldsymbol{\mu}}^{SI} + \sum_{k=1}^{K} w_k^s \vec{e}(k)$$

- Requires only a few seconds of adaptation data
- Speaker subspace dimension $K \approx 10 \rightarrow 100$

Subspace Identification – Training

- Principal Components Analysis (EigenVoices)
  - Starting from M dimensional super-vectors for each of S speakers to a K dimensional subspace



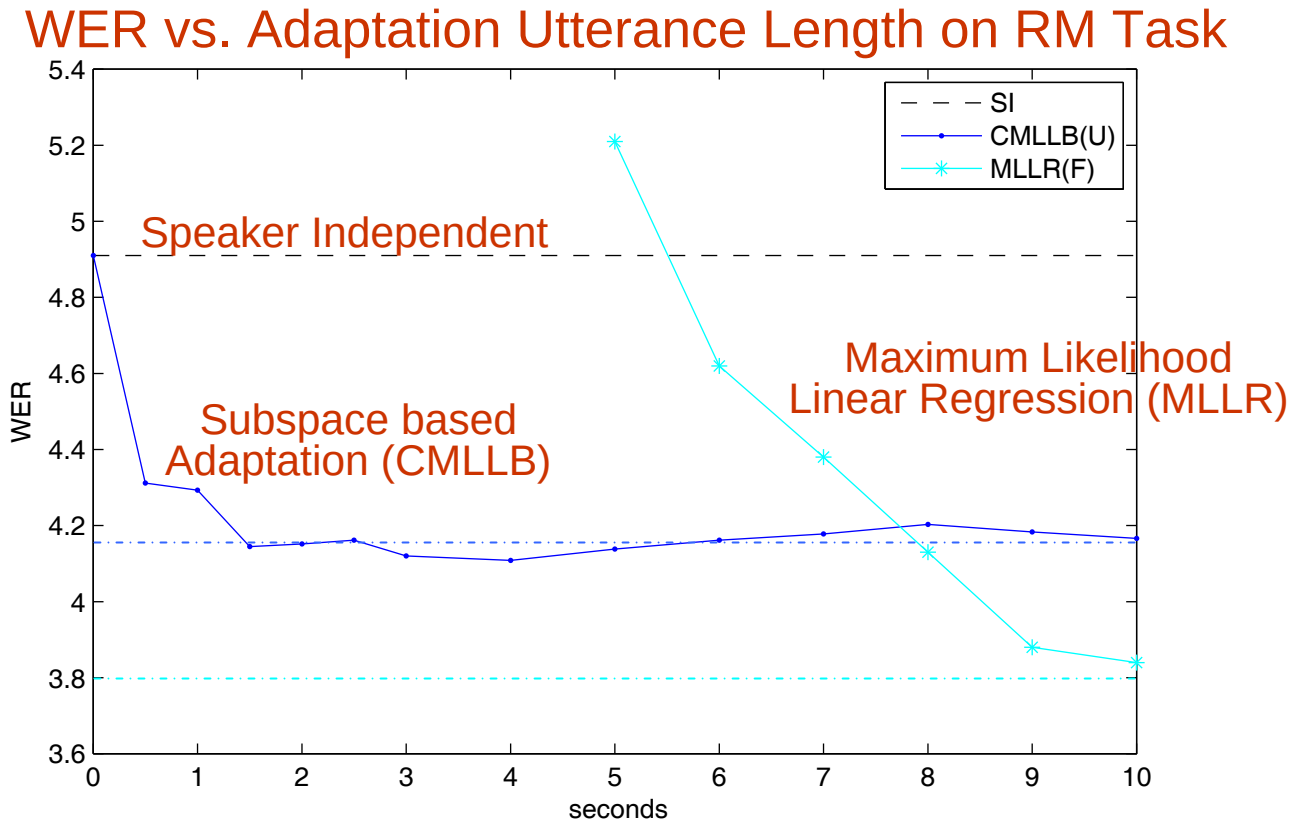
Speaker Super-vectors for Speaker 1 through Speaker S

Speaker Subspace basis vectors

- Maximum Likelihood Clustering (Cluster Adapt. Training)
  - Given SD training observation vectors, $X^s = x_1^s, \ldots, x_T^s$, SI HMM model, λ^{SI}, and initial estimate of $\mathbf{E}^{(0)} = \vec{\mathbf{e}}^1, \ldots, \vec{\mathbf{e}}^K$
  - Use EM algorithm to iteratively estimate weights and basis vectors

$$\hat{\Lambda}: \quad \hat{\vec{\boldsymbol{\mu}}}^s = \vec{\boldsymbol{\mu}}^{SI} + \sum_{k=1}^{K} w_k^s \vec{e}(k)$$

Limited Effect of "Global" Subspace Adaptation



[From Tang and Rose, 2008]

- Substantial improvement with only 1 or 2 seconds of adaptation data
- Does not exhibit desirable asymptotic behavior

Generalization to Subspace Models of Phonetic Variability

- Speaker space model is limited in the form of the variability it can represent
  - Single vector in speaker space describes speaker specific variability

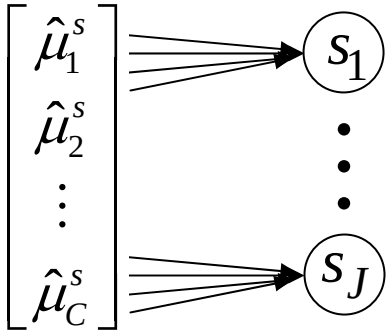
- Generalize this model in three ways:
  1. Define ***multiple model subspaces*** over different regions of the feature space
  2. Define ***state-specific***,(rather than speaker specific) ***weight vectors*** to describe phonetic variation within these subspaces
  3. Define ***joint model / speaker subspaces***

- This is conceptually a straightforward generalization of the speaker subspace approach

Generalization of Sub-Space HMM

- ***Review:*** Subspace Based Speaker Adaptation
  - Single global subspace, $\mathbf{N}$, defined over all Gaussians in HMM
  - Single weight vector, $v^{(s)}$, describes variation in subspace

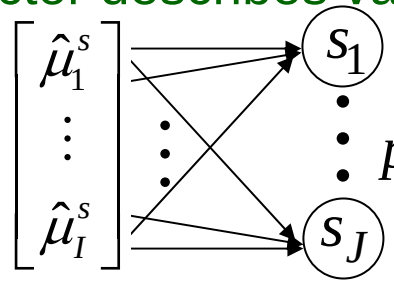
$$\hat{\mu}_{j,m}^{(s)} = \mu_{j,m}^{SI} + \mathbf{N}v^{(s)}$$



$$p(x \mid s_j) = \sum_{m=1}^{M} w_{j,m}\, p(x; \hat{\mu}_{j,m}^{(s)}, \Sigma_{j,m})$$

- ***Generalization***: Multiple Region-Specific Subspaces
  - Separate Subspaces, $\mathbf{N}_i$, defined for each Gaussian in a GMM
  - Single weight vector describes variation in subspaces

$$\hat{\mu}_i^{(s)} = \mu_i^{UBM} + \mathbf{N}_i v^{(s)}$$



$$p(x \mid s_j) = \sum_{i=1}^{I} w_{j,i}\, p(x; \hat{\mu}_i^{(s)}, \Sigma_i)$$

Electrical & Computer Engineering

Generalization to Joint Subspace HMM

- ***Generalization***: State-Specific Weight Vectors
  - Subspaces, $\mathbf{M}_i$, defined over shared pool of Gaussians
  - State-specific weight vectors, $\mathbf{v}_j$, describe phonetic var. in subspace

$$\hat{\mu}_{j,i} = \mathbf{M}_i \mathbf{v}_j \qquad p(x \mid s_j) = \sum_{i=1}^{I} w_{j.i}\, p(x; \hat{\mu}_{j,i}, \Sigma_i)$$

- ***Generalization***: Joint Model / Speaker Subspaces
  - Model and Speaker subspaces, $\mathbf{M}_i$ and $\mathbf{N}_i$
  - State-specific and speaker specific weight vectors $\mathbf{v}_j$ and $v^{(s)}$

$$\hat{\mu}_{j,i}^{(s)} = \mathbf{M}_i \mathbf{v}_j + \mathbf{N}_i v^{(s)}$$

Subspace HMM

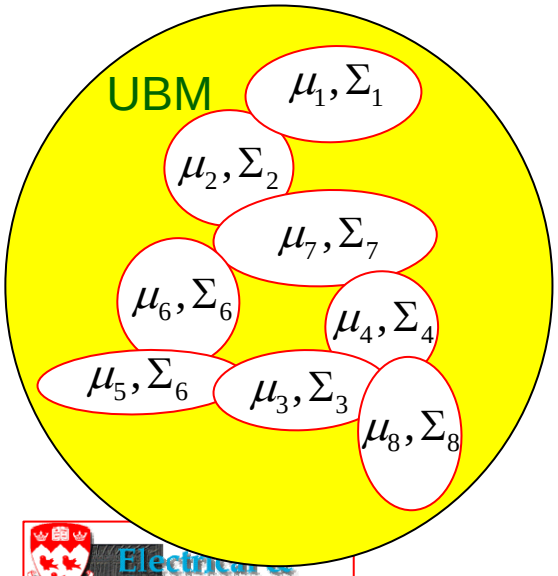
Observation Prob. for State j:

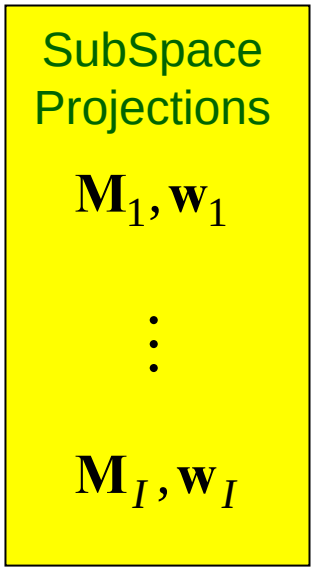
$$p(x \mid s_j) = \sum_{i=1}^{I} w_{j,i}\, p(x; \mu_{j,i}, \Sigma_i)$$

Projection of State-SpecificVectors:

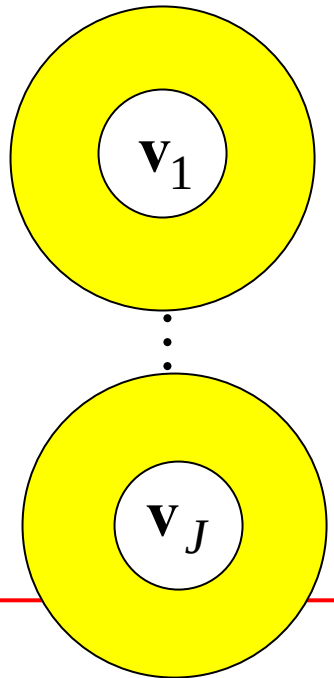
$$\mu_{j,i} = \mathbf{M}_i \mathbf{v}_j \qquad w_{j,i} = \frac{\exp \mathbf{w}_i^T \mathbf{v}_j}{\sum_{l=1}^{I} \exp \mathbf{w}_l^T \mathbf{v}_j}$$

Shared Parameters





State Specific Parameters



State 1

State J

Subspace HMM – Multiple Substates

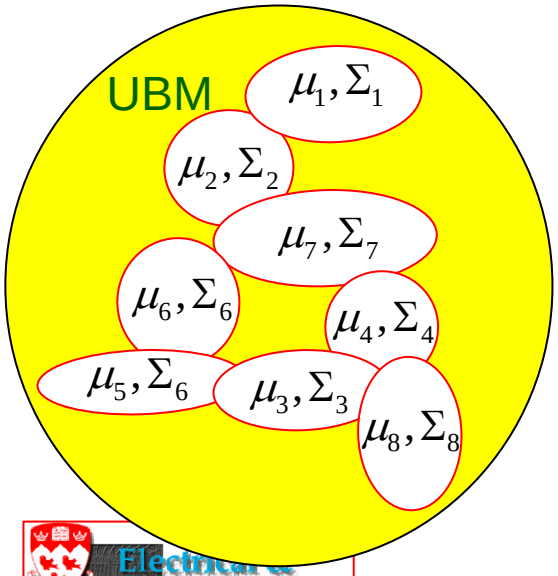
Observation Prob. for State j:

$$p(x \mid s_j) = \sum_{m=1}^{M} c_{j,m} \sum_{i=1}^{I} w_{j,m,i}\, p(x; \mu_{j,m,i}, \Sigma_i)$$

Projection of State-SpecificVectors:

$$\mu_{j,m,i} = \mathbf{M}_i \mathbf{v}_{j,m} \qquad w_{j,m,i} = \frac{\exp \mathbf{w}_i^T \mathbf{v}_{j,m}}{\sum_{l=1}^{I} \exp \mathbf{w}_l^T \mathbf{v}_{j,m}}$$

Shared Parameters

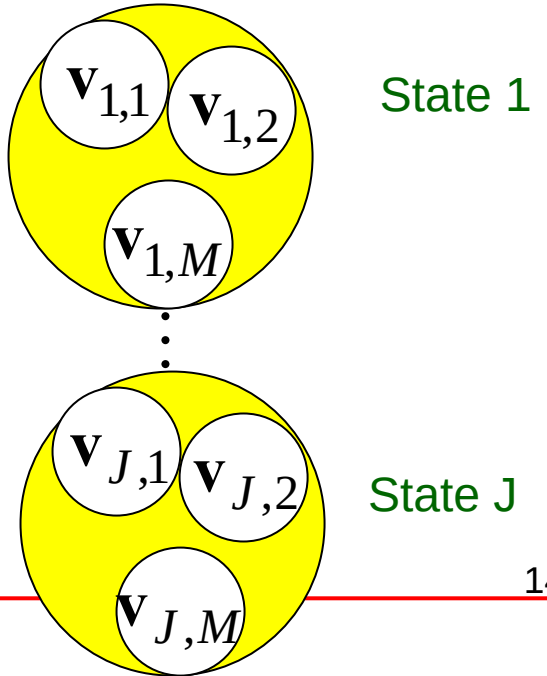


SubSpace Projections

$\mathbf{M}_1, \mathbf{w}_1$

$\mathbf{M}_I, \mathbf{w}_I$

State / Sub-state Specific Parameters



Empirical Trade-Off: Shared and State-Specific Parameters

- Example: Wall Street Journal HMM model:
- 6000 states, 100,000 Gaussians ⇨ ***~8 Million parameters***
- Possible Substate HMM Parameterizations:

Parameter Allocation			Number of parameters		
UBM Gaussians	Sub-space Dim.	Sub-States	Shared	State-Specific	Total
256	39	1	600K	235K	835K
1024	39	1	2.4M	235K	2.65M
1024	100	1	4.8M	600K	5.4M
256	39	16	600K	3.7M	4.3M

Summary

- The workshop is investigating a subspace based alternative to HMM models that includes:
  1. ***multiple model subspaces*** defined over different regions of the feature space
  2. ***state-specific weight vectors*** for describing phonetic variation within these subspaces
  3. ***joint model / speaker subspaces***

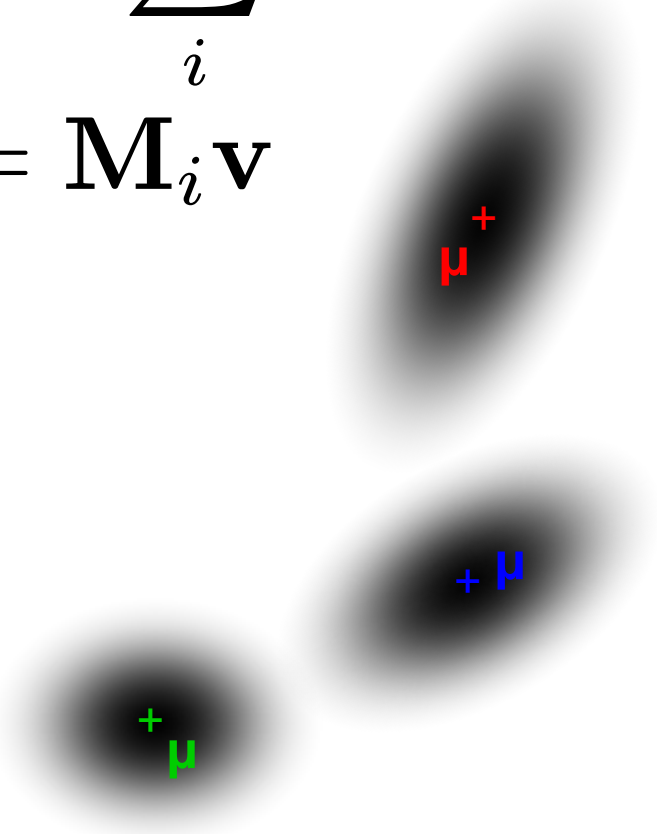
- Workshop goals are to investigate:
  - Potential for sharing training data across languages and task domains
  - Empirical trade-off between number of Gaussians, states, sub-states, and sub-state dimension
  - Effects of joint subspaces: modeling both phonetic and speaker variation

Simple Example of Subspace GMM Model

Subspace GMM Model Example

$$p(\mathbf{x}) = \sum_i w_i \mathcal{N}(\mathbf{x}; \mu_i, \mathbf{\Sigma}_i)$$

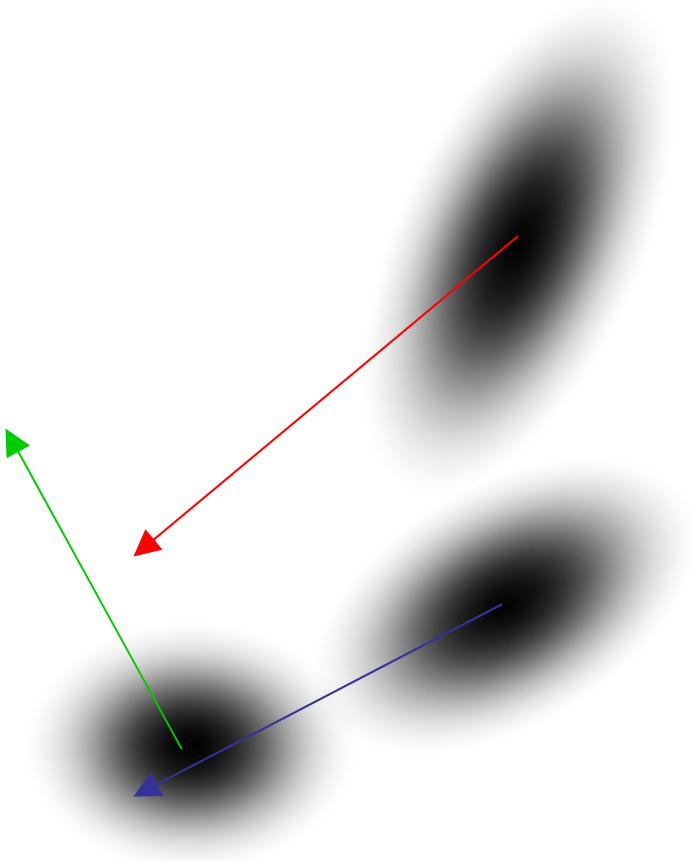
$$\mu_i = \mathbf{M}_i \mathbf{v}$$



$$\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

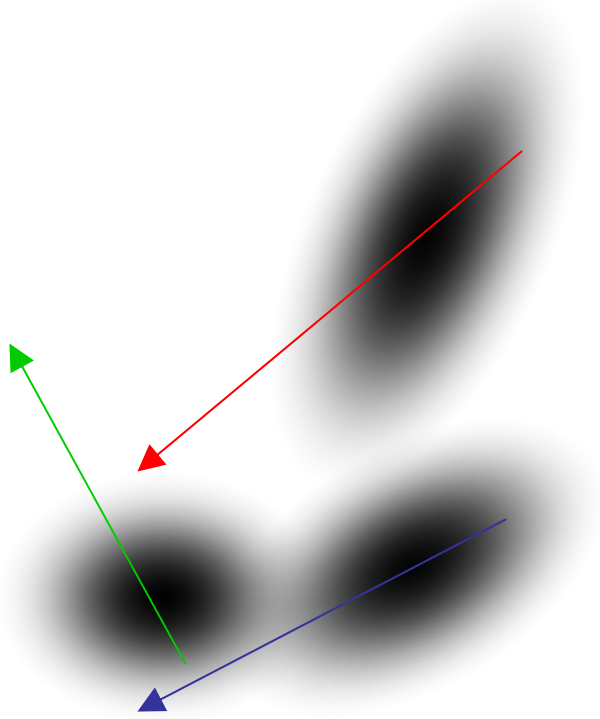
Let w_i and $\sum_i$ be fixed in our model for now.

Subspace GMM Model Example



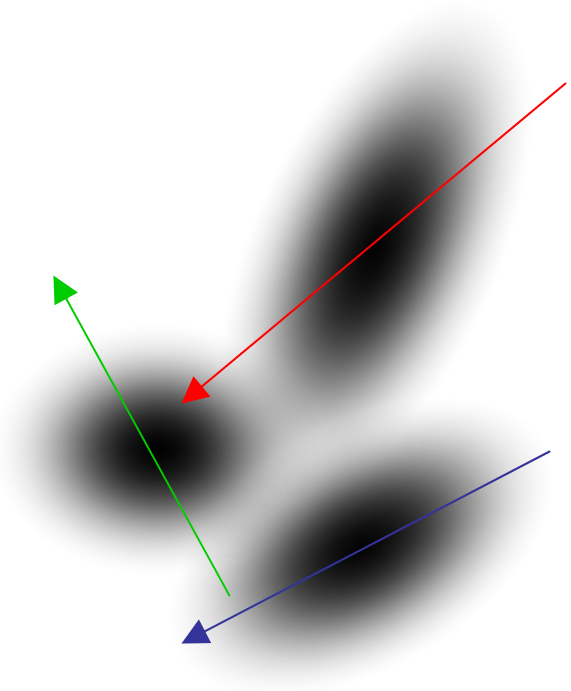
$$\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

Subspace GMM Model Example



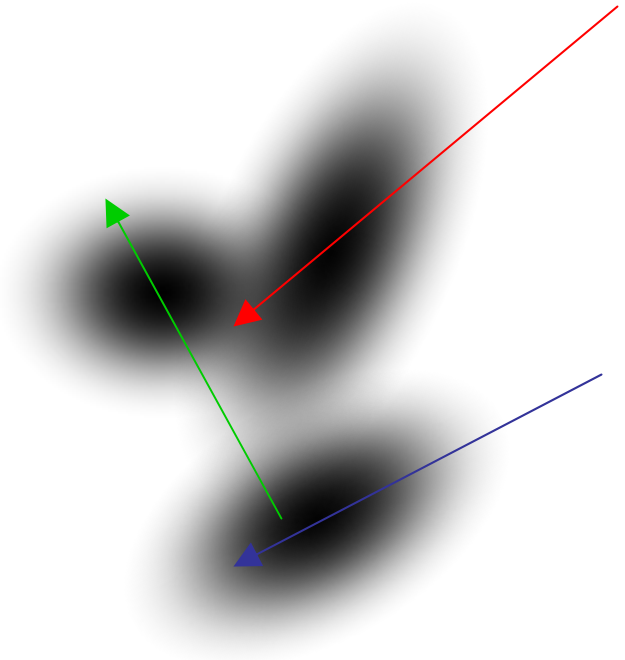
$$\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \end{bmatrix} \begin{bmatrix} \mathbf{v_1} \\ v_2 \\ v_3 \end{bmatrix}$$

Subspace GMM Model Example



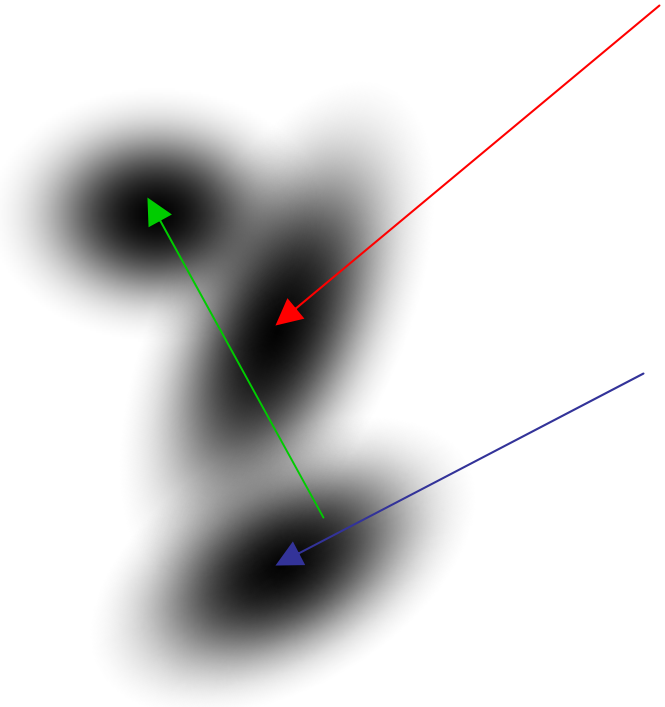
$$\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \end{bmatrix} \begin{bmatrix} \mathbf{v_1} \\ v_2 \\ v_3 \end{bmatrix}$$

Subspace GMM Model Example



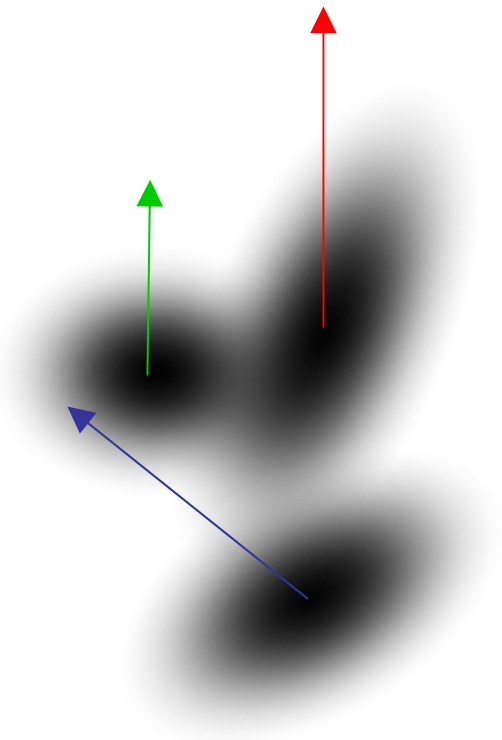
$$\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \end{bmatrix} \begin{bmatrix} \mathbf{v_1} \\ v_2 \\ v_3 \end{bmatrix}$$

Subspace GMM Model Example



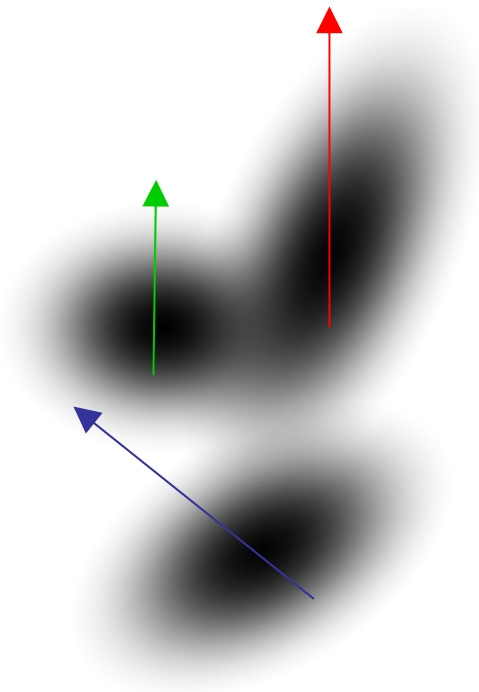
$$\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

Subspace GMM Model Example



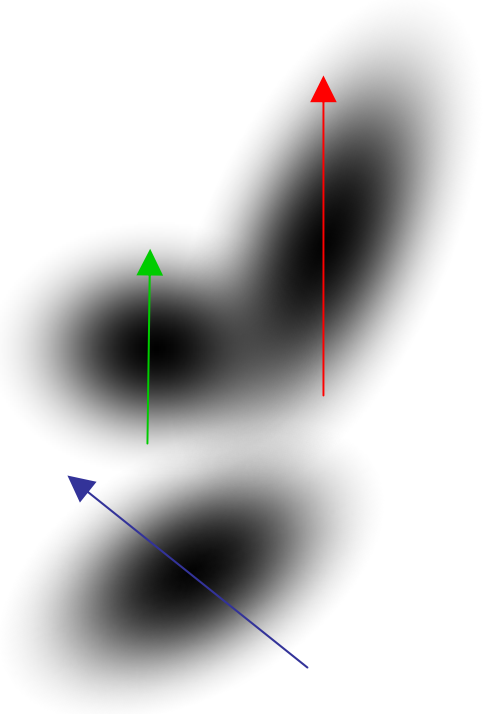
$$\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ v_2 \\ v_3 \end{bmatrix}$$

Subspace GMM Model Example



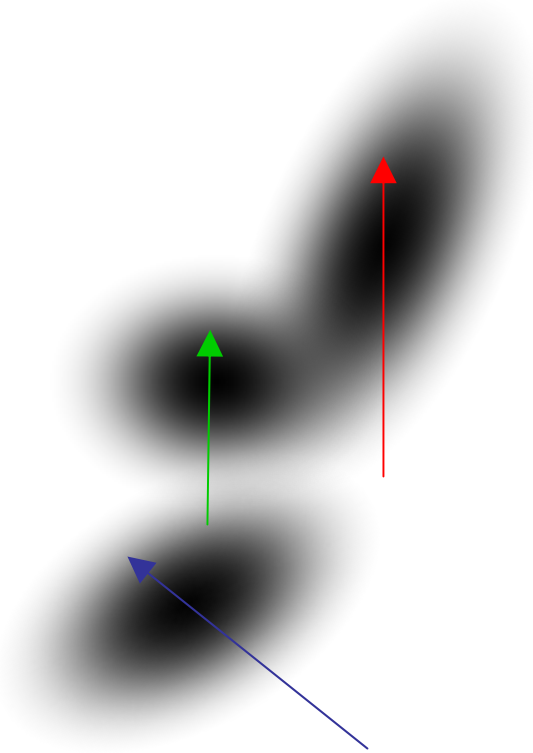
$$\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ v_3 \end{bmatrix}$$

Subspace GMM Model Example



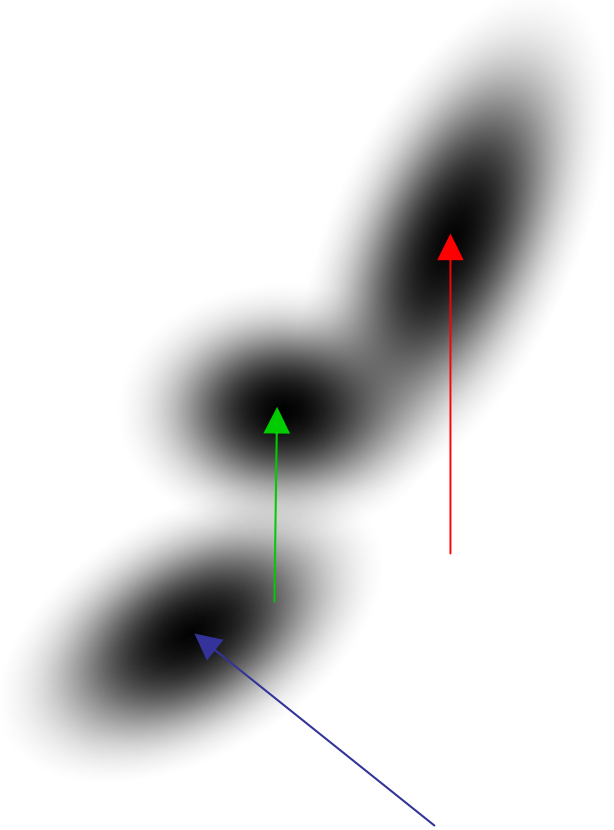
$$\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ v_3 \end{bmatrix}$$

Subspace GMM Model Example



$$\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ v_3 \end{bmatrix}$$

Subspace GMM Model Example



$$\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

Modeling mixture weights

- Log-linear model is used for modeling mixture weights

$$w_i = \frac{\exp \mathbf{w}_i^T \mathbf{v}}{\sum_k \exp \mathbf{w}_k^T \mathbf{v}}$$

$$\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \\ \log \bar{w} \\ \log \bar{w} \\ \log \bar{w} \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ w_1 & w_2 & w_3 \\ w_1 & w_2 & w_3 \\ w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

Modeling mixture weights

$$\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \\ \log \bar{w} \\ \log \bar{w} \\ \log \bar{w} \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ w_1 & w_2 & w_3 \\ w_1 & w_2 & w_3 \\ w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

Modeling mixture weights

$$\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \\ \log \bar{w} \\ \log \bar{w} \\ \log \bar{w} \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ w_1 & w_2 & w_3 \\ w_1 & w_2 & w_3 \\ w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} \mathbf{v_1} \\ v_2 \\ v_3 \end{bmatrix}$$

Modeling mixture weights

$$\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \\ \log \bar{w} \\ \log \bar{w} \\ \log \bar{w} \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ w_1 & w_2 & w_3 \\ w_1 & w_2 & w_3 \\ w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} \mathbf{v_1} \\ v_2 \\ v_3 \end{bmatrix}$$

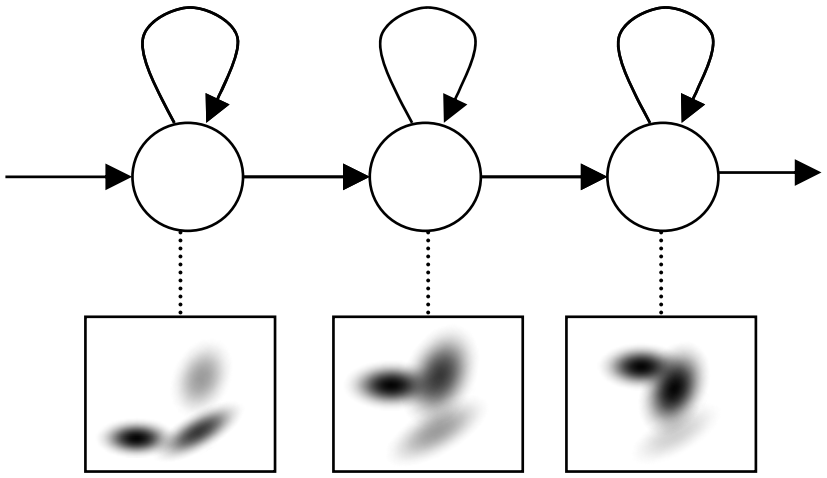
Modeling mixture weights

$$\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \\ \log \bar{w} \\ \log \bar{w} \\ \log \bar{w} \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ w_1 & w_2 & w_3 \\ w_1 & w_2 & w_3 \\ w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} \mathbf{v_1} \\ v_2 \\ v_3 \end{bmatrix}$$

Modeling mixture weights

$$\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \\ \log \bar{w} \\ \log \bar{w} \\ \log \bar{w} \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ w_1 & w_2 & w_3 \\ w_1 & w_2 & w_3 \\ w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ v_2 \\ v_3 \end{bmatrix}$$

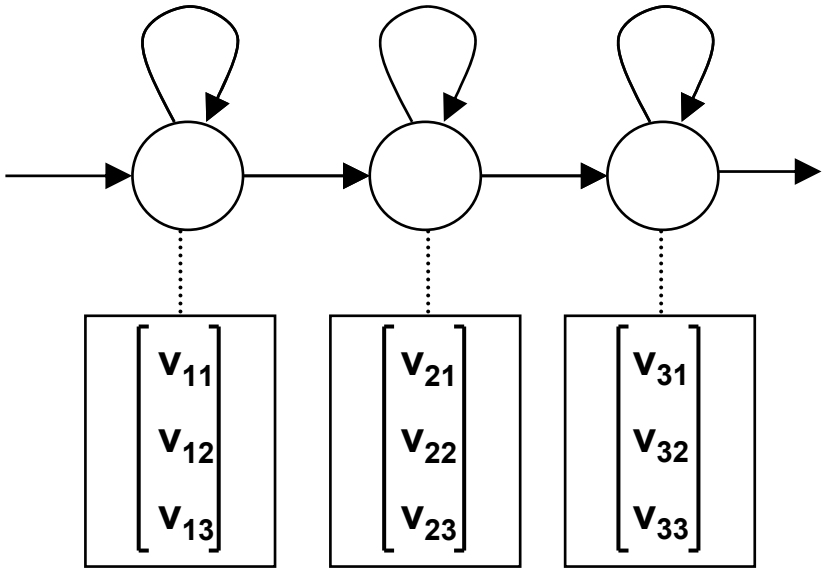
Acoustic model for speech recognition



- Speech sounds are typically modeled by HMMs with state distributions given by GMMs.
- Typically, there are thousands of such models corresponding to context dependent phonemes.
- Many state distributions are very similar and exhibit certain regularities.

Acoustic Model with Subspace GMM

Parameters shared across HMM states
(includes also covariance matrices)



$$\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \\ \log w \\ \log w \\ \log w \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ w_1 & w_2 & w_3 \\ w_1 & w_2 & w_3 \\ w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

State specific parameters
are low dimensional vector

Controlling ratio between shared and state specific parameters

- Increasing number of Gaussian component increase number of shared parameters
- Increasing size of vector **v** increase number of both shared and state specific parameters
- It would be useful to have the possibility of increasing number of state specific parameters without affecting the number of shared parameters

$$\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \\ \log w \\ \log w \\ \log w \\ \log w \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix}$$

Substates – mixture of subspace GMM distributions

- In our experiments, we keep splitting substates to reach the best performance
- Can be seen as an alternative to splitting Gaussians in standard HMM system

$$\begin{bmatrix} \mu_1 & \mu_1 \\ \mu_2 & \mu_2 \\ \mu_1 & \mu_1 \\ \mu_2 & \mu_2 \\ \mu_1 & \mu_1 \\ \mu_2 & \mu_2 \\ \log w & \log w \\ \log w & \log w \\ \log w & \log w \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ w_1 & w_2 & w_3 \\ w_1 & w_2 & w_3 \\ w_1 & w_2 & w_3 \end{bmatrix}$$

Mixture weights

$$\begin{matrix} c & c \end{matrix}$$

$$\begin{bmatrix} v_1 & v_1 \\ v_2 & v_2 \\ v_3 & v_3 \end{bmatrix}$$

Complete Model Definition (so far)

Parameters shared across HMM states

State specific parameters

Gaussian

HMM state

Substate

$$\mu_{jmi} = \mathbf{M}_i \mathbf{v}_{jm}$$

$$\mathrm{w}_{jmi} = \frac{\exp \mathbf{w}_i^T \mathbf{v}_{jm}}{\sum_h \exp \mathbf{w}_h^T \mathbf{v}_{jm}}$$

$$p(\mathbf{x}|j) = \sum_m c_{jm} \sum_i w_{jmi} \mathcal{N}(\mathbf{x}; \mu_{jmi}, \boldsymbol{\Sigma}_i)$$

Experimental part

Overview

- Baseline system
- Subspace system results
- Multilingual setup and results
- Training on very limited amount of data
- Interpreting subspace dimensions

Baseline - data

Acoustic data: CallHome databases

Language	Training set length	Evaluation set length
English	15.1h	1.8h
Spanish	16.5h	2.0h
German	14.7h	3.7h

Language model training:

- English: CallHome, Switchboard I, Switchboard Cellular, GigaWord and web data
- Spanish: CallHome and web data

Baseline systems

- PLP features
- Unadapted ML trained triphone models
- 16 Gaussians per state
- Bi-gram LM for English, tri-gram LM for Spanish
- No LVCSR build for German; results will be reported in terms of phone recognition performance
- The results are in agreement with those reported by other sites on this challenging task

	Accuracy (%)
CallHome English	45.3
CallHome Spanish	31.1

English subspace model training

- Initial configuration:
  - 1921 states
  - 400 Gaussians components
  - 39 dimensional features
  - 40 dimensional state vector – $\mathbf{v}_{jkm}$
  - **952k shared parameters**
  - **77k state specific parameters** (for single substate per state)
- Initial state alignment is taken from baseline system, later realigned by the model itself

Initial results for English

	Shared parameters	State-specific parameters	Accuracy (%)
Baseline	0	2427k	45.3
SGMM, 2k substates	952k	77k	47.5
SGMM, 9k substates	952k	363k	50.3

- For SGMM model, the number of state specific parameters is only a fraction of the number of shared parameters

Initial results for English

	Shared parameters	State-specific parameters	Accuracy (%)
Baseline	0	2427k	45.3
SGMM, 2k substates	952k	77k	47.5
SGMM, 9k substates	952k	363k	50.3

- Increasing the number of substates allow us to balance the ratio between the state specific and the shared parameters
- Still the overall number of the parameters in the SGMM model is less than half compared to the baseline

Searching for optimal configuration

- Tunable parameters:
  - number of Gaussian
  - number of tied states
  - number of substates
  - state vector dimension

- We did not find SGMM to be sensitive to exact setting of the parameters

- Best configuration found was with 3937 tied states, 16k substates, 400 Gaussians and state vector dimension 40

  Accuracy = 50.8 %

Multilingual experiments

- Can data from another languages help to estimate share parameters more precisely?
- English, Spanish and German recognizers are trained together, where
  - each language has its own state specific parameters
  - shared parameters are shared also across languages
  - shared parameters are now trained on 46.3h of training data (English: 15.1h, Spanish: 16.5h, German: 14.7h)

Word recognition experiments

- English system

System	Shared parameters	State-specific parameters	Accuracy (%)
baseline	0	2427k	45.3
English only, 400 G	952k	363k	50.3
All languages, 800 G	1904k	890k	52.1

- Spanish system

System	Shared parameters	State-specific parameters	Accuracy (%)
baseline	0	2006k	31.1
Spanish only, 400 G	952k	312k	34.8
All languages, 800 G	1904k	762k	36.0

Phoneme recognition experiments

- Bigram phonotactic language models were trained on CallHome training sets
- Phoneme recognition accuracy is evaluated

System / Language	English	Spanish	German
# phonemes	42	27	45
baseline	45.1	53.8	43.9
Language only, 400 G	48.3	56.0	46.6
All languages, 800 G	49.8	56.3	47.4

- Training shared parameters across languages results in improved recognition performance for all the languages
- We benefit from increasing the number of shared parameters, which are now trained on more data

Experiments with limited amount of training data

- Can subspace model help us to build recognizer for a language with very limited amount of training data?
- English recognizers is trained, where
  - Shared parameters are trained on Spanish (16.5h) and German (14.7h) data
  - state specific parameters are trained on 1 hour of English

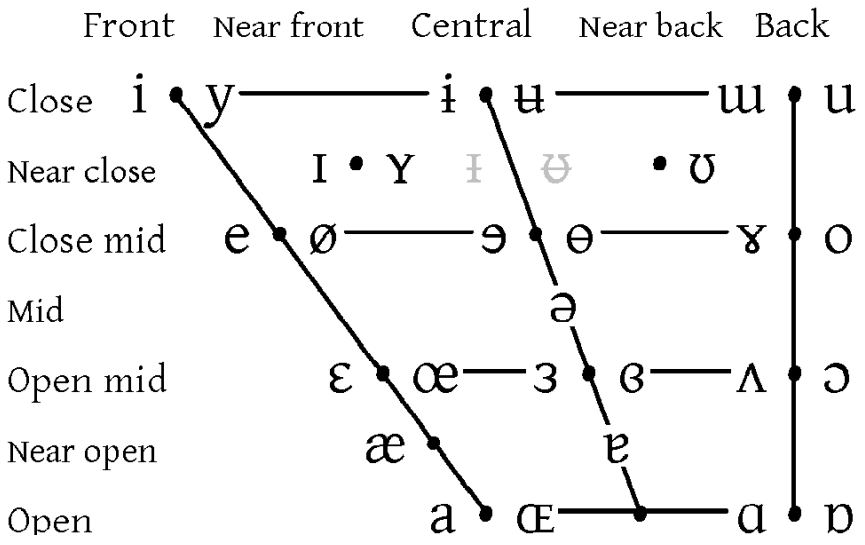
1 hour of training data

System	Accuracy (%)
HTK system, 500 tied states	27.6
SGMM, 1000 tied states, 20 dim, trained on English only	30.9
SGMM, 1500 tied states, 40 dim, shared parameters trained on Spanish + German	37.6

Interpreting subspace dimensions

- The state specific vectors $\mathbf{v}_{jkm}$ are relatively low-dimensional. Can we make the dimension even lower and visualize them?
- Substate system with 5 dimensions was trained
  - the accuracy is 34.2%
  - two most significant dimensions are shown

Vowel chart



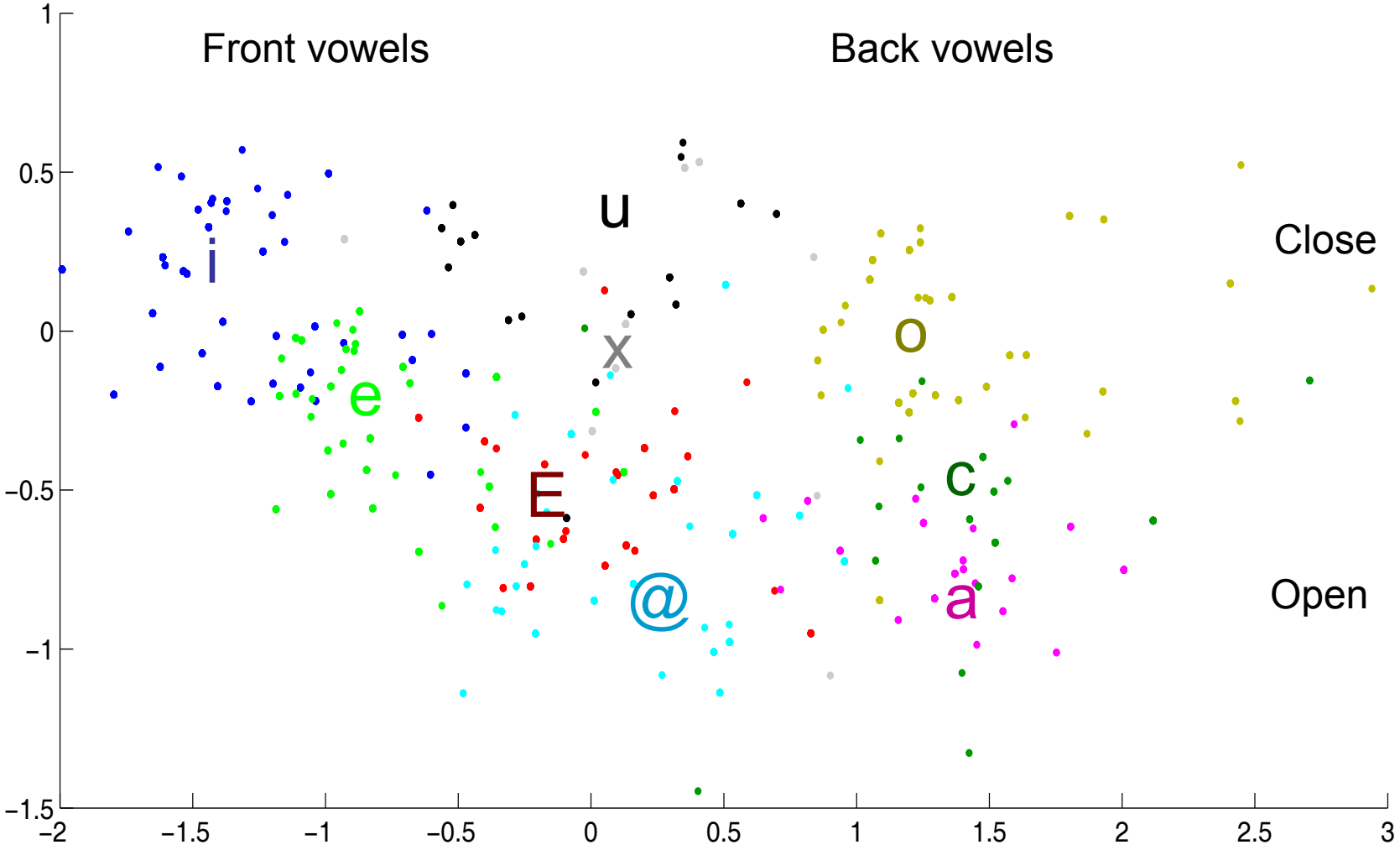
i u

e x o

E c

@ a

Phoneme (state) space

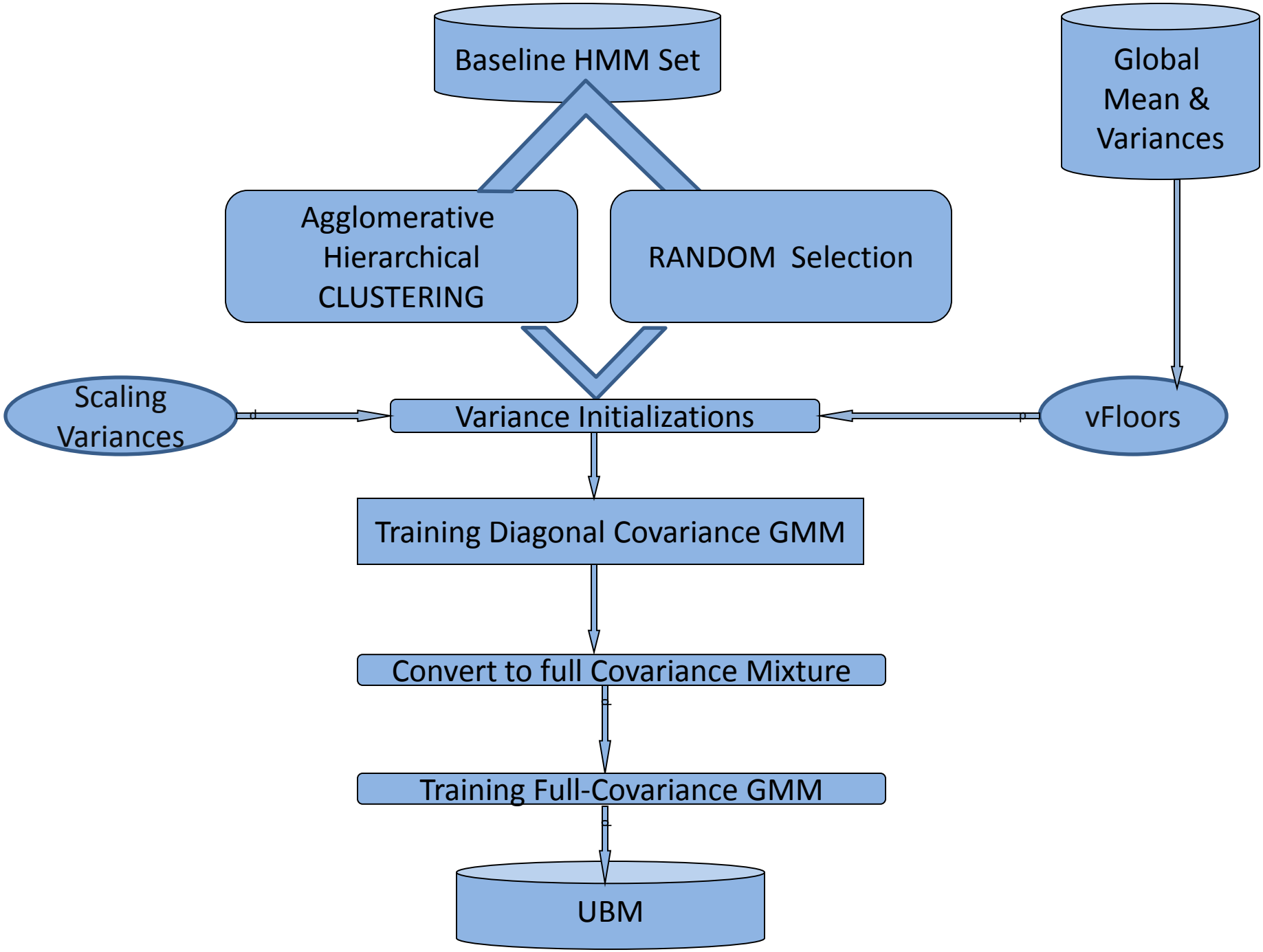


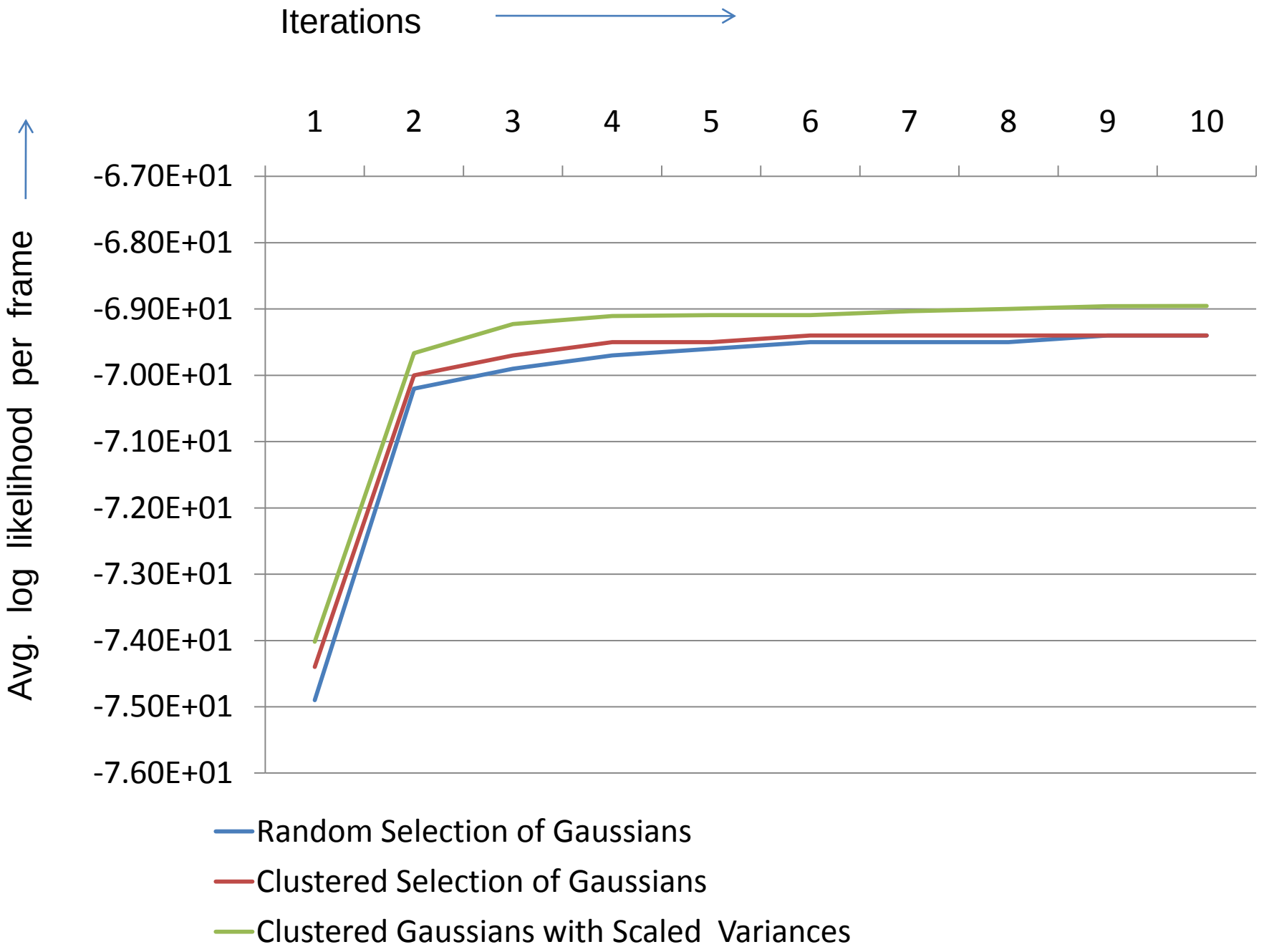
Conclusions

- Subspace GMM system outperform classical GMM system
- Training of subspace GMM shared parameters on multiple languages gives us an advantage
- Subspace GMM system can be successfully used for very limited amount of training data
- Subspace GMM system allows us to visualize state specific parameters. This gives us insight to the system and can serve as an analysis tools.

UBM Initialization

- The Universal Background model (“UBM”) is a full-covariance Gaussian Mixture Model that covers all speech states.
- It is trained on speech data without reference to classes.
- Before training, we typically initialize it from the trained Gaussians of a baseline system.
- I have investigated various ways of doing this.


Baseline HMM Set
Global Mean & Variances
Agglomerative Hierarchical CLUSTERING
RANDOM Selection
Scaling Variances
Variance Initializations
vFloors
Training Diagonal Covariance GMM
Convert to full Covariance Mixture
Training Full-Covariance GMM
UBM


Iterations
1
2
3
4
5
6
7
8
9
10
Avg. log likelihood per frame
-6.70E+01
-6.80E+01
-6.90E+01
-7.00E+01
-7.10E+01
-7.20E+01
-7.30E+01
-7.40E+01
-7.50E+01
-7.60E+01
Random Selection of Gaussians
Clustered Selection of Gaussians
Clustered Gaussians with Scaled Variances

Conclusion

- Proper initialization of variances is important.
- All initialization techniques had similar effect on likelihood.
- We must be careful when choosing likelihood as the measure of quality.
- Also, we need to evaluate this model on whole ASR system.

THANK YOU

Speaker adaptation for Subspace GMM

Arnab Ghoshal

JHU Summer Workshop, 2009

29 July 2009

Outline

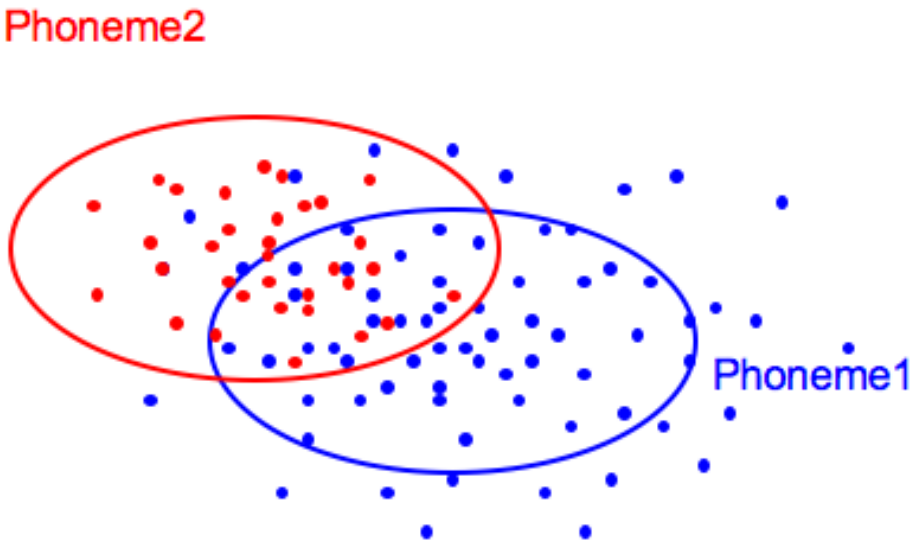
- Speaker-dependent characteristics present in acoustic data
- Modeling speaker characteristics vastly improve recognition performance

Outline

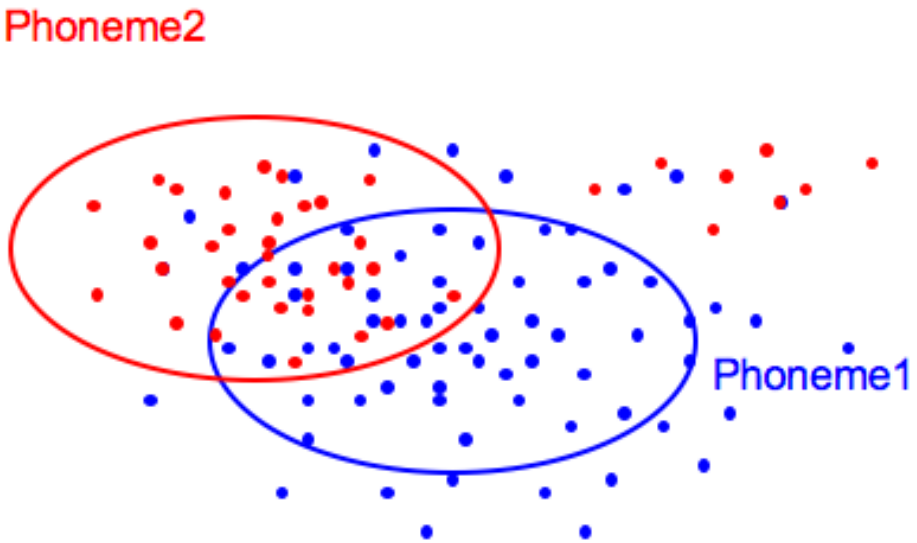
- Speaker-dependent characteristics present in acoustic data
- Modeling speaker characteristics vastly improve recognition performance

1. Speaker vectors
2. Constrained Maximum Likelihood Linear Regression (CMLLR)
3. Subspaces for CMLLR

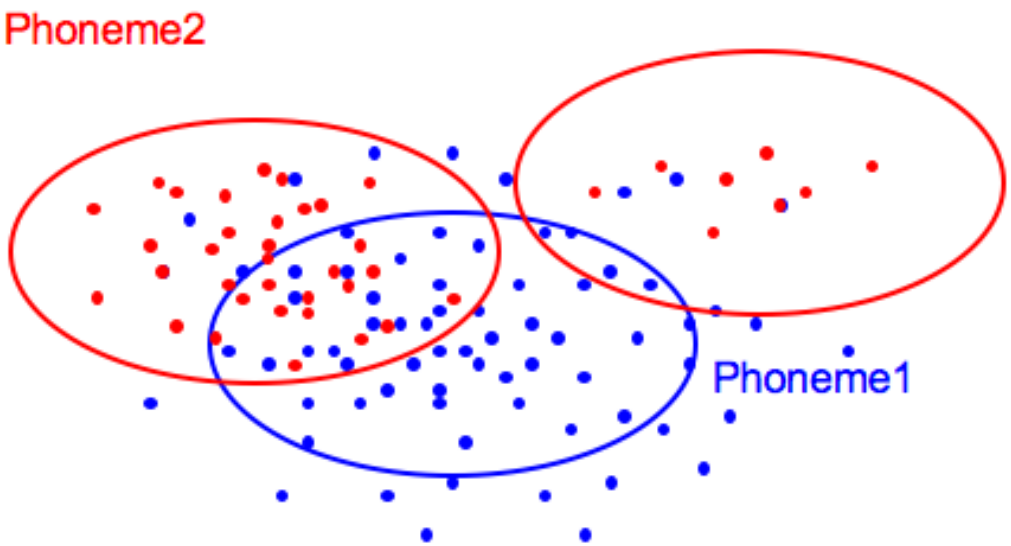
Introduction to speaker subspace



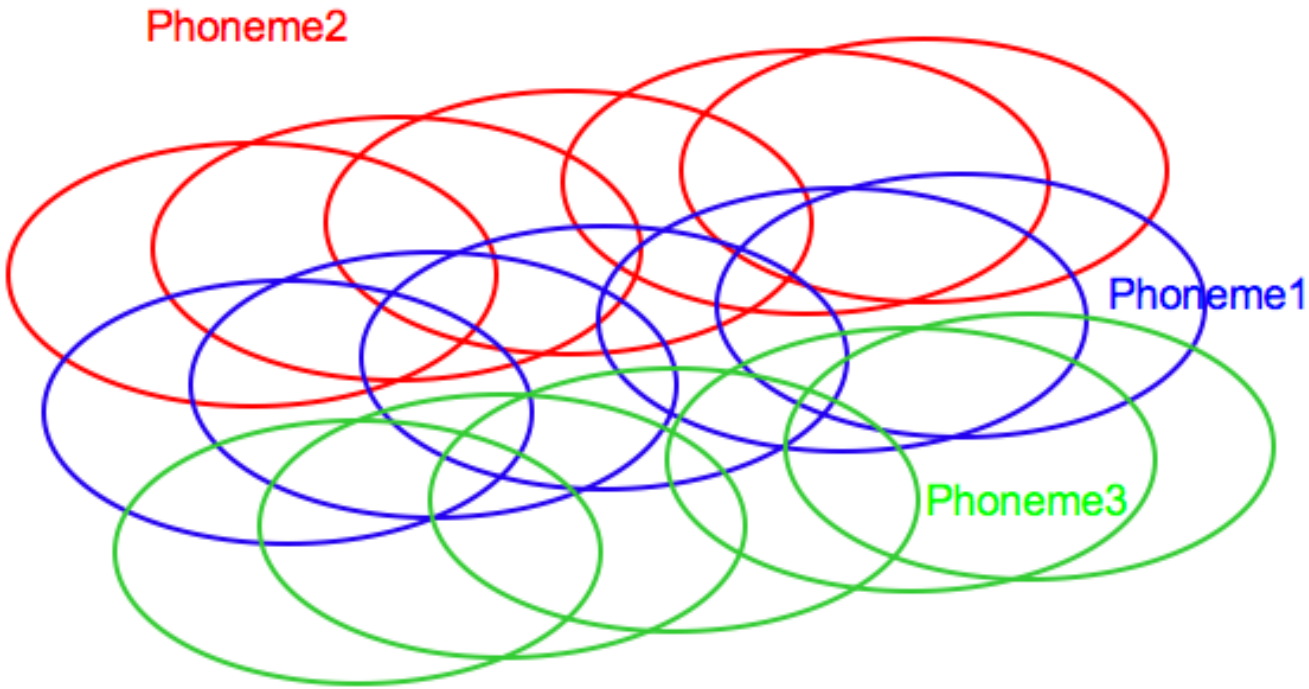
Introduction to speaker subspace



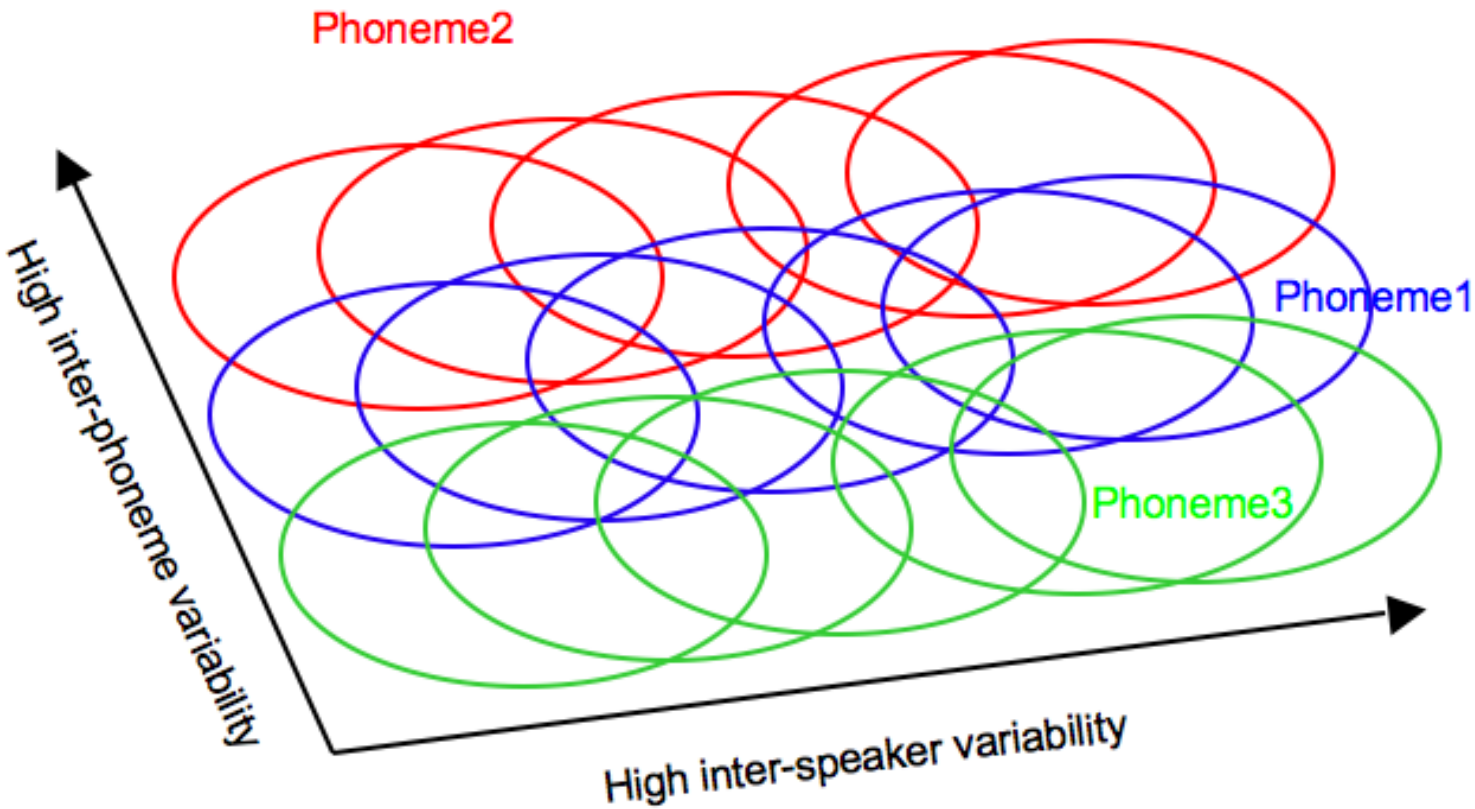
Introduction to speaker subspace



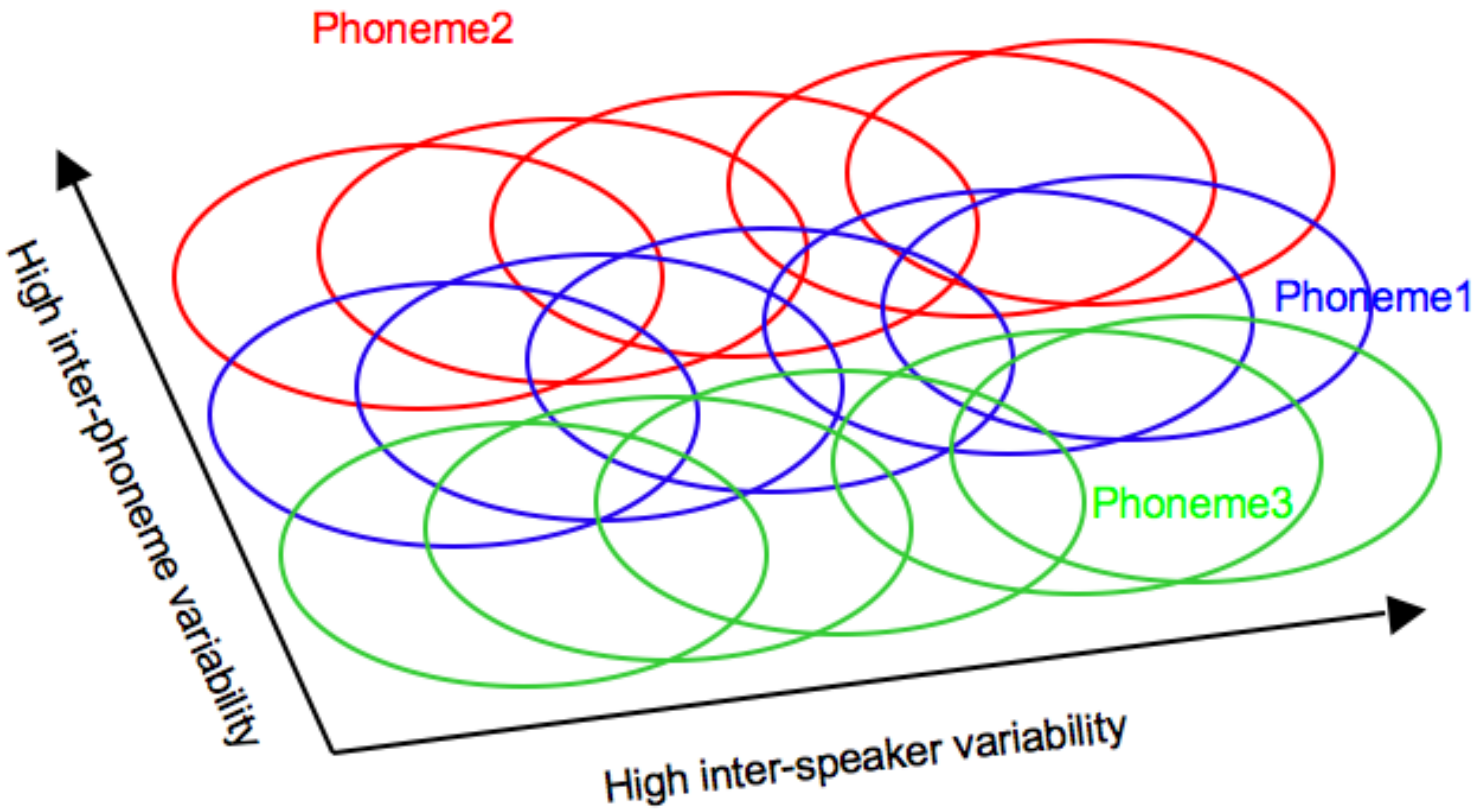
Introduction to speaker subspace



Introduction to speaker subspace



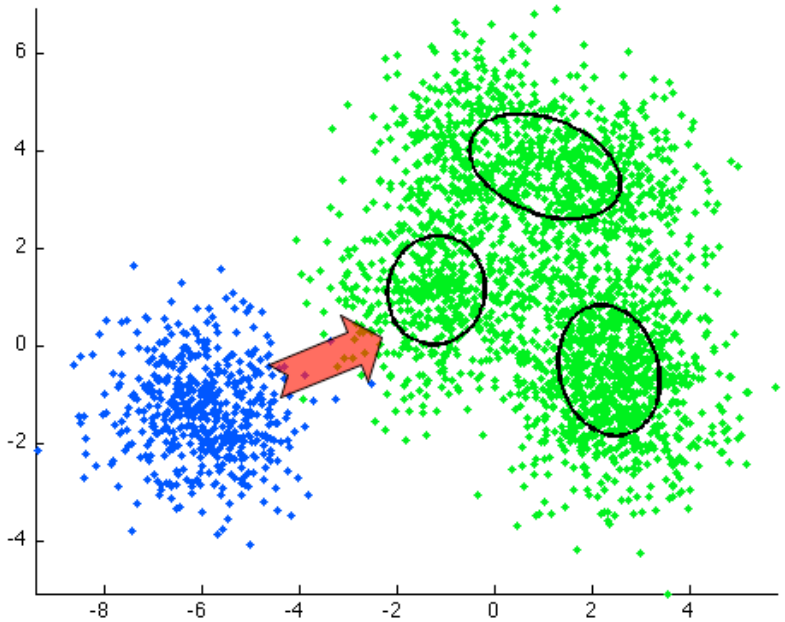
Introduction to speaker subspace



- Speaker-factor in Gaussian mean: $\mu_{jmi} = \mathbf{M}_i \, \mathbf{v}_{jm} + \mathbf{N}_i \, \mathbf{v}^{(s)}$
- Similar to Joint Factor Analysis, used widely in speaker identification systems

Constrained Maximum Likelihood Linear Regression

- Transform of the observation space to maximize likelihood under current model: $\mathbf{x}^{(s)} = \mathbf{A}^{(s)}\mathbf{x} + \mathbf{b}^{(s)}$



- Speaker-specific mean: $\mu^{(s)} = \mathbf{A}^{(s)}\mu + \mathbf{b}^{(s)}$
- Speaker-specific variance: $\Sigma^{(s)} = \mathbf{A}^{(s)}\,\Sigma\,\mathbf{A}^{(s)^\top}$

CMLLR (contd.)

- The auxiliary function is quadratic in $\mathbf{W} = [\mathbf{A}, \mathbf{b}]$
- Estimating $\mathbf{W}$ requires solving $(d^2 + d)$ simultaneous equations in $(d^2 + d)$ variables
  - Invert $(d^2 + d) \times (d^2 + d)$ matrix. $O(d^6)$ complexity.

CMLLR (contd.)

- The auxiliary function is quadratic in $\mathbf{W} = [\mathbf{A}, \mathbf{b}]$
- Estimating $\mathbf{W}$ requires solving $(d^2 + d)$ simultaneous equations in $(d^2 + d)$ variables
  - Invert $(d^2 + d) \times (d^2 + d)$ matrix. $O(d^6)$ complexity.
- Can be simplified for diagonal covariance case. Row-by-row update [Gales & Woodland, 1996].

CMLLR (contd.)

- The auxiliary function is quadratic in $\mathbf{W} = [\mathbf{A}, \mathbf{b}]$
- Estimating $\mathbf{W}$ requires solving $(d^2 + d)$ simultaneous equations in $(d^2 + d)$ variables
  - Invert $(d^2 + d) \times (d^2 + d)$ matrix. $O(d^6)$ complexity.
- Can be simplified for diagonal covariance case. Row-by-row update [Gales & Woodland, 1996].
- For full covariance case, row-by-row update still requires computing $O(d^4)$ statistics [Sim & Gales, 2005].

CMLLR (contd.)

- The auxiliary function is quadratic in $\mathbf{W} = [\mathbf{A}, \mathbf{b}]$
- Estimating $\mathbf{W}$ requires solving $(d^2 + d)$ simultaneous equations in $(d^2 + d)$ variables
  - Invert $(d^2 + d) \times (d^2 + d)$ matrix. $O(d^6)$ complexity.
- Can be simplified for diagonal covariance case. Row-by-row update [Gales & Woodland, 1996].
- For full covariance case, row-by-row update still requires computing $O(d^4)$ statistics [Sim & Gales, 2005].

- Optimal $\mathbf{W}$ can be computed using Newton's method.
- Computing inverse Hessian (matrix of second derivatives) will still require $O(d^4)$ storage, $O(d^6)$ computation.

Our approach

- Transform the *data* and *model*, such that:
  - average within-class variance is unit
  - covariance of the mean vectors is diagonal
  - average mean is zero.
- Transformation with *expected Hessian* simplifies to $O(d^2)$ computation.

Our approach

- Transform the *data* and *model*, such that:
  - average within-class variance is unit
  - covariance of the mean vectors is diagonal
  - average mean is zero.
- Transformation with *expected Hessian* simplifies to $O(d^2)$ computation.

Optimization steps

1. Compute the local gradient: $\mathbf{P}$
2. Compute gradient in transformed space: $\tilde{\mathbf{P}}$
3. Proposed change in $\mathbf{W}$ in this space: $\tilde{\Delta} = \frac{1}{\beta}\tilde{\mathbf{P}}$
4. Transform $\tilde{\Delta}$ back to the original space, and update: $\mathbf{W} \leftarrow \mathbf{W} + k\Delta$
   - Optimal value of k can be computed iteratively

CMLLR Subspaces

- Extending the idea of parameter subspaces to CMLLR transforms
- Express $\mathbf{W}^{(s)}$ as a linear combination of orthonormal basis matrices

$$\mathbf{W}^{(s)} = \mathbf{W}_0 + \sum_{b=1}^{B} \lambda_b^{(s)} \mathbf{W}_b$$

CMLLR Subspaces

- Extending the idea of parameter subspaces to CMLLR transforms
- Express $\mathbf{W}^{(s)}$ as a linear combination of orthonormal basis matrices

$$\mathbf{W}^{(s)} = \mathbf{W}_0 + \sum_{b=1}^{B} \lambda_b^{(s)} \mathbf{W}_b$$

- Helps us to perform speaker-adaptation with relatively little adaptation data

CMLLR Subspaces

- Extending the idea of parameter subspaces to CMLLR transforms
- Express $\mathbf{W}^{(s)}$ as a linear combination of orthonormal basis matrices

$$\mathbf{W}^{(s)} = \mathbf{W}_0 + \sum_{b=1}^{B} \lambda_b^{(s)} \mathbf{W}_b$$

- Helps us to perform speaker-adaptation with relatively little adaptation data
- Integrates seamlessly into our CMLLR estimation framework

Results

- Adaptation per speaker

System	% Accuracy
Baseline	50.3
+ speaker vectors	51.0
+ CMLLR	51.7
+ speaker vectors + CMLLR	52.0

Results

- Adaptation per speaker

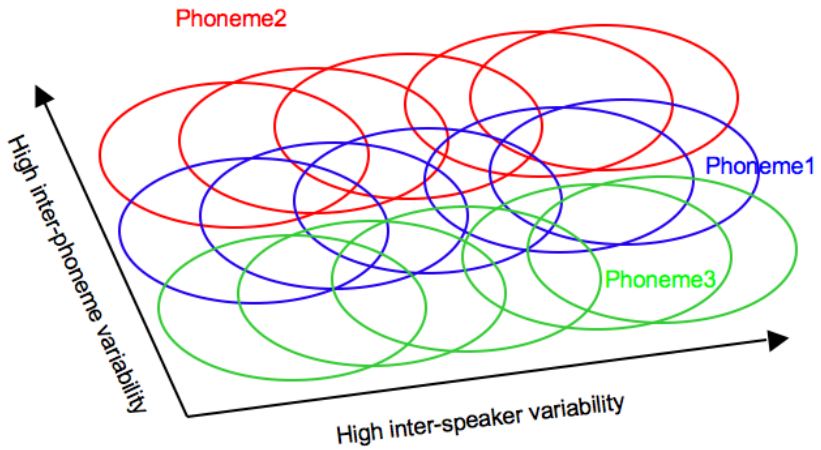
System	% Accuracy
Baseline	50.3
+ speaker vectors	51.0
+ CMLLR	51.7
+ speaker vectors + CMLLR	52.0

- Adaptation per utterance

System	% Accuracy
Baseline	50.3
+ CMLLR	50.3
+ speaker vectors	50.6
+ CMLLR subspaces	50.8

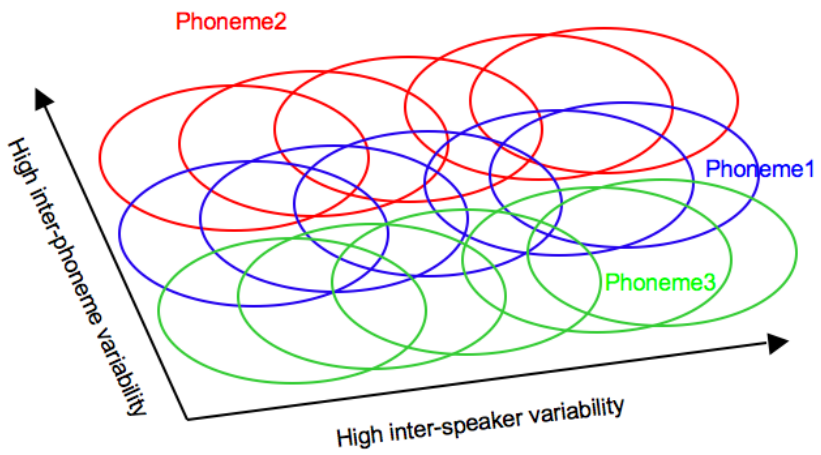
Summary

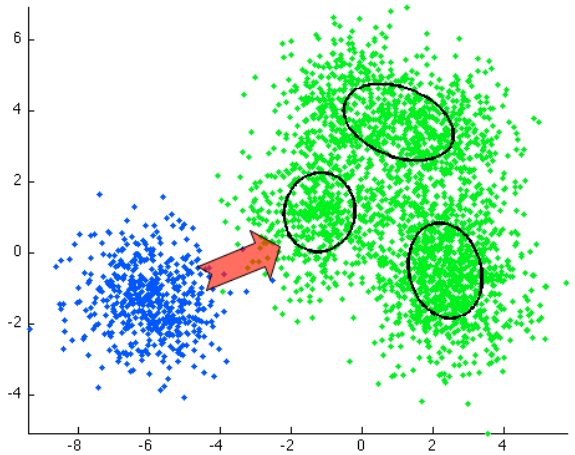
- Accounting for speaker variability using speaker vectors



Summary

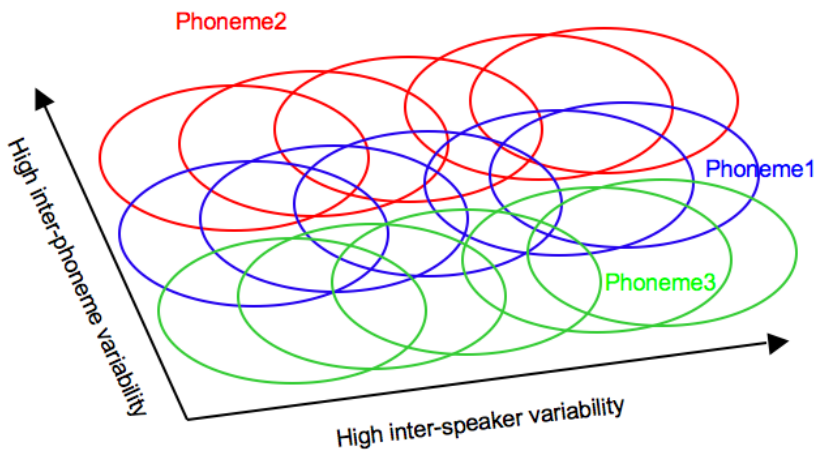
- Accounting for speaker variability using speaker vectors
- Speaker adaptation using novel full-covariance CMLLR estimation

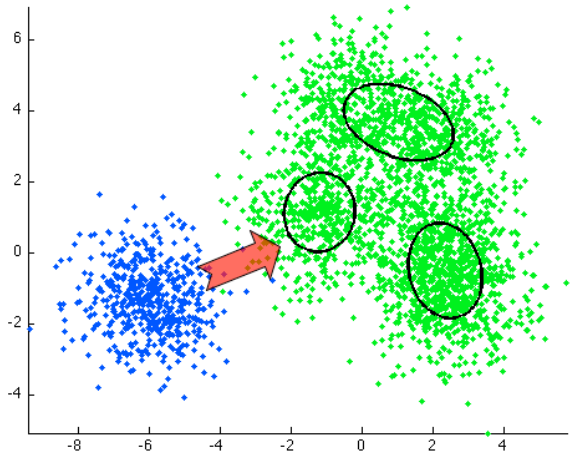




Summary

- Accounting for speaker variability using speaker vectors
- Speaker adaptation using novel full-covariance CMLLR estimation
- In the pipeline: Speaker-adaptive training, parameter tuning, study interactions of different speaker adaptation schemes





JHU CLSP Workshop 2009

An Implementation of Sub-Space Model Based ASR

July 29, 2009

OUTLINE

- Implementation of joint subspace based model (SGMM)
- Training and evaluation on the Wall Street Journal Task

$$p(x \mid s_j) = \sum_{m=1}^{M} c_{j,m} \sum_{i=1}^{I} w_{j,m,i}\, p(x; \mu_{j,m,i}^{(s)}, \Sigma_i)$$

HMM State

Substates

Shared Gaussians/ Subspaces

$$\mu_{j,m,i}^{(s)} = \mathbf{M}_i \mathbf{v}_{j,m} + \mathbf{N}_i v^{(s)}$$

Model Space Model (phonetic variability)

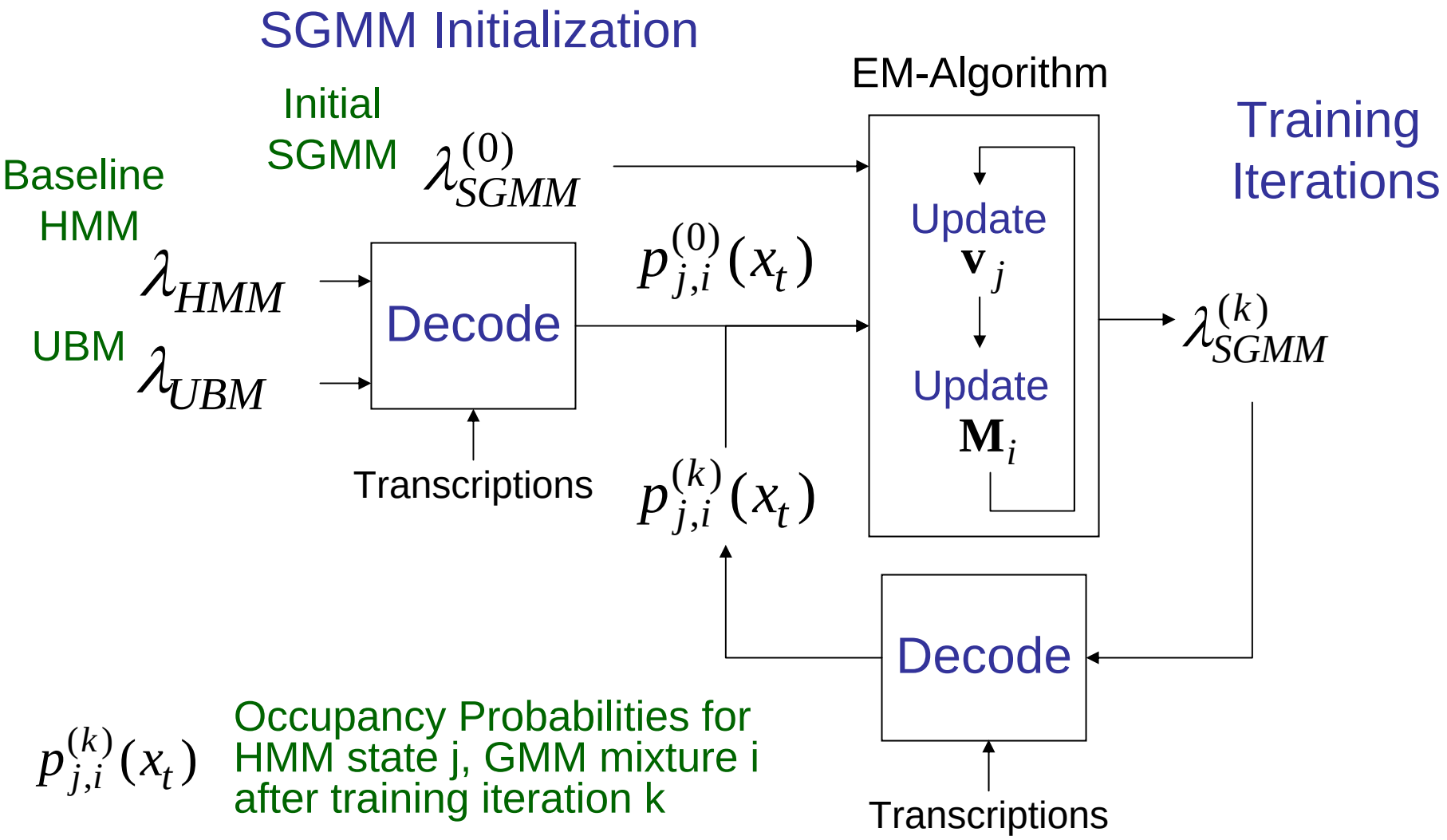
Speaker Space Model (speaker variability)

Training Joint Subspace SGMM Model

$$p(x \mid s_j) = \sum_{m=1}^{M} c_{j,m} \sum_{i=1}^{I} w_{j,m,i}\, p(x; \mu_{j,m,i}^{(s)}, \Sigma_i) \qquad \mu_{j,i}^{(s)} = \mathbf{M}_i \mathbf{v}_j + \mathbf{N}_i v^{(s)}$$

- Initialization:
  - Initial Baseline HMM model λ_{HMM}
  - Pool of full covariance Gaussian mixtures λ_{UBM}
  - Randomly initialized SGMM projection matrices $\mathbf{M}_i^{(0)}$
- Iteratively train model space model parameters
  - EM updates: $\mathbf{M}_i^{(k)}$ $\mathbf{v}_{j,m}^{(k)}$ $c_m^{(k)}$
  - Not estimating weight vectors $w_{j,m,i}$
  - Increase number of sub-states by randomly perturbing weight vectors
- Iteratively train speaker space parameters from initial model space parameters
  - EM updates: $\mathbf{N}_i^{(k)}$ $v^{(s)}$

Model Subspace Estimation



Wall Street Journal Task

- Read Speech Task
- 5K Bigram Language Model
- Close Talking Microphone Quiet Conditions
- Training Data: 988 Speakers, 80 Hours of Speech
- Baseline HMM System:
  - 6015 States
  - 96240 Gaussians
- Baseline HMM-Based ASR Performance on Nov92 WSJ test set: 5.8 % WER
- Additional model adaptation and feature normalization gives only marginal improves over this baseline performance

SGMM Model Space Results

- Baseline HMM: 5.8% WER, ~8 Million parameters
- WER with respect to SGMM Parameterizations:

Parameter Allocation			Number of parameters			Word Error Rate
UBM Gaussians	Sub-space Dim.	Sub-States	Shared	State-Specific	Total	
256	39	1	600K	235K	835K	9.5
256	39	~2	600K	470K	1.07M	8.2
256	39	~3	600K	940K	1.54M	7.6
256	39	~4	600K	1.88M	2.48M	7.3

Discussion

- Model subspace based WER is within 15% of the WSJ HMM baseline
  - Additional tuning (and bug fixes) should make up the difference
- Subspace based speaker adaptation is currently not having an effect on WER
  - Likelihoods improve, WER does not
  - Surprising because of previous success with model space adaptation on WSJ
- Post-workshop work:
  - Clean up short-cuts made in HTK based implementation (and fix bugs in the process)
  - Evaluate effects of SGMM parameterizations on other tasks – Call Home ASR and pronunciation verification task

Thanks

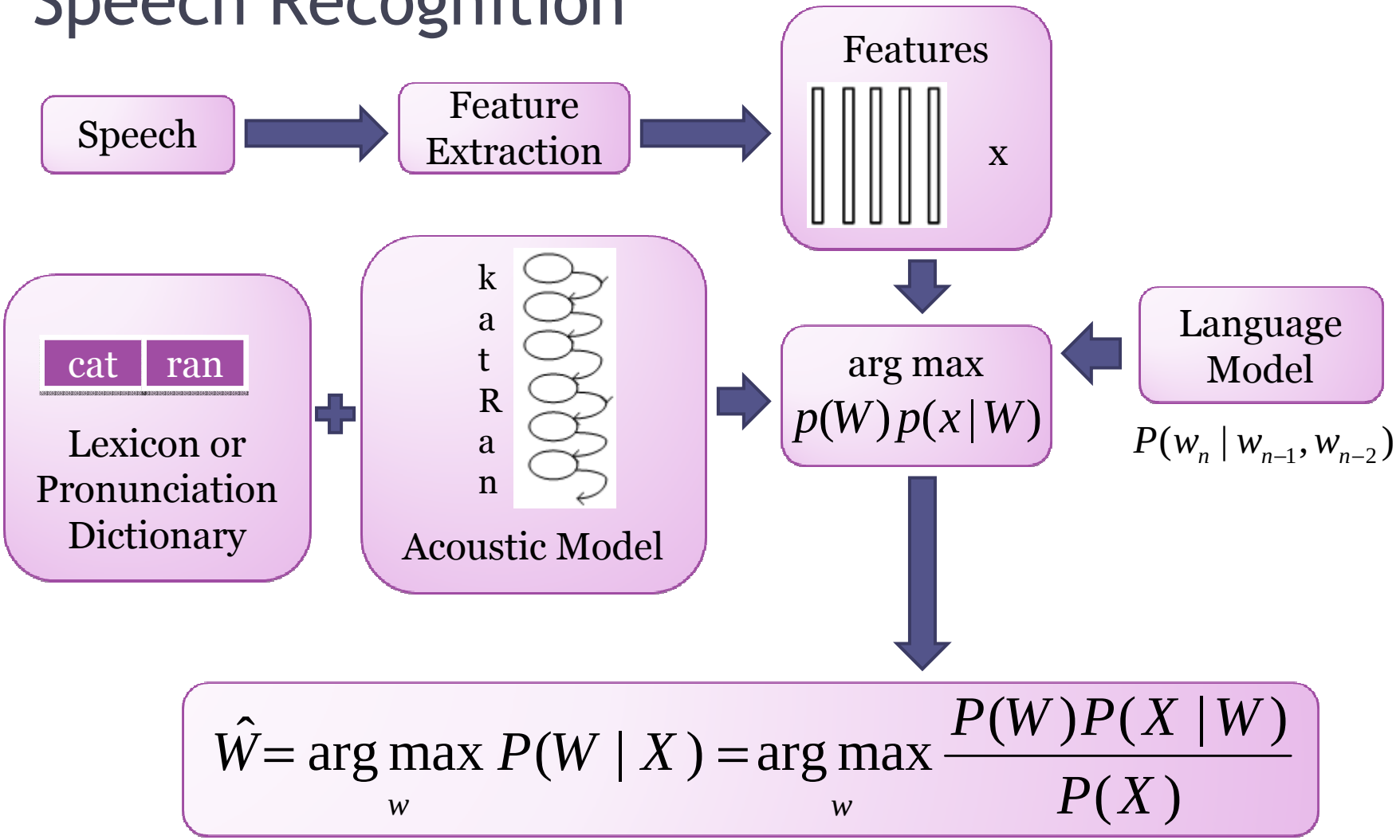
- Acknowledgment: Implementation based on Cambridge University's HTK
- Thanks to BUT for use of their cluster
- Thanks for the help and advice from all of the team members:
  - Mohit Agarwal, Pinar Akyazi, Lukas Burget, Arnab Ghoshal, Nagendra Goel, Dan Povey, Petr Schwarz, Samuel Thomas

  … and McGill colleague Yun Tang
- Thanks to our hosts at the CLSP

Low Cost Lexicon

Nagendra Kumar Goel, Samuel Thomas, Pinar Akyazi

Speech Recognition



$$\hat{W} = \arg\max_{w} P(W \mid X) = \arg\max_{w} \frac{P(W)P(X \mid W)}{P(X)}$$

Cost of Developing a New Language

- Transcribed audio data
  - Subspace acoustic models (UBM's) need less data
- Text data for language modeling
  - Obtain from the web if possible
- Pronunciation Lexicon
  - Qualified phoneticians are expensive
  - Phoneticians may make mistakes
  - Conversational (callhome) English has 4.6% OOV rate for a 5K lexicon and 0.4% for a 62K lexicon
  - Try to guess pronunciation given a limited lexicon and audio

Estimating Pronunciations

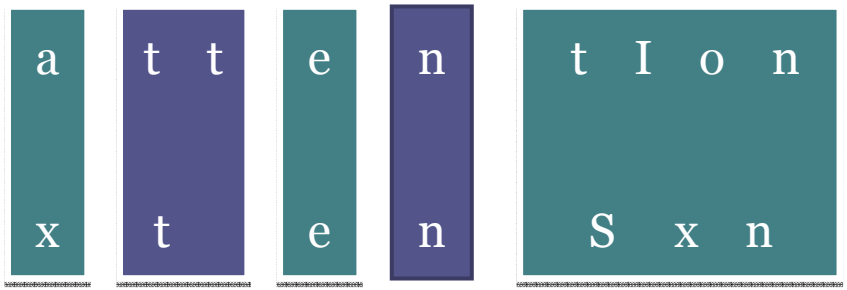
- Ideal Situation will be to just estimate all the pronunciations for the word that maximize the likelihood given the audio

$$\hat{Prn} = \arg\max_{Prn} P(X \mid Prn)$$

- There are words for which spoken audio is not available but they need to exist in the recognizer.
- Multiple pronunciations have not yet significantly inproved the performance
- This objective function needs a lot of regularization

Estimating Pronunciation from Graphemes

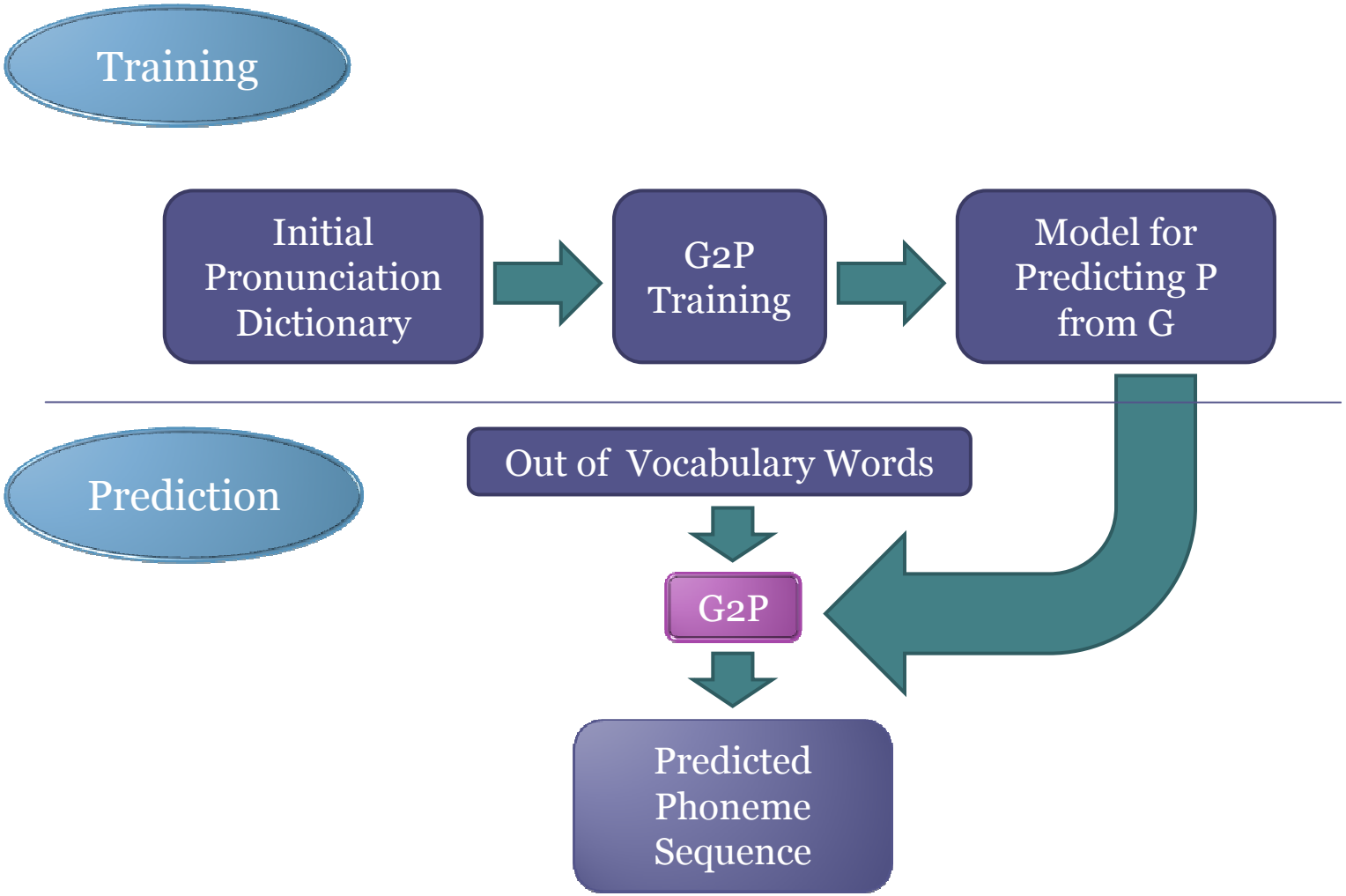
- One way is to guess the pronunciation from the orthography of the word (e.g. Bisani & Ney)
- Iterative process based on grapheme/phoneme alignment
  - Start with an initial set of graphone probabilities.



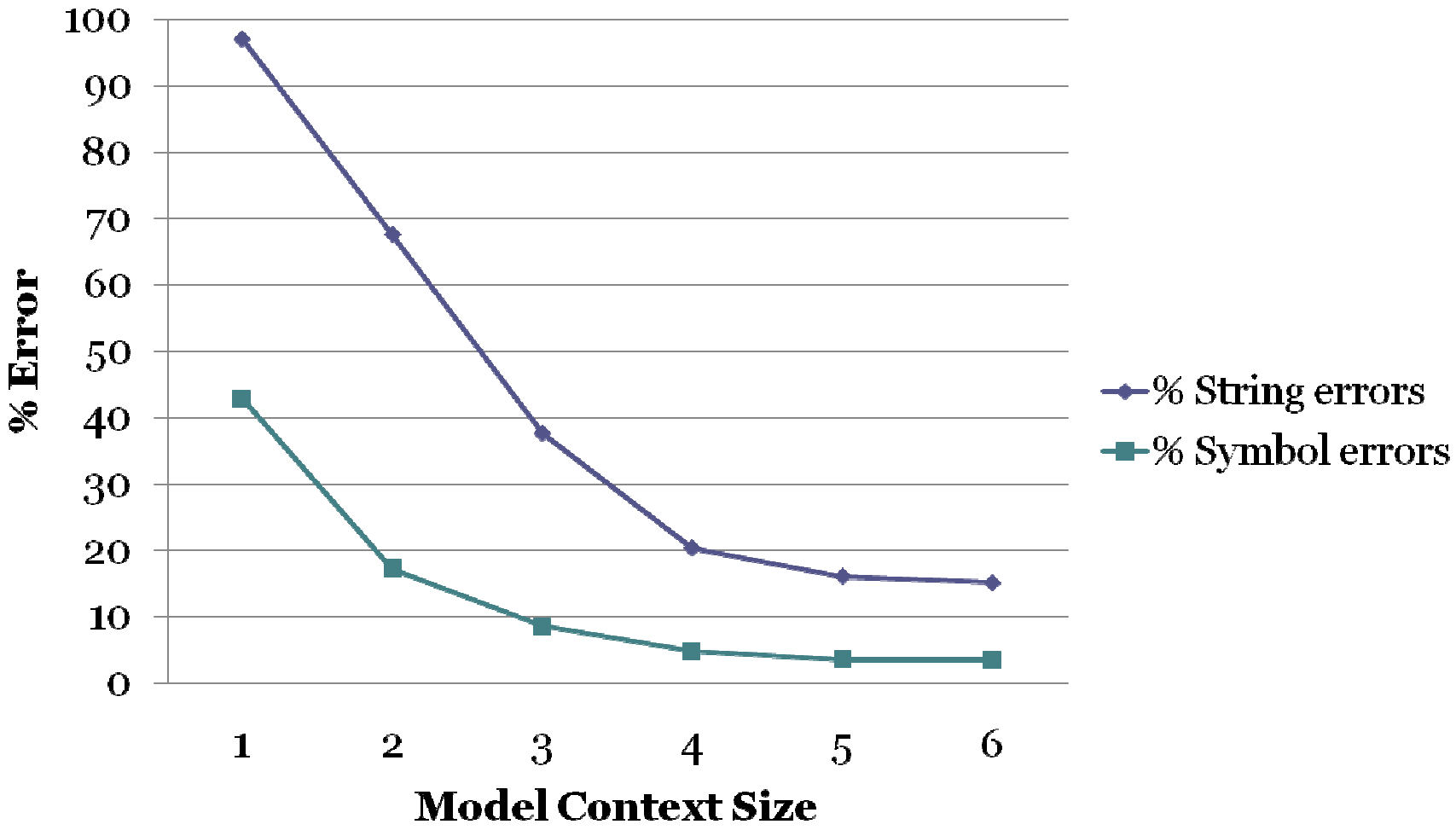
$$\hat{Prn} = \arg\max_{Prn} P(W, Prn)$$

  - Use the probabilities to realign graphones with phones on training data.
  - Re-estimate graphone probabilities from the alignments.

Training a Pronunciation Dictionary



G2P Plot for English



Estimating Pronunciations…

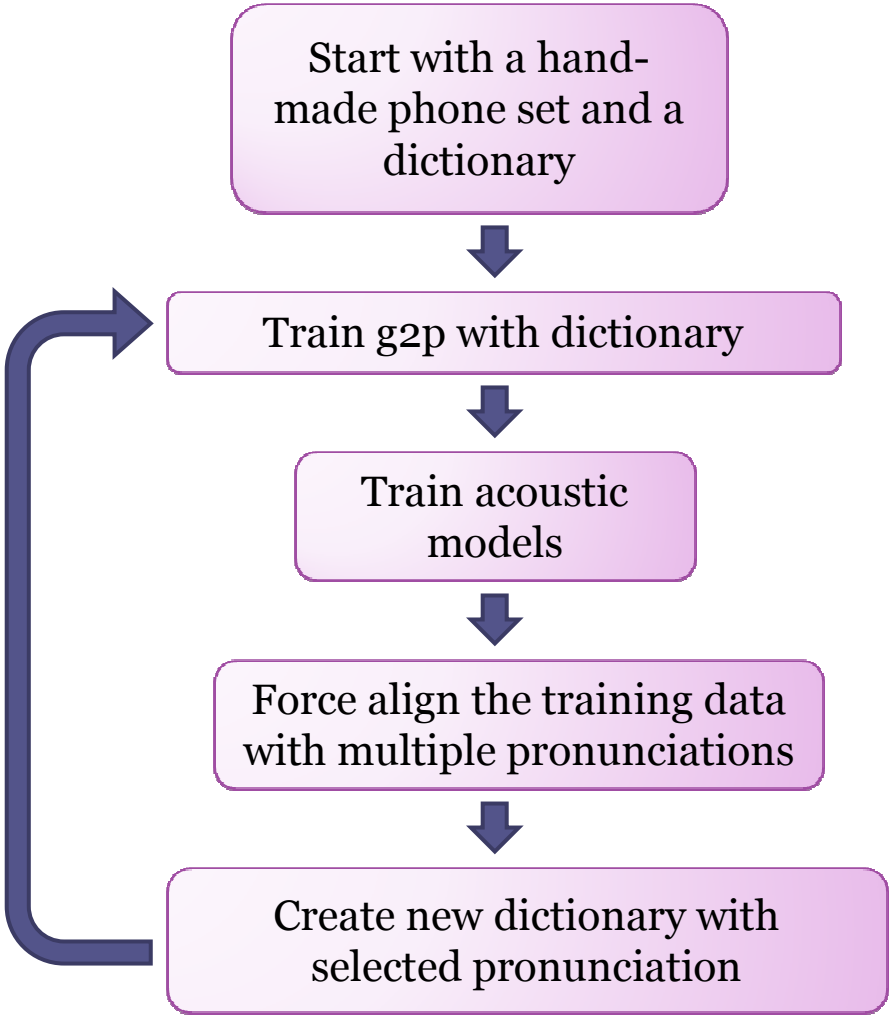
- If the audio recording is also available, that can be used to augment the estimates

$$\hat{Prn} = \arg\max_{Prn} P(X \mid Prn) P(Prn \mid W)$$

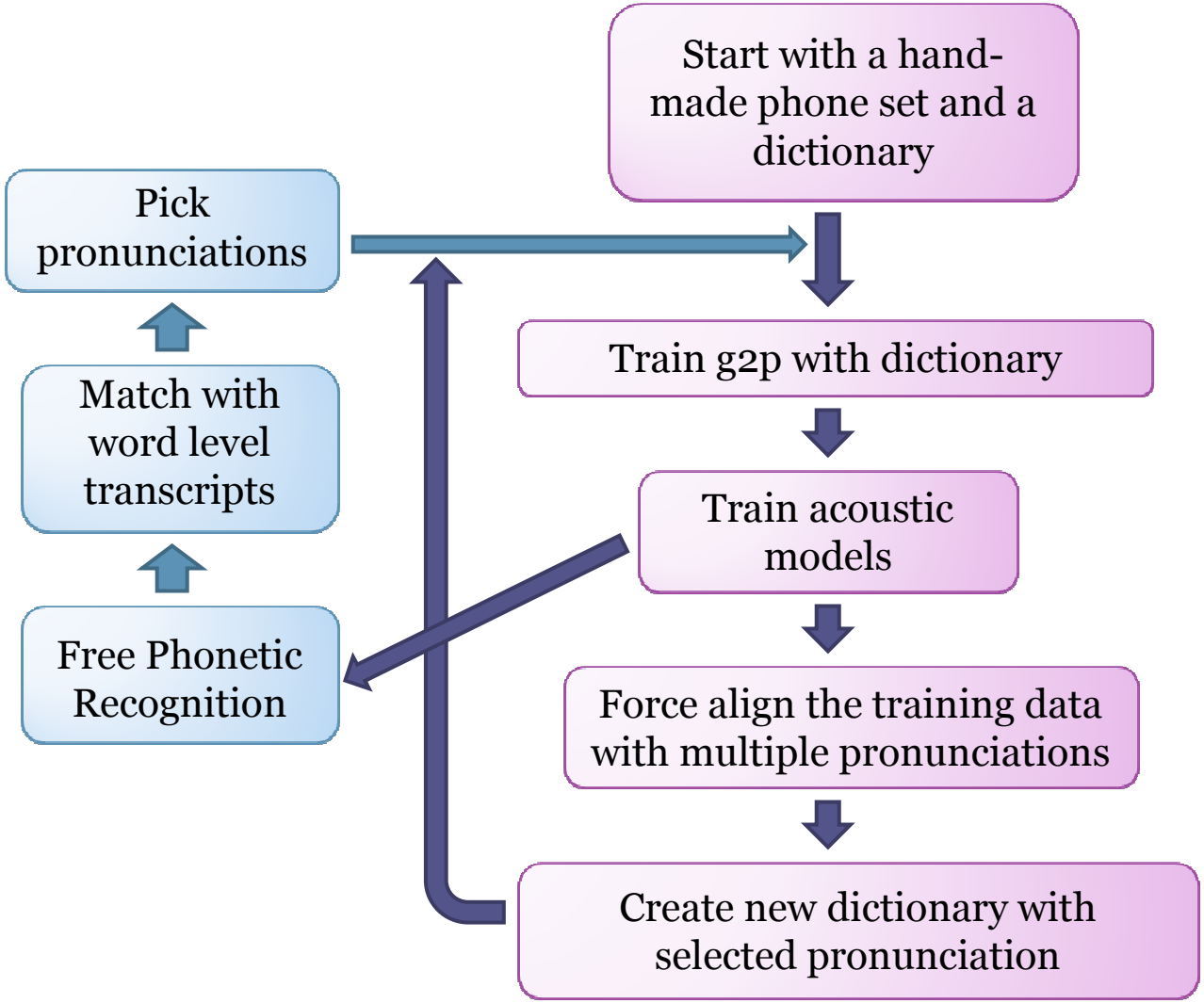
- We use an approximation to the above

$$\hat{Prn} = \arg\max_{Prn \in \{Top\ 5\ Prn\}} P(X \mid Prn)$$

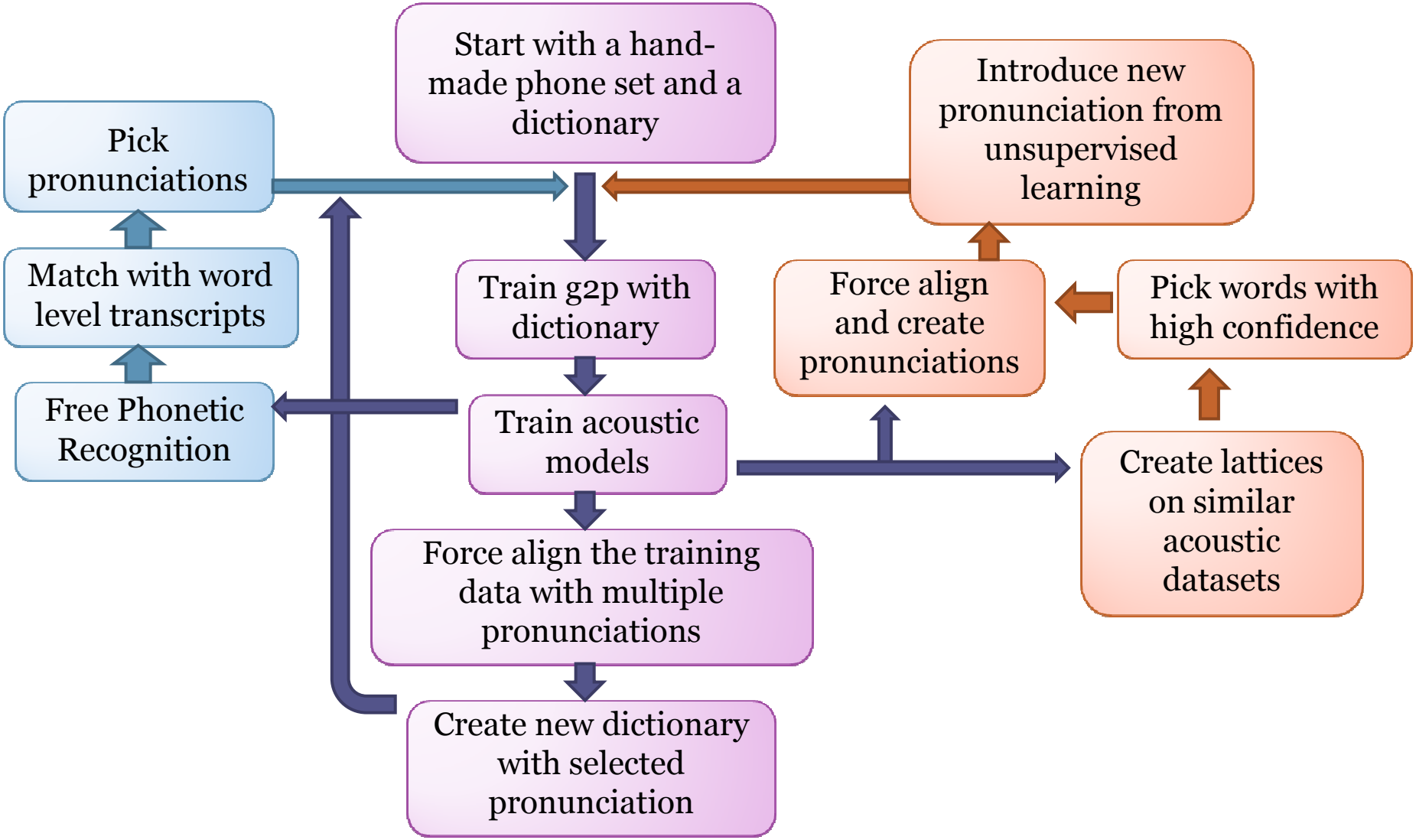
Estimating Pronunciations…



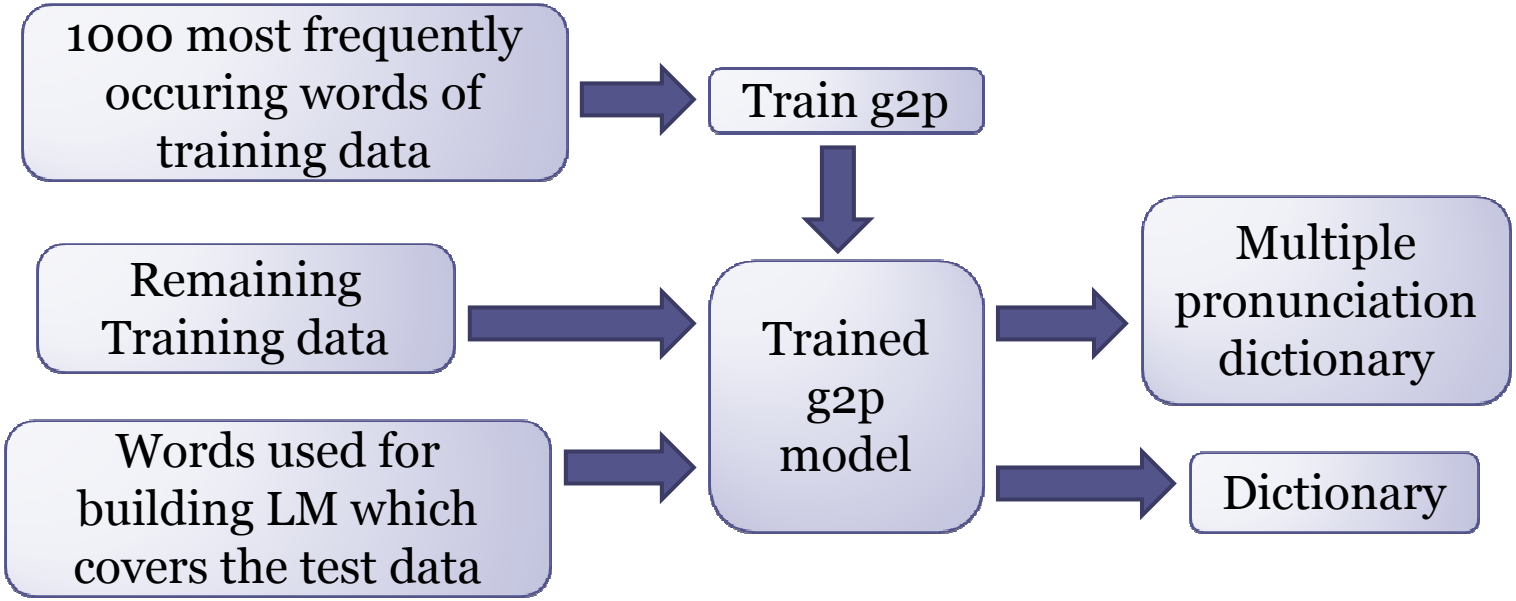
Estimating Pronunciations…



Estimating Pronunciations…

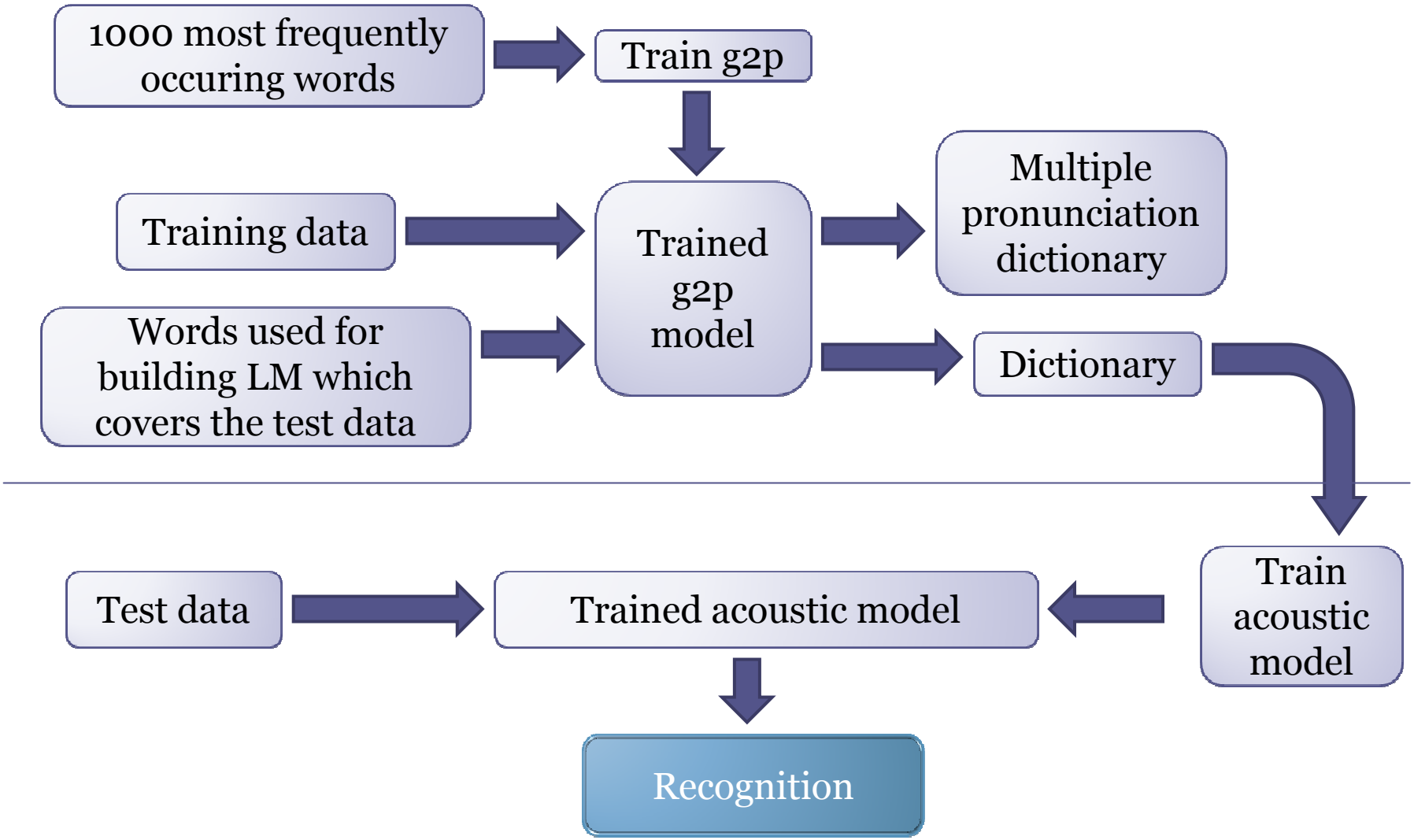


Training Procedure - Bootstrapping

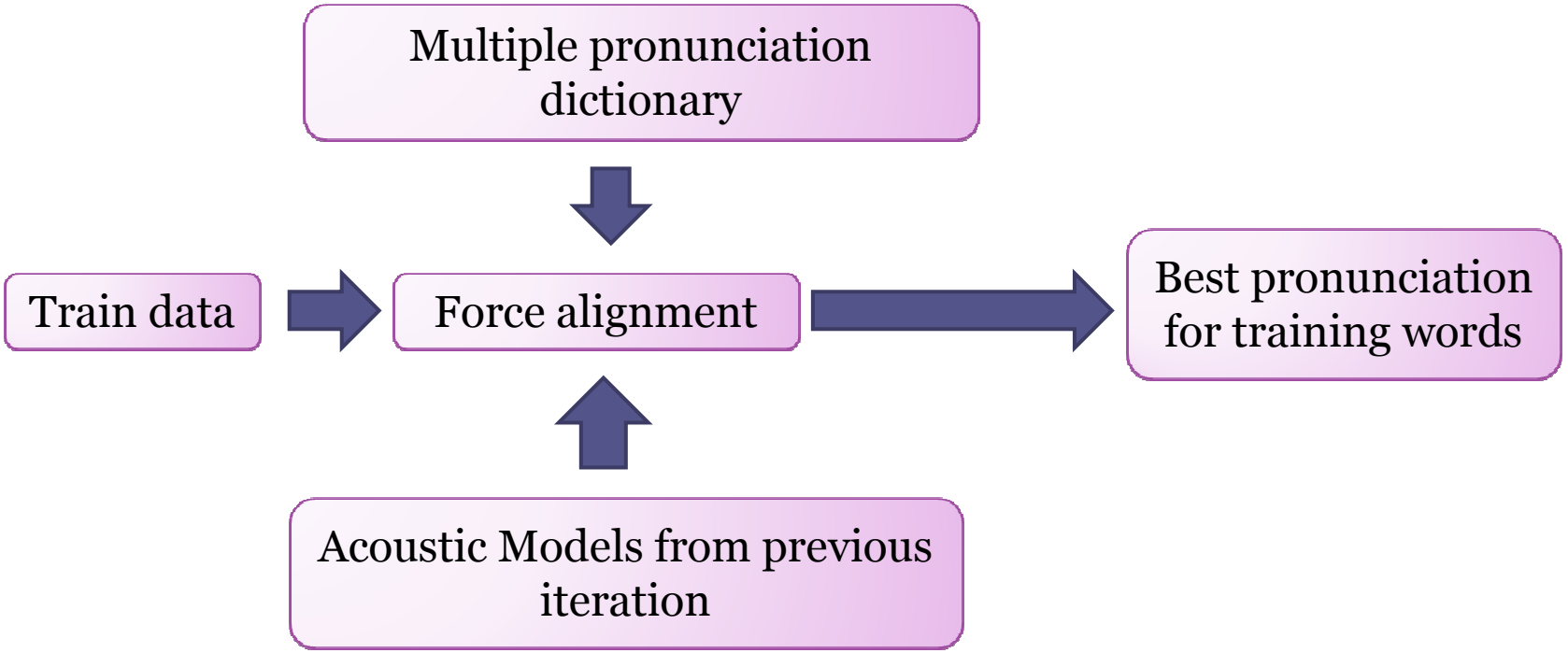


- Callhome training lexicon size – 5 K
- LM vocabulary size – 62 K
- Training acoustic data without partial words – 6 hrs
- Complete training data – 15 hrs

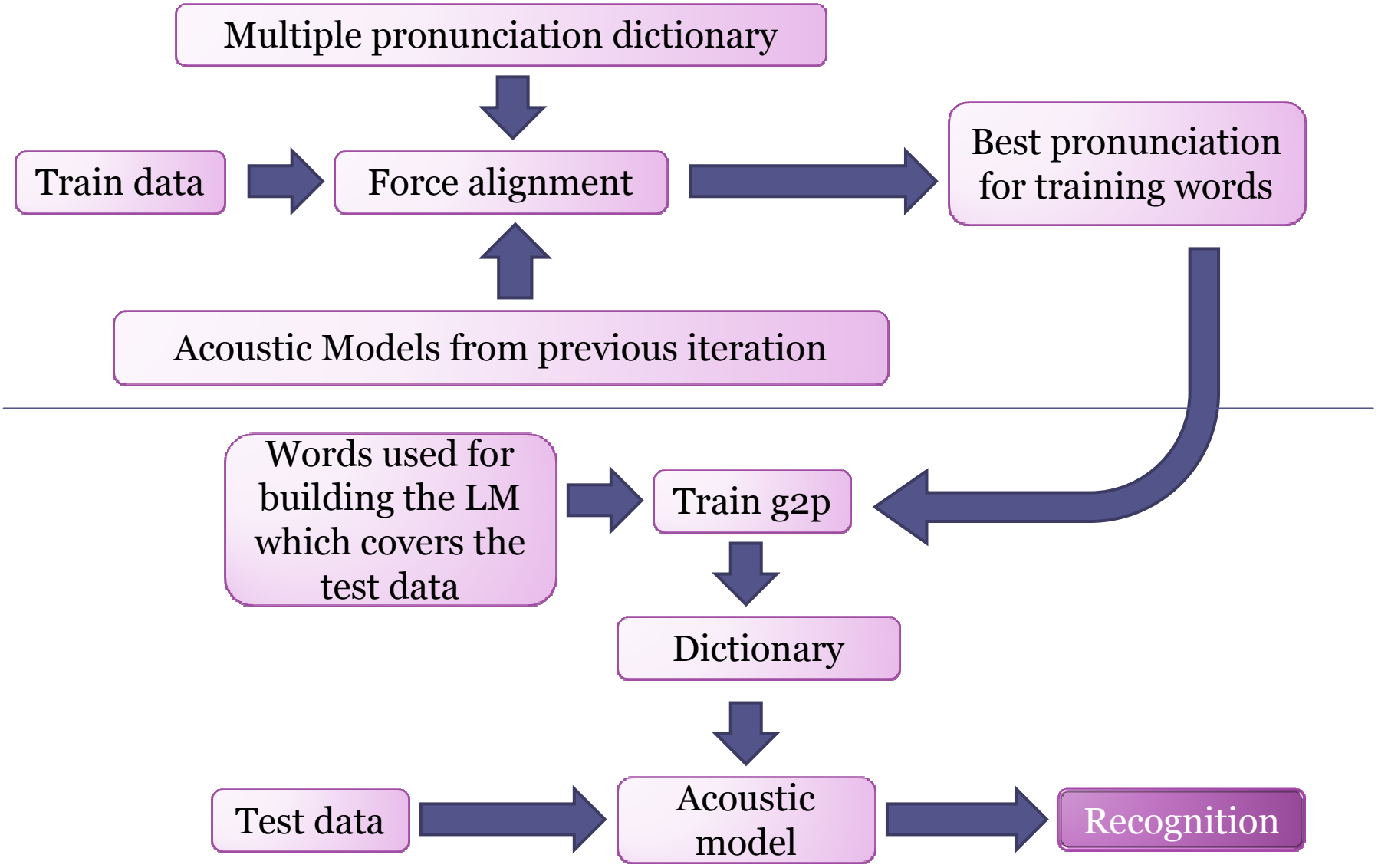
Training Procedure – Bootstrapping



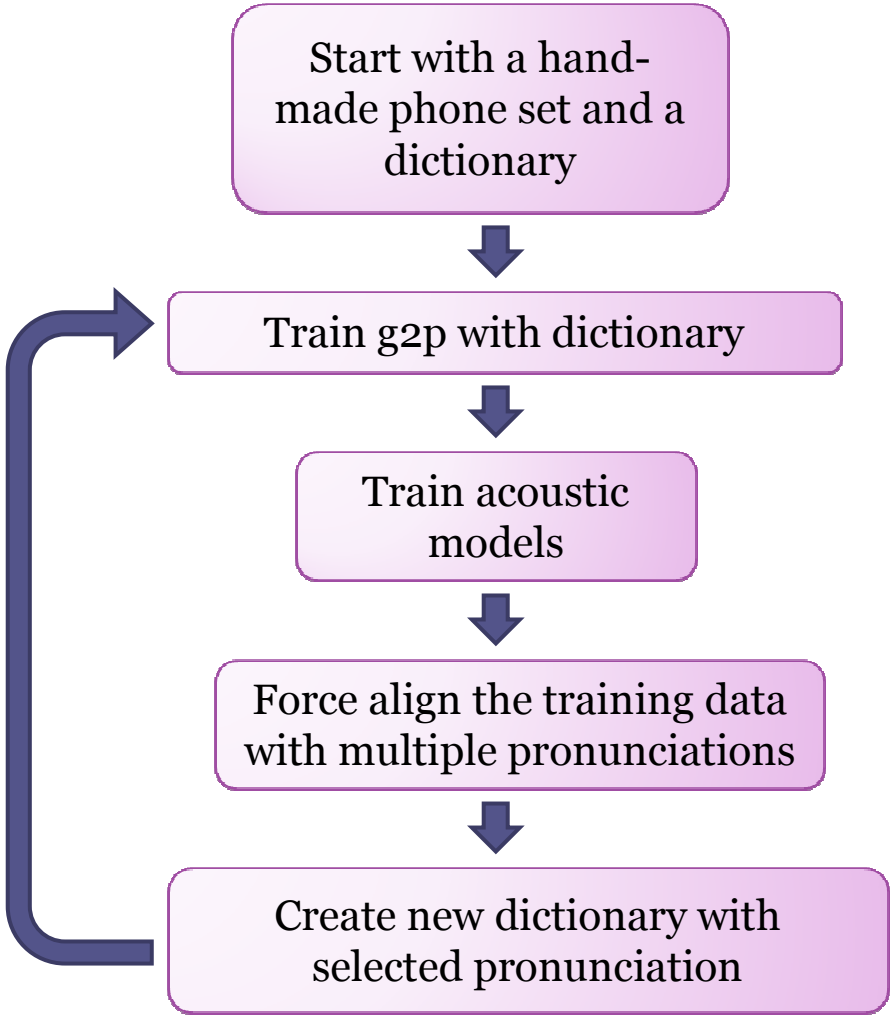
Training Procedure – Building Up



Training Procedure - Building Up



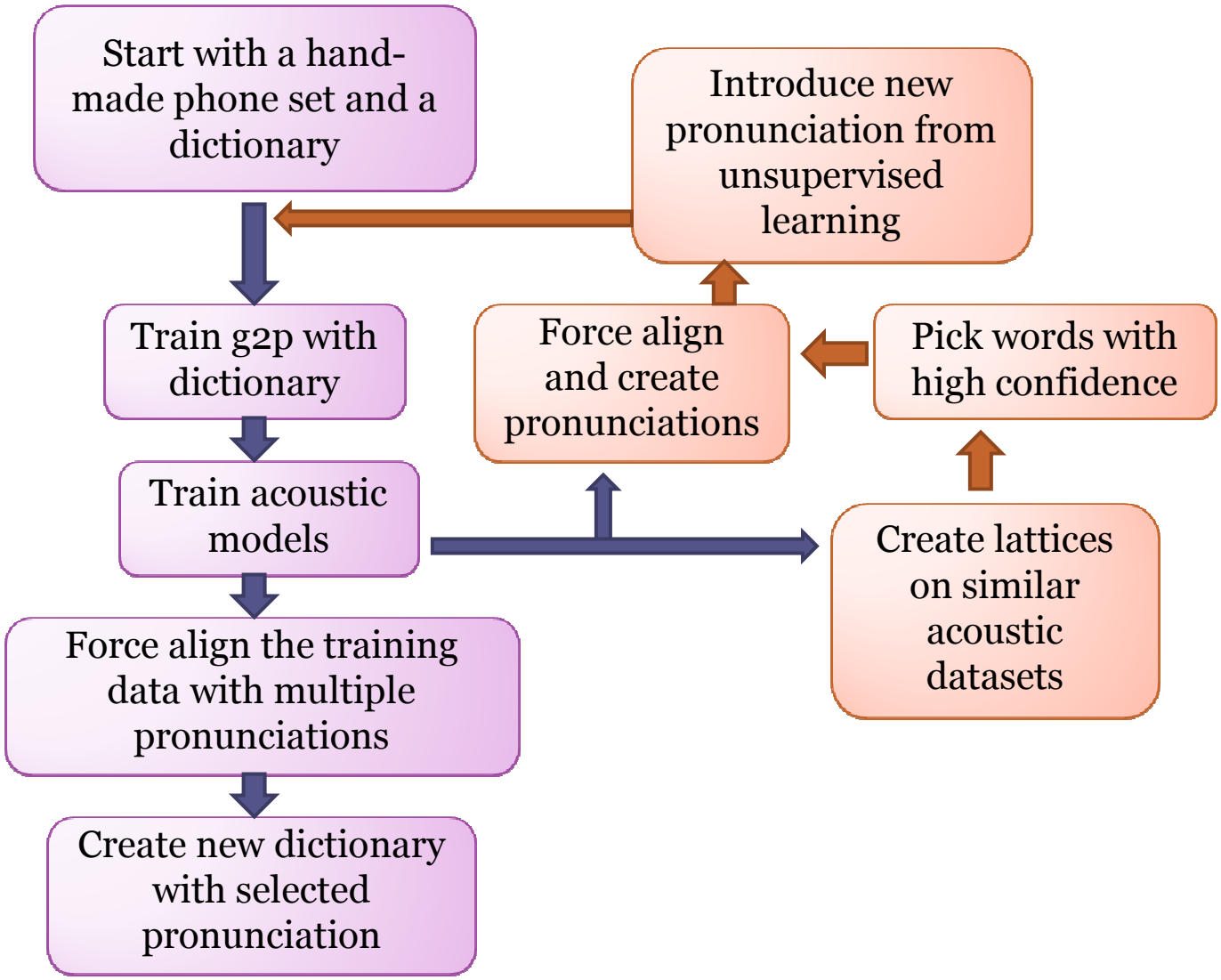
Training Procedure - Building Up



Results

Results	%
Accuracy with full dictionary available	44.35
Accuracy if 5K manual lexicon is available	40.53
Accuracy with 1000 words available	37.58
After retraining acoustic models	39.37
2nd iteration of g2p & acoustic re-train	41.60
3rd iteration of g2p & acoustic re-train	42.11
After increasing the amount of data to 15 hrs	43.56

Unsupervised Learning



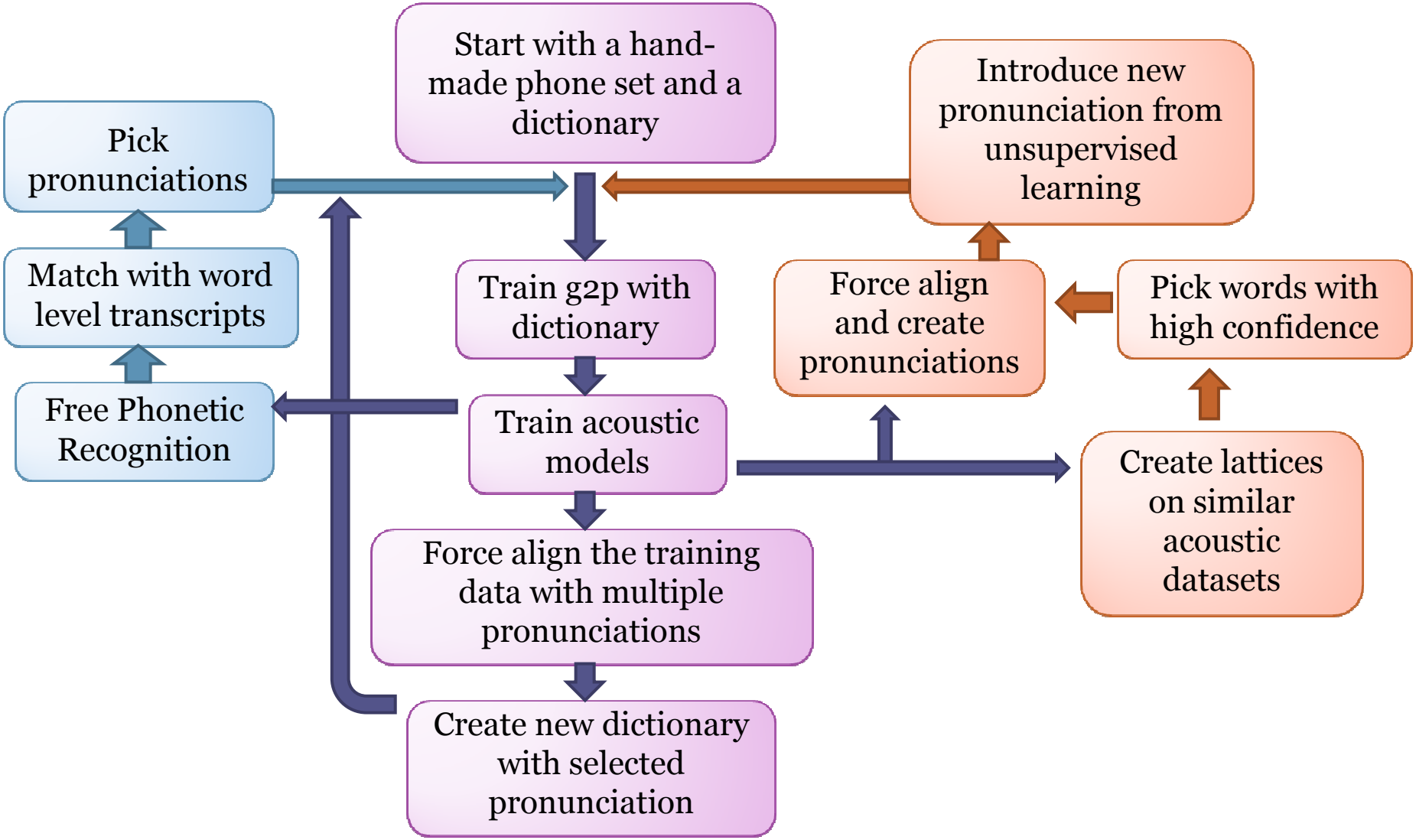
Unsupervised Lexicon Learning Results

	Baseline accuracy	After Unsupervised Learning
6 Hrs of training data	42.11	42.33
15 Hrs of training data	43.56	43.44

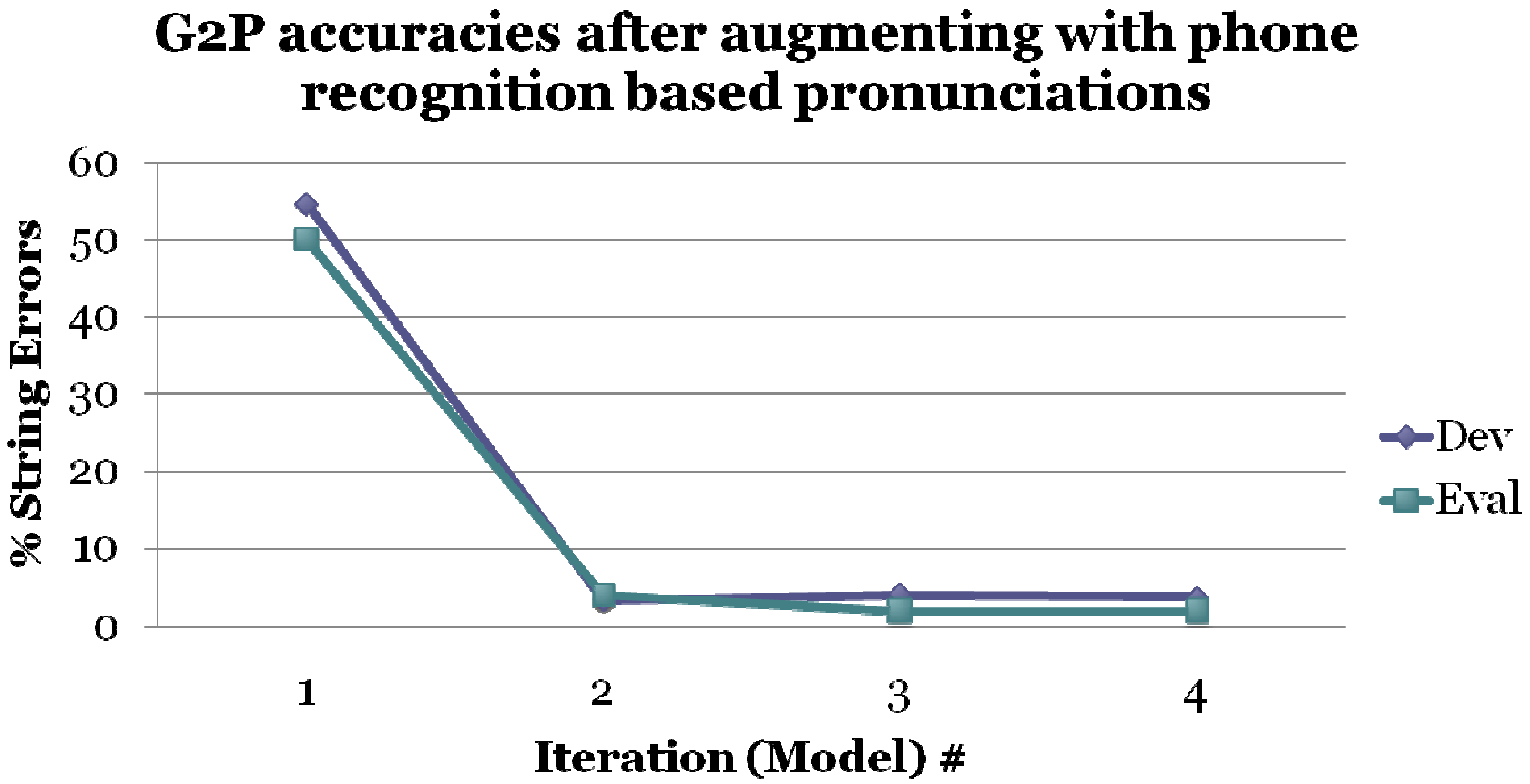
WER dilemma for Spanish Callhome

- Spanish pronunciation is very graphemic
- Accuracy for Spanish are about 31.13% (about 13% lower than callhome english)
- Phone recognition accuracy is better than callhome english
  English: 45.13% Spanish: 53.77%
- LM Perplixity is not too bad: 127
- Can learning alternate pronunciations of *reduced* words help?

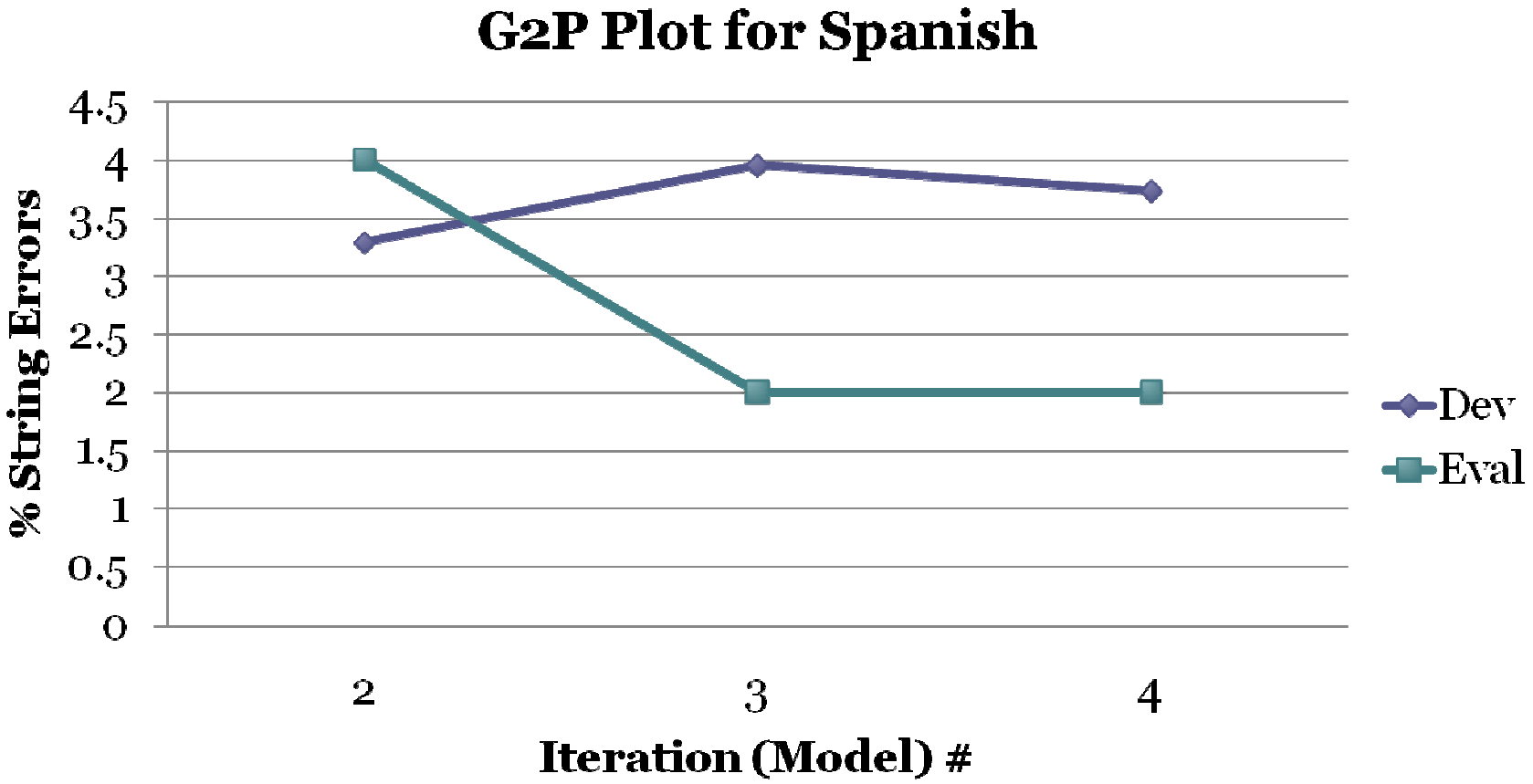
Possible lexicon training paths…



Lexicon Enhancement for Spanish



Lexicon Enhancement for Spanish



English Results and Spanish Results with unconstrained phonetic recognition approach

	Baseline	After adding pronunciations
Spanish	31.13	30.71
English	43.54	42.71

- Log likelihood of training data increases with the new lexicon.

Lexicon Enhancement

- Keep the manual Lexicon but augment with most likely pronunciation in the training data
- Affected about 250 pronunciations

- Accuracy improved from 44.33 to 45.01%

- Multiple Pronunciations had no significant impact: 45.02%

Summary

- G2p based lexicon retraining method helps in achieving accuracies close to hand made lexicons
- It can also help in improving an existing lexicon
- Unsupervised lexicon learning approach and phonetic recognition based lexicon learning approaches hold promise and need to be explored with a wider variety of smoothing and pronunciation extraction scenarios

Training Procedure

- Train g2p to generate pronunciations using your best baseline lexicon

- Generate multiple pronunciations using the g2p

- Use the training data to select the best pronunciation out of these multiple choices

- Retrain the acoustic models and iterate over the above process

Subspace GMM – the math

Dan Povey

Most basic model

$$
\begin{aligned}
p(\mathbf{x}|j) &= \sum_{i=1}^{I} w_{ji}\mathcal{N}(\mathbf{x};\boldsymbol{\mu}_{ji},\boldsymbol{\Sigma}_i) \\
\boldsymbol{\mu}_{ji} &= \mathbf{M}_i\mathbf{v}_j \\
w_{ji} &= \frac{\exp(\mathbf{w}_i^T\mathbf{v}_j)}{\sum_{i'=1}^{I}\exp(\mathbf{w}_{i'}{}^T\mathbf{v}_j)}
\end{aligned}
$$

j is speech class (phone in context)

$$p(\mathbf{x}|j) = \sum_{i=1}^{I} w_{ji}\mathcal{N}(\mathbf{x};\boldsymbol{\mu}_{ji},\boldsymbol{\Sigma}_i)$$

$$\boldsymbol{\mu}_{ji} = \mathbf{M}_i\mathbf{v}_j$$

$$w_{ji} = \frac{\exp(\mathbf{w}_i^T\mathbf{v}_j)}{\sum_{i'=1}^{I}\exp(\mathbf{w}_{i'}{}^T\mathbf{v}_j)}$$

Gaussian Mixture Model

$$p(\mathbf{x}|j) = \sum_{i=1}^{I} w_{ji}\mathcal{N}(\mathbf{x};\boldsymbol{\mu}_{ji},\boldsymbol{\Sigma}_i)$$

$$\boldsymbol{\mu}_{ji} = \mathbf{M}_i\mathbf{v}_j$$

$$w_{ji} = \frac{\exp(\mathbf{w}_i^T\mathbf{v}_j)}{\sum_{i'=1}^{I}\exp(\mathbf{w}_{i'}{}^T\mathbf{v}_j)}$$

Same number of Gaussians in each state

$$p(\mathbf{x}|j) = \sum_{i=1}^{I} w_{ji}\mathcal{N}(\mathbf{x};\boldsymbol{\mu}_{ji},\boldsymbol{\Sigma}_i)$$

$$\boldsymbol{\mu}_{ji} = \mathbf{M}_i\mathbf{v}_j$$

$$w_{ji} = \frac{\exp(\mathbf{w}_i^T\mathbf{v}_j)}{\sum_{i'=1}^{I}\exp(\mathbf{w}_{i'}{}^T\mathbf{v}_j)}$$

Full covariances shared between states (but different for each Gaussian)

$$p(\mathbf{x}|j) = \sum_{i=1}^{I} w_{ji}\mathcal{N}(\mathbf{x};\boldsymbol{\mu}_{ji},\boldsymbol{\Sigma}_i)$$

$$\boldsymbol{\mu}_{ji} = \mathbf{M}_i\mathbf{v}_j$$

$$w_{ji} = \frac{\exp(\mathbf{w}_i^T\mathbf{v}_j)}{\sum_{i'=1}^{I}\exp(\mathbf{w}_{i'}{}^T\mathbf{v}_j)}$$

Means and weights controlled by other parameters

$$p(\mathbf{x}|j) = \sum_{i=1}^{I} w_{ji}\mathcal{N}(\mathbf{x};\boldsymbol{\mu}_{ji},\boldsymbol{\Sigma}_i)$$

$$\boldsymbol{\mu}_{ji} = \mathbf{M}_i\mathbf{v}_j$$

$$w_{ji} = \frac{\exp(\mathbf{w}_i^T\mathbf{v}_j)}{\sum_{i'=1}^{I}\exp(\mathbf{w}_{i'}{}^T\mathbf{v}_j)}$$

State-specific vectors $\mathbf{v}$ determine means and weights

$$p(\mathbf{x}|j) = \sum_{i=1}^{I} w_{ji}\mathcal{N}(\mathbf{x};\boldsymbol{\mu}_{ji},\boldsymbol{\Sigma}_i)$$

$$\boldsymbol{\mu}_{ji} = \mathbf{M}_i\mathbf{v}_j$$

$$w_{ji} = \frac{\exp(\mathbf{w}_i^T\mathbf{v}_j)}{\sum_{i'=1}^{I}\exp(\mathbf{w}_{i'}{}^T\mathbf{v}_j)}$$

Globally shared parameters $\mathbf{M}_i$ and $\mathbf{w}_i$ determine the mapping

$$p(\mathbf{x}|j) = \sum_{i=1}^{I} w_{ji} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_{ji}, \boldsymbol{\Sigma}_i)$$

$$\boldsymbol{\mu}_{ji} = \mathbf{M}_i \mathbf{v}_j$$

$$w_{ji} = \frac{\exp(\mathbf{w}_i^T \mathbf{v}_j)}{\sum_{i'=1}^{I} \exp(\mathbf{w}_{i'}^T \mathbf{v}_j)}$$

Mapping of means is linear

$$p(\mathbf{x}|j) = \sum_{i=1}^{I} w_{ji}\mathcal{N}(\mathbf{x};\boldsymbol{\mu}_{ji},\boldsymbol{\Sigma}_i)$$

$$\boldsymbol{\mu}_{ji} = \mathbf{M}_i\mathbf{v}_j$$

$$w_{ji} = \frac{\exp(\mathbf{w}_i^T\mathbf{v}_j)}{\sum_{i'=1}^{I}\exp(\mathbf{w}_{i'}{}^T\mathbf{v}_j)}$$

Mapping of weights is log-linear (with renormalization)

$$p(\mathbf{x}|j) = \sum_{i=1}^{I} w_{ji}\mathcal{N}(\mathbf{x};\boldsymbol{\mu}_{ji},\boldsymbol{\Sigma}_i)$$

$$\boldsymbol{\mu}_{ji} = \mathbf{M}_i\mathbf{v}_j$$

$$w_{ji} = \frac{\exp(\mathbf{w}_i^T\mathbf{v}_j)}{\sum_{i'=1}^{I}\exp(\mathbf{w}_{i'}{}^T\mathbf{v}_j)}$$

There is a correspondence between Gaussians across states

$$p(\mathbf{x}|j) = \sum_{i=1}^{I} w_{ji}\mathcal{N}(\mathbf{x};\boldsymbol{\mu}_{ji},\boldsymbol{\Sigma}_i)$$

$$\boldsymbol{\mu}_{ji} = \mathbf{M}_i\mathbf{v}_j$$

$$w_{ji} = \frac{\exp(\mathbf{w}_i^T\mathbf{v}_j)}{\sum_{i'=1}^{I}\exp(\mathbf{w}_{i'}^T\mathbf{v}_j)}$$

Universal Background Model (UBM)

- The UBM is a single Gaussian Mixture Model that has been trained on all speech classes.
- It is used to initialize the SGMM training.
- It is used to prune the Gaussian indexes i during training and decoding.

Likelihood evaluation

- Using the UBM, prune to a subset of the indices i, e.g. the top 10 of them.
- With appropriate pre-computation per frame, each Gaussian i in each state j can have its likelihood evaluated with a single dot product in the dimension of $\mathbf{v}_j$ (typically 40 or 50).
- This can be even faster than a typical mixture-of-Gaussians system.

Prerequisites for model building

- A previously trained system (conventional or SGMM based), needed for initial state alignment.
- A phonetic context tree (use normal methods to get this).
- A trained UBM.

Model initialization

- Typical initialization:
- Set dimension of vectors $\mathbf{v}_j$ to feature-dim + 1 (e.g. 39+1 = 40).
- Set $\mathbf{v}_j$ to a unit vector [1 0 0 0 …]
- Set $\mathbf{M}_i$ to [$\boldsymbol{\mu}_i$ $\mathbf{I}$], where $\boldsymbol{\mu}_i$ is i'th mean in UBM
- Set $\mathbf{w}_i$ to zero vector (so all weights are equal)
- Effect is that the initial GMM in each state j is the same as the UBM (with equal weights).
- This is not the only way to initialize.

Model training

- Training is based on Expectation-Maximization, the same as traditional GMM training
- Each iteration, we pick a parameter type ($\mathbf{M}_i$ or $\mathbf{w}_i$ or $\mathbf{v}_j$), accumulate statistics, update it.
- Each update step is guaranteed to increase likelihood on the training data.
- Can actually combine the updates on a single iteration, within certain constraints.

Model training: $\mathbf{v}_j$

- Auxiliary function is quadratic in $\mathbf{v}_j$:

$$\mathcal{Q}(\mathbf{v}_j) = \mathbf{g}_j \cdot \mathbf{v}_j - \frac{1}{2}\mathbf{v}_j^T \mathbf{H}_j \mathbf{v}_j$$

Linear term

$$\mathcal{Q}(\mathbf{v}_j) = \mathbf{g}_j \cdot \mathbf{v}_j - \frac{1}{2}\mathbf{v}_j^T \mathbf{H}_j \mathbf{v}_j$$

Quadratic term

$$\mathcal{Q}(\mathbf{v}_j) \quad = \quad \mathbf{g}_j \cdot \mathbf{v}_j - \frac{1}{2}\mathbf{v}_j^T \mathbf{H}_j \mathbf{v}_j$$

Obtaining the auxiliary function

- $\mathbf{g}_j$ and $\mathbf{H}_j$ computed from statistics accumulated from the data on each iteration of training (don't accumulate directly: efficiency).
- The part of the auxiliary function that arises from the effect on the means is naturally quadratic
- The part that arises from the effect on the weights is not (we use a quadratic approximation).

Update equation for $\mathbf{v}_j$

$$\mathbf{v}_j = \mathbf{H}_j^{-1}\mathbf{g}_j$$

- Mathematically speaking, $\mathbf{H}_j$ always invertible
- Practically speaking, not always invertible (can have very tiny eigenvalues)
- Will discuss this problem later

Model training: M

- Auxiliary function is also quadratic:

$$\mathcal{Q}(\mathbf{M}_i) = \mathrm{tr}\left(\mathbf{M}_i^T \boldsymbol{\Sigma}_i^{-1} \mathbf{Y}_i\right) - \frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Sigma}_i^{-1} \mathbf{M}_i \mathbf{Q}_i \mathbf{M}_i^T\right)$$

- $\mathbf{Y}_i$ obtained directly from the data; $\mathbf{Q}_i$ derived from outer products of $\mathbf{v}_j$ weighted by data counts.
- Update is: $\mathbf{M}_i = \mathbf{Y}_i \mathbf{Q}_i^{-1}$
- Again we have a problem when $\mathbf{Q}_i$ is not invertible (but it will normally be invertible).

Model training: w

- Training the "weight projection vectors" $\mathbf{w}_i$ is a little less easy: there is no natural auxiliary function
- It is possible to obtain an approximate quadratic auxiliary function that leads to an update that converges quite fast.
- The sufficient statistics for updating $\mathbf{w}_i$ are the data counts for each state index j and Gaussian index i

Non-invertible matrices

- Sometimes the matrices that represent the quadratic terms in the auxiliary functions are singular.
- This situation corresponds to “don’t-care” directions in the parameter space (the linear term will also be zero in these directions).
- If we attempt to invert singular matrices we will tend to get unpredictable results.
- This is more of a problem with small datasets.

Solutions to non-invertibility

- Can introduce priors over the parameters.
  - Can make prior based on an ad-hoc formula with a τ value (like HTK-based MAP estimation)
  - OR estimate them from (estimated) parameters
- Can use a solution based on a "least squares" approach: get parameter with smallest squared value that gives maximum of auxiliary function
  - This is not possible to do exactly given the form of the statistics we accumulate (because impossible to distinguish very small from zero eigenvalues) but can approximate it in an acceptable way.
- Can just refuse to update those parameters.
- Not clear what the best solution is.

“Extra” stuff

- On top of the basic model, we can do various things. Will summarize the more important of these in the next slides:
  - Sub-states
  - Speaker subspace
  - Constrained MLLR

Sub-states

- Introduce within each state, a weighted mixture of what we previously had:

$$
\begin{aligned}
p(\mathbf{x}|j) &= \sum_{m=1}^{M_j} c_{jm} \sum_{i=1}^{I} w_{jmi} \mathcal{N}(\mathbf{x}; \mu_{jmi}, \mathbf{\Sigma}_i) \\
\mu_{jmi} &= \mathbf{M}_i \mathbf{v}_{jm} \\
w_{jmi} &= \frac{\exp \mathbf{w}_i^T \mathbf{v}_{jm}}{\sum_{i'=1}^{I} \exp \mathbf{w}_{i'}^T \mathbf{v}_{jm}}.
\end{aligned}
$$

- New parameter type: mixture weights c_{jm}
- Sub-states help more than increasing dim of $\mathbf{v}$

Speaker Subspace

- Use a similar approach to cover speaker variation:

$$
\begin{aligned}
p(\mathbf{x}|j,s) &= \sum_{m=1}^{M_j} c_{jm} \sum_{i=1}^{I} w_{jmi} \mathcal{N}(\mathbf{x}; \mu_{jmi}^{(s)}, \mathbf{\Sigma}_i) \\
\mu_{jmi}^{(s)} &= \mathbf{M}_i \mathbf{v}_{jm} + \mathbf{N}_i \mathbf{v}^{(s)} \\
w_{jmi} &= \frac{\exp \mathbf{w}_i^T \mathbf{v}_{jm}}{\sum_{i'=1}^{I} \exp \mathbf{w}_{i'}^T \mathbf{v}_{jm}},
\end{aligned}
$$

- Do not make the mixture weights depend on speaker (makes decoding too slow)
- New parameters $\mathbf{N}_i$ (globally shared), $\mathbf{v}^{(s)}$ (speaker specific)

Constrained MLLR

- A linear feature transformation

$$\mathbf{x} \rightarrow \mathbf{A}^{(s)}\mathbf{x} + \mathbf{b}^{(s)}$$

- Estimated for each speaker s
- During decoding this is independent of the model but the estimation formulas need to be specially formulated (full covariance)

Typical setup

- About 400-1000 Gaussians in the UBM
- Phonetic subspace and speaker subspace both have dimension 40-60
- Mix up to about half the number of sub-states that the baseline system had Gaussians
- Speaker-specific adaptation parameters $\mathbf{v}^{(s)}$, $\mathbf{A}^{(s)}$, $\mathbf{b}^{(s)}$ all to be estimated on speech only (not silence)
- Language model weight smaller than normal system, e.g. 8-10 vs. 13-14 with normal system
- Obtain UBM by clustering (diagonal) Gaussians from a normal system, doing full-covariance re-estimation

Issues with adaptation

- Speaker-vector adaptation not data-hungry
- Better done per segment rather than given reasonable segment lengths (not just 1 or 2 seconds)
- We developed the parameter-subspace CMLLR to enable both on per-segment basis
- Hard to get adaptation working in our setup:
  - Segments were very short (most utterances 1 or 2 seconds)
  - Adapting on silence frames bad! (needed to exclude them)
  - Not clear whether the corpus, model, or feature extraction
  - Full covariances for silence were getting floored eigenvalues (floored to 1/1000 of the largest). Strange silence features?

Software design

- This technique is more complicated than a normal mixture of Gaussians system – needs better testing.
- We separately unit-tested all easily testable code (e.g. linear algebra code)
- Printed out copious diagnostics; measured all auxiliary function changes and compared with likelihoods
- Used as much as possible blocks that can be swapped in and out with other blocks, for easier testing
- E.g. we wrote two separate versions of the update code and used each to help debug the other.
- Debugged the calling code by writing a simple GMM based acoustic model with the same C++ interface as the SGMM.

Summary of accomplishments

Dan Povey

Initial objectives (SGMM)

- I had done experiments with this style of model and had got very large improvements
  - 25% relative with ML training on 50h data
  - ~5-10% relative with ML training on 1000
  - Discriminative training -> smaller gains.
- Wanted to
  - Popularize the technique
  - Show even larger improvements with very small amounts of data (if trend held)
  - Use out-of-language data to train shared parameters

Initial goals (Lexicon)

- Nagendra had had experience with the difficultly and expense of obtaining a pronunciation dictionary (Lexicon) for new language
- Wanted to develop tools and techniques to make this easier or cheaper.
- Projects were merged under the common theme of "limited data".
- There was an open-source component: the plan to release the software developed in the project under an open source license.

SGMM accomplishments

- Demonstrated that the SGMM approach works: got better results than the baseline system.
- Developed new techniques in the SGMM framework:
  - Constrained MLLR estimation with parameter subspace
- Have worked on the basic SGMM approach:
  - E.g. looked at issues relating to poorly conditioned matrices that arise in estimation
  - Filled out the derivations

Multilingual applications

- Have demonstrated that it is possible to improve results for a group of languages by sharing non-state-specific parameters
- Better results than systems trained on the individual languages.
- Have demonstrated that with very small amounts of data (1h), we can dramatically improve results by using other languages to train shared parameters.

Open-source framework

- We have developed software that we can release.
- Software does all the required modeling, and does it quite efficiently.
- Need to do some cleanup, documentation and packaging before it would be useful to others.

Parts of the framework (1/2)

- HTK scripts to build baseline system (Martin, Samuel, Petr, Lukas…)
- Scripts to build language models based on SRILM tools (Samuel, Nagendra, Martin…)
- Scripts to improve lexicon coverage (Nagendra, Samuel, Pinar…)
- OpenFST based scripts (Lukas, Samuel) and C++ programs (Ariya) to generate WFST's for transcriptions and language models.
- C++ based code framework to read in HTK-like models and WFSTs and decode and train (Lukas, Ondrej, Petr,…)
- Matlab (Lukas) and C++ (Ariya, Tony) and HTK-based scripts (Mohit) to obtain UBM from baseline HMM set.

Parts of the framework (2/2)

- Matrix/vector template code wrapping various combinations of BLAS/ATLAS/CLAPACK (Lukas, Ondrej, Dan)
- Unit-testing code (Dan, Lukas, Ondrej,)
- C++ class for subspace GMM training and likelihood evaluation (Lukas, Dan, Arnab)
- Command line tools to use the above (Petr, Lukas, Arnab, Dan)
- Subspace GMM training and testing scripts (Petr, Lukas, Arnab, Dan)
- An entirely different HTK-based framework to implement this model (Rick, Shou-Chun)

Future of SGMM approach

- I believe the SGMM approach has a bright future
- It is a special case of a GMM which means basically all standard techniques apply.
- But extra techniques are possible with SGMM.
- It is more compact and gives better results
- But - it is more mathematically difficult which may limit the rate of uptake.
- In speaker identification, people were forced to use Factor Analysis style systems because they work better, even though they have the same issue.

Lexicon learning work

(summary of Nagendra's summary)

- Have demonstrated that it is possible to do reasonably well starting from a 1000 word pronunciation dictionary and using g2p to get the remaining words.
- Should be useful in languages where there are relatively few resources
- We plan to include the scripts and tools developed as part of the software setup we release

Further reading and resources

- For a (long) tutorial introduction to SGMMs: http://dpovey.googlepages.com/ubmtutorial.pdf
- For pre-release versions of our software, contact dpovey@microsoft.com (because I always respond promptly to email; but the software is based in Brno).
- We will most likely make two or three journal and/or conference papers out of this work.
- Many of us plan to continue research using the systems we developed.