Front-end Feature Transforms with Context Filtering for Speaker Adaptation

Jing Huang, Karthik Visweswariah
Peder Olsen, Vaibhava Goel

May 24, 2011

Outline

Front-end Speaker Adaptation

Front-end Transforms:

- **Linear transforms**
  - Feature-space MLLR (i.e. CMLLR) [Gales'98])
  - Discriminative linear transform [Wang'03]
- **Non-linear transforms** ([Olsen'03, Visweswariah'04, Saon'04])

Front-end Speaker Adaptation

Front-end Transforms:

- **Linear transforms**
  - Feature-space MLLR (i.e. CMLLR) [Gales'98])
  - Discriminative linear transform [Wang'03]
- **Non-linear transforms** ([Olsen'03, Visweswariah'04, Saon'04])

FMLLR variants:

- Q-FMLLR ([Varadarajan'08])
- Full-covariance FMLLR ([Povey06, Ghoshal'08])

Original FMLLR

Definition (Feature-space Maximum Likelihood Linear Regression)

Given adaptation data $x_t, t = 1, ..., T$ of a speaker, find an affine transform to maximize the likelihood of adaptation data given the current model.

$$y_t = Ax_t + b = W\xi_t$$

$\xi_t = \begin{bmatrix} x_t \\ 1 \end{bmatrix}$: input feature extended with 1

W: extended transformation matrix $[A \quad b]$ with square matrix A of size $d \times d$ (d is the size of x_t) and bias term b

Original FMLLR (cont.)

The objective function: log likelihood of the transformed data given the current model, plus the Jacobian compensation term

$$Q(W) = T \log \det(A) - \frac{1}{2} \sum_{j=1}^{G} \sum_{t=1}^{T} \gamma_t(j)(W\xi_t - \mu_j)^T \Sigma_j^{-1} (W\xi_t - \mu_j)$$

- j: index of Gaussian components
- μ_j, Σ_j: mean and diagonal covariance matrix
- $\gamma_t(j)$: Gaussian occupation probabilities.

Extend A to non-square matrix

Instead of just using one frame x_t, we concatenate it with its neighboring frames to make a context vector $\hat{x}_t$

Extend A to non-square matrix

Instead of just using one frame x_t, we concatenate it with its neighboring frames to make a context vector $\hat{x}_t$

Example (context size = 1)

$\hat{x}_t = [x_{t-1} \;\; x_t \;\; x_{t+1}]$. Find an affine transform to $\hat{x}_t$ to maximize the likelihood

$$y_t = A\hat{x}_t + b = W\hat{\xi}_t$$

Now the size of A is $d \times 3d$

How to estimate non-square matrix A?

Extend A to non-square matrix

Instead of just using one frame x_t, we concatenate it with its neighboring frames to make a context vector $\hat{x}_t$

Example (context size = 1)

$\hat{x}_t = [x_{t-1} \quad x_t \quad x_{t+1}]$. Find an affine transform to $\hat{x}_t$ to maximize the likelihood

$$y_t = A\hat{x}_t + b = W\hat{\xi}_t$$

Now the size of A is $d \times 3d$

How to estimate non-square matrix A?

Note: there is no direct way to compute the derivative of objective function $Q(W)$ with respect to $\hat{x}_t$.

Compensation term when A is square

Find the Jacobian compensation term to make $L_y + \mathcal{C} = L_x$:
simply assume $y = Ax$, A is square and invertible
$\implies$ compensation term $\mathcal{C} = \frac{1}{2} \log \frac{\det(\Sigma_y)}{\det(\Sigma_x)}$

Compensation term when A is square

Find the Jacobian compensation term to make $L_y + \mathcal{C} = L_x$:
simply assume $y = Ax$, A is square and invertible
$\Longrightarrow$ compensation term $\mathcal{C} = \frac{1}{2} \log \frac{\det(\Sigma_y)}{\det(\Sigma_x)}$

Proof.

- $$L_x = -\frac{1}{2}(x-\mu_x)^T \Sigma_x^{-1}(x-\mu_x) - \frac{1}{2}\log\det(\Sigma_x)$$
- $$L_y = -\frac{1}{2}(y-\mu_y)^T \Sigma_y^{-1}(y-\mu_y) - \frac{1}{2}\log\det(\Sigma_y)$$

Compensation term when A is square

Find the Jacobian compensation term to make $L_y + \mathcal{C} = L_x$:
simply assume $y = Ax$, A is square and invertible
$\Longrightarrow$ compensation term $\mathcal{C} = \frac{1}{2} \log \frac{\det(\Sigma_y)}{\det(\Sigma_x)}$

Proof.

- $$L_x = -\frac{1}{2}(x - \mu_x)^T \Sigma_x^{-1}(x - \mu_x) - \frac{1}{2}\log\det(\Sigma_x)$$
- $$L_y = -\frac{1}{2}(y - \mu_y)^T \Sigma_y^{-1}(y - \mu_y) - \frac{1}{2}\log\det(\Sigma_y)$$
- $$\begin{aligned}(y - \mu_y)^T \Sigma_y^{-1}(y - \mu_y) &= (x - \mu_x)^T A^T (A\Sigma_x A^T)^{-1} A(x - \mu_x) \\ &= (x - \mu_x)^T \Sigma_x^{-1}(x - \mu_x)\end{aligned}$$

Compensation term when A is square

Find the Jacobian compensation term to make $L_y + \mathcal{C} = L_x$:
simply assume $y = Ax$, A is square and invertible
$\Longrightarrow$ compensation term $\mathcal{C} = \frac{1}{2} \log \frac{\det(\Sigma_y)}{\det(\Sigma_x)}$

Proof.

- $$L_x = -\frac{1}{2}(x-\mu_x)^T \Sigma_x^{-1}(x-\mu_x) - \frac{1}{2}\log\det(\Sigma_x)$$
- $$L_y = -\frac{1}{2}(y-\mu_y)^T \Sigma_y^{-1}(y-\mu_y) - \frac{1}{2}\log\det(\Sigma_y)$$
- $$\begin{aligned}(y-\mu_y)^T \Sigma_y^{-1}(y-\mu_y) &= (x-\mu_x)^T A^T (A\Sigma_x A^T)^{-1} A(x-\mu_x) \\ &= (x-\mu_x)^T \Sigma_x^{-1}(x-\mu_x)\end{aligned}$$
- Set $L_x = L_y + \mathcal{C}$, $\mathcal{C} = \frac{1}{2} \log \frac{\det(\Sigma_y)}{\det(\Sigma_x)} = \frac{1}{2} \log \frac{\det(A\Sigma_x A^T)}{\det(\Sigma_x)} = \log\det(A)$.

Compensation term when A is not square

we assume the compensation term $\mathcal{C}$ remains the same:
$\mathcal{C} = \frac{1}{2}\log \det(A\Sigma_{\hat{x}}A^T)$
(drop out $\log\det(\Sigma_{\hat{x}})$ because it does not depend on A)

Compensation term when A is not square

we assume the compensation term $\mathcal{C}$ remains the same:
$\mathcal{C} = \frac{1}{2}\log\det(A\Sigma_{\hat{x}}A^T)$
(drop out $\log\det(\Sigma_{\hat{x}})$ because it does not depend on A)
the objective function becomes:

$$Q(W) = \frac{1}{2}T\log\det(A\Sigma_{\hat{x}}A^T) - \frac{1}{2}\sum_{j=1}^{N}\sum_{t=1}^{T}\gamma_t(j)(W\hat{\xi}_t - \mu_i)^T\Sigma_j^{-1}(W\hat{\xi}_t - \mu_j)$$

The first term is a replacement of the Jacobian term when A is not a square matrix, while $\Sigma_{\hat{x}}$ is the covariance matrix computed from $\hat{x}_t$, $t = 1, ..., T$.

Compute the obj. function using stats files

Definition (mean and variance stats)

$$K = \sum_{j=1}^{N}\sum_{t=1}^{T}\gamma_t(j)\Sigma_j^{-1}\mu_j\hat{\xi}_t^{T}, \textit{size } d \times (3d+1)$$

$$G_i = \sum_{j=1}^{N}\sum_{t=1}^{T}\gamma_t(j)\sigma_{j,l}^{-1}\hat{\xi}_t\hat{\xi}_t^{T}, \textit{size } (3d+1) \times (3d+1), i = 1, ..., d$$

Compute the obj. function using stats files

Definition (mean and variance stats)

$$K = \sum_{j=1}^{N}\sum_{t=1}^{T}\gamma_t(j)\Sigma_j^{-1}\mu_j\hat{\xi}_t^{\,T}, \textit{size } d \times (3d+1)$$

$$G_i = \sum_{j=1}^{N}\sum_{t=1}^{T}\gamma_t(j)\sigma_{j,I}^{-1}\hat{\xi}_t\hat{\xi}_t^{\,T}, \textit{size } (3d+1)\times(3d+1), i = 1, ..., d$$

the objective function for context filtering and its gradient

$$Q(W) = \frac{1}{2}T\log\det(A\Sigma_{\hat{x}}A^T) + tr(W^TK) - \frac{1}{2}\sum_{j=1}^{n} tr(W^TE_{j,j}WG_j)$$

$$\frac{\partial Q}{\partial W} = T \times [(A\Sigma_{\hat{x}}A^T)^{-1}A\Sigma_{\hat{x}}, \quad \mathbf{0}] + K - \sum_{j=1}^{n} E_{j,j}WG_j$$

Solve the optimization problem

1. row-by-row iterative update algorithm in [Gales 98] cannot be applied here
    - the determinant of a square matrix equals the dot product of any given row with the corresponding row of cofactors.
    - It is not obvious how to extend this algorithm to non-square matrices

Solve the optimization problem

1. row-by-row iterative update algorithm in [Gales 98] cannot be applied here
   - the determinant of a square matrix equals the dot product of any given row with the corresponding row of cofactors.
   - It is not obvious how to extend this algorithm to non-square matrices
2. use limited memory BFGS algorithm along with line search (HCL package)
   - only need functions to evaluate the objective function and its gradient
   - gradient magnitude/maximum number of iterations are set to stop the opt. module

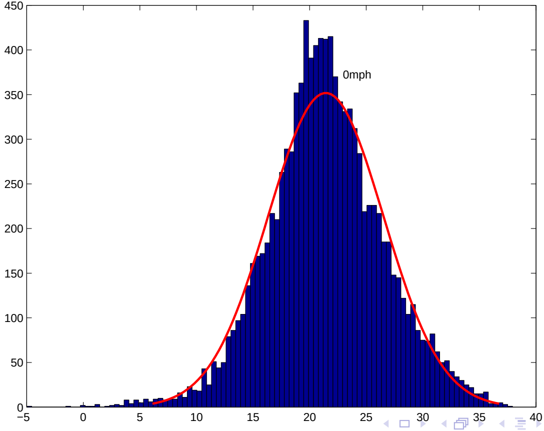
Training data and Models

- Majority of the training data was collected in stationary cars
- Total 800K training utterances/800 hours
- Word-internal with pentaphone context, with 830 context-dependent states and 10K Gaussians
- With LDA 40-dim features, built a ML model, a discriminative trained BMMI (boosted maximum mutual information) model, and a FMMI (feature-space maximum mutual information) model.

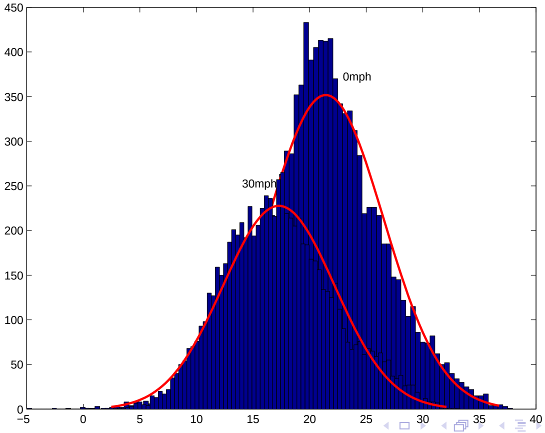
Test data

- Recorded in cars at three different speeds: 0mph (idling), 30mph and 60mph.
- Four tasks are selected in the test set: addresses, digits, commands and radio control
- Total about 26K utterances and 130K words

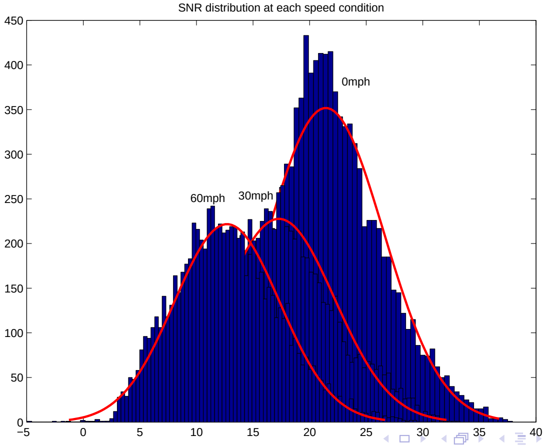
SNR distribution



SNR distribution



SNR distribution



Experiments

- Unsupervised speaker adaptation on the corresponding ML/BMMI/FMMI models
- MLCF-n: maximum likelihood context filtering with context size n

Experiments

- Unsupervised speaker adaptation on the corresponding ML/BMMI/FMMI models
- MLCF-n: maximum likelihood context filtering with context size n
- MLCF transform initialization: zero matrices for all the frames/identity matrix for the center
- MLCF-n-init: uses FMLLR as the starting point for the center frame

Adaptation results on the ML model

WER/SER	0mph	30mph	60mph
baseline	0.77/3.34	1.28/5.15	2.65/8.94

Adaptation results on the ML model

WER/SER	0mph	30mph	60mph
baseline	0.77/3.34	1.28/5.15	2.65/8.94
FMLLR	0.57/2.42	0.94/3.82	1.87/6.29

Adaptation results on the ML model

WER/SER	0mph	30mph	60mph
baseline	0.77/3.34	1.28/5.15	2.65/8.94
FMLLR	0.57/2.42	0.94/3.82	1.87/6.29
MLCF-2	0.55/2.41	0.93/3.84	**1.44/5.49**

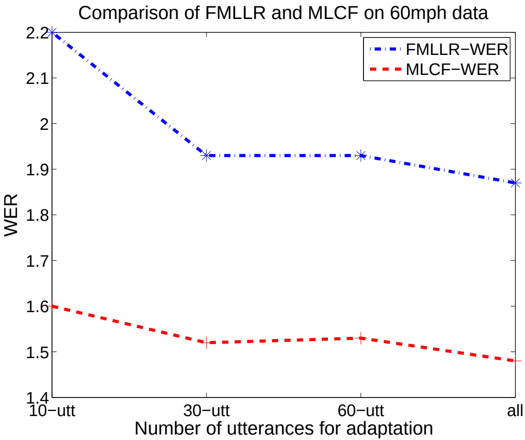
Adaptation results on the ML model

WER/SER	0mph	30mph	60mph
baseline	0.77/3.34	1.28/5.15	2.65/8.94
FMLLR	0.57/2.42	0.94/3.82	1.87/6.29
MLCF-2	0.55/2.41	0.93/3.84	**1.44/5.49**
MLCF-1	0.54/2.31	0.95/3.91	**1.48/5.54**
MLCF-1-init	0.54/2.32	0.96/3.89	**1.50/5.58**

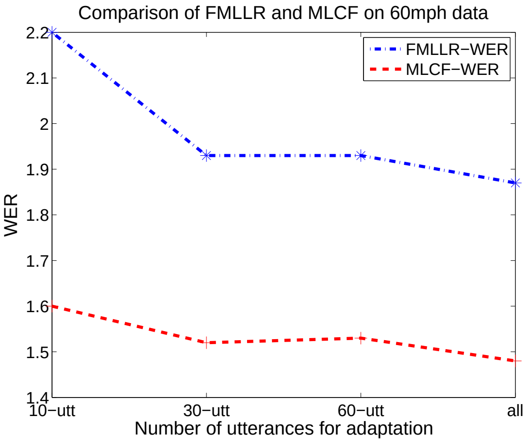
Table: Comparison of FMLLR and MLCF adapted on the ML model.

- Compared to FMLLR, on noisy **60mph** data, 23%/13% relative gain on WER/SER over FMLLR, tiny gains on 0mph/30mph
- Starting with FMLLR for the center frame does not provide any advantage over the identity matrix

Effect of adaptation data

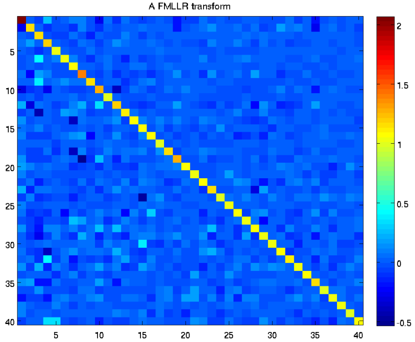


Effect of adaptation data

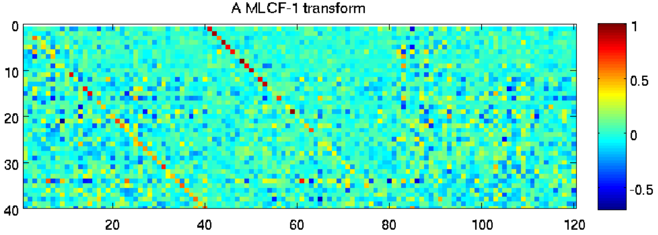


- in 10-utt case, MLCF-1 gains even more over FMLLR — 30% relative
- for FMLLR there is 15% degradation from all-utterance to 10-utterance, while for MLCF-1, only 7% relative degradation.

Visual comparison of FMLLR and MLCF



Visual comparison of FMLLR and MLCF



Adaptation results on the BMMI model

WER/SER	0mph	30mph	60mph
baseline	0.63/2.76	0.96/3.82	2.02/6.95
FMLLR	0.46/1.98	0.75/3.06	1.47/5.24
MLCF-1	0.45/1.91	0.74/3.05	**1.33/4.68**
MLCF-1-init	0.43/1.86	0.74/3.04	1.33/4.77

Table: Comparison of FMLLR and MLCF adapted on the BMMI model.

- 9%/11% relative improvement of WER/SER over FMLLR on the noisy 60mph data
- starting with FMLLR transform for the central frame does not provide any advantage over the identity transform.

Adaptation results on the FMMI model

WER/SER	0mph	30mph	60mph
baseline	0.45/1.90	0.76/3.23	1.30/5.05
FMLLR	0.33/1.40	0.60/2.52	1.00/4.06
MLCF-1	0.32/1.31	0.61/2.56	0.96/3.84
MLCF-1-init	0.32/1.34	0.59/2.47	**0.93/3.75**

Table: Comparison of FMLLR and MLCF adapted on the FMMI model.

This time MLCF-1-init is better than MLCF-1, and gains 7%/9% relative on WER/SER over FMLLR.

Summary

- MLCF: extend the full-rank square matrix of FMLLR to a non-square matrix that uses neighboring feature vectors to estimate the adapted central feature vector
- MLCF is shown outperform FMLLR on noisy 60mph data: 23% on WER over FMLLR with adapted ML model, and 7%/9% on the FMMI/BMMI models.

Summary

- MLCF: extend the full-rank square matrix of FMLLR to a non-square matrix that uses neighboring feature vectors to estimate the adapted central feature vector
- MLCF is shown outperform FMLLR on noisy 60mph data: 23% on WER over FMLLR with adapted ML model, and 7%/9% on the FMMI/BMMI models.

Future work includes

- use disc. ojective function or smoothing of disc. and ML objective functions
- check the interaction of context filtering with other front-end noise robustness techniques (e.g. Spectral Substraction, Dynamic Noise Adaptation)