ELEN E6884 - Topics in Signal Processing
Topic: Speech Recognition
Lecture 9

Stanley F. Chen, Michael A. Picheny, and Bhuvana Ramabhadran
IBM T.J. Watson Research Center
Yorktown Heights, NY, USA
stanchen@us.ibm.com, picheny@us.ibm.com
bhuvana@us.ibm.com

10 November 2009

Outline of Today's Lecture

- Administrivia
- Cepstral Mean Removal
- Spectral Subtraction
- Code Dependent Cepstral Normalization
- Parallel Model Combination
- Some Comparisons
- Break
- MAP Adaptation
- MLLR and fMLLR Adaptation

Robustness - Things Change

- Background noise can increase or decrease
- Channel can change
  - Different microphone
  - Microphone placement
- Speaker characteristics vary
  - Different glottal waveforms
  - Different vocal tract lengths
  - Different speaking rates
- Heaven knows what else can happen

Robustness Strategies

Basic Acoustic Model: $P(O|W, \theta)$

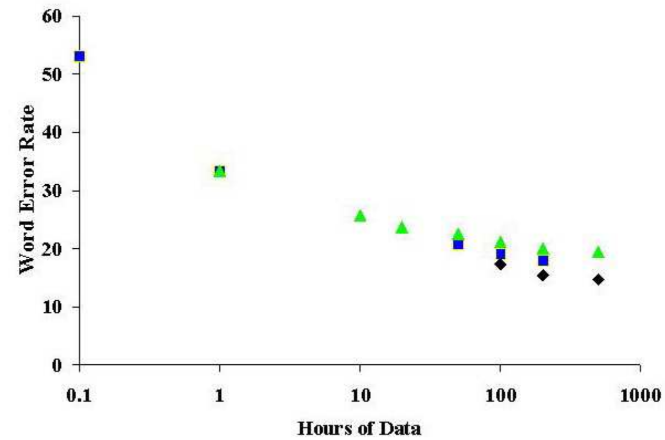
- **Robust features:** Features O that are independent of noise, channel, speaker, etc. so θ does not have to be modified.
  - More an art than a science but requires little/no data
- **Noise Modeling:** Explicit models for the effect background noise has on speech recognition parameters $\theta' = f(\theta, N)$
  - Works well when model fits, requires less data
- **Adaptation:** Update estimate of θ from new observations
  - Very powerful but often requires the most data $\theta' = f(N, p(O|W, \theta))$

Robustness Outline

- General Adaptation Issues - Training and Retraining
- Features
  - PLP
- Robust Features
  - Cepstral Mean Removal
  - Spectral Subtraction
  - Codeword Dependent Cepstral Normalization (CDCN) - Noise Modeling
  - Parallel Model Combination
  - Some comparisons of various noise immunity schemes
- Adaptation
  - Maximum A Posteriori (MAP) Adaptation
  - Maximum Likelihood Linear Regression (MLLR)
  - feature-based MLLR (fMLLR)

Adaptation - General Training Issues

Most systems today require $>$ 200 hours of speech from $>$ 200 speakers to train robustly for a new domain.



Adaptation - General Retraining

- If the environment changes, retrain system from scratch in new environment
  - Very expensive - cannot collect hundreds of hours of data for each new environment
- Two strategies
  - Environment simulation
  - Multistyle Training

Environment Simulation

- Take training data
- Measure parameters of new environment
- Transform training data to match new environment
  - Add matching noise to the new test environment
  - Filter to match channel characteristics of new environment
- Retrain system, hope for the best.

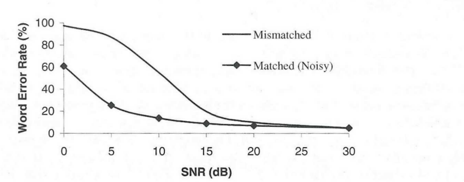


Figure 10.31 Word error rate as a function of the testing data SNR (dB) for Whisper trained on clean data and a system trained on noisy data at the same SNR as the testing set as in Figure 10.30. White noise at different SNRs is added.

Multistyle Training

- Take training data
- Corrupt/transform training data in various representative fashions
- Collect training data in a variety of representative environments
- Pool all such data together; retrain system

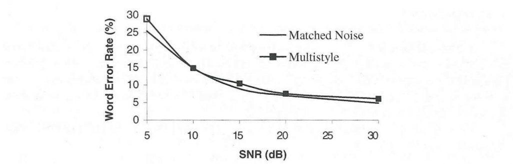


Figure 10.32 Word error rates of multistyle training compared to matched-noise training as a function of the SNR in dB for additive white noise. Whisper is trained as in Figure 10.30. The error rate of multistyle training is between 12% (for low SNR) and 25% (for high SNR) higher in relative terms than that of matched-condition training. Nonetheless, multistyle training does better than a system trained on clean data for all conditions other than clean speech.

Issues with System Retraining

- Simplistic models of noise and channel
  - e.g. telephony degradations more than just a decrease in bandwidth
- Hard to anticipate every possibility
  - In high noise environment, person speaks louder with resultant effects on glottal waveform, speed, etc.
- System performance in clean envrironment can be degraded.
- Retraining system for each environment is very expensive
- Therefore other schemes - noise modeling and general forms of adaptation - are needed and sometimes used in tandem with these other schemes.

Cepstral Mean Normalization

We can model a large class of environmental distortions as a simple linear filter:

$$\hat{y}[n] = \hat{x}[n] * \hat{h}[n]$$

where $\hat{h}[n]$ is our linear filter and $*$ denotes convolution (Lecture 1). In the frequency domain we can write

$$\hat{Y}(k) = \hat{X}(k)\hat{H}(k)$$

Taking the logarithms of the amplitudes:

$$\log \hat{Y}(k) = \log \hat{X}(k) + \log \hat{H}(k)$$

that is, the effect of the linear distortion is to add a constant vector to the amplitudes in the log domain.

Now if we examine our normal cepstral processing, we can write

this as the following processing sequence.

$$\begin{aligned} \mathbf{O}[k] &= \text{Cepst}(\log \text{Bin}(\text{FFT}(\hat{x}[n] * \hat{h}(n)))) \\ &= \text{Cepst}(\log \text{Bin}(\hat{X}(k)\hat{H}(k))) \end{aligned}$$

We can essentially ignore the effects of binning. Since the mapping from mel-spectra to mel cepstra is linear, from the above, we can essentially model the effect of linear filtering as just adding a constant vector in the cepstral domain:

$$\mathbf{O}'[k] = \mathbf{O}[k] + h[k]$$

so robustness can be achieved by estimating $h[k]$ and subtracting it from the observed $\mathbf{O}'[k]$.

Cepstral Mean Normalization - Estimation

Given a set of cepstral vectors $\mathbf{O}_t$ we can compute the mean:

$$\bar{\mathbf{O}} = \frac{1}{N}\sum_{t=1}^{N}\mathbf{O}_t$$

"Cepstral mean normalization" produces a new output vector $\hat{\mathbf{O}}_t$

$$\hat{\mathbf{O}}_t = \mathbf{O}_t - \bar{\mathbf{O}}$$

Say the signal correponding to $\mathbf{O}_t$ is processed by a linear filter. Say $\mathbf{h}$ is a cepstral vector corresponding to such a linear filter. In such a case, the output after linear filtering will be

$$\mathbf{y}_t = \mathbf{O}_t + \mathbf{h}$$

The mean of $\mathbf{y}_t$ is

$$\bar{\mathbf{y}} = \frac{1}{N}\sum_{t=1}^{N}\mathbf{y}_t = \frac{1}{N}\sum_{t=1}^{N}(\mathbf{O}_t + \mathbf{h}) = \bar{\mathbf{O}} + \mathbf{h}$$

so after "Cepstral Mean Normalization"

$$\hat{\mathbf{y}}_t = \mathbf{y}_t - \bar{\mathbf{y}} = \hat{\mathbf{O}}_t$$

That is, the influence of $\mathbf{h}$ has been eliminated.

Cepstral Mean Normalization - Issues

- Error rates for utterances even in the same environment improves (Why?)
- Must be performed on both training and test data.
- Bad things happen if utterances are very short (Why?)
- Bad things happen if there is a lot of variable length silence in the utterance (Why?)
- Cannot be used in a real time system (Why?)

Cepstral Mean Normalization - Real Time Implementation

Can estimate mean dynamically as

$$\bar{\mathbf{O}}_t = \alpha\mathbf{O}_t + (1-\alpha)\bar{\mathbf{O}}_{t-1}$$

In real-life applications, it is useful run a silence detector in parallel and turn adaptation off (set α to zero) when silence is detected, hence:

$$\bar{\mathbf{O}}_t = \alpha(s)\mathbf{O}_t + (1-\alpha(s))\bar{\mathbf{O}}_{t-1}$$

Cepstral Mean Normalization - Typical Results

From "Environmental Normalization for Robust Speech Recognition Using Direct Cepstral Compensation" F. Liu, R. STern, A. Acero and P. Moreno Proc. ICASSP 1994, Adelaide Australia

	CLOSE	OTHER
BASE	8.1	38.5
CMN	7.6	21.4
Best	8.4	13.5

Task is 5000-word WSJ LVCSR

Spectral Subtraction - Background

Another common type of distortion is additive noise. In such a case, we may write

$$y[i] = x[i] + n[i]$$

where $n[i]$ is some noise signal. Since we are dealing with linear operations, we can write in the frequency domain

$$Y[k] = X[k] + N[k]$$

The power spectrum (Lecture 1) is therefore

$$|Y[k]|^2 = |X[k]|^2 + |N[k]|^2 + X[k]N^*[k] + X^*[k]N[k]$$

If we assume $n[i]$ is zero mean and uncorrelated with $x[i]$, the last two terms on the average would also be zero. By the time we window the signal and also bin the resultant amplitudes of the

spectrum in the mel filter computation, it is also reasonable to assume the net contribution of the cross terms will be zero.

In such a case we can write

$$|Y[k]|^2 = |X[k]|^2 + |N[k]|^2$$

Spectral Subtraction - Basic Idea

In such a case, it is reasonable to estimate $|X[k]|^2$ as:

$$|\hat{X}[k]|^2 = |Y[k]|^2 - |\hat{N}[k]|^2$$

where $|\hat{N}[k]|^2$ is some estimate of the noise. One way to estimate this is to average $|Y[k]|^2$ over a sequence of frames known to be silence (by using a silence detection scheme):

$$|\hat{N}[k]|^2 = \frac{1}{M}\sum_{t=0}^{M-1} |Y_t[k]|^2$$

Note that $Y[k]$ here can either be the FFT output (when trying to actually reconstruct the original signal) or, in speech recognition, the output of the FFT after Mel binning.

Spectral Subtraction - Issues

The main issue with Spectral Subtraction is that $|\hat{N}[k]|^2$ is only an estimate of the noise, not the actual noise value itself. In a given frame, $|Y[k]|^2$ may be less than $|\hat{N}[k]|^2$. In such a case, $|\hat{X}[k]|^2$ would be negative, wreaking havoc when we take the logarithm of the amplitude when computing the mel-cepstra.

The standard solution to this problem is just to "floor" the estimate of $|\hat{X}[k]|^2$:

$$|\hat{X}[k]|^2 = \max(|Y[k]|^2 - |\hat{N}[k]|^2, \beta)$$

where β is some appropriately chosen constant.

Given that for any realistic signal, the actual $|X(k)|^2$ has some amount of background noise, we can estimate this noise during training similarly to how we estimate $|N(k)|^2$. Call this estimate

$|N_{\text{train}}[k]|^2$. In such a case our estimate for $|X(k)|^2$ becomes

$$|\hat{X}[k]|^2 = \max(|Y[k]|^2 - |\hat{N}[k]|^2, |N_{\text{train}}[k]|^2)$$

Even with this noise flooring, because of the variance of the noise process, little "spikes" come through generating discontinuities in in time in low-noise regions with disastrous effects on recognition. To deal with this, sometimes "oversubtraction" is used:

$$|\hat{X}[k]|^2 = \max(|Y[k]|^2 - \alpha|\hat{N}[k]|^2, |N_{\text{train}}[k]|^2)$$

where α is some constant chosen to minimize the noise spikes when there is no speech.

Spectral Subtraction - Performance

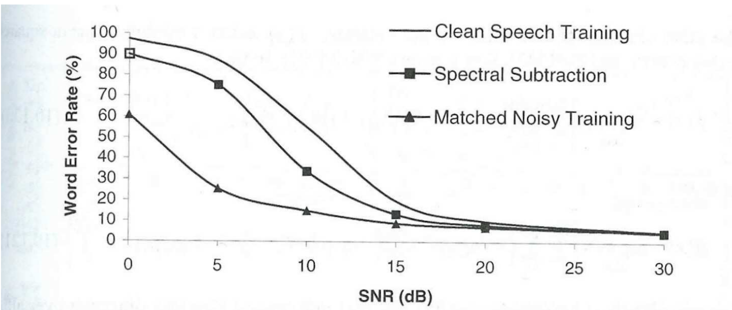


Figure 10.28 Word error rate as a function of SNR (dB) using Whisper on the *Wall Street Journal* 5000-word dictation task. White noise was added at different SNRs. The solid line represents the baseline system trained with clean speech, the line with squares the use of spectral subtraction with the previous clean HMMs. They are compared to a system trained on the same speech with the same SNR as the speech tested on.

%

Combined Noise and Channel Degradations

Spectral subtraction assumes degradation due to additive noise and Cepstral Mean Removal assumes degradation due to multiplicative noise. Combining both, we get

$$Y = HX + N$$

or taking logarithms

$$\ln Y = \ln X + \ln H + \ln(1 + \frac{N}{HX})$$

switching to the log domain we get

$$y^l = x^l + h^l + \ln(1 + e^{n^l - x^l - h^l})$$

or using the notation $y = Cy^l$ to move to the cepstral domain we get

$$y = x + h + C\ln(1 + e^{C^{-1}(n-x-h)}) = x + h + r(x, n, h)$$

or

$$x = y - h - r(x, n, h)$$

So, this is not an easy expression to work with. What do we do?

Generalization: Minimum Mean Square Error Estimation

Assume the vector y is some corrupted version of the vector x. We get to observe y and wish to devise an estimate for x. It would appear that a reasonable property would be to find some estimate $\hat{x}$ such that the average value of $(x-\hat{x})^2$ is minimized. It can easily be shown that the best estimator $\hat{x}$ in such a case is just:

$$\hat{x} = E(x|y) = \int xp(x|y)dx$$

Spectral subtraction can be shown to be a special case of MMSE with a set of restrictive assumptions. In general, a goal of MMSE modeling is to look for relatively simple functional forms for $p(x|y)$ so that a closed form expression for $\hat{x}$ in terms of y can be found.

Modeling $p(x|y)$ via Gaussian Mixtures

Now

$$p(x|y) = p(y, x)/p(y)$$

Let us model $p(x, y)$ as a function of a sum of K distributions:

$$p(x, y) = \sum_{k=1}^{K} p(x, y|k)p(k)$$

Let us write

$$p(x, y|k) = p(y|x, k)q(x|k)$$

where

$$q(x|k) = \frac{1}{K}\frac{1}{\sqrt{2\pi}\sigma}\, e^{-\frac{(x-\mu_k)^2}{2\sigma^2}}$$

From above, our noise model is

$$x = y - h - r(x, n, h)$$

which is equivalent to saying

$$p(y|x, k) = \delta(x - (y - h - r(x, n, h)))$$

where $\delta(t)$ is a delta (impulse) function

CDCN - Codeword Dependent Cepstral Normalization

The main assumption in CDCN is that the correction vector $r(x, n, h)$ is constant given mixture component k and can just be computed directly from the mean of mixture component k.

$$r[k] = C \ln(1 + e^{C^{-1}(n - \mu_k - h)})$$

In this case we can write

$$p(y|x, k) = \delta(x - (y - h - r[k]))$$

Note also that because of the nature of the delta function:

$$\begin{aligned}
p(y) &= \int p(x, y) dx \\
&= \int \sum_{k=1}^{K} \delta(x - (y - h - r[k])) q(x|k) dx \\
&= \sum_{k=1}^{K} \int \delta(x - (y - h - r[k])) q(x|k) dx \\
&= \sum_{k=1}^{K} q(y - h - r[k]|k)
\end{aligned}$$

Estimation Equations

We now may write the estimate for x as

$$\begin{aligned}
\hat{x} &= \int x p(x|y) dx \\
&= \int x \frac{p(x, y)}{\sum_{l=1}^{K} q(y - h - r[l]|l)} dx \\
&= \int x \frac{\sum_{k=1}^{K} \delta(x - (y - h - r[k])) q(x|k)}{\sum_{l=1}^{K} q(y - h - r[l]|l)} dx \\
&= \sum_{k=1}^{K} \frac{(y - h - r[k]) q(y - h - r[k]|k)}{\sum_{l=1}^{K} q(y - h - r[l]|l)}
\end{aligned}$$

Note the term involving q is just the mixture of gaussian posterior probability we saw in Lecture 3. Iterative equations for estimating h and n can also be developed; refer to the reading for more

information.

Vector Taylor Series is a CDCN variant in which r is approximated as a linearized function with respect to x and μ_k rather than assumed constant.

Algonquin is a more sophistcated CDCN variant in which $p(y|x, k)$ is assumed to have an actual probability distribution (e.g., Normal) to model noise phase uncertainty.

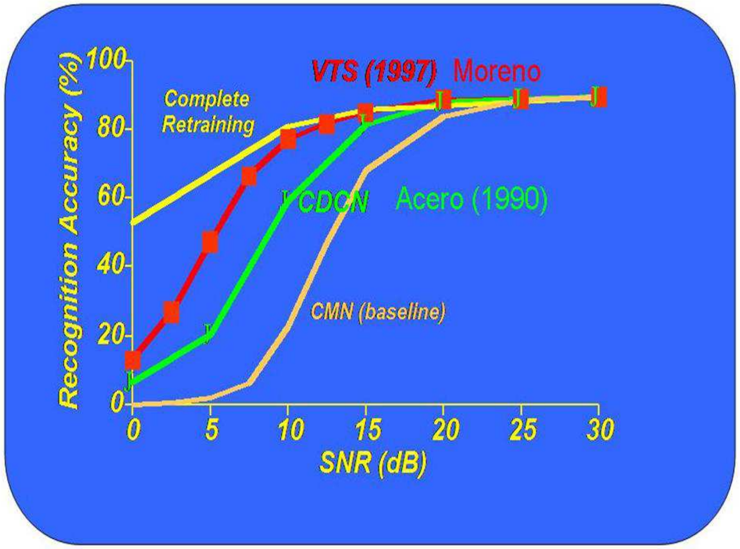
CDCN Performance

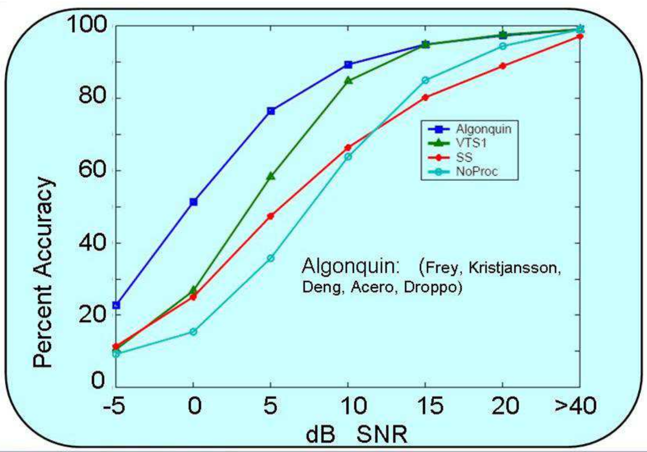
From Alex Acero's PhD Thesis "Acoustical and Environmental Robustness in Automatic Speech Recognition" CMU (1990):

TRAIN/TEST	CLS/CLS	CLS/PZM	PZM/CLS	PZM/PZM
BASE	14.7	81.4	63.1	23.5
CMR	N/A	61.7	49.1	23.5
PSUB	N/A	61.4	29.4	29.9
MSUB	N/A	37.4	28.3	28.7
CDCN	14.7	25.1	26.3	22.1

Error rates for a SI alphanumeric task recorded on two different microphones.

Additional Performance Figures





Parallel Model Combination - Basic Idea

Idea: Incorporate model of noise directly into our GMM-based HMMs.

If our observations were just the FFT outputs this would be straightforward. In such a case, the corrupted version of our signal x with noise n is just:

$$y = x + n$$

If $x \sim N(\mu_x, \sigma_x^2)$ and $n \sim N(\mu_n, \sigma_n^2)$ then $y \sim N(\mu_x + \mu_n, \sigma_x^2 + \sigma_n^2)$

But our observations are cepstral parameters - extremely nonlinear transformations of the space in which the noise is additive. What do we do?

Parallel Model Combination - One Dimensional Case

Let us make a Very Simple approximation to Cepstral parameters: $X = \ln x, N = \ln n$.

Pretend we are modeling these "cepstral" parameters with "HMMs" in the form of univariate Gaussians. In such a case, let us say $X \sim N(\mu_X, \sigma_X^2)$ and $N \sim N(\mu_N, \sigma_N^2)$. We can then write:

$$Y = \ln(e^X + e^N)$$

What is the probability distribution of Y?

Parallel Model Combination - Log Normal Distribution

If X is a Gaussian random variable with mean μ and variance σ^2 then $x = e^X$ follows the *lognormal* distribution:

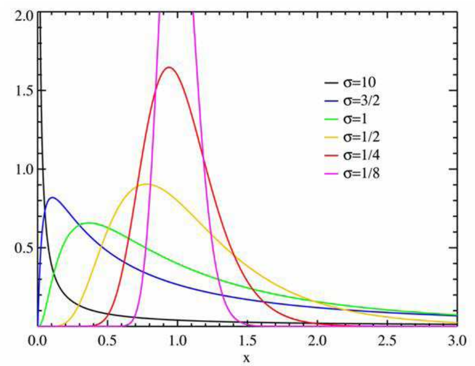
$$p(x) = \frac{1}{x\sigma\sqrt{2\pi}} \exp(-\frac{(\ln x - \mu)^2}{2\sigma^2})$$

The mean of this distribution can be shown to be

$$E(x) = \int xp(x)dx = \exp(\mu + \sigma^2/2)$$

and the variance

$$E((x - E(x))^2) = \int (x - E(x))^2 p(x)dx = \mu^2(\exp(\sigma^2) - 1)$$



Parallel Model Combination - Lognormal Approximation

Since back in the linear domain

$$y = x + n$$

the distribution of y will correspond to the distribution of a sum of two lognormal variables x and n.

$$\begin{aligned}
\mu_x &= \exp(\mu_X + \sigma_X^2/2) \\
\sigma_x^2 &= \mu_X^2(\exp(\sigma_X^2) - 1) \\
\mu_n &= \exp(\mu_N + \sigma_N^2/2) \\
\sigma_n^2 &= \mu_N^2(\exp(\sigma_N^2) - 1)
\end{aligned}$$

If x and n are uncorrelated, we can write:

$$\begin{aligned}\mu_y &= \mu_x + \mu_n \\ \sigma_y^2 &= \sigma_x^2 + \sigma_n^2\end{aligned}$$

Unfortunately, although the sum of two Gaussian variables is a Gaussian, the sum of two lognormal variables is not lognormal.

As good engineers, we will promptly ignore this fact and act as if y DOES have a lognormal distribution (!). In such a case, $Y = \ln y$ is Gaussian and the mean and variance are given by:

$$\begin{aligned}\mu_Y &= \ln \mu_y - \frac{1}{2}\ln\left[\frac{\sigma_y^2}{\mu_y^2}+1\right] \\ \sigma_Y^2 &= \ln\left[\frac{\sigma_y^2}{\mu_y^2}+1\right]\end{aligned}$$

The matrix and vector forms of the modified means and variances, similar to the unidimensional forms above, can be found in HAH pg. 533

Parallel Model Combination - Actual Cepstra

Remember that the mel-cepstra are computed from mel-spectra by the following formula:

$$c[n] = \sum_{m=0}^{M-1} X[m]\cos(\pi n(m-1/2)/M)$$

We can view this as just a matrix multiplication:

$$\mathbf{c} = \mathbf{C}\mathbf{x}$$

where $\mathbf{x}$ is just the vector of the $x[m]$s and the components of matrix $\mathbf{C}$ are

$$C_{ij} = \cos(\pi j(i-1/2)/M)$$

In such a case, the mean and covariance matrix in the mel-

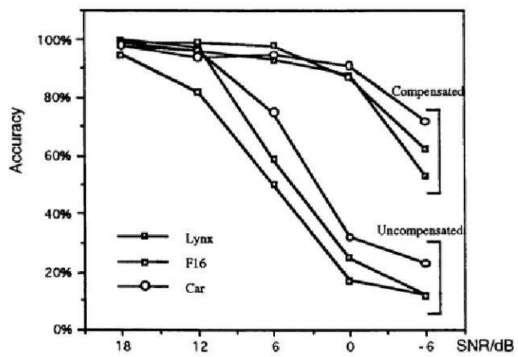
spectral domain can be computed as

$$\begin{aligned}\boldsymbol{\mu}_{\mathbf{x}} &= C^{-1}\boldsymbol{\mu}_c \\ \boldsymbol{\Sigma}_{\mathbf{x}} &= C^{-1}\boldsymbol{\Sigma}_c(C^{-1})^T\end{aligned}$$

and similarly for the noise cepstra.

Parallel Model Combination - Performance

From "PMC for Speech Recognition in Convolutional and Additive Noise" by Mark Gales and Steve Young, (modified by Martin Russell) TR-154 Cambridge U. 1993.



Although comparisons seem to be rare, when PMC is compared to schemes such as VTS, VTS seems to have proved somewhat superior in performance. However, the basic concepts of PMC have been recently combined with EM-like estimation schemes to significantly enhance performance (more later).

Maximum A Posteriori Parameter Estimation - Basic Idea

Another way to achieve robustness is to take a fully trained HMM system, a small amount of data from a new domain, and combine the information from the old and new systems together. To put everything on a sound framework, we will utilize the parameters of the fully-trained HMM system as *prior* information.

In Maximum Likelihood Estimation (Lecture 3) we try to pick a set of parameters $\hat{\theta}$ that maximize the likelihood of the data:

$$\hat{\theta} = \arg\max_{\theta} \mathcal{L}(O_1^N|\theta)$$

In Maximum A Posterior Estimation we assume there is some prior probability distribution on θ, $p(\theta)$ and we try to pick $\hat{\theta}$ to

maximize the a posteriori probability of θ given the observations:

$$\begin{aligned}\hat{\theta} &= \arg\max_{\theta} p(\theta|O_1^N) \\ &= \arg\max_{\theta} \mathcal{L}(O_1^N|\theta)p(\theta)\end{aligned}$$

Maximum A Posteriori Parameter Estimation - Conjugate Priors

What form should we use for $p(\theta)$? To simplify later calculations, we try to use an expression so that $\mathcal{L}(O_1^N|\theta)p(\theta)$ has the same functional form as $\mathcal{L}(O_1^N|\theta)$. This type of form for the prior is called a *conjugate prior*.

In the case of a univariate Gaussian we are trying to estimate μ and σ. Let $r = 1/\sigma^2$. An appropriate conjugate prior is:

$$p(\theta) = p(\mu, r) \propto r^{(\alpha-1)/2} exp(-\frac{\tau r}{2}(\mu - \mu_p)^2) exp(-(\sigma_p^2 r/2)$$

where μ_p and σ_p^2 are prior estimates/knowledge of the mean and variance from some initial set of training data. Note how ugly the functional forms get even for a relatively simple case!

Maximum A Posteriori Parameter Estimation - Univariate Gaussian Case

Without torturing you with the math, we can plug in the conjugate prior expression and compute μ and r to maximize the a posteriori probability. We get

$$\hat{\mu} = \frac{N}{N+\tau}\mu_O + \frac{\tau}{N+\tau}\mu_p$$

where μ_O is the mean of the data computed using the ML procedure.

$$\hat{\sigma^2} = \frac{N}{N+\alpha-1}\sigma_O^2 + \frac{\tau(\mu_O - \hat{\mu})^2 + \sigma_p^2}{N+\alpha-1}$$

Maximum A Posteriori Parameter Estimation - General HMM Case

Through a set of similar manipulations, we can move generalize the previous formula to the HMM case. As before, c_{ik} is the mixture weight k for state i, $\nu_{ik}, \boldsymbol{\mu}_{ik}, \Sigma_{ik}$ are the prior estimates for the mixture weight, mean and covariance matrix of mixture component k for state i from a previously trained HMM system. In this case:

$$\hat{c}_{ik} = \frac{\nu_{ik} - 1 + \sum_t C_t(i,k)}{\sum_l(\nu_{il} - 1 + \sum_t C_t(i,l))}$$

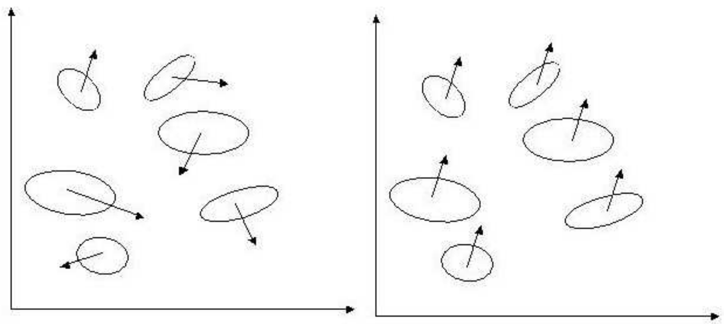
$$\hat{\boldsymbol{\mu}}_{ik} = \frac{\tau_{ik}\boldsymbol{\mu}_{ik} + \sum_{t=1}^N C_t(i,k)\mathbf{O}_t}{\sum_l(\tau_{ik} + \sum_t C_t(i,l))}$$

$$\begin{aligned}\hat{\Sigma}_{ik} &= \frac{(\alpha_{ik} - D)\Sigma_{ik}}{\alpha_{ik} - D + \sum_{t=1}^N C_t(i,k)} \\ &+ \frac{\tau_{ik}(\hat{\boldsymbol{\mu}}_{ik} - \boldsymbol{\mu}_{ik})(\hat{\boldsymbol{\mu}}_{ik} - \boldsymbol{\mu}_{ik})^t}{\alpha_{ik} - D + \sum_{t=1}^N C_t(i,k)} \\ &+ \frac{\sum_{t=1}^N C_t(i,k)(\mathbf{O}_t - \hat{\boldsymbol{\mu}}_{ik})(\mathbf{O}_t - \hat{\boldsymbol{\mu}}_{ik})^t}{\alpha_{ik} - D + \sum_{t=1}^N C_t(i,k)}\end{aligned}$$

Both τ and α are balancing parameters that can be tuned to optimize performance on different test domains. In practice, a single τ is adequate across all states and Gaussians, and variance adaptation rarely has been successful, at least in speech recognition, to improve performance. We will save discussions of MAP performance on adaptation until the end of the MLLR section, which is next.

Maximum Likelihood Linear Regression - Basic Idea

In MAP, the different HMM Gaussians are free to move in any direction. In Maximum Likelihood Linear Regression the means of the Gaussians are constrained to only move according to an affine transformation ($Ax + b$).



Simple Linear Regression - Review

Say we have a set of points $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$ and we want to find coefficients a, b so that

$$\sum_{t=1}^{N} (O_t - (ax_t + b))^2$$

is minimized. Define $\mathbf{w}$ to be the column vector consisting of (a, b), and the column vector $\mathbf{x}_t$ corresponding to $(x_t, 1)$ We can then write the above set of equations as

$$\sum_{t=1}^{N} (O_t - \mathbf{x}_t^T \mathbf{w})^2$$

Taking derivatives with respect to $\mathbf{w}$ we get

$$\sum_{t=1}^{N} 2\mathbf{x}_t (O_t - \mathbf{x}_t^T \mathbf{w}) = 0$$

so collecting terms we get

$$\mathbf{w} = \left[\sum_{t=1}^{N} \mathbf{x}_t \mathbf{x}_t^T \right]^{-1} \sum_{t=1}^{N} \mathbf{x}_t O_t$$

In MLLR, the x values will turn out to correspond to the means of the Gaussians.

MLLR for Univariate GMMs

We can write the likelihood of a string of observations $O_1^N = O_1, O_2, \ldots, O_N$ from a Gaussian Mixture Model as:

$$\mathcal{L}(O_1^N) = \prod_{t=1}^{N} \sum_{k=1}^{K} \frac{p_k}{\sqrt{2\pi}\sigma_k} e^{-\frac{(O_t - \mu_k)^2}{2\sigma_k^2}}$$

It is usually more convenient to deal with the log likelihood

$$L(O_1^N) = \sum_{t=1}^{N} \ln \left[\sum_{k=1}^{K} \frac{p_k}{\sqrt{2\pi}\sigma_k} e^{-\frac{(O_t - \mu_k)^2}{2\sigma_k^2}} \right]$$

Let us now say we want to transform all the means of the Gaussian by $a\mu_k + b$. It is convenient to define $\mathbf{w}$ as above, and to the augmented mean vector $\boldsymbol{\mu}_k$ as the column vector corresponding to $(\mu_k, 1)$. In such a case we can write the overall likelihood as

$$L(O_1^N) = \sum_{t=1}^{N} \ln \left[\sum_{k=1}^{K} \frac{p_k}{\sqrt{2\pi}\sigma_k} e^{-\frac{(O_t - \boldsymbol{\mu}_k^T \mathbf{w})^2}{2\sigma_k^2}} \right]$$

To maximize the likelihood of this expression we utilize the E-M algorithm we have briefly alluded to in our discussion of the Forward-Backward (aka Baum-Welch) algorithm.

E-M Review

The E-M Theorem states that if

$$\begin{aligned} Q(\mathbf{w}, \mathbf{w}') &= \sum_x p_{\mathbf{w}}(X_1^N | O_1^N) \ln p_{\mathbf{w}'}(X_1^N, O_1^N) \\ &> \sum_x p_{\mathbf{w}}(X_1^N | O_1^N) \ln p_{\mathbf{w}}(X_1^N, O_1^N) \end{aligned}$$

then

$$p_{\mathbf{w}'}(O_1^N) > p_{\mathbf{w}}(O_1^N)$$

Therefore if we can find

$$\hat{\mathbf{w}}' = \arg\max_{\mathbf{w}'} Q(\mathbf{w}, \mathbf{w}')$$

we can iterate to find a $\mathbf{w}$ that maximizes $p_{\mathbf{w}}(O_1^N)$ or equivalently $L(O_1^N)$

E-M for MLLR

For a Gaussian mixture, it can be shown that

$$\begin{aligned} Q(\mathbf{w}, \mathbf{w}') &= \sum_{k=1}^{K} \sum_{t=1}^{N} C_t(k) [\ln p_k \\ &\quad - \ln \sqrt{2\pi}\sigma_k - (O_t - \boldsymbol{\mu}_k^T \mathbf{w}')^2 / 2\sigma_k^2] \end{aligned}$$

where

$$C_t(k) = p_{\mathbf{w}}(k|O) = \frac{\frac{p_k}{\sqrt{2\pi}\sigma_k} e^{-\frac{(O_t - \boldsymbol{\mu}_k^T \mathbf{w})^2}{2\sigma_k^2}}}{\sum_{l=1}^{K} \frac{p_l}{\sqrt{2\pi}\sigma_l} e^{-\frac{(O_t - \boldsymbol{\mu}_l^T \mathbf{w})^2}{2\sigma_l^2}}}$$

We can maximize $Q(\mathbf{w}, \mathbf{w}')$ by computing it's derivative and setting it equal to zero:

$$\sum_{t=1}^{N}\left[\sum_{k=1}^{K} C_t(k)\frac{\boldsymbol{\mu}_k}{\sigma_k^2}(O_t - \boldsymbol{\mu}_k^T\mathbf{w}')\right]$$

Setting to zero and collecting terms we get

$$\sum_{t=1}^{N}\sum_{k=1}^{K} C_t(k)\frac{\boldsymbol{\mu}_k\boldsymbol{\mu}_k^T}{\sigma_k^2}\mathbf{w}' = \sum_{t=1}^{N}\sum_{k=1}^{K} C_t(k)\frac{\boldsymbol{\mu}_k}{\sigma_k^2}O_t$$

Define $C(k) = \sum_t C_t(k)$ and $\bar{O}(k) = \frac{1}{C(k)}\sum_t C_t(k)O_t$, we can rewrite the above as

$$\left[\sum_{k=1}^{K} C(k)\frac{\boldsymbol{\mu}_k\boldsymbol{\mu}_k^T}{\sigma_k^2}\right]\mathbf{w}' = \sum_{k=1}^{K} C(k)\frac{\boldsymbol{\mu}_k}{\sigma_k^2}\bar{O}(k)$$

so we may compute $\mathbf{w}'$ as just:

$$\mathbf{w}' = \left[\sum_{k=1}^{K} C(k)\frac{\boldsymbol{\mu}_k\boldsymbol{\mu}_k^T}{\sigma_k^2}\right]^{-1}\sum_{k=1}^{K} C(k)\frac{\boldsymbol{\mu}_k}{\sigma_k^2}\bar{O}(k)$$

compare to the expression for simple linear regression:

$$\mathbf{w} = \left[\sum_{t=1}^{N}\mathbf{x}_t\mathbf{x}_t^T\right]^{-1}\sum_{t=1}^{N}\mathbf{x}_tO_t$$

In actual speech recognition systems, the observations are vectors, not scalars, so the transform to be estimated is of the form

$$\mathbf{A}\boldsymbol{\mu} + \mathbf{b}$$

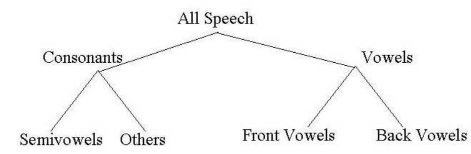
where $\mathbf{A}$ is a matrix and $\mathbf{b}$ is a vector. The resultant MLLR equations are somewhat more complex but follow the same basic form. We refer you to the readings for the details.

MLLR - Additional Considerations

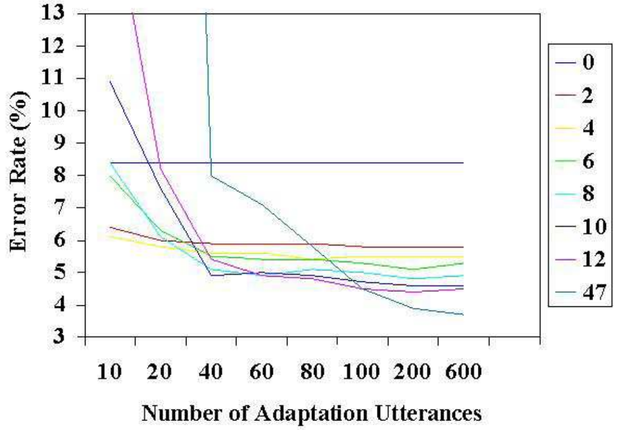
Since the typical parameter vector being processed is 39 dimensional (13 cepstral parameters, and the associated deltas and double-deltas) the number of matrix parameters to be estimated is roughly 1600. As a rule of thumb, if one frame of data gives you enough information to estimate one parameter, then we need at least 16 seconds of speech to estimate a full 39x39 MLLR matrix.

MLLR - Multiple Transforms

A single MLLR transform for all of speech is very restrictive. Multiple transforms can be created by grouping HMM states into larger classes, for example, at the phone level. Sometimes these classes can be arranged hierarchically, in the form of a tree. The number of speech frames at each node in the tree is examined, and if there are enough frames at a node, a separate transform is estimated for all the phones at the node.



MLLR - Performance



Feature Based MLLR

Let's say we now want to transform all the means by

$$\mu_k/a - b/a$$

and the variances by

$$\sigma_k^2/a^2$$

Define $\mathbf{O}_t$ as the augmented column observation vector (O_t,1), $\mathbf{w} = (a,b)^T$ as above, and $\mathbf{r} = (1,0)^T$ We can therefore write

$$\begin{aligned} Q(\mathbf{w},\mathbf{w}') &= \sum_{k=1}^{K}\sum_{t=1}^{N} C_t(k)[\ln p_k \\ &- \ln\sqrt{2\pi}\sigma_k + \ln \mathbf{r}^T\mathbf{w}' - (\mathbf{O}_t^T\mathbf{w}' - \mu_k)^2/2\sigma_k^2] \end{aligned}$$

with $C_t(k)$ defined similarly as in the MLLR discussion.

The primary advantage is that the likelihood computation can be written as purely as a transformation to the input features so if solvable it is very easy to implement.

Solving fMLLR

If we take the derivative, we now get

$$\sum_{k=1}^{K}\sum_{t=1}^{N} C_t(k)[\mathbf{r}/\mathbf{r}^T\mathbf{w}' - \mathbf{O}_t(\mathbf{O}_t^T\mathbf{w}' - \mu_k)^2/2\sigma_k^2]$$

which can be rewritten as

$$\beta\mathbf{r}/\mathbf{r}^T\mathbf{w}' - [\sum_{k=1}^{K} 1/\sigma_k^2 \sum_{t=1}^{T} C_t(k)\mathbf{O}_t\mathbf{O}_t^T]\mathbf{w}' + \sum_{k=1}^{K}\mu_k/\sigma_k^2\sum_{t=1}^{T} C_t(k)\mathbf{O}_t$$

or

$$\beta\mathbf{r}/\mathbf{r}^T\mathbf{w}' - G\mathbf{w}' + \mathbf{s} = 0$$

collecting terms we can rewrite this as

$$\mathbf{w}' = G^{-1}(\beta/\mathbf{r}^T\mathbf{w}' + \mathbf{s})$$

if we premultiply by $\mathbf{r}^T$ we get

$$\mathbf{r}^T\mathbf{w}' = \mathbf{r}^T G^{-1}(\beta/\mathbf{r}^T\mathbf{w}' + \mathbf{s})$$

let $\alpha = \mathbf{r}^T\mathbf{w}'$ then we can write

$$\alpha = \mathbf{r}^T G^{-1}(\beta/\alpha + \mathbf{s})$$

one can then solve for α and $\mathbf{w}'$ and pick the value of α that maximizes $Q(\mathbf{w}, \mathbf{w}')$ The details on how to do this for the vector observations are given in the paper in the readings ("Maximum Likelihood Linear Transformations for HMM-Based Speech Recognition", Mark Gales, Computer Speech and Language 1998 Volume 12).

Performance of MLLR and fMLLR

	Test1	Test2
BASE	9.57	9.20
MLLR	8.39	8.21
fMLLR	9.07	7.97
SAT	8.26	7.26

Task is Broadcast News with a 65K vocabulary.

SAT refers to "Speaker Adaptive Training". In SAT, a transform is computed for each speaker during test and training; it is a very common training technique in ASR today.

MLLR and MAP - Performance

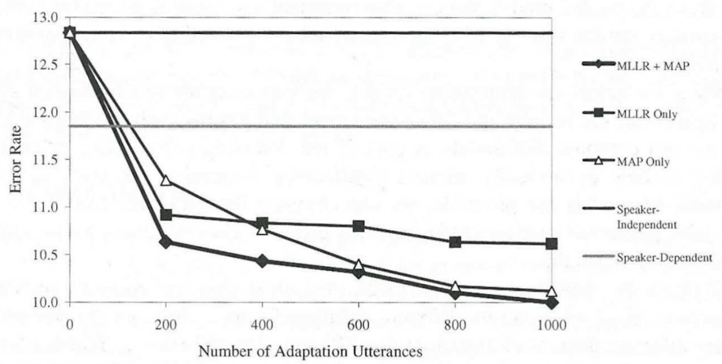


Figure 9.11 Comparison of Whisper with MLLR, MAP, and combined MLLR and MAP. The error rate is shown for a different amount of adaptation data. The speaker-dependent and speaker-independent models are also included. The speaker-dependent model was trained with 1000 sentences.

MLLR - Comments on Noise Immunity Performance

Last but not least, one can also apply MLLR and fMLLR as a noise compensation scheme in "HIGH-PERFORMANCE HMM ADAPTATION WITH JOINT COMPENSATION OF ADDITIVE AND CONVOLUTIVE DISTORTIONS VIA VECTOR TAYLOR SERIES Jinyu Li1, Li Deng, Dong Yu, Yifan Gong, and Alex Acero" presented at ASRU 2007 in Japan, it is claimed that MLLR/fMLLR alone is inferior to schemes that use the E-M algorithm to directly estimate PMC-like noise compensation model based parameters at very low SNRs but comprehensive comparisons across all SNRs were not provided.

COURSE FEEDBACK

- Was this lecture mostly clear or unclear? What was the muddiest topic?
- Other feedback (pace, content, atmosphere)?