ELEN E6884 - Topics in Signal Processing
Topic: Speech Recognition
Lecture 10

Stanley F. Chen, Ellen Eide, and Michael A. Picheny

IBM T.J. Watson Research Center

Yorktown Heights, NY, USA

stanchen@us.ibm.com, eeide@us.ibm.com,

picheny@us.ibm.com

10 November 2005

Outline of Today's Lecture

- Administrivia
- Robustness Review
- PMC
- Adaptation
- Linear Discriminant Analysis
- Maximum Mutual Information Training
- ROVER
- Consensus Decoding

Administrivia

- main feedback from last lecture
  - pace a little slow?
- Lab 3 not graded yet, will be handed back next week
- Lab 4 out, due Sunday after next

Robustness Review

- Transformation of data to match new environment
- Multi-style training
- Cepstral Mean Removal
- Spectral Subtraction
- Codeword-Dependent Cepstral Normalization

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's just look at the one-dimensional case. Let $X = \ln x, N = \ln n$.

Let us say $X \sim N(\mu_X, \sigma_X^2)$ and $N \sim N(\mu_N, \sigma_N^2)$.

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

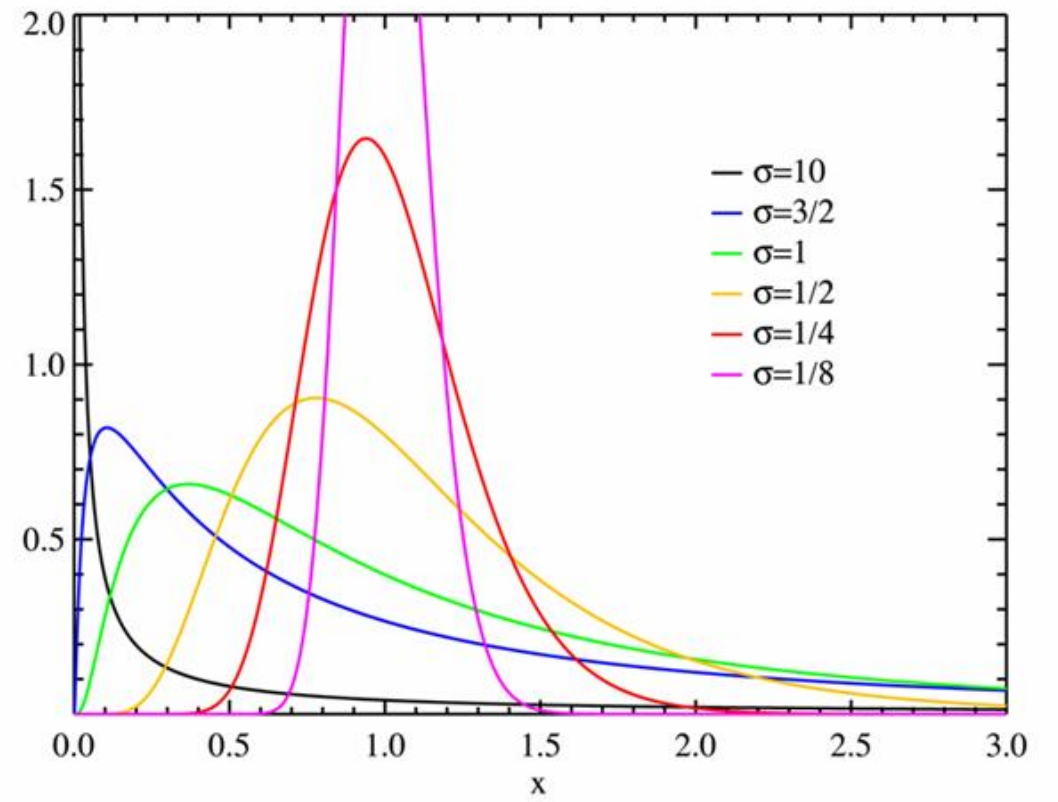
$$p(x) = \frac{1}{x\sigma\sqrt{2\pi}} \exp(-\frac{(\ln x - \mu)^2}{2\sigma^2})$$

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

Let us now assume that in the linear domain

$$y = x + n$$

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

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 (!).

If x and n are uncorrelated, we can write:

$$\mu_y = \mu_x + \mu_n$$

$$\sigma_y^2 = \sigma_x^2 + \sigma_n^2$$

In such a case, $Y = \ln y$ is Gaussian and the mean and variance are given by:

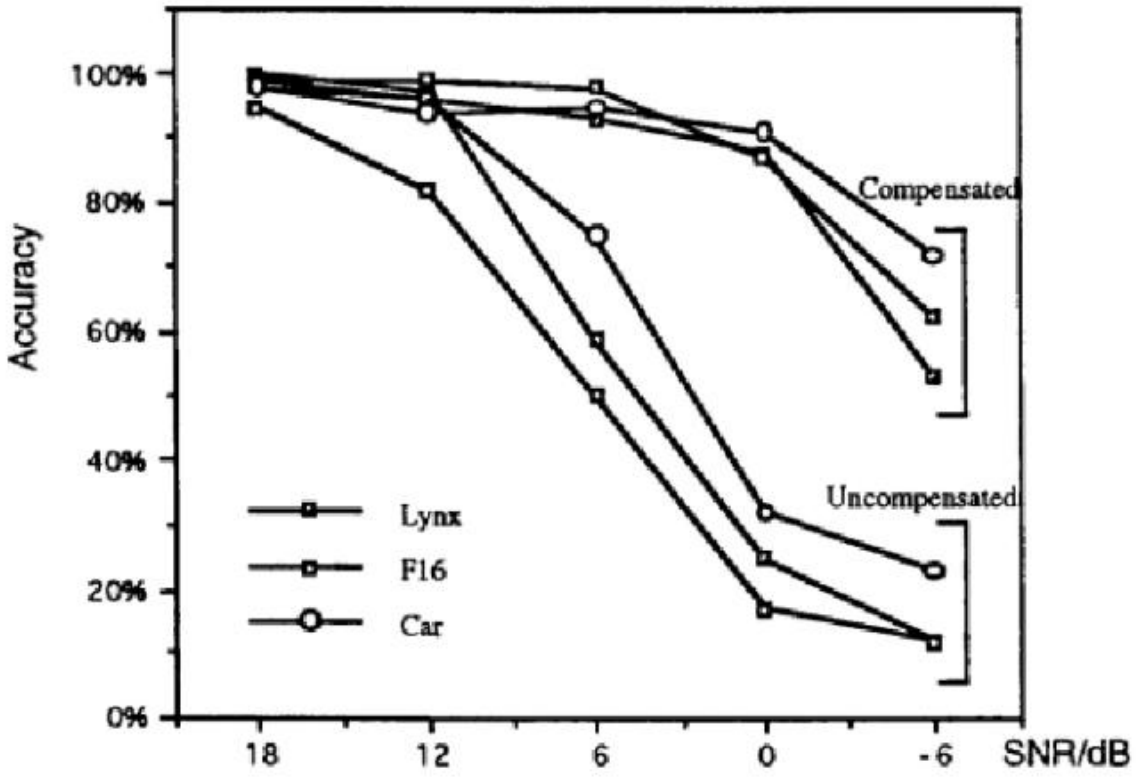
$$\mu_Y = \ln \mu_y - \frac{1}{2} \ln \left[\frac{\sigma_y^2}{\mu_y^2} + 1 \right]$$

$$\sigma_Y^2 = \ln \left[\frac{\sigma_y^2}{\mu_y^2} + 1 \right]$$

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 - 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.



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

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

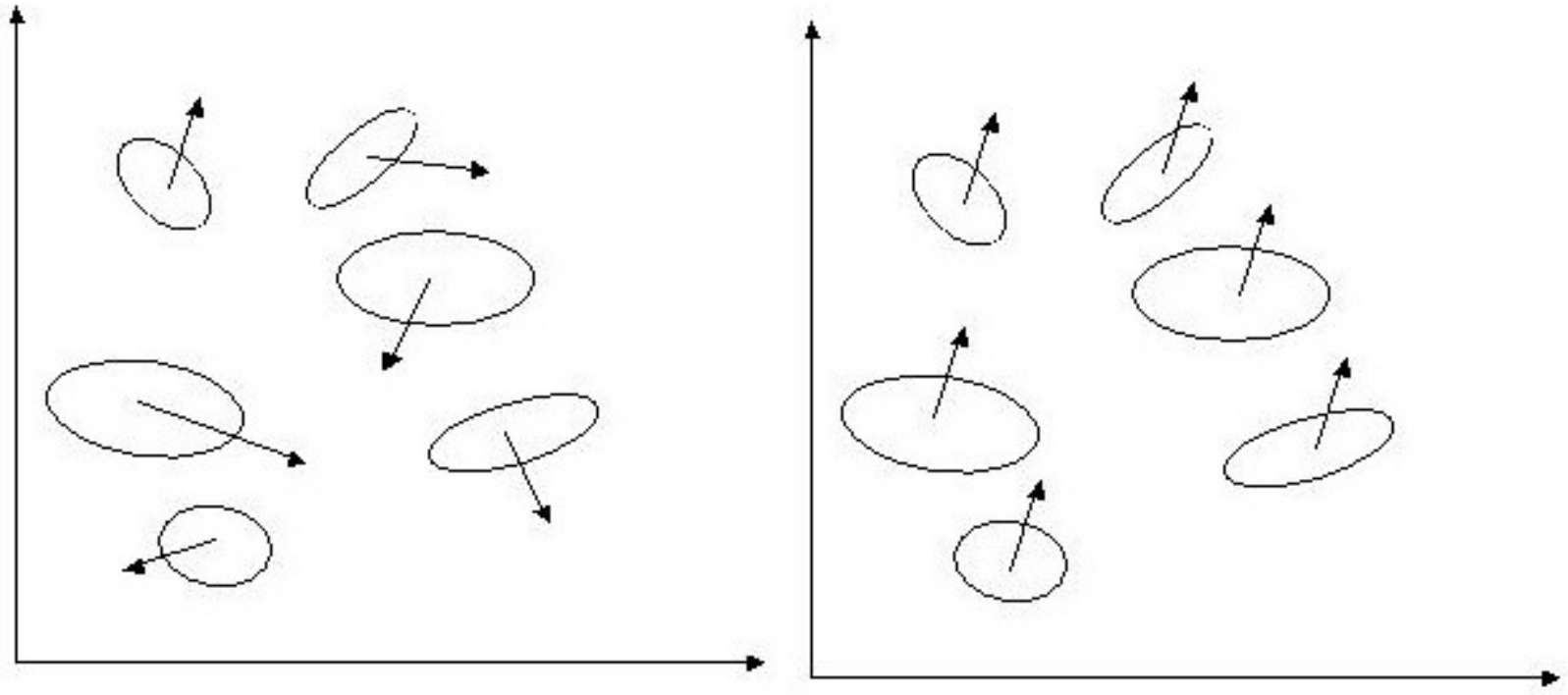
where μ_O is the mean of the data computed using the ML procedure.

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

τ is a balancing parameters that can be tuned to optimize performance on different test domains.

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 ($Ax + b$).



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 μ_k as the column vector

corresponding to $(\mu_k, 1)$. In such a case we can write the overall likelihood as

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

To maximize the likelihood of this expression we take the derivative with respect to $\mathbf{w}$ and solve the resultant linear equations. In actual speech recognition systems, the observations are vectors, not scalars, so the transform to be estimated is of the form

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

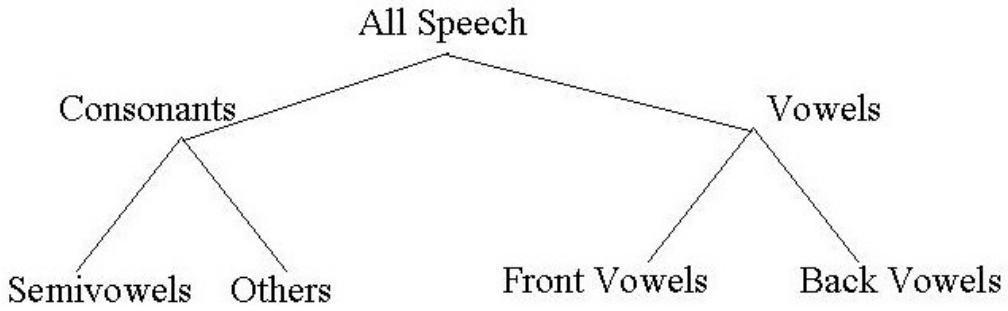
where $\mathbf{A}$ is a matrix and $\mathbf{b}$ is a vector. The resultant MLLR equations are somewhat more complex but follow the same basic form. We refer you to the readings for the actual formulas.

MLLR - Additional Considerations

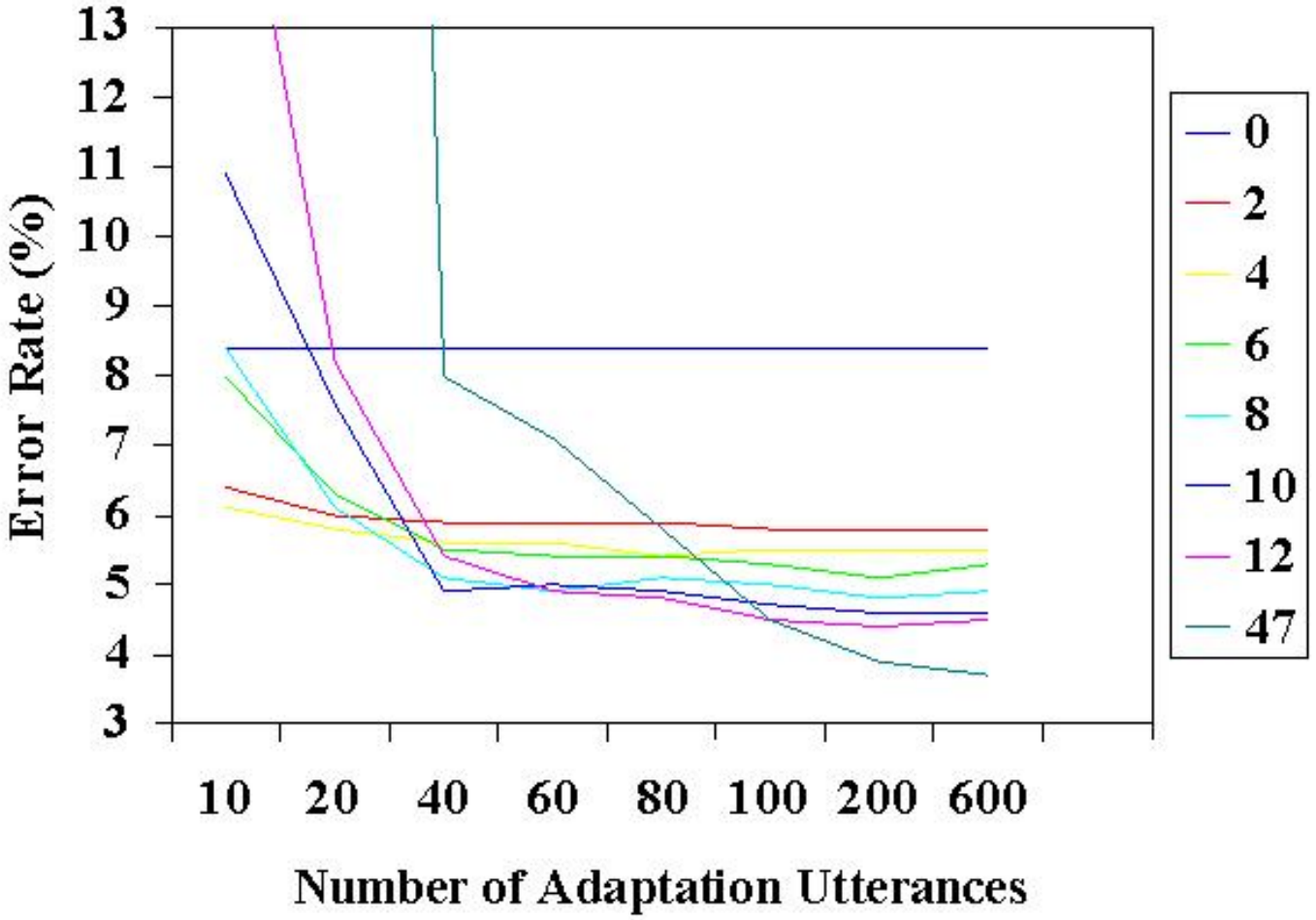
Since the typical parameter vector being processed is 39 dimensional (13 cepstral parameters, and the associated deltas and double-deltas) the number of matrix parameters to be estimated is roughly 1600. As a rule of thumb, if one frame of data gives you enough information to estimate one parameter, 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



MLLR and MAP - Performance

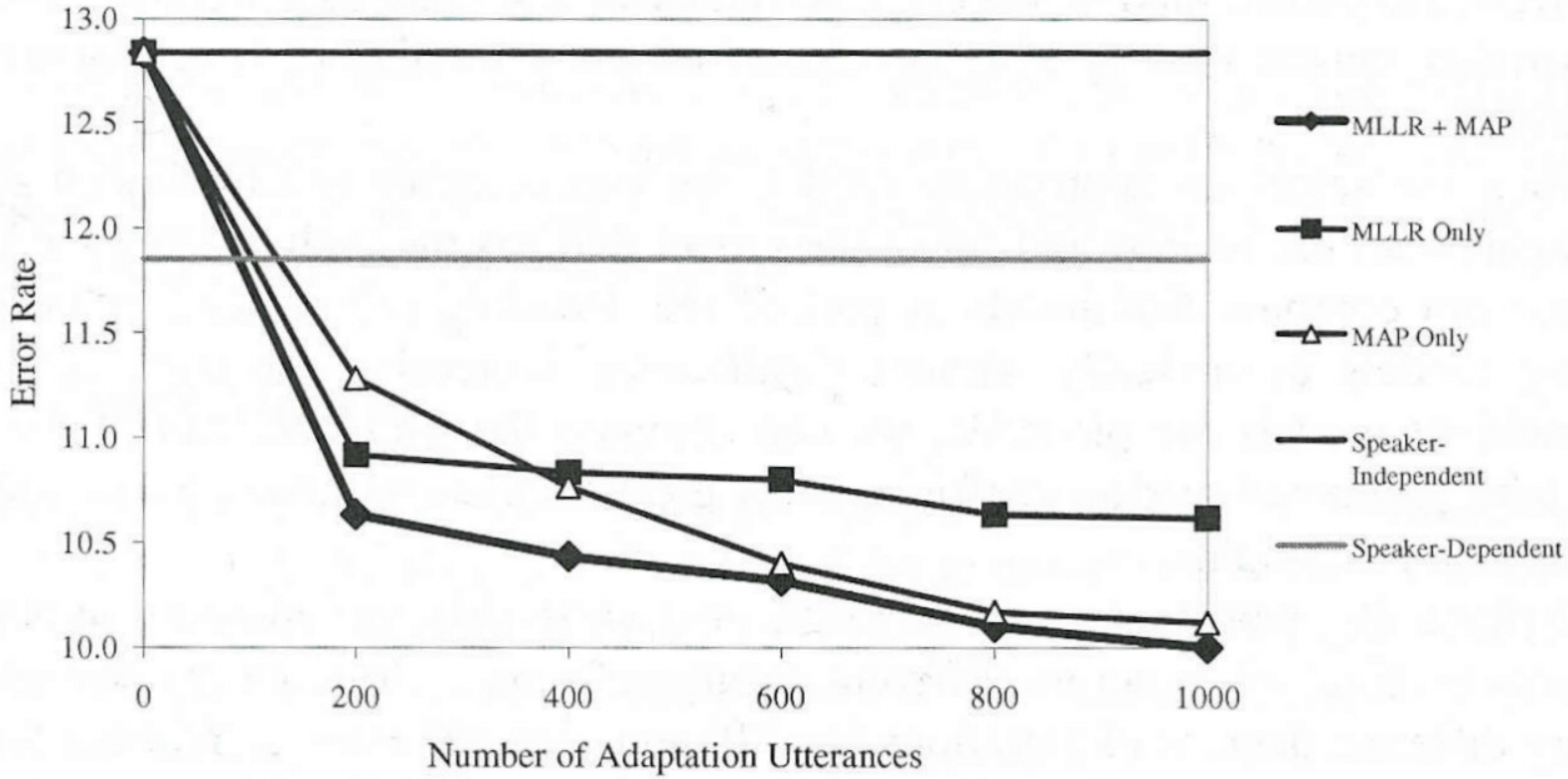
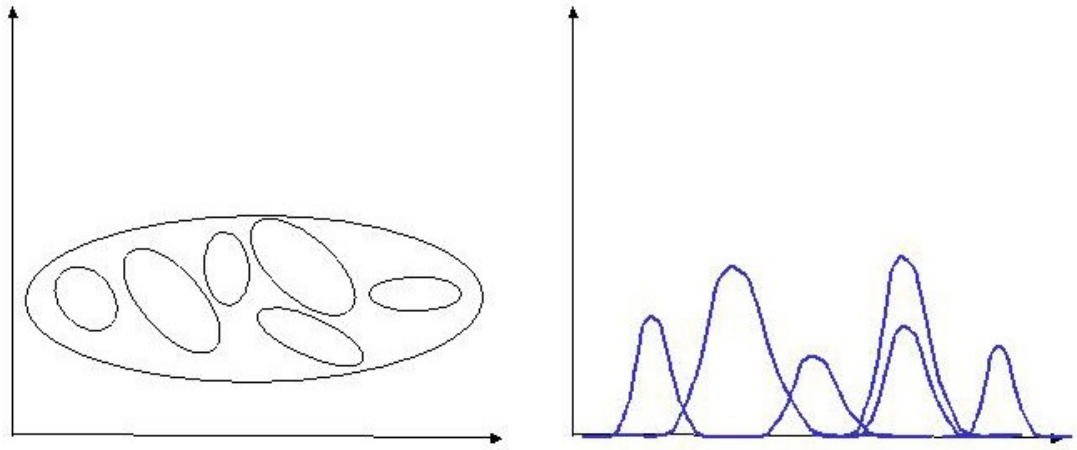


Figure 9.11 Comparison of Whisper with MLLR, MAP, and combined MLLR and MAP. The error rate is shown for a different amount of adaptation data. The speaker-dependent and speaker-independent models are also included. The speaker-dependent model was trained with 1000 sentences.

Linear Discriminant Analysis

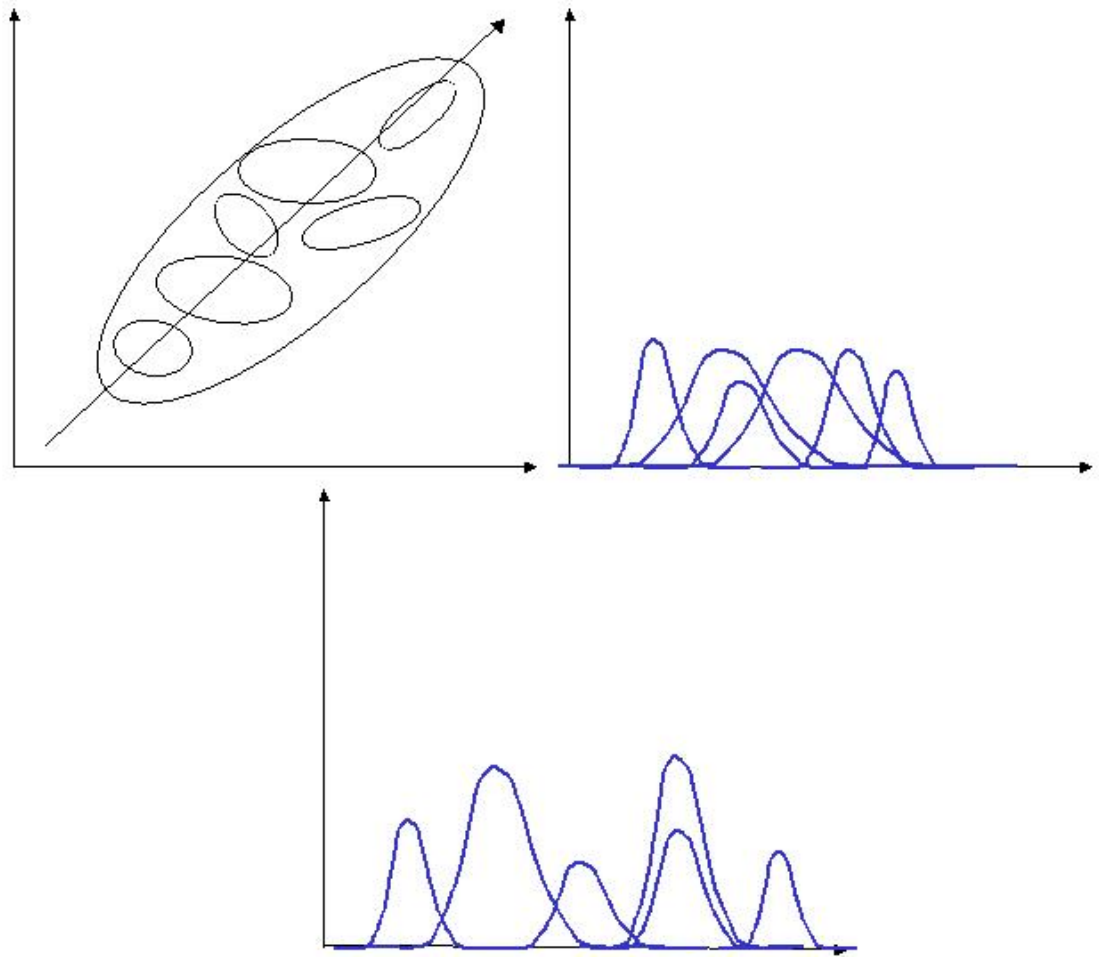
A way to achieve robustness is to extract features that emphasize sound discriminability and ignore irrelevant sources of information. LDA tries to achieve this via a linear transform of the feature data.

If the main sources of class variation lie along the coordinate axes there is no need to do anything even if assuming a diagonal covariance matrix (as in most HMM models):



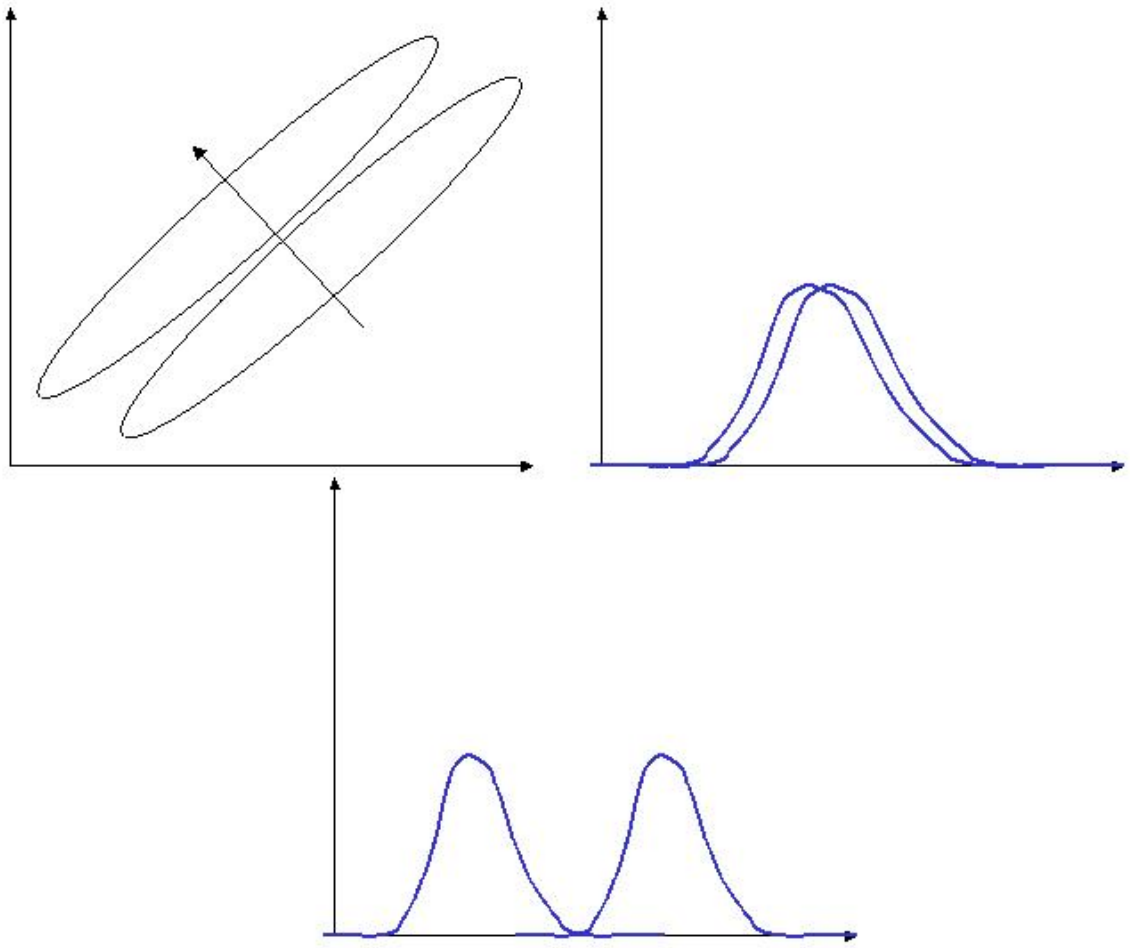
Principle Component Analysis-Motivation

If the main sources of class variation lie along the main source of variation we may want to rotate the coordinate axis (if using diagonal covariances):



Linear Discriminant Analysis - Motivation

If the main sources of class variation do NOT lie along the main source of variation we need to find the best directions:



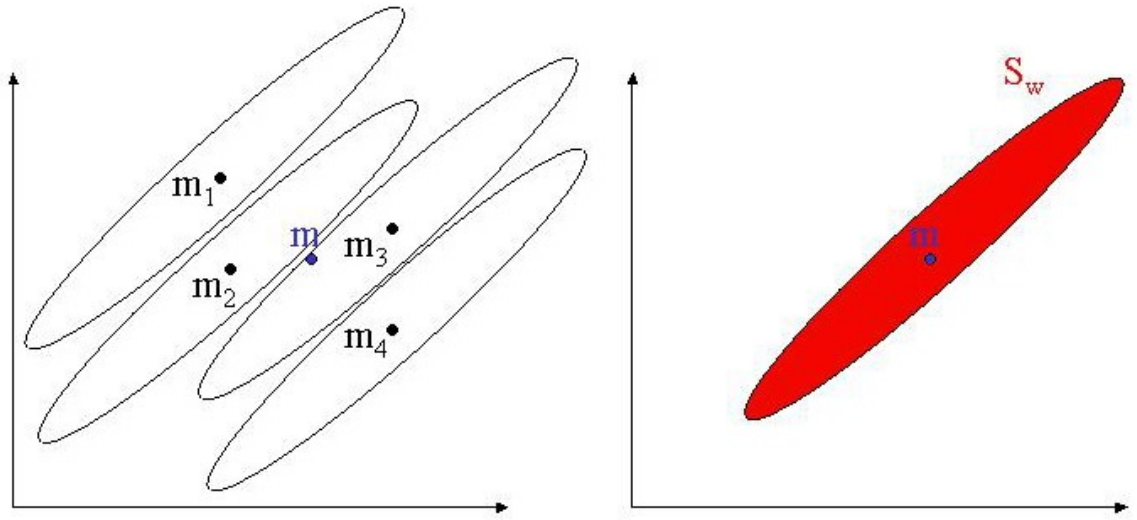
Linear Discriminant Analysis - Derivation

Let us say we have vectors corresponding to c classes of data. We can define a set of covariance matrices as

$$\mathbf{S}_i = \sum_{x \in \mathcal{D}_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^T$$

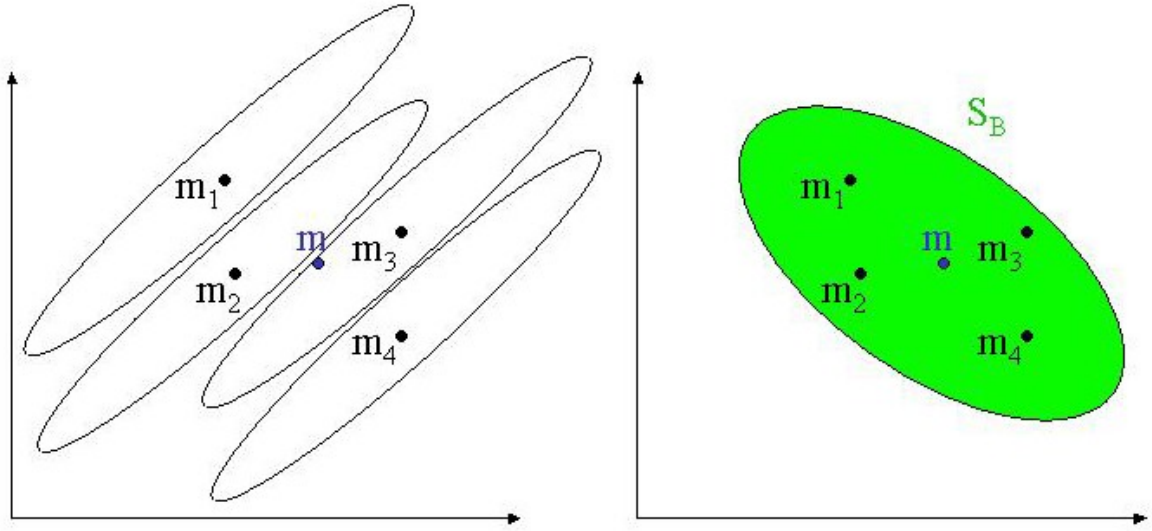
where $\mathbf{m}_i$ is the mean of class i. In this case we can define the within-class covariance (essentially the average covariance across the classes relative to the mean of each class) as just:

$$\mathbf{S}_W = \sum_{i=1}^{c} \mathbf{S}_i$$



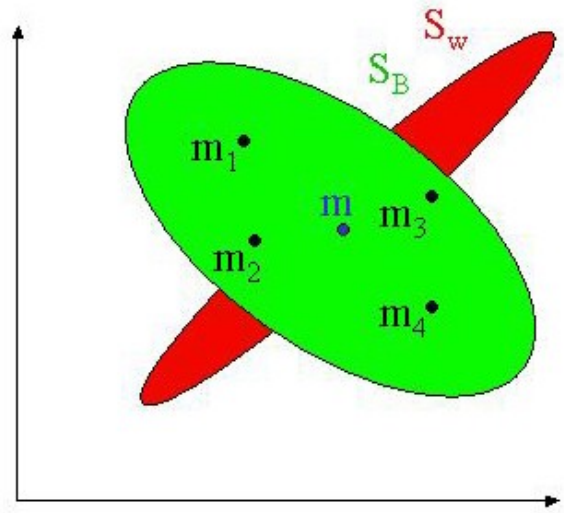
Another useful covariance matrix is the between class covariance matrix, defined as

$$\mathbf{S}_B = \sum_{i=1}^{c} (\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^T$$



We would like to determine a set of projection directions $\mathbf{V}$ such that the classes c are maximally discriminable in the new coordinate space given by

$$\tilde{\mathbf{x}} = \mathbf{V}\mathbf{x}$$



A reasonable measure of discriminability is the ratio of the volumes represented by the covariance matrices. Since the determinant of a matrix is a measure of the corresponding volume, we can use the ratio of determinants as a measure:

$$J = \frac{|\mathbf{S}_B|}{|\mathbf{S}_W|}$$

So we want to find a set of directions that maximize this expression. In the new space, we can write the above expression

as:

$$\begin{aligned}\tilde{\mathbf{S}}_B &= \sum_{i=1}^{c}(\tilde{\mathbf{m}}_i - \tilde{\mathbf{m}})(\tilde{\mathbf{m}}_i - \tilde{\mathbf{m}})^T \\ &= \sum_{i=1}^{c}\mathbf{V}(\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^T\mathbf{V}^T \\ &= \mathbf{V}\mathbf{S}_B\mathbf{V}^T\end{aligned}$$

and similarly for $\mathbf{S}_W$ so the discriminability measure becomes

$$J(\mathbf{V}) = \frac{|\mathbf{V}\mathbf{S}_B\mathbf{V}^T|}{|\mathbf{V}\mathbf{S}_W\mathbf{V}^T|}$$

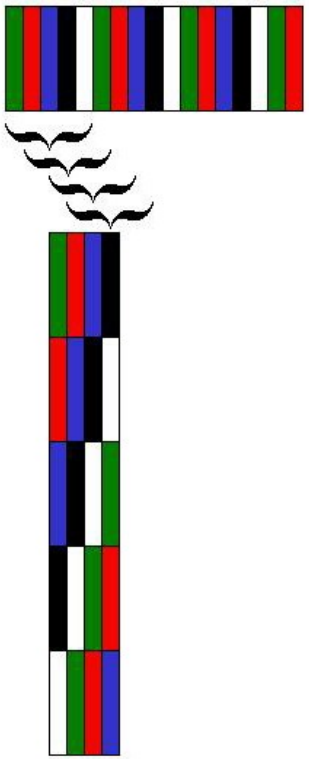
With a little bit of manipulation, it turns out that the solution are the eigenvectors of the matrix

$$\mathbf{S}_W^{-1}\mathbf{S}_B$$

which can be generated by most common mathematical packages.

Linear Discriminant Analysis in Speech Recognition

- Speech recognition training data are aligned against the underlying words using the Viterbi alignment algorithm described in Lecture 4.
- Using this alignment, each cepstral vector is tagged with a different phone or sub-phone. For English this typically results in a set of 156 (52x3) classes.
- For each time t the cepstral vector $\mathbf{x}_t$ is spliced together with $N/2$ vectors on the left and right to form a "supervector" of N cepstral vectors. (N is typically 5-9 frames.) Call this "supervector" $\mathbf{y}_t$.



- The LDA procedure is applied to the supervectors $\mathbf{y}_t$.
- The top M directions (usually 40-60) are chosen and the supervectors $\mathbf{y}_t$ are projected into this lower dimensional space.
- The recognition system is retrained on these lower dimensional vectors.
- Performance improvements of 10%-15% are typical.

Main Problem with Maximum Likelihood Estimation

The true distribution of speech is (probably) not generated by an HMM, at least not of the type we are currently using. (How might we demonstrate this?)

Therefore, the optimality of the ML estimate is not guaranteed and the parameters estimated may not result in the lowest error rates.

A reasonable criterion is rather than maximizing the likelihood of the data given the model, we try to maximize the a posteriori probability of the model given the data (Why?):

$$\theta_{\text{MAP}} = \arg\max_{\theta} p_{\theta}(S|O)$$

MMI Estimation

Let's look at the previous equation in more detail. It is more convenient to look at the problem as maximizing the logarithm of the a posteriori probability across all the sentences:

$$\begin{aligned}
\theta_{\text{MMI}} &= \arg\max_{\theta} \sum_i \log p_\theta(S_i|\mathbf{O}_i) \\
&= \arg\max_{\theta} \sum_i \log \frac{p_\theta(\mathbf{O}_i|S_i)p(S_i)}{p_\theta(\mathbf{O}_i)} \\
&= \arg\max_{\theta} \sum_i \log \frac{p_\theta(\mathbf{O}_i|S_i)p(S_i)}{\sum_j p_\theta(\mathbf{O}_i|S_i^j)p(S_i^j)}
\end{aligned}$$

where S_i^j refers to the jth possible sentence hypothesis given a set of acoustic observations $\mathbf{O}_i$

MMI Training Algorithm

A big breakthrough in the MMI area occured when it was shown that a forward-backward-like algorithm existed for MMI training [2]. The derivation is complex but the resulting esitmation formulas are surprisingly simple. We will just give the results for the estimation of the means in a Gaussian HMM framework.

The MMI objective function is

$$\sum_i \log \frac{p_\theta(\mathbf{O}_i|S_i)p(S_i)}{\sum_j p_\theta(\mathbf{O}_i|S_i^j)p(S_i^j)}$$

We can view this as comprising two terms, the numerator, and the denominator. We can increase the objective function in two ways:

- Increase the contribution from the numerator term
- Decrease the contribution from the denominator term

Basic idea:

- Collect estimation counts from both the numerator and denominator terms
- Increase the objective function by subtracting the denominator counts from the numerator counts.

More specifically, let:

$$\theta_{mk}^{num} = \sum_{i,t} \mathbf{O}_i(t)\gamma_{mki}^{num}(t)$$

$$\theta_{mk}^{den} = \sum_{i,t} \mathbf{O}_i(t)\gamma_{mki}^{den}(t)$$

where $\gamma_{mki}^{num}(t)$ are the counts for state k, mixture component m, computed from running the forward-backward algorithm on the "correct" sentence S_i and $\gamma_{mki}^{den}(t)$ are the counts computed across all the sentence hypotheses corresponding to S_i The MMI

estimate for μ_{mk} is:

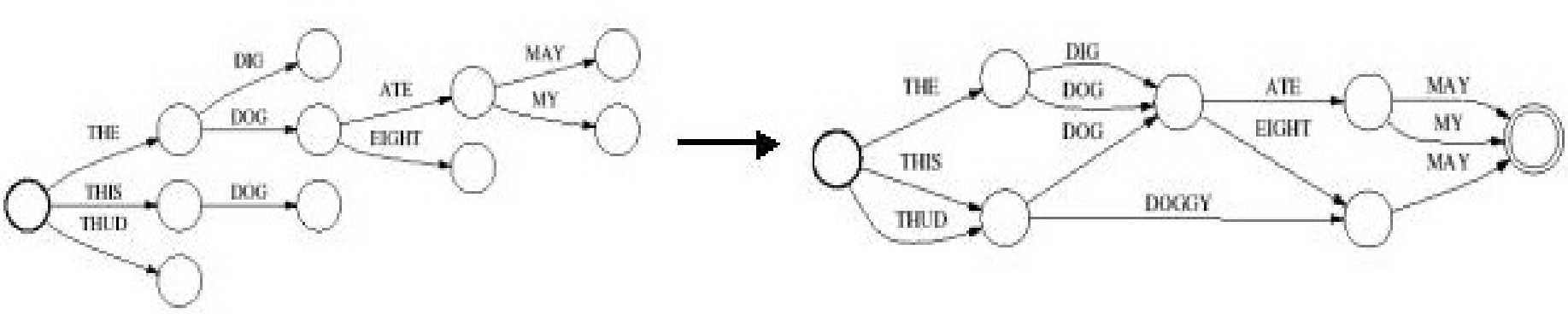
$$\mu_{km} = \frac{\theta_{mk}^{num} - \theta_{mk}^{den} + D_{mk}\mu'_{mk}}{\gamma_{mk}^{num} - \gamma_{mk}^{den} + D_{mk}}$$

The factor D_{mk} is chose large enough to avoid problems with negative count differences. Notice that ignoring the denominator counts results in the normal mean estimate. A similar expression exists for variance estimation.

Computing the Denominator Counts

The major component of the MMI calculation is the computation of the denominator counts. Theoretically, we must compute counts for every possible sentence hypotheis. How can we reduce the amount of computation?

1. From the previous lectures, realize that the set of sentence hypotheses are just captured by a large HMM for the entire sentence:



Counts can be collected on this HMM the same way counts are collected on the HMM representing the sentence corresponding to the correct path.

2. Use a ML decoder to generate a “reasonable” number of sentence hypotheses and then use FST operations such as determinization and minimization to compactify this into an HMM graph (*lattice*).

3. Do not regenerate the lattice after every MMI iteration.

Other Computational Issues

Because we ignore correlation, the likelihood of the data tends to be dominated by a very small number of lattice paths (Why?). To increase the number of confusable paths, the likelihoods are scaled with an exponential constant:

$$\sum_i \log \frac{p_\theta(\mathbf{O}_i|S_i)^\kappa p(S_i)^\kappa}{\sum_j p_\theta(\mathbf{O}_i|S_i^j)^\kappa p(S_i^j)^\kappa}$$

For similar reasons, a weaker language model (unigram) is used to generate the denominator lattice. This also simplifies denominator lattice generation.

Results

MMIE Iteration	%WER	
	eval97sub	eval98
0 (MLE)	44.4	45.6
1	42.4	43.7
1 (3xCHE)	42.0	43.5
2	41.8	42.9
2 (3xCHE)	41.9	42.7

Table 6: Word error rates when using h5train00 training with and without CHE data weighting (3xCHE).

Adaptation	% WER eval98	
	MLE	MMIE
None	44.6	42.5
MLLR	42.1	39.9

Table 8: Effect of MLLR on MLE and MMIE trained models.

Note that results hold up on a variety of other tasks as well.

References

[1] P. Brown (1987) “The Acoustic Modeling Problem in Automatic Speech Recognition”, PhD Thesis, Dept. of Computer Science, Carnegie-Mellon University.

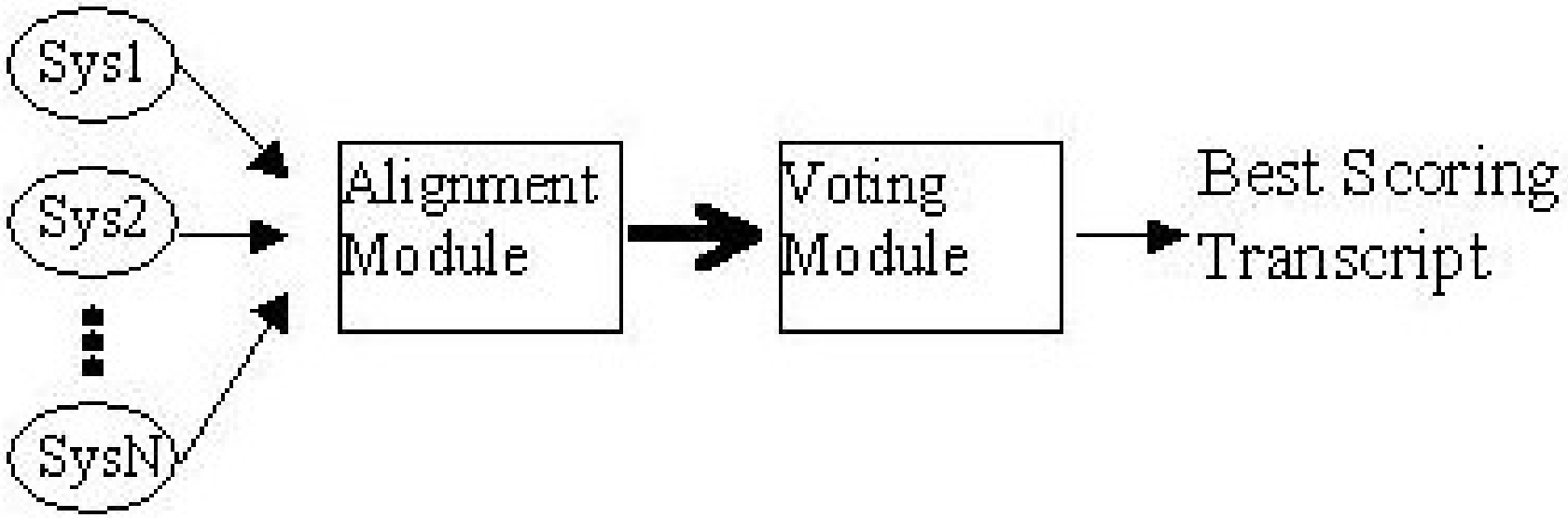
[2] P.S. Gopalakrishnan, D. Kanevsky, A. Nadas, D. Nahamoo (1991) “ An Inequality for Rational Functions with Applications to Some Statistical Modeling Problems”, IEEE Trans. on Acoustics, Speech and Signal Processing, 37(1) 107-113, January 1991

ROVER - Recognizer Output Voting Error Reduction[1]

ROVER is a technique for combining recognizers together to improve recognition accuracy. The concept came from the following set of observations about 7 years ago:

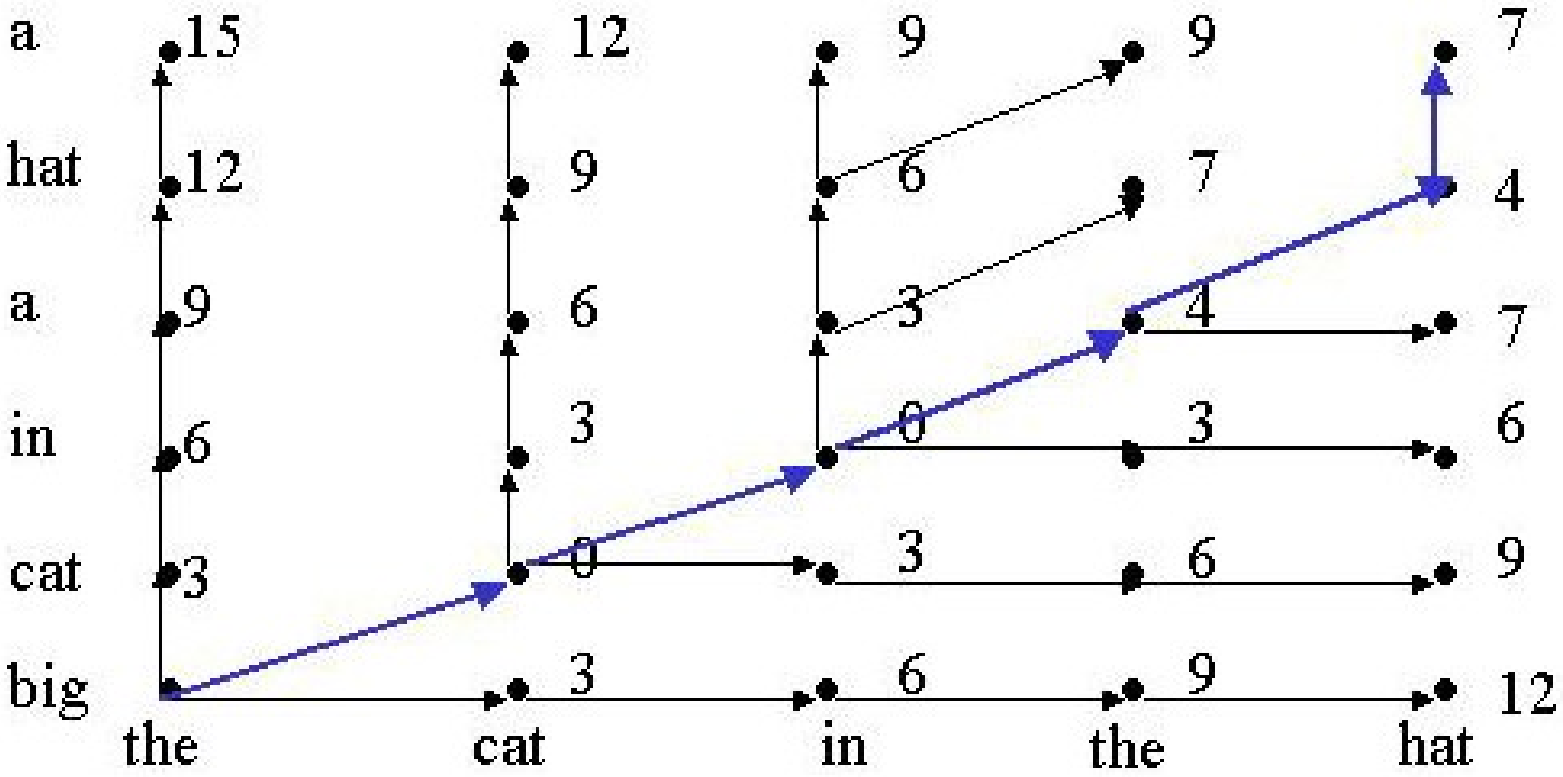
- Compare errors of recognizers from two different sites
- Error rate performance similar - 44.9% vs 45.1%
- Out of 5919 total errors, 738 are errors for only recognizer A and 755 for only recognizer B
- Suggests that some sort of voting process across recognizers might reduce the overall error rate

ROVER - Basic Architecture



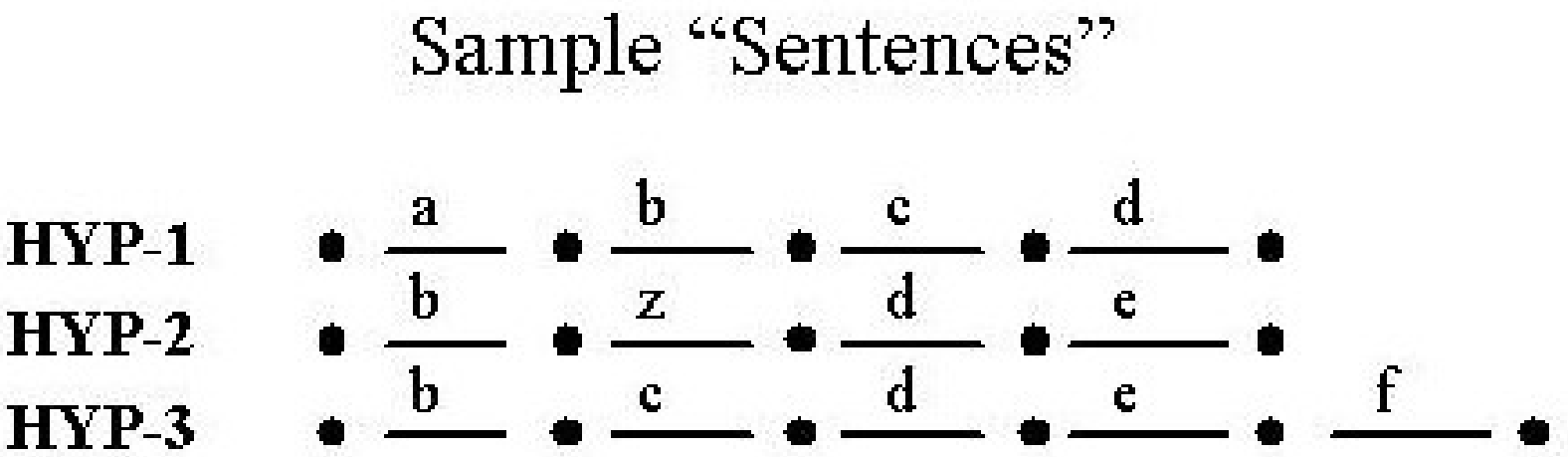
- Systems may come from multiple sites
- Can be a single site with different processing schemes

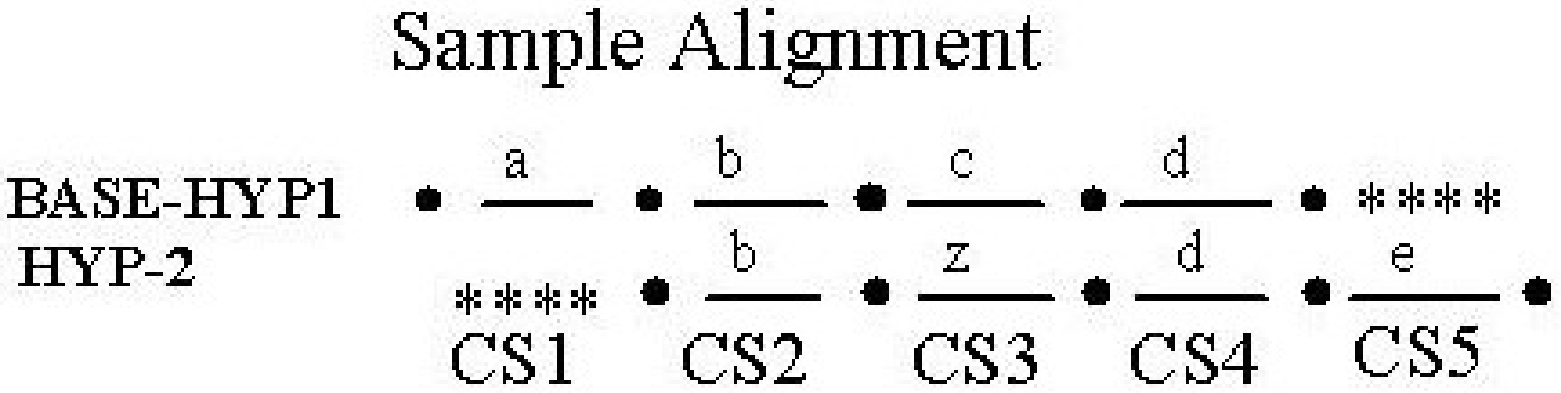
ROVER - Text String Alignment Process



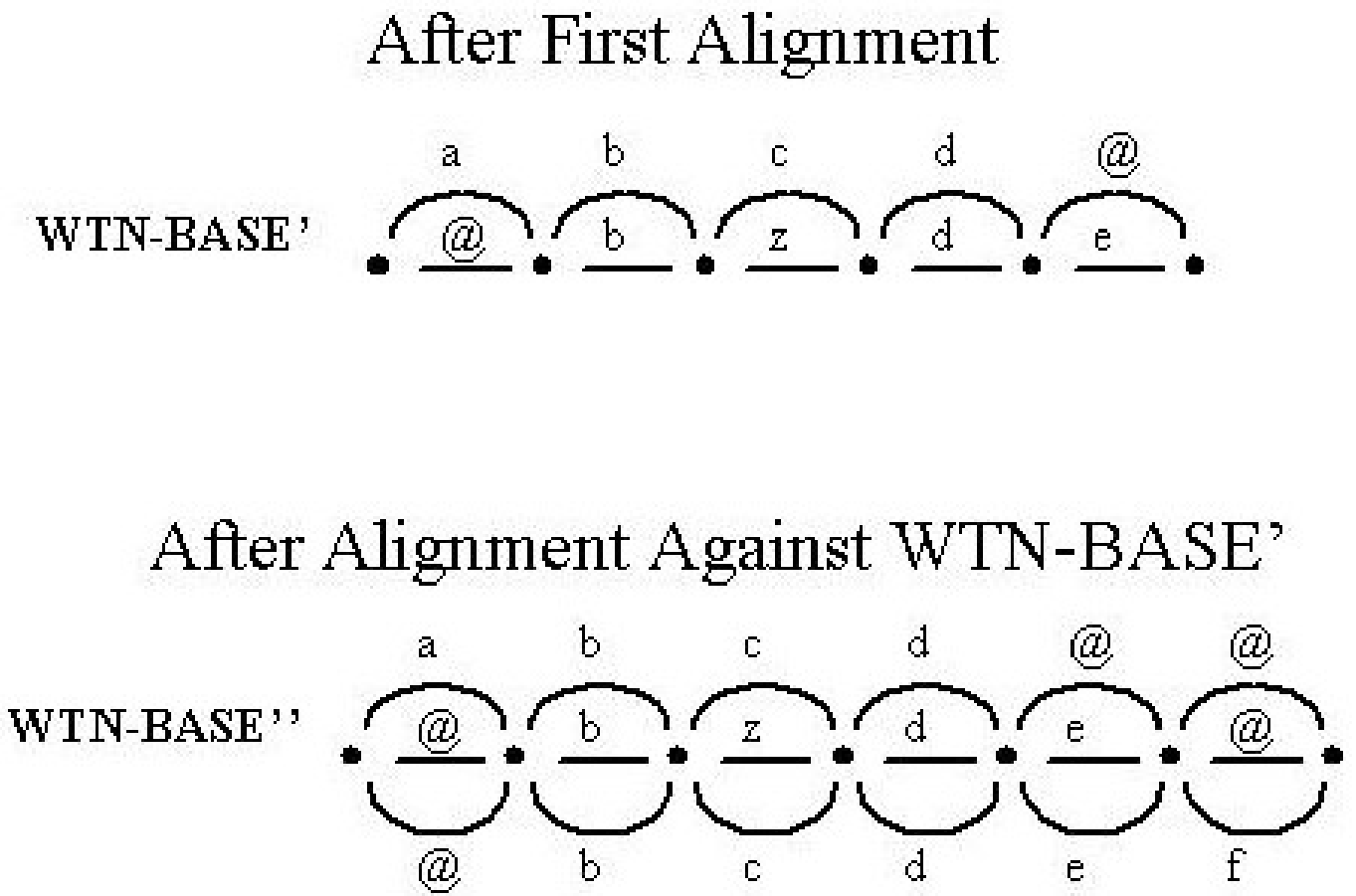
$$score(m,n) = \min \{ score(m-1,n-1) + 4*no\_match(m,n),\ score(m-1,n) + 3,\ score(m, n-1) + 3 \}$$

ROVER - Example





ROVER - Form Confusion Sets



ROVER - Vote

- Main Idea: for each confusion set, take word with highest frequency

SYS1	SYS2	SYS3	SYS4	SYS5	**ROVER**
44.9	45.1	48.7	48.9	50.2	**39.7**

- Improvement very impressive - as large as any significant algorithm advance.

ROVER - Example

Example Confusion Set

bbn1.ctm	there's	a	lot	of	@	like	societies	@	@	ruin	engineers	and	lakes
cmu-isl1.ctm	there's	the	labs	@	@	like	societies	@	for	women	engineers	i	think
cu-htk2.ctm	there's	the	last	@	@	like	societies	@	true	of	engineers	and	like
dragon1.ctm	was	@	alive	@	the	legal	society	is	for	women	engineers	and	like
sri1.ctm	there's	a	lot	of	@	like	society's	@	@	through	engineers	@	like

- Error not guaranteed to be reduced.
- Sensitive to initial choice of base system used for alignment - typically take the best system.

ROVER - As a Function of Number of Systems [2]

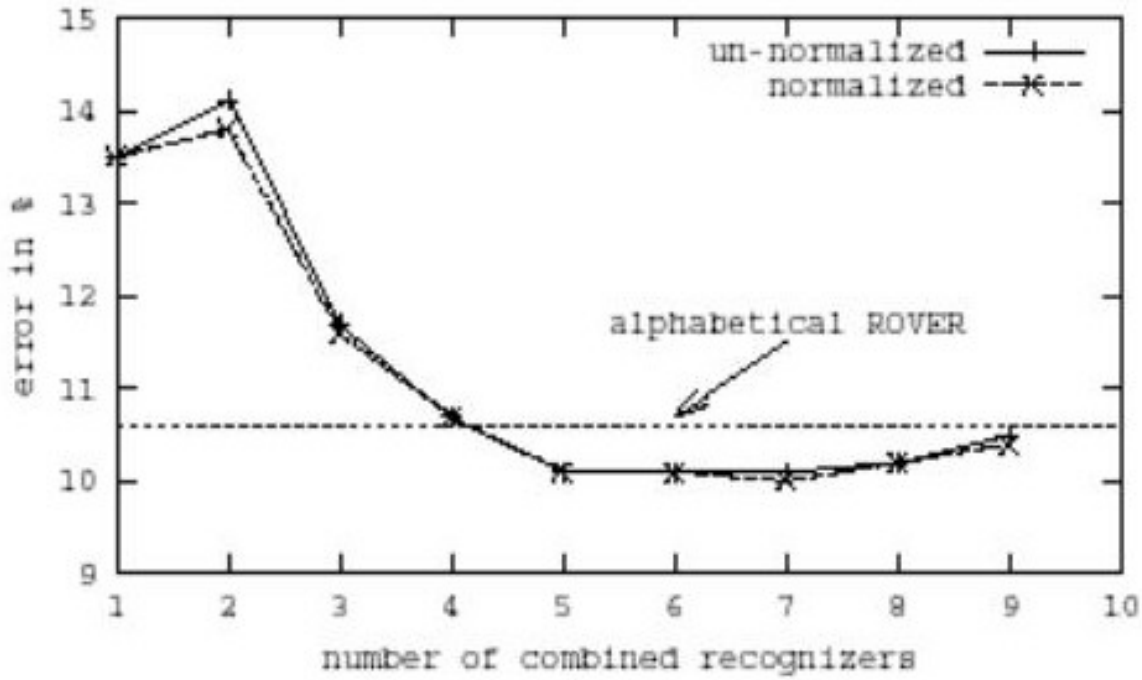


Figure 1: 1998 Broadcast News word error rates in function of the number of combined systems (individual error ranked order).

- Alphabetical: take systems in alphabetical order.
- Curves ordered by error rate.
- Note error actually goes up slightly with 9 systems

ROVER - Types of Systems to Combine

- ML and MMI
- Varying amount of acoustic context in pronunciation models (Triphone, Quinphone)
- Different lexicons
- Different signal processing schemes (MFCC, PLP)
- Anything else you can think of!

Rover provides an excellent way to achieve cross-site collaboration and synergy in a relatively painless fashion.

References

[1] J. Fiscus (1997) “A Post-Processing System to Yield Reduced Error Rates”, IEEE Workshop on Automatic Speech Recognition and Understanding, Santa Barbara, CA

[2] H. Schwenk and J.L. Gauvain (2000) “Combining Multiple Speech Recognizers using Voting and Language Model Information” ICSLP 2000, Beijing II pp. 915-918

Consensus Decoding[1] - Introduction

Problem

- Standard SR evaluation procedure is word-based
- Standard hypothesis scoring functions are sentence-based

Goal

- Explicitly minimize word error metric:

$$\widehat{W} = \arg\min_{W} E_{P(R|A)}[WE(W,R)] = \arg\min_{W} \sum_{R} P(R|A)\, WE(W,R)$$

- For each candidate word, sum the word posteriors and pick the word with the highest posterior probability.

Consensus Decoding - Motivation

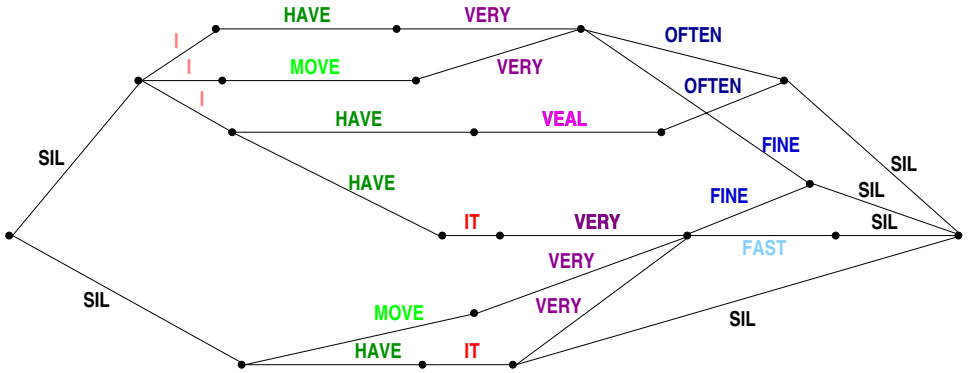
TABLE I: Example illustrating the difference between sentence and word error measures.

Hypothesis (H)							
w_1	w_2	w_3	$P(H\|A)$	$P(w_1\|A)$	$P(w_2\|A)$	$P(w_3\|A)$	E[correct]
I	DO	INSIDE	0·16	0·34	0·29	0·16	0·79
I	DO	FINE	0·13	0·34	0·29	0·28	0·91
BY	DOING	FINE	0·11	0·45	0·49	0·28	1·22
BY	DOING	WELL	0·11	0·45	0·49	0·11	1·05
BY	DOING	SIGHT	0·10	0·45	0·49	0·10	1·04
BY	DOING	BYE	0·07	0·45	0·49	0·07	1·01
BY	DOING	THOUGHT	0·05	0·45	0·49	0·07	0·99
I	DOING	FINE	0·04	0·34	0·49	0·28	1·11
I	DON'T	BUY	0·01	0·34	0·01	0·01	0·36
BY	DOING	FUN	0·01	0·45	0·49	0·01	0·95

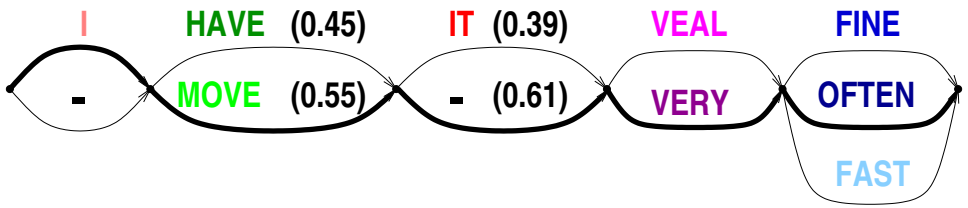
- Original work was done off N-best lists
- Lattices much more compact and have lower oracle error rates

Obtaining the Consensus Hypothesis

Input:



Output:



Consensus Decoding on Broadcast News

	Word Error Rate (%)							
	Avg	F0	F1	F2	F3	F4	F5	FX
C-	16.5	8.3	18.6	27.9	26.2	10.7	22.4	23.7
C+	16.0	8.5	18.1	26.1	25.8	10.5	18.8	22.5

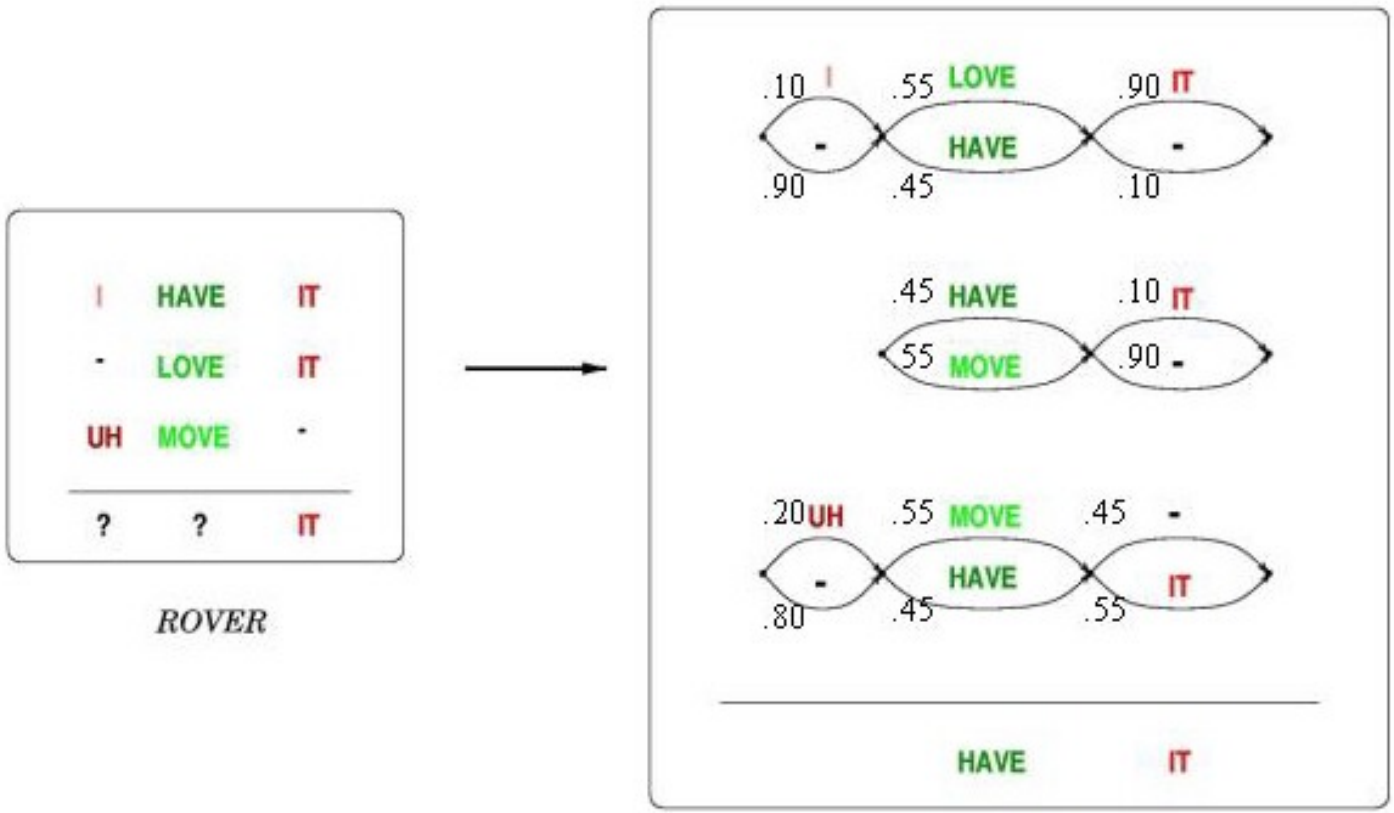
	Word Error Rate (%)							
	Avg	F0	F1	F2	F3	F4	F5	FX
C-	14.0	8.6	15.8	19.4	15.3	16.0	5.7	44.8
C+	13.6	8.5	15.7	18.6	14.6	15.3	5.7	41.1

System Combination Using Confusion Networks

If we have multiple systems, we can combine the concept of ROVER with confusion networks as follows:

- Use the same process as ROVER to align confusion networks
- Take the overall confusion network and add the posterior probabilities for each word.
- For each confusion set, pick the word with the highest summed posteriors.

System Combination Using Confusion Networks



(b) System Combination

Results of Confusion-Network-Based System Combination

		eval98	eval00	
		WER	WER	NCE
single system	Quin MMIE	36.0	26.5	0.284
2-way (MMIE)	Rover conf	35.6	25.7	0.267
	CNC	35.2	25.6	0.278
4-way	Rover vote	35.8	25.9	
	Rover conf	35.4	25.5	0.262
	CNC	35.0	25.4	0.271

Table 4: System Combination Results

	Word Error Rate (%)				
Meetings task	MMI PLP	ML max PLP	ML mean PLP	ML max MFCC	SPAM
MAP	36.5	36.4	38.9	35.9	36.3
CONS	**34.8**	**34.6**	**38.8**	**34.7**	**35.3**
Δ WER	-1.7	-1.8	-1.1	-1.2	-1.0

WER Extended ROVER : **33.6%**

References

[1] L. Mangu, E. Brill and A. Stolcke (2000) “Finding consensus in speech recognition: word error minimization and other applications of confusion networks”, Computer Speech and Language 14(4), 373-400.

COURSE FEEDBACK

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