EECS E6870 - Speech Recognition Lecture 11

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

24 November 2009

Outline of Today's Lecture

- Administrivia
- Linear Discriminant Analysis
- Maximum Mutual Information Training
- ROVER
- Consensus Decoding

Administrivia

See

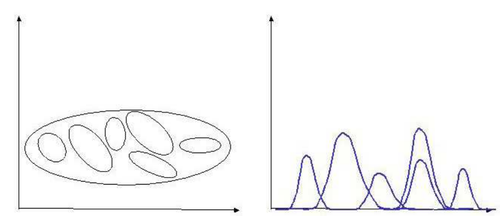
http://www.ee.columbia.edu/ stanchen/fall09/e6870/readings/project_f09.html

for suggested readings and presentation guidelines for final project.

Linear Discriminant Analysis

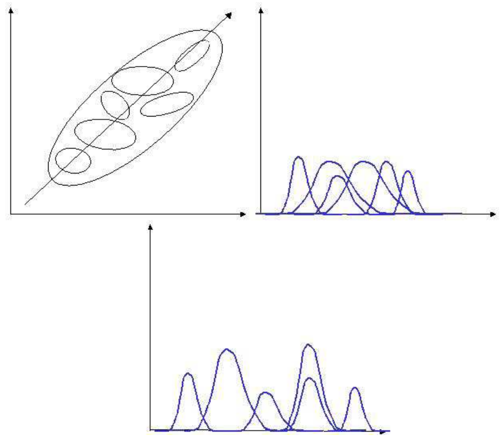
A way to achieve robustness is to extract features that emphasize sound discriminability and ignore irrelevant sources of information. LDA tries to achieve this via a linear transform of the feature data.

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):



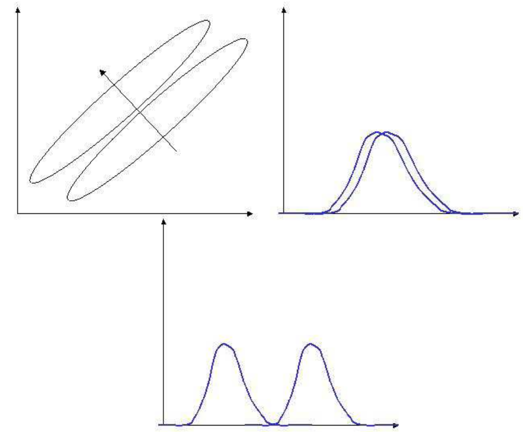
Principle Component Analysis-Motivation

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:



Eigenvectors and Eigenvalues

A key concept in feature selection are the eigenvalues and eigenvectors of a matrix.

The eigenvalues and eigenvectors of a matrix are defined by the following matrix equation:

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$$

For a given matrix $\mathbf{A}$ the eigenvectors are defined as those vectors $\mathbf{x}$ for which the above statement is true. Each eigenvector has an associated eigenvalue, λ. To solve this equation, we can rewrite it as

$$(\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = 0$$

If $\mathbf{x}$is non-zero, the only way this equation can be solved is if the determinant of the matrix $(\mathbf{A} - \lambda\mathbf{I})$ is zero. The determinant of

this matrix is a polynomial (called the *characteristic polynomial*) $p(\lambda)$. The roots of this polynomial will be the eigenvalues of $\mathbf{A}$. For example, let us say

$$\mathbf{A} = \begin{bmatrix} 2 & -4 \\ -1 & -1 \end{bmatrix}.$$

In such a case,

$$\begin{aligned} p(\lambda) &= \begin{vmatrix} 2-\lambda & -4 \\ -1 & -1-\lambda \end{vmatrix} \\ &= (2-\lambda)(-1-\lambda) - (-4)(-1) \\ &= \lambda^2 - \lambda - 6 \\ &= (\lambda - 3)(\lambda + 2) \end{aligned}$$

Therefore, $\lambda_1 = 3$ and $\lambda_2 = -2$ are the eigenvalues of $\mathbf{A}$.

To find the eigenvectors, we simply plug in the eigenvalues into

$(\mathbf{A} - \lambda \mathbf{I})\mathbf{x} = 0$ and solve for $\mathbf{x}$. For example, for $\lambda_1 = 3$ we get

$$\begin{bmatrix} 2-3 & -4 \\ -1 & -1-3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Solving this, we find that $x_1 = -4x_2$, so all the eigenvector corresponding to $\lambda_1 = 3$ is a multiple of $[-4\ 1]^T$. Similarly, we find that the eigenvector corresponding to $\lambda_1 = -2$ is a multiple of $[1\ 1]^T$.

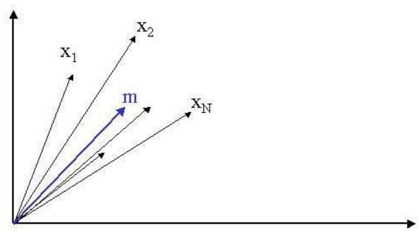
Principle Component Analysis-Derivation

First consider the problem of best representing a set of vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ by a single vector $\mathbf{x}_0$. More specifically let us try to minimize the sum of the squared distances from $\mathbf{x}_0$

$$J_0(\mathbf{x}_0) = \sum_{k=1}^{N} |\mathbf{x}_k - \mathbf{x}_0|^2$$

It is easy to show that the sample mean, $\mathbf{m}$, minimizes J_0, where $\mathbf{m}$ is given by

$$\mathbf{m} = \mathbf{x}_0 = \frac{1}{N} \sum_{k=1}^{N} \mathbf{x}_k$$



Now, let $\mathbf{e}$ be a unit vector in an arbitrary direction. In such a case, we can express a vector $\mathbf{x}$ as

$$\mathbf{x} = \mathbf{m} + a\mathbf{e}$$

For the vectors $\mathbf{x}_k$ we can find a set of a_ks that minimizes the

mean square error:

$$J_1(a_1, a_2, \ldots, a_N, \mathbf{e}) = \sum_{k=1}^{N} |\mathbf{x}_k - (\mathbf{m} + a_k \mathbf{e})|^2$$

If we differentiate the above with respect to a_k we get

$$a_k = \mathbf{e}^T(\mathbf{x}_k - \mathbf{m})$$

i.e. we project $\mathbf{x}_k$ onto the line in the direction of $\mathbf{e}$ that passes through the sample mean $\mathbf{m}$. How do we find the best direction $\mathbf{e}$? If we substitute the above solution for a_k into the formula for the overall mean square error we get after some manipulation:

$$J_1(\mathbf{e}) = -\mathbf{e}^T \mathbf{S} \mathbf{e} + \sum_{k=1}^{N} |\mathbf{x}_k - \mathbf{m}|^2$$

where $\mathbf{S}$ is called the *Scatter* matrix and is given by:

$$\mathbf{S} = \sum_{k=1}^{N} (\mathbf{x}_k - \mathbf{m})(\mathbf{x}_k - \mathbf{m})^T$$

Notice the scatter matrix just looks like N times the sample covariance matrix of the data. To minimize J_1 we want to maximize $\mathbf{e}^T\mathbf{S}\mathbf{e}$ subject to the constraint that $|\mathbf{e}| = \mathbf{e}^T\mathbf{e} = 1$. Using Lagrange multipliers we write

$$u = \mathbf{e}^T\mathbf{S}\mathbf{e} - \lambda\mathbf{e}^T\mathbf{e}$$

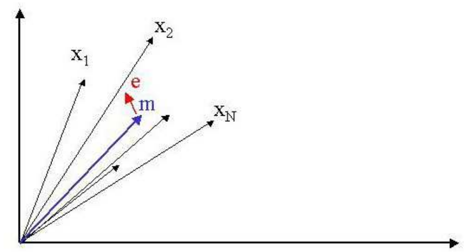
. Differentiating u w.r.t $\mathbf{e}$ and setting to zero we get:

$$2\mathbf{S}\mathbf{e} - 2\lambda\mathbf{e} = 0$$

or

$$\mathbf{S}\mathbf{e} = \lambda\mathbf{e}$$

So to maximize $\mathbf{e}^T\mathbf{S}\mathbf{e}$ we want to select the eigenvector of $\mathbf{S}$ corresponding to the largest eigenvalue of $\mathbf{S}$.



If we now want to find the best d directions, the problem is now to express $\mathbf{x}$ as

$$\mathbf{x} = \mathbf{m} + \sum_{i=1}^{d} a_i\mathbf{e}_i$$

In this case, we can write the mean square error as

$$J_d = \sum_{k=1}^{N} |(\mathbf{m} + \sum_{i=1}^{d} a_{ki}\mathbf{e}_i) - \mathbf{x}_k|^2$$

and it is not hard to show that J_d is minimized when the vectors $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_d$ correspond to the d largest eigenvectors of the scatter matrix $\mathbf{S}$.

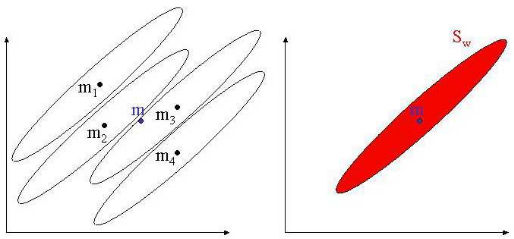
Linear Discriminant Analysis - Derivation

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

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

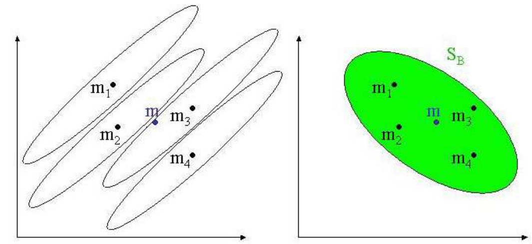
where $\mathbf{m}_i$ is the mean of class i. In this case we can define the within-class scatter (essentially the average scatter across the classes relative to the mean of each class) as just:

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



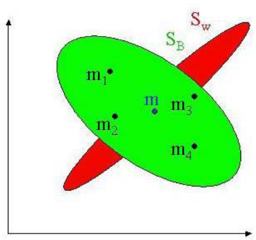
Another useful scatter matrix is the between class scatter matrix, defined as

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



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 scatter 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 similar to that in PCA, it turns out that the solution are the eigenvectors of the matrix

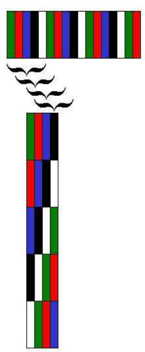
$$\mathbf{S}_W^{-1}\mathbf{S}_B$$

which can be generated by most common mathematical packages.

Linear Discriminant Analysis in Speech Recognition

The most successful uses of LDA in speech recognition are achieved in an interesting fashion.

- 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.

Training via Maximum Mutual Information

The **Fundamental Equation of Speech Recognition** states that

$$p(S|O) = p(O|S)p(S)/P(O)$$

where S is the sentence and O are our observations. We model $p(O|S)$ using Hidden Markov Models (HMMs). The HMMs themselves have a set of parameters θ that are estimated from a set of training data, so it is convenient to write this dependence explicitly: $p_\theta(O|S)$.

We estimate the parameters θ to maximize the likelihood of the training data. Although this seems to make some intuitive sense, is this what we are after?

Not really! (Why?). So then, why is ML estimation a good thing?

Maximum Likelihood Estimation Redux

ML estimation results in a function that allows us to estimate parameters of the desired distribution from observed samples of the distribution. For example, in the Gaussian case:

$$\hat{\mu}_{\text{MLE}} = \frac{1}{n}\sum_{k=1}^{n}\mathbf{x}_k$$

$$\hat{\Sigma}_{\text{MLE}} = \frac{1}{n}\sum_{k=1}^{n}(\mathbf{x}_k - \hat{\mu}_{\text{MLE}})(\mathbf{x}_k - \hat{\mu}_{\text{MLE}})^T$$

Since μ and $\mathbf{\Sigma}$ themselves are computed from the random variables $\mathbf{x}_k$ we can consider them to be random variables as well.

More generally we can consider the estimate of the parameters θ as a random variable. The function that computes this estimate is called an *estimator*.

Any estimator, maximum likelihood or other, since it is a random variable, has a mean and a variance. It can be shown that if

- The sample is actually drawn from the assumed family of distributions
- The family of distributions is well-behaved
- The sample is large enough

then, the maximum likelihood estimator has a Gaussian distribution with the following good properties:

- The mean converges to the true mean of the parameters (consistent)
- The variance has a particular form and is just a function of the true mean of the parameters and the samples (Fisher information)
- No other consistent estimator has a lower variance

This means the ML estimate on the average will produce the closest estimate to the true parameters of the system.

If we assume that the system has its best performance when the parameters match the true parameters, then the ML estimate will, on average, perform as good as or better than any other estimator.

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 $\mathbf{O}_i$

Why is this Called MMI Estimation?

There is a quantity in information theory called the Mutual Information. It is defined as:

$$E\left[\log \frac{p(X,Y)}{p(X)p(Y)}\right]$$

Since $p(S_i)$ does not depend on θ, the term can be dropped from the previous set of equations, in which case the estimation formula looks like the expression for mutual information, above.

When originally derived by Brown[1], the formulation was actually in terms of mutual information, hence the name. However, it is easier to quickly motivate in terms of maximizing the a posteriori probability of the answers.

Comparison to ML Estimation

In ordinary ML estimation, the objective is to find θ :

$$\theta_{\text{ML}} = \arg\max_{\theta} \sum_i \log p_\theta(\mathbf{O}_i|S_i)$$

Therefore, in ML estimation, for each i we only need to make computations over the correct sentence S_i. In MMI estimation, we need to worry about computing quanitities over all possibile sentence hypotheses - a much more computationally intense process.

Another advantage of ML over MMI is that there exists a relatively simple algorithm - the forward-backward, or Baum-Welch, algorithm, for iteratively estimating θ that is guaranteed to converge. When originally formulated, MMI training had to be done by painful gradient search.

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:

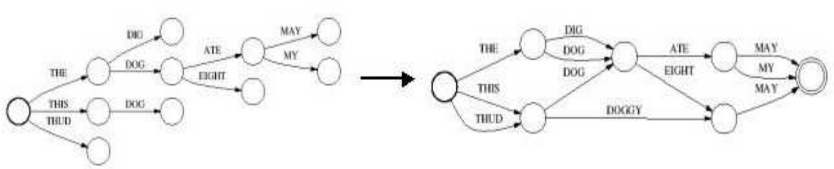
$$\mu_{km} = \frac{\theta_{mk}^{num} - \theta_{mk}^{den} + D_{mk}\mu'_{mk}}{\gamma_{mk}^{num} - \gamma_{mk}^{den} + D_{mk}}$$

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	%WER 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.

Variations and Embellishments

- MPE - Minimum Phone Error
- bMMI - Boosted MMI
- MCE - Minimum Classification Error
- fMPE/fMMI - feature-based MPE and MMI

MPE

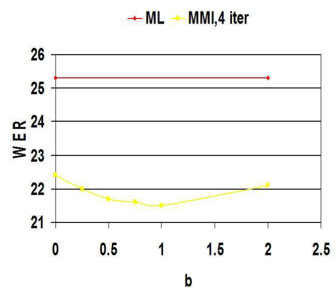
$$\sum_i \frac{\sum_j p_\theta(\mathbf{O}_i|S_j)^\kappa p(S_j)^\kappa A(S_{ref}, S_j)}{\sum_j p_\theta(\mathbf{O}_i|S_i^j)^\kappa p(S_i^j)^\kappa}$$

- $A(S_{ref}, S_j$ is a phone-frame accuracy function. A measures the number of correctly labeled frames in S
- Povey [3] showed this could be optimized in a way similar to that of MMI.
- Usually works somewhat better than MMI itself

bMMI

$$\sum_i \log \frac{p_\theta(\mathbf{O}_i|S_i)^\kappa p(S_i)^\kappa}{\sum_j p_\theta(\mathbf{O}_i|S_i^j)^\kappa p(S_i^j)^\kappa \exp(-bA(S_i^j, S_{ref}))}$$

- A is a phone-frame accuracy function as in MPE.
- Boosts contribution of paths with lower phone error rates.



Various Comparisons

Language	Arabic	English	English	English
Domain	Telephony	News	Telephony	Parliament
Hours	80	50	175	80
ML	43.2	25.3	31.8	8.8
MPE	36.8	19.6	28.6	7.2
bMMI	35.9	18.1	28.3	6.8

MCE

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

where $f(x) = \frac{1}{1+e^{2\rho x}}$

- The sum over competing models explicitly excludes the correct class (unlike the other variations)
- Approximates sentence error rate on training data
- Originally developed for grammar-based applications
- Comparable to MPE, never compared to bMMI

fMPE/fMMI

$$y_t = O_t + Mh_t$$

- h_t are the set of Gaussian likelihoods for frame t. May be clustered into a smaller number of Gaussians, may also be combined across multiple frames.
- The training of M is exceedingly complex involving both the gradients of your favorite objective function with respect to M as well as the model parameters θ with multiple passes through the data.
- Rather amazingly gives significant gains both with and without MMI.

fMPE/fMMI Results

English BN 50 Hours, SI models

	RT03	DEV04f	RT04
ML	17.5	28.7	25.3
fBMMI	13.2	21.8	19.2
fbMMI+ bMMI	12.6	21.1	18.2

Arabic BN 1400 Hours, SAT Models

	DEV07	EVAL07	EVAL06
ML	17.1	19.6	24.9
fMPE	14.3	16.8	22.3
fMPE+ MPE	12.6	14.5	20.1

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

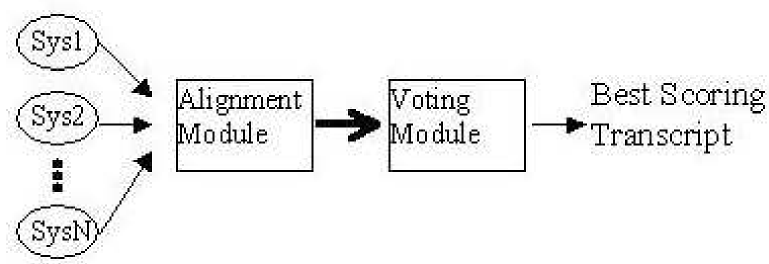
[3] D. Povey and P. Woodland (2002) “Minimum Phone Error and i-smoothing for improved discriminative training”, Proc. ICASSP vol. 1 pp 105-108.

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 11 years ago:

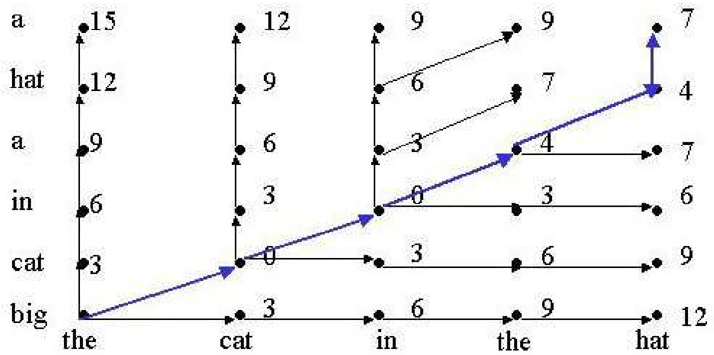
- Compare errors of recognizers from two different sites
- Error rate performance similar - 44.9% vs 45.1%
- Out of 5919 total errors, 738 are errors for only recognizer A and 755 for only recognizer B
- Suggests that some sort of voting process across recognizers might reduce the overall error rate

ROVER - Basic Architecture



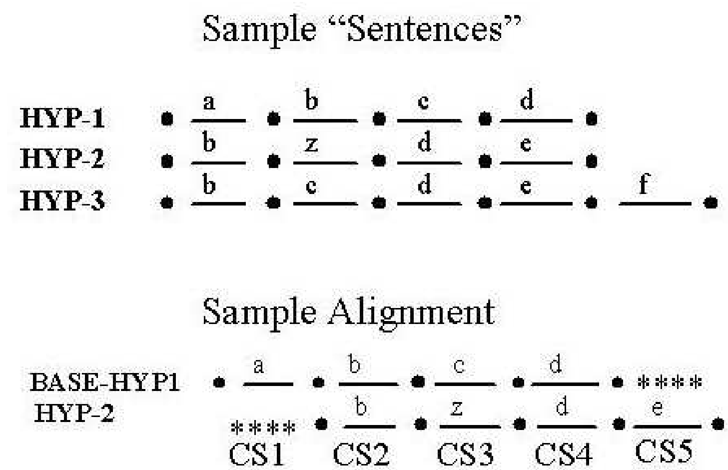
- Systems may come from multiple sites
- Can be a single site with different processing schemes

ROVER - Text String Alignment Process

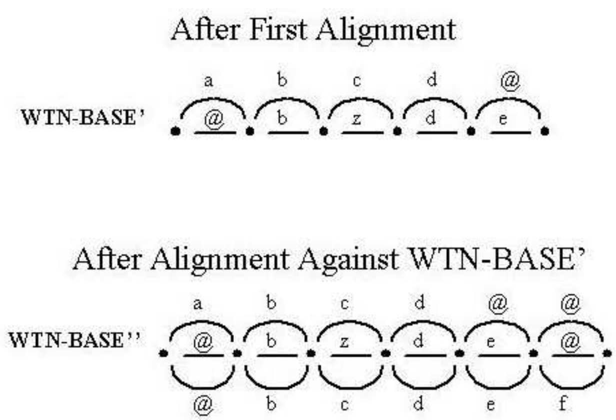


score(m,n)= min { score(m-1,n-1) + 4*no_match(m,n), score(m-1,n) + 3, score(m, n-1) + 3}

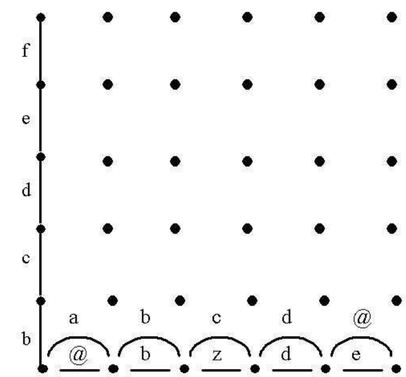
ROVER - Example



ROVER - Form Confusion Sets

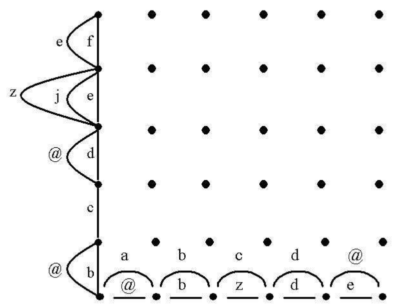


ROVER - Aligning Strings Against a Network



Solution: Alter cost function so that there is only a substitution cost if no member of the reference network matches the target symbol.

ROVER - Aligning Networks Against Networks



No so much a ROVER issue but will be important for confusion networks.
Problem: How to score relative probabilities and deletions?
Solution: cost_subst(s_1,s_2)= (1 - p_1(winner(s_2)) + 1 - p_2(winner(s_1))/2

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]

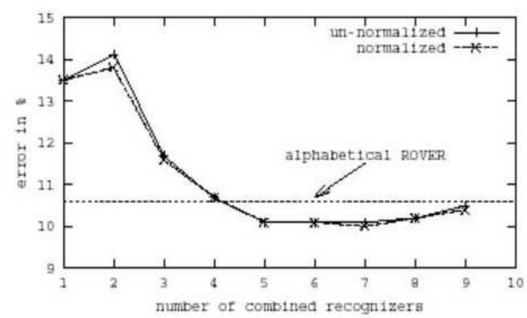


Figure 1: 1998 Broadcast News word error rates in function of the number of combined systems (individual error ranked order).

- 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.

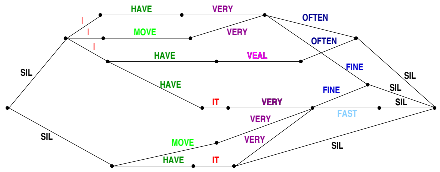
Hypothesis (H)							
w_1	w_2	w_3	$P(H\|A)$	$P(w_1\|A)$	$P(w_2\|A)$	$P(w_3\|A)$	E[correct]
I	DO	INSIDE	0·16	0·34	0·29	0·16	0·79
I	DO	FINE	0·13	0·34	0·29	0·28	0·91
BY	DOING	FINE	0·11	0·45	0·49	0·28	1·22
BY	DOING	WELL	0·11	0·45	0·49	0·11	1·05
BY	DOING	SIGHT	0·10	0·45	0·49	0·10	1·04
BY	DOING	BYE	0·07	0·45	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

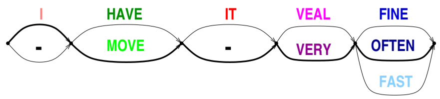
Consensus Decoding - Approach

Find a multiple alignment of all the lattice paths

Input Lattice:



Multiple Alignment:



Consensus Decoding Approach - cont'd

- Compute the word error between two hypotheses according to the multiple alignment:

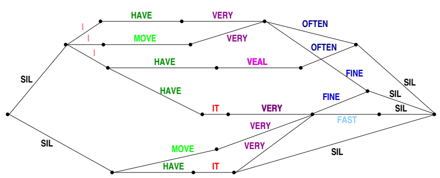
$$WE(W,R) \approx MWE(W,R)$$

- Find the *consensus* hypothesis:

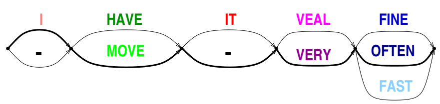
$$W_C = \arg\min_{W \in \mathcal{W}} \sum_{R \in \mathcal{W}} P(R|A) * MWE(W,R)$$

Consensus Decoding Approach - cont'd

Input:



Output:



Consensus Decoding Approach - Multiple Alignment

- Equivalence relation over word hypotheses (links)
- Total ordering of the equivalence classes

Mathematical problem formulation:

- Define a partial order on sets of links which is consistent with the precedence order in the lattice
- Cluster sets of links in the partial order to derive a total order

Consensus Decoding Approach - Clustering Algorithm

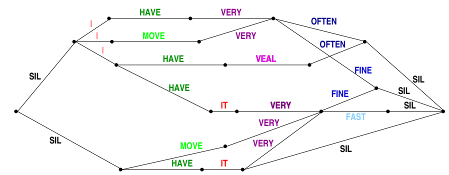
Initialize Clusters: a cluster consists of all the links having the same starting time, ending time and word label

Intra-word Clustering: merge only clusters which are not in relation and correspond to the same word

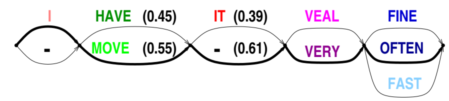
Inter-word Clustering: merge heterogeneous clusters which are not in relation

Obtaining the Consensus Hypothesis

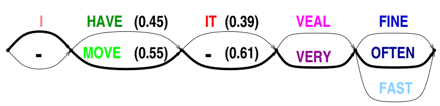
Input:



Output:

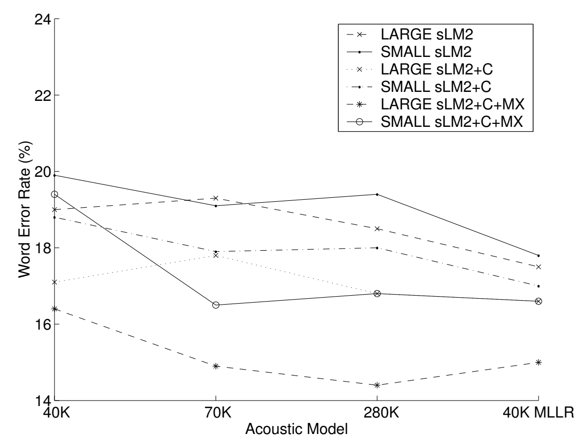


Confusion Networks



- Confidence Annotations and Word Spotting
- System Combination
- Error Correction

Consensus Decoding on DARPA Communicator



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

Consensus Decoding on Voice Mail

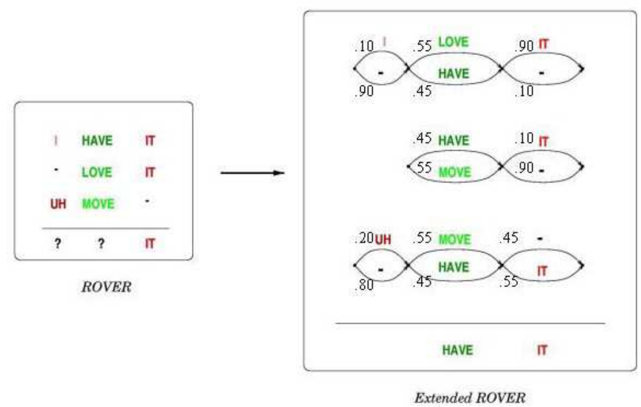
	Word Error Rate (%)	
System	Baseline	Consensus
S-VM1	30.2	28.8
S-VM2	33.7	31.2
S-VM3	42.4	41.6
ROVER	29.2	28.5

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 WER	eval00 WER	eval00 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)?