

SPEAKER ADAPTATION BY CORRELATION (ABC)

Scott Shaobing Chen & Peter DeSouza
IBM T.J. Watson Research Center
email: schen@watson.ibm.com

ABSTRACT

This paper describes a new rapid speaker adaptation algorithm using a small amount of adaptation data. This algorithm, termed adaptation by correlation (ABC), exploits the intrinsic correlation among speech units to update the speech models. The algorithm updates the means of each Gaussian based on its correlation with means of the Gaussians which are observed in the adaptation data; the updating formula is derived from the theory of least squares. Our experiments on the ARPA NAB-94 evaluation (Eval-94) and the ARPA Hub4-96 (Hub4-96) tasks indicate that ABC seems more stable than MLLR when the amount of data for adaptation is very small (~ 5 seconds), and that ABC seems to enhance MLLR when they are combined.

1. INTRODUCTION

The problem of speaker adaptation is to adjust the parameters of a speech recognizer according to a certain amount of adaptation data. In recent years, considerable amount of research effort has been invested in this area; various techniques have been proposed, such as MAP [6], MLLR [5] and CT (clustered transformation) [7].

In this paper, we are interested in the problem of *rapid* speaker adaptation, the problem of adapting speech systems using a very limited amount of data, e.g., ~ 10 seconds of speech. In such a situation, it is important that the limited amount of information in the data is *fully exploited* for the purpose of adaptation.

We assume that the basic speech recognition system uses HMM's to model the speech production process, and mixtures of continuous-density Gaussians to model the output distributions of the HMM's. Based on the adaptation data, counts can be obtained by running the forward-backward algorithm; since the amount of adaptation data is small, some Gaussians are observed, while most Gaussians are not. The challenge of rapid speaker adaptation is to determine how to adjust the unobserved Gaussians.

In the MLLR approach, Gaussians are usually tied into classes. Each class contains some observed Gaussians, based on which a rotation and a shift is computed for that

class; then every Gaussian in that class is transformed by that rotation and that shift.

In our approach, we exploit the intrinsic *correlations* among speech units to update those unobserved Gaussians: for each unobserved Gaussian, a shift is computed by linear regression on the shifts of the observed Gaussians.

We comment that the correlations between speech units has been used to construct tying structures for speaker adaptation [8]. We have recently learned that our approach is closely related to the quasi-Bayesian learning scheme of correlated continuous density HMMS proposed by Huo and Lee [4].

This paper goes as follows: section 2 explains correlations among speech units; section 3 describes our formulation; in section 4, we present experiments with the Wall Street Journal task and 1996 Hub4 task.

2. CORRELATIONS AMONG SPEECH UNITS

Certain speech units are intrinsically correlated, since they are produced by similar positionings of the articulators. This can be verified by examining the correlations among the cepstral values of speech units. In the IBM speech recognition systems, there are 52 phones. Each phone is decomposed into a sequence of 3 sub-phones corresponding to the beginning, middle and end of the phone. Thus the phone AA is replaced as a concatenation of the models AA_1, AA_2 and AA_3. There are thus 156 subphonetic units. Each subphonetic unit is further decomposed into context-dependent allophones. For our experiments, we used IBM systems which have 5471 allophones. Acoustic features are generated by performing Mel-cepstral analysis and linear discriminant analysis. The Wall Street Journal corpus is used to compute the correlations. The training set consists of $M = 309$ speakers, including 284 short-term speakers and 25 long-term speakers. For simplicity, we show here the correlations among the *subphonetic units*. First, counts are obtained for each speaker by running the forward-backward algorithm on the training data. A single Gaussian is computed for each subphonetic unit. For each acoustic dimension,

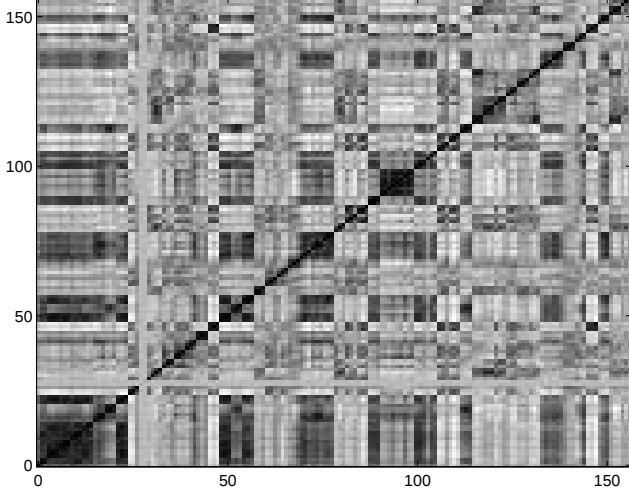


Figure 1. Correlations among subphonetic units

subphonetic units		correlation
AO_2	: AO_3	0.970
AE_2	: AE_3	0.964
N_2	: M_2	0.937
NG_2	: N_2	0.934
AY_2	: AE_2	0.932
EH_2	: AE_2	0.927
IX_2	: AX_2	0.914

Table 1. Correlated subphonetic units

the correlation between unit i and j is obtained in the conventional way:

$$r_{ij} = \text{cor}(\mu_i^{(1)}, \dots, \mu_i^{(M)} \quad , \quad (\mu_j^{(1)}, \dots, \mu_j^{(M)}) \quad)$$

where $\mu_i^{(m)}$ is the Gaussian mean of unit i of speaker m .

Figure 1 plots the correlation matrix in acoustic dimension 1, which corresponds to the biggest eigen value in the linear discriminant analysis; it shows the negative correlation in gray scale, i.e. black corresponds to +1 and white correspond to -1. Table 1 lists some of the top correlated units. Clearly we observe that the same subphonetic units from the same phone are often highly correlated, such as AO_2 and AO_3; similar phones, such as N_2 and M_2, are also highly correlated.

3. ABC FORMULATION

We exploit the correlations among speech units for speaker adaptation through linear regression. For each dimension of the feature space, the algorithm goes as follows.

For simplicity of discussion, we assume each HMM state l has only one Gaussian $\mathcal{N}(\mu_l, \sigma_l)$, $l = 1, \dots, L$. Let $(c_l, \bar{x}_l)$ be the counts by running the forward-backward algorithm on the adaptation data (c_l is the total count

of state l and $\bar{x}_l$ is the mean). The correlation structure among the Gaussian means is modeled as

$$\mu_{L \times 1} \sim \mathcal{N}(\mathbf{a}_{L \times 1}, \mathbf{S}_{L \times L}). \quad (1)$$

Denote o (*observed*) the subscripts of the Gaussians are observed and m (*missing*) otherwise; the mean vector $\mathbf{a}$ and the covariance matrix $\mathbf{S}$ are partitioned accordingly:

$$\mathbf{a} = \begin{pmatrix} \mathbf{a}_o \\ \mathbf{a}_m \end{pmatrix} \quad ; \quad \mathbf{S} = \begin{pmatrix} \mathbf{S}_{oo} & \mathbf{S}_{om} \\ \mathbf{S}_{mo} & \mathbf{S}_{mm} \end{pmatrix}$$

Then from the theory of least squares [1],

$$\mathbf{E}(\mu | (\mathbf{c}, \bar{\mathbf{x}})) = \mathbf{a} + \mathbf{S}(\mathbf{S}_{oo} + \Sigma_o)^{-1}(\bar{\mathbf{x}}_o - \mathbf{a}_o). \quad (2)$$

where

$$\Sigma_o = \begin{pmatrix} \ddots & & 0 \\ & \sigma_o^2/c_o & \\ 0 & & \ddots \end{pmatrix}.$$

The mean vector $\mathbf{a}$ in (1) can be considered as the mean vector associated to a *canonical* speaker. It is intuitively reasonable to use the mean vector of the speaker independent system. That leads to:

$$\Delta \mu_m = \mathbf{S}_{mo}(\mathbf{S}_{oo} + \Sigma_o)^{-1} \Delta \mu_o. \quad (3)$$

Clearly the shifts on the missing Gaussians are computed via linear regression with the shifts on the observed Gaussians.

Equation (3) is a weighted least square scheme. Note that Σ_o is a diagonal matrix of the speaker independent variances σ^2 scaled by the counts c ; thus the shift on a observed Gaussian receives more weight when the count associated to that Gaussian is high.

The covariance matrix $\mathbf{S}$ among the means of HMM states can be computed in similar fashion as in section 2. However, in big speech systems, there are thousands of HMM states; many speakers in the training corpus, such as WSJ corpus, would have many states which are not observed in the training data. To fully utilize the training corpus, first, the variances S_{ii} of μ_i can be computed from all the training speakers who have observations on state i ; then the correlation r_{ij} can be computed from all the training speakers who have observations on both state i and j , and the covariance can then be obtained by

$$S_{ij} = R_{ij} \cdot \sqrt{S_{ii} S_{jj}}.$$

It is necessary to repeat the above process for every dimension of the feature space, because different dimensions can have very different correlation structures.

The ABC algorithm is summarized as follows:

- Estimate the covariances $\mathbf{S}$ from the WSJ corpus.
- Obtain counts by running forward-backward algorithm on the adaptation data.

- For every dimension of the feature space:
 - For observed states, compute shifts.
 - For missing states, estimate shifts by ABC(3).
 - For each state, equally shift all Gaussians in the mixture.

4. EXPERIMENTS

In this section, we present adaptation experiments on two test sets; we compare ABC adaptation, in particular, with MLLR adaptation. Unsupervised adaptation was used in all cases.

4.1. ARPA NAB-94 Evaluation

The test data consists of about 15 sentences each, from 20 speakers; this is the Nov'94 evaluation data in the ARPA Wall Street Journal task. The base system is a scaled down version of the IBM system used in the Nov'94 evaluation [2]; it had 5471 context-dependent states and $\approx 17K$ Gaussians; the official 20K language model that was provided by NIST for the Nov'94 ARPA evaluation was used.

In this experiment, the base system was adapted using only 1 sentence; this is to study the stability of adaptation schemes when the amount of data is very small. For each speaker, the base system was adapted using the first sentence, then all sentences were decoded with the adapted system; the decoding results are shown in Table 2. We observe that overall MLLR increased the error rate by absolute 1% while ABC reduced the error rate by absolute 1%; In particular, MLLR performed poorly for speaker h1c4t402 and speaker h1c4t802, for which the amount of adaptation is very small (3.9 seconds and 5.3 seconds); ABC seems relatively more stable, and can sometimes reduce the errors, even when the amount of data for adaptation is very small.

4.2. ARPA Hub4-96 Evaluation

We applied ABC adaptation on the F0 condition (clean and prepared speech) of the Hub4 1996 evaluation test set. This test data consists of 102 sentences, from 16 speakers. The base system is the so-called conglomerate model M96H4 described in [3]; it had 5471 context-dependent states and $\approx 160K$ Gaussians; a 65K language model was used [3]. Auto adaptation was performed on every test sentences. Table 3 shows the overall decoding results. ABC performed slightly worse than MLLR, and the combined scheme of ABC+MLLR gave the most error reduction.

5. DISCUSSION

We discuss in this section the performance and computation of various adaptation schemes.

speaker	adaptation data in seconds	total # words	base	MLLR	ABC
h1c4t002	9.1	457	122	94	91
h1c4t102	7.9	543	55	61	57
h1c4t202	11.2	404	55	44	43
h1c4t302	8.0	391	20	19	23
h1c4t402	3.9	334	45	133	41
h1c4t502	11.1	413	36	48	39
h1c4t602	4.9	382	132	135	116
h1c4t702	11.2	476	54	58	52
h1c4t802	5.3	320	41	85	44
h1c4t902	11.8	371	25	25	25
h1c4ta02	7.9	379	31	28	29
h1c4tb02	13.8	322	56	51	51
h1c4tc02	15.7	418	27	27	28
h1c4td02	15.2	519	183	161	176
h1c4te02	5.9	429	53	62	46
h1c4tg02	7.3	336	63	66	68
h1c4th02	9.6	509	42	45	42
h1c4ti02	20.3	454	76	70	68
h1c4tj02	10.6	342	50	50	47
h1c4tk02	9.3	390	43	34	41
total		8189	1209	1296	1127
wer			14.8%	15.8%	13.8%

Table 2. Adaptation on Eval-94 using the first sentence

	Base	MLLR	ABC	ABC+MLLR
# W.E. Rate	22.5%	21.7%	22.0%	21.4%

Table 3. Auto adaptation on Hub4-96

5.1. Performance

Overall, ABC adaptation seems effective in reducing the decoding error rate, as shown in the above experiments. Here we make three observations.

First, when the amount of adaptation data is very small, ABC adaptation seems more stable than MLLR. In the MLLR adaptation, since only a few Gaussians are observed, all Gaussians were forced to share the same linear transformation. In this case, it is likely that a bad transformation could be obtained; such a transformation has to be shared on all the Gaussians, and the adapted system could perform badly. On the other hand, in ABC adaptation, many Gaussians remain unchanged, since only those Gaussians which are correlated with the observed Gaussians are updated. Such adjustments are relatively stable, and can often improve the performance, even though the amount of adjustment is small.

Second, ABC adaptation seems to enhance MLLR. The combined scheme of ABC+MLLR gives the most error reduction in the Hub4-96 task. One possible explanation is that ABC explicitly utilizes the correlations among HMM states, which is probably not fully exploited in MLLR adaptation.

Third, ABC adaptation alone is worse than MLLR in both experiments. The reason is that MLLR adaptation uses the information across different acoustic dimensions, whereas our current implementation of ABC adaptation does not. In MLLR adaptation, information across different acoustic dimensions is used to compute the rotations, and each Gaussian is both rotated and shifted properly. Conceptually, one can utilize correlations across different acoustic dimensions in ABC adaptation; however, due to the limitation of computation and storage requirements, different dimensions are adapted independently and each Gaussian is only shifted properly. In our future study, we would like to investigate strategies of utilizing such correlations across different acoustic dimensions for ABC adaptation.

5.2. Computation

At first glance, it seems very hard to implement ABC adaptation. naively, one would like to pre-compute and store the covariance matrix S for each acoustic dimension. However, that is not realistic for big systems: the IBM systems in the previous experiments has 5471 allophones and uses 60 dimensional features; that would require 8GB to store the 60 covariance matrices!

However, in rapid speaker adaptation, only small number of Gaussians are observed. Thus only a small part of the covariance matrix S is actually used. Our strategy is to *compute the covariances by demand*. The resulting implementation of ABC adaptation is very efficient; it was as fast as MLLR in the previous experiments.

It is reported [7] that CT adaptation can outperform

MLLR; however it has a much higher computational complexity. In our future study, we would like to compare ABC with CT and to see if ABC can also enhance CT.

6. ACKNOWLEDGEMENT

We thank P.S. Gopalakrishnan for constant encouragements and helps in our implementation of ABC; he made suggestions which significantly speeded up the computation. We thank Mukund Padmanabhan for helps in setting up the experiments.

REFERENCES

- [1] T.W. Anderson, An introduction to multivariate statistical analysis, 2nd edition, Wiley, 1984.
- [2] L. R. Bahl et al., "Performance of the IBM large vocabulary continuous speech recognition system on the ARPA Wall Street Journal task", Proceedings of the ICASSP, pp 41-44, 1995.
- [3] R. Bakis, S. Chen, P.S. Gopalakrishnan, R. Gopinath, S. Maes, L. Polymenakos, "Transcription of broadcast news shows with the IBM large vocabulary speech recognition system", elsewhere in this proceedings.
- [4] Q. Huo and C. H. Lee, "On-line adaptive learning of the correlated continuous density hidden Markov models for speech recognition", ICSLP-96, vol. II, pp 985-988.
- [5] C. J. Legetter and P. C. Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density HMM's", Computer Speech and Language, vol. 9, no. 2, pp 171-186.
- [6] J. L. Gauvain and C. H. Lee, "Maximum-a-Posteriori estimation for multivariate Gaussian observations of Markov chains", IEEE Trans. Speech and Audio Processing, vol. 2, no. 2, pp 291-298, Apr 1994.
- [7] M. Padmanabhan, L. R. Bahl, D. Nahamoo, M. A. Picheny, "Speaker clustering and transformation for speaker adaptation in large vocabulary speech recognition systems", ICASSP-96, vol. II, pp 701-704.
- [8] S. Takahashi and S. Sagayama, "Tied-structure HMM based on parameter correlation for efficient model training", ICASSP-96, vol. II, pp 467-470.