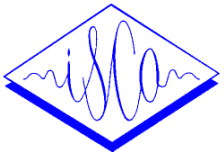

7th International Conference on Spoken Language Processing [ICSLP2002]
Denver, Colorado, USA
September 16-20, 2002

INCREMENTAL ON-LINE FEATURE SPACE MLLR ADAPTATION FOR TELEPHONY SPEECH RECOGNITION

Yongxin Li, Hakan Erdogan, Yuqing Gao, Etienne Marcheret

IBM Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY10598, USA
{yongxin,erdogan,yuqing,etiennem}@us.ibm.com

10.21437/ICSLP.2002-64

ABSTRACT

In this paper, we present a method for incremental on-line adaptation based on feature space Maximum Likelihood Linear Regression (FMLLR) for telephony speech recognition applications. We explain how to incorporate a feature space MLLR transform into a stack decoder and perform on-line adaptation. The issues discussed are as follows: collecting adaptation data on-line and in real time; mapping adaptation data from previous feature space to the present feature space; and smoothing adaptation statistics with initial statistics based on original acoustical model to achieve stability. Testing results on various systems demonstrate that on-line incremental FMLLR adaptation could be an effective and stable method when the adaptation statistics are mapped and smoothed.

1. INTRODUCTION

With speech recognition systems focusing more on telephony applications, it is imperative to develop an on-line speaker adaptation suited for telephony speech recognition applications. Speaker adaptation in spoken dialog systems is challenging, because there is limited adaptation data available in a telephony application. In a conversational telephony system, it is possible to have five or more utterances from a single speaker, enabling us to perform speaker adaptation. The goal of this work is to develop practical techniques to perform adaptation with small amounts of speaker dependent data in real time.

MLLR is the most efficient adaptation method with small amounts of adaptation data, when effective smoothing is employed. Usually MLLR is applied to adapt the means of the Gaussians in an HMM model, in which case we call it model space MLLR. Since model space MLLR adapts all means even with sparse data, the computational burden might exceed the resources available to a real time telephony system. In addition, some speech recognition systems employ fast labeling approximations that might require to be updated with the new set of models. For example, in IBM ViaVoice, the means and variances of the Gaussians are quantized into bins, enabling some terms in the likelihood computation to be precomputed. This speeds up the overall Gaussian likelihood calculations. Re-quantization after model space MLLR adaptation becomes unacceptable in ViaVoice for a real time system. Although some of these problems could be avoided, by for example adapting the quantized values directly [4], we would like to focus on feature space MLLR (or FMLLR) to have the least headaches. In FMLLR, a transform matrix is applied to the speaker dependent feature vectors to bring them closer to the speaker independent models. As opposed to model space MLLR, the main benefit of FMLLR is that once the transform matrix is computed, it can be applied immediately to following utterance without any delay or extra computation. However, in FMLLR, it is slightly more computationally complex to compute the transform itself.

In this work we apply a single feature space transformation matrix. In a standard telephony application the speech data from a single speaker is sparse, making it difficult to compute multiple transforms. Another reason we do not apply multiple transform FMLLR is that with multiple feature space transforms, there will be multiple feature streams, bringing additional significant computation.

As for incremental on-line FMLLR adaption for telephony applications, we have to address issues such as: efficient collection of un-supervised adaptation data; incrementally integrating adaptation data derived from different feature spaces; reducing iterations to compute FMLLR; and smoothing adaptation data for robustness. The above concerns are covered in following sections. Section 2 briefly reviews the feature space MLLR algorithm in [1]. We discuss how to collect unsupervised adaptation data on-line and in real time in section 3. Section 4 discusses why and how to map adaptation data between feature spaces. In Section 5, a smoothing scheme similar to DLLR[3] is proposed and testing results are presented. In Section 6 we discuss the details of implementation and present testing results showing improvement in accuracy on various test sets. We draw conclusions and discuss further directions in Section 7.

2. FEATURE SPACE MAXIMUM LIKELIHOOD LINEAR REGRESSION

In this section, we quickly review the algorithm to compute the FMLLR transform. Following notation in [1], A is the FMLLR linear transform on the feature space. $O(\tau)$ is the observation in the original feature space at time τ and $\hat{O}(\tau)$ is the observation vector in transformed feature space.

$$\hat{O}(\tau) = AO(\tau) + b, \tag{1}$$

b being the bias term. Denoting ξ as the extended observation vector, $\xi_\tau = [1, o(\tau)]^T$. W is the extended transformation matrix on the extended observation, $W = [b^T, A^T]^T$. With these notations, Equation 1 is expressed as

$$\hat{O}(\tau) = W\xi(\tau). \tag{2}$$

The objective function, or, the likelihood of the transformed observation vectors, is

$$\beta log(p_i w_i^T) - \frac{1}{2}\sum_{i=1}^{n}\left(w_i G_i w_i^T - 2w_i k_i^T\right) \tag{3}$$

where we ignore terms independent of W; n is the dimension of feature space; $\beta = \sum_{m=1}^{M}\sum_{\tau=1}^{T}\gamma_m(\tau)$ is the total count; subindex m indicates states or Gaussians, depending on the context; $\gamma_m(\tau)$ is the posterior probability of being in state m at time τ; w_i is the i-th row of the expanded transformation

matrix W; p_i is the cofactor $[0, c_{i1}, \ldots, c_{in}]$, $c_{ij} = cof(A_{ij})$; and

$$G_i = \sum_{m=1}^{M} \frac{1}{\sigma_{m,i}^2} \sum_{\tau=1}^{T} \gamma_m(\tau)\xi(\tau)\xi(\tau)^T, \quad (4)$$

$$k_i = \sum_{m=1}^{M} \frac{1}{\sigma_{m,i}^2} \mu_{m,i} \sum_{\tau=1}^{T} \gamma_m(\tau)\xi(\tau)^T. \quad (5)$$

In this paper, G_i and k_i are referred to as adaptation data or FMLLR data.

Because of the term $log(p_i w_i^T) = log(\det A)$, a closed-form solution for W does not exist. A row by row iteration method is proposed in [1]. We describe it briefly as follows. Set

$$a = p_i G_i^{-1} p_i^T \quad (6)$$

$$b = p_i G_i^{-1} k_i^T \quad (7)$$

Then solve the following quadratic equation on α.

$$a\alpha^2 + b\alpha - \beta = 0 \quad (8)$$

There are two solutions for α. In order to maximize the objective function, we select α such that

$$\alpha = \begin{cases} \frac{1}{2a}(-b + \sqrt{b^2 + 4a\beta}) & \text{if } b \geq 0 \\ \frac{1}{2a}(-b - \sqrt{b^2 + 4a\beta}) & \text{if } b < 0 \end{cases} \quad (9)$$

Finally

$$w_i = \Big(\alpha p_i + k_i\Big) G_i^{-1} \quad (10)$$

is the approximating solution for i-th row of W. By doing this for all the rows we will find a better W. By iterating the process we will approximate the optimal solution of W.

The number of iterations necessary for W to converge is highly dependent on the data. In testing, it converges usually after 30 iterations, while in some cases, we also observe that it does not converge even after 100 iterations. In each iteration, we need to compute the cofactors of W, which is equivalent to finding the inverse of the matrix W. With respect to computational cost, it is the most expensive part of each iteration. Fortunately, the dimension of feature space is usually around 40 and so the inversion of W does not take long. Nevertheless, we want to reduce the number of iterations. This is addressed further in Section 4 and 5.

3. COLLECTING UNSUPERVISED FMLLR STATISTICS ON-LINE

Ideally, the on-line adaptation should be accomplished seamlessly without noticeable latency. The first step in the process is to collect labeled adaptation data. IBM ViaVoice is a stack decoder and we do not have the strict alignment information in decoding as in a Viterbi decoder. Thus after we finish decoding a short phrase, we run a Viterbi alignment of the speech data with the decoded text. After the alignment, we know which state an observation vector belongs to. We spread the counts over the individual Gaussians which comprise a state. That is how we obtain the posterior probability $\gamma_m(\tau)$. We can then accumulate the statistics G_i and k_i as defined in Equation 4 and Equation 5

In IBM ViaVoice, the state likelihoods for each frame is available at the end of decoding. We reuse the state likelihoods to run Viterbi alignment. Because of this, the resources needed to run Viterbi is significantly reduced. In the entire process, the most computationally expensive step is accumulating G_i and k_i. Taking a 40-dim feature space, for each frame, we have to do rank one updates on 40 matrices of 41×41. This requires excessive number of multiplications, even if we use the symmetry of G_i. One way to speed up accumulating is pooling together those observations belonging to the same state and accumulating only once for all of them. In fact, for faster speed, we use ESSL, a computing package offered by IBM with all of the necessary linear algebra routines to compute the FMLLR matrix.

As for an un-supervised adaptation, we must address methods to avoid bad data. Particularly, in telephony systems, the quality of speech varies significantly. We implemented an algorithm to find confidence scores for the decoded words [6] and use it to reject weak hypotheses. A trade-off needs to be made between losing good data and allowing corrupted data to enter into adaptation. On this issue, we are conservative and we tune the rejection threshold high. Rejection is done on a word basis, therefore only those words with low confidence score are rejected, while other words in the same phrase are retained.

In telephony speech recognition, due to a variety of reasons, it is very likely to get long periods of silence, which essentially is channel noise. Thus, silence data is helpful to adapt to the channel. However, if the adaptation data is dominated by silence, the new transform could be biased towards adapting silence only, which is not desirable. With this concern in mind, we attempt to keep the balance between silence data and speech data. In testing, we got improvement of accuracy by balancing data from silence and speech.

4. MAPPING THE STATISTICS TO NEW FEATURE SPACES

The scenario of incremental adaptation is that after decoding of each typically short utterance, the statistics are collected, a new transformation matrix is computed and applied immediately for the next decoding. This process is continued incrementally after each utterance. We would like to accumulate all statistics from the speaker. However, a potential problem is that they come from different feature spaces. It does not make much sense to directly accumulate the data from different spaces.

There are two ways to solve this problem. One is to keep all the data in the original space and compute a new FMLLR transformation matrix from the original feature space to the new feature space. To do this, we will need either to keep using feature vectors in the original feature space when we accumulate G_i and k_i, or, to map new data back into the original feature space with an inverse transform. Both of the approaches are cumbersome. Another approach is to map adaptation data of previous feature space to present feature space then compute FMLLR transform from the latest feature space to a new feature space. For this purpose, we do the following update each time we compute a new FMLLR transformation matrix A and bias b. Denote

$$\hat{W} = \begin{pmatrix} 1 & 0 \\ b & A \end{pmatrix} \quad (11)$$

then

$$\hat{G}_i = \hat{W} G_i \hat{W}^T \quad (12)$$

$$\hat{k}_i = \hat{W} k_i \quad (13)$$

With the above update, $\hat{G}_i$ and $\hat{k}_i$ are equivalent to some data computed on the observations in the transformed feature space, with the assumption that the alignment remains unchanged.

Now the transform and bias term are computed to transform the latest feature space to another feature space. Suppose the previous transform and bias are A_0 and b_0, then we have to update them as follows.

$$\hat{A} = AA_0 \tag{14}$$

$$\hat{b} = Ab_0 + b \tag{15}$$

The transformation matrix and bias applied to the original feature space should be $\hat{A}$ and $\hat{b}$.

For very sparse adaptation data, mapping $\hat{G}_i$ and $\hat{k}_i$ does not make much of a difference. In case there is more data available, it could hurt the accuracy if the data is not mapped. We will show results in Section 6 on this issue. A nice side effect of mapping data is that it reduces the number of iterations drastically. With the above update, MLLR transform is computed from latest feature space to a new feature space. The MLLR transform will be close to the identity matrix and therefore it takes less iterations to converge.

5. SMOOTHING

In practise, it is impossible to reject corrupted adaptation data completely. In some cases, an FMLLR transform could result in disaster. We test the on-line FMLLR on a numeric-alphabet application. The grammar is simple with just digits and English letters. The acoustic model size is 30K Gaussians. Each utterance is as long as sixteen digits or letters, equivalent to five hundred frames. We tested on 50 speakers and on an average of 40 utterances per speaker. The second column is the decoding word error rates without smoothing.

	Without Smoothing	With Smoothing
baseline	2.3	2.3
One Utt.	12.38	2.13
Three Utt.	2.36	1.89
Incr.	1.82	1.76

Table 1: Comparison on smoothing

One Utt. means the first utterance is used as adaptation data to compute the FMLLR transform and then apply it to all the remaining utterances. We also tried using the first three utterances as adaptation data. *Incr.* means we keep collecting adaptation data all the time, at the end of each utterance, a new MLLR transformation matrix is computed and applied to the next decoding. Obviously, in this testing, FMLLR is not stable. The breaking down of the accuracy on individual speakers shows a few speakers got seriously hurt and their error rate increased a lot. To avoid these errors, smoothing is required.

There might be different ways to conduct smoothing. One is linearly interpolating the FMLLR matrix with identity matrix as a post processing step. We did not take this approach because in the testing mentioned above, for some speakers, with the first utterance as adaptation data, the MLLR matrix computed could be far from the identity matrix. To make such a matrix more reasonable by interpolating it with an identity matrix, we would have to use a big weight on the identity matrix. Doing that offsets the benefit of MLLR dramatically.

In the context of model space MLLR, one of the cost effective and successful smoothing techniques is discounted likelihood linear regression (DLLR). Another method is called maximum aposteriori linear regression (MAPLR) which introduces a prior term in the objective function. MAPLR requires finding a prior distribution for the transform matrix and could be very expensive to compute. In this paper, we use a technique similar to DLLR because of the ease of use and simple computation. In DLLR, the adaptation statistics are interpolated with speaker independent statistics to avoid overtraining [3]. For FMLLR, we perform a similar technique that we describe below. We start accumulating our statistics not from zero, but from terms that depend on the speaker independent models. This method has the nice effect of having the influence of speaker independent statistics reduce as we get more and more adaptation data. Assuming we observe some feature vectors drawn from the acoustic model. For a Gaussian g_m, with mean $\mu_m = (\mu_{m,1}, \ldots, \mu_{m,n})^T$ and diagonal covariance $\Sigma_m = (\sigma^2_{m,1}, \ldots, \sigma^2_{m,n})$, if a random variable $\{X_t\}_{t=1}^T$ is drawn from g_m, then by definition,

$$\lim_{T\to\infty} \sum_{t=1}^{T} X_t = \mu_m \tag{16}$$

$$\lim_{T\to\infty} \sum_{t=1}^{T} (X_{t,i} - \mu_{m,i})^2 = \sigma^2_{m,i} \tag{17}$$

Now pretend X_t is a sequence of real observations and denote $\xi_t = [1, X_t]^T$. Following Equation 5,

$$k_i = \sum_{t=1}^{T} \frac{1}{\sigma^2_{m,i}} \mu_{m,i} X_t^T \tag{18}$$

taking the average on t and letting $T \to \infty$, we have k_i with respect to g_m as

$$k_{i,m} = \frac{1}{\sigma^2_{m,i}} \mu_{m,i} \begin{pmatrix} 1 \\ \mu_m \end{pmatrix} \tag{19}$$

Similarly, for G_i, first we compute G_i corresponding to X_t,

$$\begin{aligned} G_i &= \sum_{t=1}^{T} \frac{1}{\sigma^2_{m,i}} \xi_t \xi_t^T \\ &= \frac{1}{\sigma^2_{m,i}} \sum_{t=1}^{T} \begin{pmatrix} 1 & X_t^T \\ X_t & X_t X_t^T \end{pmatrix} \end{aligned}$$

Taking the average on t and letting $T \to \infty$, G_i with respect to g_m is

$$G_{i,m} = \frac{1}{\sigma^2_{m,i}} \begin{pmatrix} 1 & \mu_m^T \\ \mu_m & \mu_m \mu_m^T + \Sigma_m \end{pmatrix}$$

where $\Sigma_m = \mathrm{diag}(\sigma^2_{m,1}, \ldots, \sigma^2_{m,n})$. Now sum up G_i and k_i corresponding to all Gaussian g_m with weight p_m, we get the initial FMLLR statistics.

$$\begin{aligned} G_i &= \sum_{m=1}^{M} p_m \frac{1}{\sigma^2_{m,i}} \begin{pmatrix} 1 & \mu_m^T \\ \mu_m & \mu_m \mu_m^T + \Sigma_m \end{pmatrix} \\ k_i &= \sum_{m=1}^{T} p_m \frac{\mu_{m,i}}{\sigma^2_{m,i}} \begin{pmatrix} 1 \\ \mu_m \end{pmatrix} \end{aligned}$$

By tuning $p = \sum_{m=1}^{T} p_m$, we can control the smoothing weight. The greater p, the more stable FMLLR is, but less effective.

In testing, we found $p = 1000$ works well. In case the priors of states are unknown we can just apply a uniform weight.

With the above initial G_i and k_i, the algorithm cited in Section 2 from [1] generates an identity matrix at first iteration. This confirms the way we compute the initial data is correct.

The third column of Table 1 is the word error rate with smoothing. Comparing it with the results without smoothing, we see that smoothing is beneficial.

6. IMPLEMENTATION AND EXPERIMENTAL RESULTS

In the implementation, we accumulate adaptation data on top of the initial G_i and k_i as computed in Section 5. When an additional 5% more data is collected, FMLLR adaptation starts. A new transformation matrix is computed and applied to next decoding. With the updating described in Section 4 and smoothing in Section 5, the average iteration numbers for FMLLR to converge is approximately 5. Without mapping and smoothing, the number of iterations it takes is somewhere between 50 and 100 or even more in some cases.

We test the adaptation on a dictation system of 40K Gaussians with clean testing data from ten speakers, 1K words per speaker. The dictation system is not a telephony application but we include it here since it is informative from an algorithm point of view, how much improvement we can get with this adaptation approach with clean, sufficient data.

Baseline	1/4 Incr.	Half.Incr.	Incr.	NoMap.
9.67	8.97	8.76	8.67	9.05

Table 2: Testing on dictation

In Table 2, *Incr.* means on-line adaptation is done incrementally all the way for each speaker on a 1000-word-dictation. *Half.Incr.* means the incremental adaptation is stopped at midway and FMLLR transform is fixed and applied to the rest testing of the same speaker. *1/4 Incr.* means only the first quart of test data is used for incremental adaptation. All the above are done with mapping data except the last column. *NoMap.* means it is done without mapping adaptation data as in Section 4. With an equivalent amount of adaptation data and the same testing data, we can reduce error rate by 20% with 20 model space MLLR transform adaptations. With only one feature space transform, we have seen 10% improvement. It also shows that, mapping the adaptation data is helpful on the accuracy.

We have also tested the method on a telephony dialog system in the air travel domain. It is a medium vocabulary system with more than 40K Gaussians. Besides the smoothing scheme of Section 5, we use a threshold, called minimal-counts (referred as M in Table 3) to stabilize the adaptation. FMLLR adaptation is kicked off only when adaptation data exceeds minimal-counts. It helps for the error rate but not as much as the smoothing scheme of Section 5. In the table, P refers to the weight given to the smoothing term. A count of 1000 represents a total of 10 seconds of speech data. In the testing showed below, we use both M and P. We tested the case $P = 0$, observed degrading of accuracy on some test sets and so we set $P = 500$ or higher. The column *Overall* is the word-based average of test 1 through 4. In test1, there are 74 speakers and 53 words per speaker on average. There are less words per speaker on other tests. The results show that smoothing makes adaptation more robust without the minimal count threshold.

	test1	test2	test3	test4	overall
baseline	19.80	16.73	11.12	25.63	19.12
M=0,P=1000	17.81	15.62	10.48	23.52	18.44
M=0,P=500	18.07	16.35	9.90	23.95	18.57
M=500,P=1000	17.81	15.62	9.88	23.81	18.41
M=500,P=500	18.07	16.35	9.30	23.95	18.47
M=1000,P=1000	17.81	15.62	10.48	23.52	18.44
M=1000,P=500	18.07	16.35	9.30	23.68	18.34

Table 3: FMLLR performance applied to DARPA communicator.

7. CONCLUSION AND DISCUSSION

Testing on different systems demonstrates incremental on-line FMLLR adaptation is effective. It reduces the word error rate 4% to 10% in a large vocabulary system depending on the amount of adaptation data. Even more reduction is observed on a small vocabulary system. Mapping adaptation data from the previous feature space to the present feature space helps on stability and speed.

There might be numerous arguments on how to handle insufficient adaptation data. One approach is computing a diagonal transformation matrix instead of a full matrix. We tried this approach but we did not see any gain in the case of a small amount of adaptation data. We use the smoothing technique of Section 5 to handle insufficiency of data.

As for any un-supervised adaptation, we need a better rejection scheme using better confidence scores. We plan to continue working on this issue in the future.

As reported in [7], the gains from FMLLR and standard model space MLLR could be more or less additive. As future work, we are planning to implement a fast model space MLLR, compare it with FMLLR or even try running it together with FMLLR adaptation.

REFERENCES

[1] M. J. F. Gales, "Maximum Likelihood Linear Transformations for HMM-Based Speech Recognition", Technical Report, Cambridge University Engineering Department, May 97.

[2] Y. Gao, Y. Li and M. Picheny, "Maximum Rank Likelihood as an Objective function for Speech Recognition", ICSLP 2000.

[3] W. Byrne and A. Gunawardana, "Discounted Likelihood Linear Regression for rapid adaptation", EUROSPEECH 1999.

[4] J. Huang and M. Padmanabhan, "Fast Adaptation of Band-Quantized Speech Decoding System", submitted to Special Issue on Speech Technologies for Mobile and Portable Devices, IEEE Trans. on Speech and Audio Processing.

[5] C. J. Legetter and P. C. Woodland, "Maximal Likelihood Linear Regression for Speaker Adaptation of Continuous density HMM's", Computer Speech and Language, vol. 9, no. 2, 171-186.

[6] B. Maison and R. Gopinath, "Robust Confidence Annotation and Rejection for Continuous Speech Recognition", SPEECH L4, ICASSP 2001.

[7] G. Saon, G. Zweig and M. Padmanabhan, "Linear Feature Space Projections for Speaker Adaptation", ICASSP 2001.