EFFICIENT, LOW LATENCY ADAPTATION FOR SPEECH RECOGNITION

Suleyman S. Kozat, Karthik Visweswariah and Ramesh Gopinath

IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
{kozat, kv1, rameshg}@us.ibm.com

ABSTRACT

Constrained or feature space Maximum Likelihood Linear Regression (FMLLR) is known to be an effective algorithm for adaptation to a new speaker or environment. It employs a single transformation matrix and bias vector to linearly transform the test speaker's features. FMLLR makes no assumption on the underlying noise, environment or speaker and estimates parameters to maximize likelihood of the test data. The standard implementation needs considerable computational power, requires significant amounts of storage, and requires a first pass decoding before adaptation can begin. In this paper, we propose a simplified implementation of FMLLR for embedded applications to address these problems. Here, we employ a simple speech/silence segmentation to estimate parameters. We operate in the 13 dimensional cepstral space, hence resource requirements are low. The algorithm does not require a first pass decoding (parameter estimation is accomplished entirely in the front end) and can be applied with low latency as compared to FMLLR. The algorithms we describe here provide an attractive tradeoff between the power of FMLLR and the computational simplicity of Cepstral Mean Subtraction. With minimal cost, we achieve nearly 15% relative gains on an embedded speech recognition task.

Index Terms— Speech recognition, Speech enhancement

1. INTRODUCTION

Adaptation of system parameters or features to a new speaker or environment is employed by most of the state of the art speech recognizers. Several different algorithms are employed with considerable success to compensate the mismatch due to a new speaker or environment. These algorithms include, Cepstral Mean Subtraction (CMS) for compensating for a convolutive channel; RASTA for rectifying the channel mismatch; CDCN for additive noise and convolutive channel compensation; maximum likelihood linear regression (MLLR) or its contrained version FMLLR, for rectifying speaker variability and channel mismatch. FMLLR, unlike other types of feature adaptation techniques (including CDCN, RASTA or CMS), there is no assumption on the underlying noise, model or channel. In FMLLR, the test features are linearly transformed such that the features are better matched to the original model. In this paper we are interested in simplifying FMLLR so that all the calculations required to estimate the parameters can be carried out in the front end, without the need for a first pass decoding. The other objective we had was to reduce the amount of computation and memory required to estimate parameters. We note that Cepstral Mean Subtraction can be viewed as an extremely simple version of FMLLR, where we transform the data to match a zero mean Gaussian model. In this paper we are interested in making a tradeoff between the simplicity of CMS and the power of FMLLR.

The standard implementation of FMLLR was introduced in [1]. The affine transform is estimated by maximizing the likelihood of the transformed features with respect to the full speech recognition model assuming that the decoding is correct. An auxiliary function for the likelihood [1] results in a fairly efficient algorithm for parameter estimation. Parameters can also be estimated by direct maximization of likelihood [2] which is less efficient but more flexible. The standard implementation of FMLLR requires $O(n^3)$ operations per frame to collect statistics, $O(n^3)$ space for storage and $O(n^4)$ operations for calculation of the transform matrix. In standard features space sizes (n is around 50), the computations required can be prohibitive for commerical speech recognition engines which often run more than 10 times faster than real time. Since estimation requires decoding results, adaptation is not possible for the first utterance (unless we decode twice). Furthermore, the effectiveness of FMLLR is reduced when there are only a few utterances from a given speaker/condition. In order to overcome these deficiencies several different methods are proposed in the literature. One option is to simplify the transform matrix; using a block diagonal [3] or just a diagonal transform matrix. Restricting the matrix to be in a subspace can be effective in reducing the amount of data required to estimate the transform [4], [3]. [5] proposes a stochastic gradient estimation technique that is computationally very efficient. Although these methods reduce the computational load and/or are more stable with relatively small amounts of data, they still require a first pass decoding and are still much more expensive than techniques like CMS.

In this paper we employ a different approach to tackle

these problems. Instead of maximizing likelihood under the full speech recognition model, we use a simplified model comprised solely of speech and silence models. We use speech silence labels generated by a low latency speech detector to split the data into two classes: speech and silence. For each of the classes (speech and silence) we build Gaussian mixture models in the cepstral space, and use these models to calculate the likelihood of the data. Since we calculate transforms in the cepstral space, we only deal with 13 dimensional features, which results in smaller amount of computation. The transform can be estimated and applied before the first decoding. We find that diagonal transforms and even just simple scaling can give significant gains. These simple transforms can be estimated very efficiently. This can be seen as generalization of CMS where a bias is subtracted to make the data zero mean (Maximization of likelihood assuming the model is zero mean Gaussian) and cepstral variance normalization (Maximization of likelihood assuming the model is zero mean unit variance Gaussian). Increasing the complexity of the model gives a smooth tradeoff between techniques like FMLLR and techniques like CMS. We could choose a model which allows more acoustic classes and not just speech and silence.

The rest of the paper is organized as follows. In Section 2, we explain the estimation of the feature transformation and generation of the speech/silence model. In Section 3, we present the experimental results on several different databases. We then conclude our paper with some directions for future research.

2. ALGORITHM DESCRIPTION

If we denote the feature vectors generated for a test speaker as $\vec{x}_t$ at time t, then the transformed features are given by

$$\vec{y}_t = A\vec{x}_t + \vec{b},$$

where A is the linear transform matrix and $\vec{b}$ is the added bias vector. The transform matrix A and the bias vector $\vec{b}$ are estimated by maximizing the likelihood of the feature vectors for a test speaker $Y = \{\vec{y}_1, \ldots, \vec{y}_T\}$, given a corresponding graph $\mathcal{G}$ which specifies the set of allowed state sequences. The graph used in estimation can be either the full decoding graph (which is like standard fMLLR) or a reduced silence-speech graph. We fix split the data into two classes and calculate the likelihood of the data models built on the corresponding class. The objective function is given as

$$l(A, \vec{b}) = \log \frac{dY}{dX} + P(\vec{y}_1^T | G, A, \vec{b}).$$

where $Y = \{\vec{y}_1, \ldots, \vec{y}_T\}$ and $X = \{\vec{x}_1, \ldots, \vec{x}_T\}$ are the corresponding transformed and original feature vectors for a test utterance respectively. We directly optimize this function by using the limited memory BFGS algorithm [6] with the More-Thuente line search algorithm [7]. Since we use a fairly simple speech/silence graph for estimation, it is fairly efficient to directly optimize the objective function instead of using an auxiliary function. In a real application one would still use the auxilliary function based method [1] for further gains in efficiency. This search algorithm requires both computation of the likelihood l and the gradient of the likelihood with respect to A and $\vec{b}$. If we can calculate the gradient of l w.r.t. $\vec{y}_t$, then we can propagate this gradient using the chain rule to calculate all the required gradients as follows

$$\frac{dl}{dA} = \frac{dl}{dY}\frac{dY}{dA} \tag{1}$$

and

$$\frac{dl}{d\vec{b}} = \frac{dl}{dY}\frac{dY}{d\vec{b}} \tag{2}$$

Let $\mathcal{G}$ be the set of allowed Gaussian sequences determined by the speech/silence alignment and the mixture models. Then we can write

$$l(A, \vec{b}) = \log P(Y|G, A, \vec{b}) = \log \sum_{g^n \in \mathcal{G}} P(g^n) P(Y|g^n).$$

The gradient l w.r.t. a given frame $\vec{y}_t$ is given by

$$\begin{aligned}
\frac{d \log P(Y|G)}{d\vec{y}_t} &= \frac{\log \sum_{g^n \in \mathcal{G}} P(g^n) P(Y|g^n)}{d\vec{y}_t} \\
&= \frac{\sum_{g^n \in \mathcal{G}} P(g^n) dP(Y|g^n)/d\vec{y}_t}{\sum_{g^n \in \mathcal{G}} P(g^n) P(Y|g^n)} \\
&= \sum_{g^n \in \mathcal{G}_t} \gamma_g(t) \frac{d \log P(\vec{y}_t|g)}{d\vec{y}_t} \\
&= \sum_{g^n \in \mathcal{G}_t} \gamma_g(t) \Sigma_g^{-1} (\mu_g - \vec{y}_t),
\end{aligned}$$

where $\mathcal{G}_t$ is the set of Gaussians that are allowed at time t according to the set of Gaussian sequences in $\mathcal{G}$; Σ, μ are the covariance matrix and the mean vector for the corresponding Gaussian. Plugging this result in Equations (1) and (2) we get

$$\frac{dg}{dA} = \sum_t \sum_{g \in \mathcal{G}_t} \gamma_g(t) \Sigma_g^{-1} (\mu_g - \vec{y}_t) \vec{x}_t^T,$$

and

$$\frac{dg}{d\vec{b}} = \sum_t \sum_{g \in \mathcal{G}_t} \gamma_g(t) \Sigma_g^{-1} (\mu_g - \vec{y}_t).$$

The Jocabian term can be easily calculated as

$$\log |\det \frac{dY}{dX}| = \sum_t \log |\det \frac{d\vec{y}_t}{d\vec{x}_t}| = \sum_t \log |\det(A)|,$$

since for a simple linear transformation, the Jacobian term reduces to the determinant of the transform matrix.

3. SYSTEM

The experiments for this paper are carried on two different IBM internal databases. The first database is comprised of utterances recorded in a car environment at three different speeds; idling, 30 mph, 60 mph. This database is the primary test bed for our experiments since we plan to use this algorithm for embedded applications. The tasks in this test set includes commands, digits, addresses. The second database is comprised of utterances collected by a telephony deployment of IBM. For both test sets several different grammars are used depending on the tasks.

The embedded test set is comprised of 36537 words and each speaker has nearly 5.2 minutes of data corresponding to 100 utterances per speaker. We test the performance of our algorithm both using a single utterance or all the utterances for a particular speaker in the optimization algorithm. The training data for the embedded application is also collected in a car with three different speeds. Since most of the training data is collected in a stationary car, extra noise collected in a moving car is added to simulate noisy data. The database used for training consisted of 887110 utterances. The baseline acoustic model was word internal with 826 states and 10001 diagonal Gaussians. The front end is fairly standard; 13 dim MFCC with mean normalization (max normalization) and delta and double deltas. The final feature is then 39 dimensional.

The speech silence models are also generated from the same training database. The speech silence labels are generated by using the silence detector of IBM ViaVoice speech recognition engine. The silence detector is based on the cepstral features. The silence detector is part of the front end and it is computationally cheap and low latency. A GMM for speech and silence is built from the speech/silence tagged vectors.

The telephony test set comprised 10126 utterances (approximately 4.2 hours). The test set has several different tasks including, yesno, digits, orders and navigation and was collected from several different telephony applications. There is approximately 2 seconds of data per utterance. All estimation was done using a single utterance since we did not have speaker boundaries for some of the test data. The base line acoustic model is left context with 2335 states and 167929 diagonal Gaussians. Feature vectors are generated by concetenating 9 consecutive MFCC features, which are then transformed to 60 dimensional feature vectors with an LDA transformation. The MFCC features are also normalized with CMS both at test time and training. We apply our algorithm either to 13 dimensional MFCC features before LDA or to 60 dimensional features after LDA. For these experiments we use the speech detector described in [8].

4. RESULTS

We first report results for the embedded test set. For the embedded test set, the base error rate is 1.70. We first test the linear transform part of the FMLLR, i.e., only A no $\vec{b}$. In the first set of experiments, we use all the data available for each speaker in estimation of the transform matrix A, i.e., nearly 5.2 minutes of data per speaker. In Table 1, we provide results when we maximize likelihood of a model with a single Gaussian per class. Since we are transforming the features to match a simple model (which is different from the model used for recognition) it is possible that we learn transforms that cause a degradation in performance. Indeed from Table 1 (row 2) we see that learning an unconstrained matrix causes the error rate to go up by a factor of 4. We then tried constraining the transform A to improve generalization. The best performance (row 5) is obtained when we constrain A to be diagonal with all the diagonal entries the same, i.e., $A = aI$ where $a \in R$ and I is the identity matrix of appropriate dimension. In the result in row 3, we only scale c_0 leaving all the other cepstral coefficients fixed. In row 4, we scale c_0 and the rest of the cepstral coefficients separately with the coeffiecients c_1 to c_{12} sharing the same scale. These results indicate that when matching to simple models it is best to severly constrain the transforms.

We then moved to experiments wehere we estimate the transform based on only one utterance (which would be the way in which these techniques would be used in practice). As expected, based on our previous experiments, we get the best performance when $A = aI$. Hence, we report our experiments for this configuration only. To approximate the original full model better, we increase the number of Gaussians in the GMM per class. In Table 2 we provide results for different number of Gaussians per class. Contrary to expectations, the performance of the algorithm remained effectively the same until 4 Gaussians and degraded when we increased the number of Gaussians further.

We then experimented with including the bias term $\vec{b}$ in the feature transformation. We expect only a small potential gain by using $\vec{b}$, since the features are already normalized (CMS) during training and testing. In Table 3, we present the results with different constraints of A and $\vec{b}$.

N_G	Configuration	WER
-	Baseline	1.70%
1	Unconstrained	8.91%
1	Diag. A, only c_0	2.69%
1	Diag. A, c_0, c_1-c_{12}	4.57 %
1	Diag. A, aI	1.50%

Table 1. Adaptation with different configuration of A where A are trained with using all the test data per speaker, i.e., approximately 5.2 minutes.

All the experiments for $\vec{b}$ are done per utterance. Once

N_G	Configuration	WER
-	Baseline	1.70%
1	Diag. $A = aI$	1.52%
2	Diag. $A = aI$	1.59%
4	Diag. $A = aI$	1.60 %
8	Diag. $A = aI$	1.80%
16	Diag. $A = aI$	6.22%

Table 2. Adaptation with different configuration of A, where A are trained using test data per utterance, i.e., approximately 3 seconds.

N_G	Configuration	WER
-	Baseline	1.70%
1	no A, $\vec{b}$	2.90%
1	Diag. $A = aI$, $\vec{b}$	2.49%
1	A, $\vec{b}$	22.99 %
1	Diag. $A = aI$, $\vec{b} = b\vec{1}$	1.46%

Table 3. Adaptation with different configuration of $\vec{b}$ where $\vec{b}$ are trained with using only the test data per utterance, i.e., approximately 3 seconds.

again we observe that we need to constrain the transform to be simple to achieve perfomance improvements. When we use 2 parameters ($A = aI$, $\vec{b} = b\vec{1}$ $a, b \in R$) we achieve a WER of 1.46% which is a relative improvement of 15 percent over the baseline.

N_G	Performance	Baseline	$A = aI$, $\vec{b} = b\vec{1}$
1	CA	71%	72 %
1	FAin	5%	5 %
1	CR	6 %	6 %
1	FAout	17 %	16 %

Table 4. Experiments with telephony set. Adaptation with $A = aI$, $\vec{b} = b\vec{1}$, $a, b \in R$ where parameters are trained with using only the test data per utterance. CA:correct acceptance, FAin: false acceptance for in grammar utterances, CR: correct rejection, FAout: false acceptance for out of grammar utterances.

Next we report results of our experiments on our telephony database. For these test sets we have several out of grammar utterances so we use correct/false acceptance and correct/false rejection rate as the comparison criteria. The results for the test set is given in Table 4 for $A = aI$, $\vec{b} = b\vec{1}$, $a, b \in R$ which (as before) is the best configuration in terms of performance. Unlike the earlier test set, we observe no improvement in performance. We believe that the proposed algorithm works for embedded environment since the embedded environment is fairly noisy unlike our telephony test set.

5. CONCLUSION

In this paper we introduced a adaptation technique using FMLLR with a simplified probabilistic model for estimation of transformation parameters. The algorithm can be directly implemented by the front end since it needs only binary tagging of feature vectors by a speech/silence classifier which is usually part of the front end in standard speech recognizers. The algorithm can also be viewed as maximum likelihood generalizations of CMS. We observe that even using a simple speech/silence model, we obtain a relative improvement of 15% over the base rate. The algorithm requires minimal computation and storage. The algorithm is effective mainly for noisy environments. The performance of the algorithm can be improved by increasing the complexity of the model used in optimization. We are in the process of trying to use simplified graphs based on vowels and consonants, which will approximate the full model better than a speech/silence graph. We also believe that the algorithm will give better performance when it is used both for training and testing.

6. REFERENCES

[1] M. J. F. Gales, "Maximum likelihood linear transformations for HMM-based speech recognition," *Technical report, TR 291, Cambridge University*, 1997.

[2] K. Visweswariah and R. A. Gopinath, "Adaptation of front end parameters in a speech recognizer," in *Proceedings of ICSLP*, 2004.

[3] P. Olsen and R. A. Gopinath, "Modeling inverse covariances in gaussian mixture models," in *Proceedings of ICASSP*, 2002.

[4] V. Goel, K. Visweswariah, and R. Gopinath, "Rapid adaptation with linear combinations of rank-one matrices," in *Proceedings of ICASSP*, 2002.

[5] S. V. Balakrishnan, "Fast incremental adaptation using maximum likelihood regression and stochastic gradient descent," in *Proceedings of Eurospeech*, 2003.

[6] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization problems," *Mathematical Programming*, vol. 45, pp. 503–528, 1989.

[7] T. More, "Line search algorithms with guaranteed sufficient decrease," *ACM TOMS*, vol. 20, no. 3, pp. 286–307, 1994.

[8] E. Marcheret, K. Visweswariah, and G. Potamianos, "Speech activity detection fusing acoustic phonetic and energy features.," in *Proceedings of Eurospeech*, 2005.