Fast Speaker Adaptive Training for Speech Recognition

Daniel Povey, Hong-Kwang J. Kuo, Hagen Soltau

IBM T.J. Watson Research Center, Yorktown Heights, NY, USA
{dpovey,hkuo,hsoltau}@us.ibm.com

Abstract

In this paper we describe various fast and convenient implementations of Speaker Adaptive Training (SAT) for use in training when Maximum Likelihood Linear Regression (MLLR) is to be used in test time to adapt Gaussian means. The memory and disk requirements for most of these are similar to those for normal ML training; the computation in all cases is dominated by the need to compute the MLLR transforms. Commonly MLLR is combined with Constrained MLLR (CMLLR) which can be viewed as a feature space affine transform and has its own form of SAT (we will call this CMLLR-SAT); we experiment with combining the two forms of SAT. We find that even on top of CMLLR-SAT, MLLR-SAT gives improvements.

Index Terms: speech recognition, speaker adaptation, speaker adaptive training, linear regression

1. Introduction

Maximum Likelihood Linear Regression (MLLR) [1] is a commonly used speaker adaptation technique used for adapting Gaussian mean vectors in Hidden Markov Model based speech recognition. The means are adapted per speaker using an affine transform:

$$\hat{\mu}_j^{(s)} = \mathbf{A}^{(s)}\mu_j + \mathbf{b}^{(s)}. \quad (1)$$

This may be combined with the use of regression classes [2], in which the transform parameters $\mathbf{A}^{(s)}$ and $\mathbf{b}^{(s)}$ depend on the particular Gaussian j; sometimes there are just two transforms for silence and non-silence, sometimes a variable number based on a tree of clustered phones.

Speaker Adaptive Training as used for MLLR [4] is a method of maximizing the likelihood of the training data given the MLLR-adapted models. It involves maximizing a quadratic objective function for each Gaussian mean μ_j. Various approaches have been proposed to implement it [5] but they all appear to involve storing statistics per speaker on disk at least up to Gaussian counts, and they all appear to have excessive disk and/or memory requirements when the number of speakers becomes very large.

The techniques that we present here avoid storing on disk any speaker-specific statistics during training. We demonstrate different approaches, one (the most exact) that stores statistics similar in size to the statistics needed to train full-covariance Gaussians, and others that store statistics similar in size to statistics used for normal ML training with diagonal Gaussians. The time taken by all of these methods is dominated by the computation of the MLLR transform itself which is always going to be necessary for any SAT implementation.

We also investigate the combination of two forms of SAT, one corresponding to MLLR (the algorithm which we describe here) and the other corresponding to Constrained MLLR (CMLLR) [3] which can be viewed as a feature space transform analogous to Equation 1. CMLLR-SAT simply consists of training on the adapted features. It has been reported [7] that MLLR-SAT gives improvements on top of CMLLR-SAT; we confirm this, although our improvements are somewhat smaller.

Section 2 introduces existing methods for SAT training; Section 3 describes an exact method for SAT training which avoids storing any per-speaker statistics (but involves statistics equivalent in size to full-covariance statistics), and Section 4 describes some more efficient but less exact techniques. Section 5 describes our experimental conditions; Section 6 describes our results and Section 7 concludes.

2. Implementations of SAT

In [5], various implementation methods for SAT are discussed. The first one (standard SAT) involves storing on disk standard diagonal statistics for each speaker independently. In the update phase, all speaker statistics are read in (they must either be stored in memory or read in twice, once for the mean update and once for the variance update). First the updated means are computed by maximizing a quadratic objective function for each mean, and then the variances are computed given the updated means and the stored statistics. This method is impractical for large training corpora with many speakers due to excessive use of disk space and excessive per-speaker computation during the update phase.

Another method (2-pass SAT) is also discussed. This method updates the means and variances on two separate passes over the data. For the means, as in all SAT methods we are maximizing a quadratic objective function in the feature dimension d. The mean-update phase of 2-pass SAT stores on disk the linear term in this objective function for each mean and also stores the speaker-specific adaptation matrices and the speaker-specific counts for each Gaussian. These last two are read from disk and used together in the mean-update phase to compute the quadratic term in the objective function for each mean, which is a matrix to be inverted. From this and the stored linear term the means are computed. The variance-update phase is simple; we go over the data and accumulate the diagonal variance of the observations around the speaker-adapted means. This method is also inconvenient when the number of speakers is large as the disk space per speaker is still substantial. The "Fast SAT" (FSAT) method [6] combines these two passes into one via an approximation.

Other methods, called Inverse Transform SAT and Least Squares SAT, are also discussed in [5] but they are not exact solutions and do not give as good results. Note that the efficient solutions we propose here are also inexact but in a different sense: if they do converge they will converge to a maximum of the likelihood.

3. Direct 2-pass SAT

We will first introduce a fast method of SAT training we call direct 2-pass SAT. This is an exact method that avoids dumping any speaker-specific statistics at all. The 2 passes of SAT training relate to the mean and variance respectively. Mixture weights can be updated on both passes if desired but we will not describe this since it is trivial. This method is like 2-pass SAT [5] except we compute the quadratic term in the objective function for each mean during the accumulation phase rather than the update phase.

We will write the observations for each speaker s and time $1 \leq t \leq T_s$ as $\mathbf{x}_s(t)$. The Gaussian posteriors for speaker s and time t are $\gamma_j(s,t)$. We always compute these posteriors using the most recent speaker-adapted parameters possible (we could make this explicit using iteration indices but it should be clear). The count, mean and variance statistics (i.e. the zeroth, first and second order statistics) for speaker s are:

$$\gamma_j^{(s)} = \sum_{t=1}^{T_s} \gamma_j^{(s)}(t) \quad (2)$$

$$\mathbf{x}_j^{(s)} = \sum_{t=1}^{T_s} \gamma_j^{(s)}(t)\mathbf{x}_s(t) \quad (3)$$

$$\mathbf{S}_j^{(s)} = \sum_{t=1}^{T_s} \gamma_j^{(s)}(t)\mathbf{x}_s(t)\mathbf{x}_s(t)^T. \quad (4)$$

We only store the diagonal of $\mathbf{S}_j^{(s)}$. In the mean-accumulation phase, we are accumulating a vector $\mathbf{v}_j$ and a matrix $\mathbf{M}_j$ for each Gaussian j which are the terms in a quadratic auxiliary function for its mean:

$$f(j) = \mu_j^T \mathbf{v}_j - 0.5\mu_j^T \mathbf{M}_j \mu_j \quad (5)$$

$$\mathbf{v}_j = \sum_{s=1}^{S} \mathbf{A}^{(s)^T} \Sigma_j^{-1} \left(\mathbf{x}_j^{(s)} - \gamma_j^{(s)}\mathbf{b}^{(s)}\right) \quad (6)$$

$$\mathbf{M}_j = \sum_{s=1}^{S} \gamma_j^{(s)} \mathbf{A}^{(s)^T} \Sigma_j^{-1} \mathbf{A}^{(s)} \quad (7)$$

$$\mu_j := \mathbf{v}_j \mathbf{M}_j^{-1}. \quad (8)$$

As we process each speaker we store the speaker-specific count and mean statistics in memory and then at the end of the speaker's data we directly increment $\mathbf{v}_j$ and $\mathbf{M}_j$. Each parallel process will dump out $\mathbf{v}_j$ and $\mathbf{M}_j$ to disk. The disk and memory usage is the same as for full-covariance training. The computation is $O(d^3)$ for each active Gaussian for each speaker, because of the matrix multiply needed to accumulate $\mathbf{M}_j$. This is not excessive because we need the same time to compute the speaker's MLLR transform itself, using standard approaches (although note that we could reduce this to $O(d^2)$ using ideas we previously introduced in [10]).

The variance update in this direct 2-pass SAT is the same as in normal SAT, and is quite simple: we just accumulate the variance around the adapted mean (we only need the diagonal).

$$\gamma_j = \sum_{s=1}^{S} \gamma_j^{(s)} \quad (9)$$

$$\mathbf{S}_j = \sum_{s=1}^{S} \mathbf{S}_j^{(s)} - 2\mathbf{x}_j^{(s)}\ \hat{\mu}_j^{(s)^T} + \gamma_j^{(s)}\hat{\mu}_j^{(s)}\ \hat{\mu}_j^{(s)^T} \quad (10)$$

$$\mathbf{\Sigma}_{j\,d,d}^2 := (1/\gamma_j)\mathbf{S}_{j\,d,d} \quad (11)$$

3.1. Direct single-pass SAT

To save a factor of two in speed we can combine the two passes into a single pass and accumulate the mean and variance statistics simultaneously, as in FSAT [6, 7]. This makes the technique not provably correct (at least using normal methods), but the interaction between the mean and variance updates is weak enough it is hard to construct an example where it would fail to converge. We can, however show that close to convergence it would converge the same as a fully correct approach because the change in the mean affects the updated variance in a quadratic, not linear way.

4. Diagonal SAT

Diagonal SAT is a modification of the mean update of the above algorithm (Direct single-pass SAT). In Diagonal SAT we only store the diagonal of the quadratic term in the mean's objective function. This requires that we formulate the quadratic objective function in terms of the change in the mean rather than the mean itself, to maintain the correct fixed point. This algorithm does have the potential for divergence but in practice it seems very stable. The auxiliary function for each mean is based on Δ_j which is the change in μ_j.

$$f(j) = \Delta_j^T \mathbf{v}_j - 0.5\Delta_j^T \mathbf{M}_j \Delta_j \quad (12)$$

$$\mathbf{v}_j = \sum_{s=1}^{S} \mathbf{A}^{(s)^T} \Sigma_j^{-1} \left(\mathbf{x}_j^{(s)} - \gamma_j^{(s)}\hat{\mu}_j^{(s)}\right) \quad (13)$$

$$\mathbf{M}_j = \mathrm{diag}\left(\sum_{s=1}^{S} \gamma_j^{(s)} \mathbf{A}^{(s)^T} \Sigma_j^{-1} \mathbf{A}^{(s)}\right) \quad (14)$$

$$M_{j\,d,d} = \sum_{s=1}^{S} \gamma_j^{(s)} \sum_{e=1}^{D} {A_{e,d}^{(s)}}^2 / \sigma_{j\,e}^2 \quad (15)$$

$$\mu_j := \mu_j + \mathbf{v}_j \mathbf{M}_j^{-1} \quad (16)$$

Here the notation diag means keeping the diagonal only of a matrix and setting other elements to zero. The variance accumulation and update are unchanged (Equations 9 to 11); we do the variance accumulation at the same time as the mean accumulation. Diagonal SAT requires one and a half times the disk space of normal ML accumulation and the computation required is negligible - after storing the diagonal statistics we have to do an $O(d^2)$ computation for each Gaussian active for the speaker which is less than the $O(d^3)$ computation we require on each iteration to estimate the MLLR transform itself.

4.1. ML-like SAT

ML-like SAT is a very simple form of SAT where we compute modified mean and variance statistics as for normal ML training that are constructed so our normal auxiliary function would have the same gradient w.r.t. the parameters as the SAT auxiliary function. This is based on the notion of "weak-sense auxiliary functions" introduced in [8].

$$\tilde{\mathbf{x}}_j^{(s)} = \gamma_j^{(s)}\mu + \Sigma_j^{(s)} \mathbf{A}^{(s)^T} {\Sigma^{(s)}}^{-1} \left(\mathbf{x}_j^{(s)} - \gamma_j^{(s)}\hat{\mu}_j^{(s)}\right) \quad (17)$$

$$\tilde{\mathbf{S}}_j^{(s)} = \left(\mathbf{S}_j^{(s)} - 2\mathbf{x}_j \hat{\mu}_j^{(s)} + \hat{\mu}_j^{(s)}\ \hat{\mu}_j^{(s)^T}\right) + 2\mu_j \tilde{\mathbf{x}}_j^{(s)} - \mu_j {\mu_j}^T \quad (18)$$

$$\mathbf{x}_j = \sum_{s=1}^{S} \tilde{\mathbf{x}}_j^{(s)} \quad (19)$$

$$\mathbf{S}_j = \sum_{s=1}^{S} \tilde{\mathbf{S}}_j^{(s)}. \quad (20)$$

Note that our variance statistics are constructed such that the "fake" statistics have the same variance around the Speaker Independent (SI) mean as the real features do around the SA mean. The update we use for ML-like SAT is the same as the normal ML update. Note that there is a potential for negative variances. In the experiments we report here, we floor these to a very small value, but it happens relatively rarely (around 1 in 1000 Gaussians if we start from a trained system, or more if we start from newly initialized mixtures). Note that ML-like SAT has a natural extension to Extended Baum-Welch based discriminative training as we can apply this transformation of statistics to the numerator and denominator statistics independently; SAT is known to give improvements when combined with discriminative training [6, 7].

4.2. Modified ML-like SAT

We also experiment with a modification to the update equations used in ML-like SAT in which we set the updated variance to the variance around the old mean rather than the new mean. In the context of normal ML training this is a valid but suboptimal update. In ML-like SAT it has the advantage of avoiding negative variances and possibly associated instabilities. We set the variance to:

$$\sigma^2_{j\,d} := \frac{S_{j\,d,d}}{\gamma_j} - 2\mu_{j\,d} x_{j\,d} + {\mu_j}^2_d, \tag{21}$$

where μ_j refers to the un-updated mean.

4.3. Overall procedure

The overall procedure for SAT training is as follows. On each iteration of SAT training, we split the speakers up into sets to be processed in parallel. A particular parallel process operates as follows: for each speaker, starting from an unadapted system it does several iterations (typically 2) of re-computing the speaker's MLLR transform and re-computing Gaussian posteriors. We then compute diagonal speaker-specific statistics (Equations 2 to 4) and use these to store SAT statistics using one of the various processes described above. Each process dumps the SAT statistics to disk when it has finished all its speakers. When all processes are done, the update process sums up the stored SAT statistics and computes the new model parameters.

5. Experimental Conditions

We test on two different setups, using English and Mandarin Broadcast News.

On the English side our training data is 50 hours of English news broadcasts obtained by subsampling the 1996 and 1997 Hub4 training sets (LDC97S44 and LDC98S71 respectively). Our test set is the Dev04f test set from the DARPA EARS project, which comprises 3 hours of speech from 6 broadcasts collected between 15 November and 1 December 2003, and includes 22.6K words. The language model used for testing is a 3.3M 4-gram LM trained on a corpus of 335M words. The features are 13 PLP coefficients spliced across nine frames and projected using LDA and then STC. We report results for SAT training on top of speaker-independent (SI) systems and systems that are speaker-adaptively trained with respect to VTLN and CMLLR, i.e. systems trained on adapted features. In both cases we have a small and a large system: the small ones both have 700 quinphone context dependent states and 5k mixtures and the larger ones have respectively 1K and 30K for the SI system and 3K and 50K for the SAT system.

We test with MLLR multiple regression classes; we use a regression tree with a minimum count of 3000 to estimate up to 16 transforms. There is no variance adaptation. All SAT training is with a single MLLR transform with a minimum count of 300 and minimum number of Gaussians seen of 80. In test we adapt (unsupervised) on text output from a previous phase of decoding which did not use SAT models, e.g. on an otherwise speaker independent system this would be an SI model, or in a system with VTLN and CMLLR this would be a model trained on VTLN and CMLLR features.

Our Mandarin system is an ML system adapted with VTLN and CMLLR built for the DARPA GALE program; the system architecture is broadly similar to the above but there are 20K states and 800K Gaussians and 1700 hours of training data. We report results on the basis of character error rate; the test sets are not official releases so the numbers may not be comparable to other sites' results.

6. Experimental Results

SAT	WER, 1 iter MLLR			
Iter	ML-like SAT	Modified ML-like	Diagonal SAT	Direct 1-pass SAT
(SI)	41.1%	41.1%	41.1%	41.1%
0	38.4%	38.4%	38.4%	38.4%
1	37.1%	37.2%	37.4%	37.4%
2	37.0%	37.1%	37.3%	37.2%
3	37.1%	37.1%	36.9%	37.0%
4	37.2%	37.1%	36.8%	36.9%
(all)	36.6%	36.6%	37.0%	36.8%
	WER, 2 iter MLLR			
0	38.4%	38.4%	38.4%	38.4%
4	36.8%	36.9%	36.8%	36.8%
(all)	36.4%	36.3%	36.6%	36.2%
	Train likelihood/frame			
1	-49.63	-49.63	-49.63	-49.63
2	-48.97	-48.98	-49.01	-49.00
3	-48.83	-48.83	-48.86	-48.85
4	-48.77	-48.76	-48.79	-48.78
(all)	-48.62	-48.61	-48.63	-48.62

Table 1: Speaker Independent, small (5k Gaussian)

SAT	WER, 1 iter MLLR			
Iter	ML-like SAT	Modified ML-like	Diagonal SAT	Direct 1-pass SAT
(SI)	34.1%	34.1%	34.1%	34.1%
0	31.9%	31.9%	31.9%	31.9%
1	30.9%	30.9%	30.8%	31.0%
2	30.9%	30.8%	30.8%	30.9%
3	30.9%	30.8%	30.8%	30.9%
4	30.7%	30.7%	30.7%	30.7%
(all)	30.2%	-	-	-

Table 2: Speaker Independent, normal (30K Gaussian)

Tables 1 and 2 show results from SAT training and MLLR on otherwise speaker independent systems, small and large ones respectively. In Table 1 we get very impressive improvements from SAT, up to about 2% absolute. Without SAT, MLLR reduces WER from 41.1% to 38.4%, and this is the same whether we do one iteration of MLLR in test time or two iterations. For SAT-adapted system, the iteration number on the left shows how many iterations of SAT training we did, between 1 and 4 iterations starting from a non-SAT system; "(all)" means we trained the system from fixed state alignments (20 iterations), doing

SAT on each iteration except the first. We can see that doing SAT training from the start this way gives more improvement. We can also see that when we are using SAT, it helps to do two iterations of MLLR adaptation in test time; the text used for adaptation is not recomputed, only the state and Gaussian-level alignment are changed as we re-adapt the system. The most improvement we get from SAT is 2.2% absolute, from a non-SAT system at 38.4% to a Direct 1-pass SAT system trained with SAT from the start with two iterations of MLLR in test time, which takes us to 36.2%.

With a more normal sized otherwise unadapted system in Table 2, we also get substantial improvements from SAT. We do not test multiple MLLR iterations here, but we get 1.2% absolute improvement from doing 4 iterations of SAT using any of the methods we test, or 1.7% absolute if we train with ML-like SAT from the start.

SAT	WER, 1 iter MLLR			
Iter	ML-like SAT	Modified ML-like	Diagonal SAT	Direct 1-pass SAT
(SI)	33.9%	33.9%	33.9%	33.9%
0	33.2%	33.2%	33.2%	33.2%
1	33.1%	33.1%	30.8%	33.0%
2	33.0%	32.9%	32.8%	33.0%
3	32.8%	32.8%	32.9%	32.9%
4	32.8%	32.7%	32.8%	32.8%

Table 3: Speaker Adapted, small (5k Gaussian)

SAT	WER, 1 iter MLLR			
Iter	ML-like SAT	Modified ML-like	Diagonal SAT	Direct 1-pass SAT
(SI)	26.0%	26.0%	26.0%	26.0%
0	25.3%	25.3%	25.3%	25.3%
1	25.2%	25.1%	25.2%	25.2%
2	25.2%	25.2%	25.2%	25.1%
3	25.1%	25.1%	25.1%	25.0%
4	25.1%	25.1%	25.1%	25.0%
(all)	24.9%	-	24.8%	-
	WER, 2 iter MLLR			
0	25.4%	25.4%	25.4%	25.4%
(all)	24.9%	-	24.9%	-

Table 4: Speaker Adapted, normal (50K Gaussian)

Tables 3 and 4 show SAT on a system with constrained MLLR (CMLLR) and VTLN applied both in train and test. With the smaller system (Table 3), MLLR gives us 0.7% absolute improvement without SAT, and with four iterations of SAT we get another 0.4% of improvement, regardless of the training technique. With the larger system, MLLR gives us 0.7% absolute improvement but four iterations of SAT give us only a further 0.2% to 0.3% absolute improvement. However, by doing SAT on all iterations the improvement from SAT increases to 0.4% to 0.5%. In this case, doing two iterations of MLLR in test do not seem to give us any further improvement (in fact, it hurts slightly). This may be because MLLR is not making enough difference to affect the alignments substantially.

Table 5 shows Diagonal SAT on a Mandarin system, also with VTLN and CMLLR. Averaging over the test sets, we get about 0.2% improvement from SAT. In another experiment (not shown), we saw very little improvement ($<$0.1%) from doing one iteration of ML-like SAT on a system built from scratch on top of discriminatively trained features.

SAT	CER, 2 iter MLLR, Diagonal SAT		
Iter	Dev07	Eval06	Eval07
0	15.2%	20.7%	13.5%
1	15.1%	20.6%	13.3%
2	15.1%	20.4%	13.2%
3	15.1%	20.4%	13.2%
4	15.0%	20.4%	13.2%

Table 5: Mandarin, speaker adapted (800K Gaussian)

7. Conclusions

We have presented various efficient and simple methods of SAT training. All the approaches give roughly the same performance so for efficiency and simplicity our recommendation is to use what we call Modified ML-like SAT or the more exact Diagonal SAT. We have shown the importance of SAT training where MLLR is the only form of adaptation used, and have confirmed the result from [7] that even when a system is trained and tested with VTLN and CMLLR, SAT can still give improvements. We have also shown that if possible SAT should be applied from early stages of training, and that on SAT systems we may need two iterations of MLLR estimation in test time. The compute time is dominated by the computation of the MLLR transform, so in future work we may try to speed this up [10] or recompute it less often. One unanswered question is whether it is important to use multiple regression classes in train as well as test.

8. Acknowledgements

This work was funded by DARPA contract HR0011-06-2-0001

9. References

[1] Leggetter, C. J. and Woodland, P. C., "Maximum Likelihood Linear Regression for Speaker Adaptation of Continuous Density Hidden Markov Models," Computer Speech and Language, v. 9, pp. 171-185, 1995.

[2] Gales M.J.F., "The generation and use of regression class trees for MLLR adaptation," Technical Report CUED/F-INFENG/TR263, Cambridge University, 1996.

[3] Gales, M. J. F., "Maximum Likelihood Linear Transformations for HMM-based Speech Recognition," Computer Speech and Language, v. 12, 1998.

[4] Anastasakos, T., McDonough, J., Schwartz R. and Makhoul, J., A Compact Model for Speaker-Adaptive Training," Proc. ICSLP, 1996.

[5] Matsoukas, S., Schwartz, R., Jin H. and Nguyen, L., "Practical Implementations of Speaker-Adaptive Training," 1997 DARPA Speech Recognition Workshop, Chantilly VA, Feb. 1997.

[6] McDonough, J., Schaaf, T., and Waibel, A., "On Maximum Mutual Information Speaker-Adapted Training," ICASSP 2002.

[7] McDonough, J., Wolfel, M., Stoimenov, E., "On Maximum Mutual Information Speaker-Adapted Training," "Computer Speech and Language," v. 22, Issue 2, pp. 130-147, 2008.

[8] Povey, D., "Discriminative Training for Large Vocabulary Speech Recognition," PhD Thesis, Cambridge University, 2003.

[9] Povey, D., "Improvements to fMPE for discriminative training of features," Proc. Interspeech, 2005.

[10] Varadarajan, B., Povey D., Chu, S., "Quick fMLLR for speaker adaptation in Speech Recognition," Proc. ICASSP, 2008.