

Fast incremental adaptation using maximum likelihood regression and stochastic gradient descent

Sreeram V. Balakrishnan

IBM T. J. Watson Research Center, Yorktown Heights, NY - 10598
sreevb@us.ibm.com

Abstract

Adaptation to a new speaker or environment is becoming very important as speech recognition systems are deployed in unpredictable real world situations. Constrained or Feature space Maximum Likelihood Regression (fMLLR) [1] has proved to be especially effective for this purpose, particularly when used for incremental unsupervised adaptation [2]. Unfortunately the standard implementation described in [1] and used by most authors since, requires statistics that require $O(n^3)$ operations to collect per frame. In addition the statistics require $O(n^3)$ space for storage and the estimation of the feature transform matrix requires $O(n^4)$ operations. This is an unacceptable cost for most embedded speech recognition systems. In this paper we show the fMLLR objective function can be optimized using stochastic gradient descent in a way that achieves almost the same results as the standard implementation. All this is accomplished with an algorithm that requires only $O(n^2)$ operations per frame and $O(n^2)$ storage requirements. This order of magnitude savings allows continuous adaptation to be implemented in most resource constrained embedded speech recognition applications.

1. Introduction

Constrained or Feature space Maximum Likelihood Linear regression fMLLR [1] has proved to be highly effective as a method for unsupervised incremental adaptation to a new speaker or environment. It requires only a single transform matrix and bias vector to be estimated, and is effective even with just a few seconds of data.

Adaptation is implemented through a feature space transform of the form

$$\hat{\mathbf{o}} = \mathbf{A}\mathbf{o} + \mathbf{b}$$

where \mathbf{o} are the speech frames, \mathbf{A} is the transformation and \mathbf{b} , the bias. Since there is only a single transform to estimate with only $n(n+1)$ parameters, adaptation works well even with little data. The standard implementation first outlined in [1] requires three steps.

1. Do an alignment of the adaptation speech with either a known transcript (supervised adaptation) or the results from recognition. The alignment results in the generation of probabilities $\gamma_m(t)$ which are the fractional allocation of frame $\mathbf{o}(t)$ to the m th Gaussian
2. Using the allocation probabilities compute the statistics matrices \mathbf{G}_i , $i = 1 \dots n$ where n is the size of the feature vector as follows:

$$\mathbf{G}_i = \sum_m \sum_t \gamma_m(t) P_{ii}^m \mathbf{f}(t) \mathbf{f}(t)^T \quad (1)$$

where \mathbf{P}^m is the precision matrix of the m th Gaussian and $\mathbf{f}(t)$ is the extended feature vector

$$f_1(t) = 1 \quad f_i(t) = o_{i-1}(t) \quad i = 2 \dots n + 1$$

Note: we assume diagonal covariance throughout this paper, however all the equations used can be extended to handle the full covariance case.

3. Use an iterative update scheme to estimate \mathbf{A} and \mathbf{b} from the \mathbf{G}_i . This scheme is based on maximizing the auxiliary function

$$Q(\mathbf{W}) = \log(\det(\mathbf{A})) - \sum_i \frac{1}{2} \mathbf{w}_i \mathbf{G}_i \mathbf{w}_i^T + \mathbf{w}_i \mathbf{k}_i \quad (2)$$

where \mathbf{w}_i is the i th row of $\mathbf{W} = [\mathbf{b} \ \mathbf{A}]$. Details of this scheme are given in [1]

The main drawback of the standard implementation is that to compute the n statistics matrices \mathbf{G}_i in step 2 requires $O(n^3)$ operations per frame of speech and each matrix requires storage for n^2 values. The cost of doing the alignment in step 1 and the final computation of the transform in step 3 are generally a fraction of the cost of step 2.

Several authors have tackled this problem by simplifying the transform matrix to either a block diagonal [1] or just diagonal form [3]. Although these reduce the order of computation, the overall effectiveness of the adaptation is sometimes compromised compared to using the full matrix [4]. Alternatives to block diagonal using rank-one basis matrices are explored in [4].

This paper presents an approach to reducing the computation by changing the method of optimizing the auxiliary function $Q(\mathbf{W})$. Instead of collecting statistics and then performing the row by row update of \mathbf{W} , $Q(\mathbf{W})$ is optimized directly using stochastic gradient descent. We show that this requires only $O(n^2)$ computations and $O(n^2)$ storage per frame of speech and results in minimal reduction in the effectiveness of the adaptation compared to the standard implementation.

The rest of this paper is organized as follows. In Section 2 we presented the basic idea and implementation of the stochastic gradient based optimization of $Q(\mathbf{W})$. For all gradient descent methods, rate of convergence is a critical issue. Section 3 tackles this issue and describes modifications to the basic implementation to improve the speed of convergence to a point that the stochastic gradient based method almost matches the performance of the standard iterative update scheme. Finally, in Section 4 we present the results on our car test database [5].

2. Stochastic Gradient Descent

The Stochastic Gradient Descent optimization of $Q(\mathbf{W})$ is based on estimating the gradient ∇Q with respect to the transform \mathbf{W} for each frame for speech. This gradient is accumulated over a block of B frames and used to update \mathbf{W} at the end of each block. If B is equal to the total number of frames of adaptation data T , then stochastic gradient descent reduces to simple gradient descent. Stochastic Gradient Descent is widely used in the Machine Learning community, where it is also known as online learning and is one of the primary methods

of learning the parameters of a feed-forward neural network. There is extensive work comparing its performance to standard gradient descent [6, 7].

In spite of its known drawbacks, the reason for using it for optimizing the fMLLR objective function is that $\nabla_{\mathbf{w}} Q$ can be computed in only $O(n^2)$ operations per frame. This can be seen by expanding $\nabla_{\mathbf{w}} Q$.

$$\nabla_{w_i} Q = \begin{bmatrix} 0 \\ \mathbf{a}_i^{-1} \end{bmatrix} - \mathbf{G}_i \mathbf{w}_i^T + \mathbf{k}_i \quad (3)$$

with \mathbf{a}_i^{-1} being the i th column of \mathbf{A}^{-1} and

$$\begin{aligned} \mathbf{G}_i \mathbf{w}_i^T &= \sum_m \sum_t \gamma_m(t) P_{ii}^m \mathbf{f}(t) \mathbf{f}(t)^T \mathbf{w}_i^T \\ &= \sum_t \left(\left(\sum_m \gamma_m(t) P_{ii}^m \right) \mathbf{f}(t) \left(\mathbf{f}(t)^T \mathbf{w}_i^T \right) \right) \\ &= \sum_t c_i(t) d(t) \mathbf{f}(t) \\ \mathbf{k}_i &= \sum_m \gamma_m(t) \mu_i^m P_{ii}^m \mathbf{f}(t) \\ &= e_i(t) \mathbf{f}(t) \\ \nabla_{w_i} Q &= \sum_t (e_i(t) - c_i(t) d(t)) \mathbf{f}(t) \\ &= \sum_t h_i(t) \mathbf{f}(t) \end{aligned} \quad (4)$$

where

$$\begin{aligned} c_i(t) &= \sum_m \gamma_m(t) P_{ii}^m \\ d(t) &= \mathbf{f}(t)^T \mathbf{w}_i^T \\ e_i(t) &= \sum_m \gamma_m(t) \mu_i^m P_{ii}^m \\ h_i(t) &= e_i(t) - c_i(t) d(t) \end{aligned} \quad (5)$$

The cost of computing $c_i(t)$ and $e_i(t)$ depends on the alignment, but typically the order of \sum_m can be kept quite small by ignoring all Gaussians in the alignment for which $\gamma_m(t) < \epsilon$. The cost of $d(t)$ is just a dot product of two vectors of size n . In addition, $d(t)$ is only computed once per frame. Thus the overall cost of computing the contribution to $\nabla_{\mathbf{w}_i} Q$ from frame t , is dominated by cost of accumulating $h_i(t) \mathbf{f}(t)$ which is of $O(n)$. Therefore, computation of all rows of $\nabla_{\mathbf{w}_i} Q$ requires $O(n^2)$ operations.

2.1. Details of Implementation

Let $\nabla_{w_i} Q(t_1, t_2)$ be the component of $\nabla_{w_i} Q$ for frames $t_1 \dots t_2$ where $t_1 < t_2$. The stochastic approximation consists of computing $\nabla_{w_i} Q(t_1, t_2)$ for small blocks of feature vectors, and updating \mathbf{W} using $\mathbf{w}_i = \mathbf{w}_i + \beta \nabla_{w_i} Q(t_1, t_2)$. The scale factor β is set to be sufficiently small to ensure stability, but is large enough to get meaningful gain in likelihood. Figure 1 shows the Pseudo-Code that describes the algorithm for Stochastic Gradient Adaptation.

The term $\alpha \Delta^{q-1} \mathbf{w}_i$ is required to include the effect of previous blocks. It is equivalent to the momentum term used in back-propagation training of neural networks. It serves to both smooth out some of the randomness in the stochastic gradient descent and account for the earlier gradients from previous blocks [7]. The values of $\alpha = 0.9$ and $\beta = 0.00001$ were found to give good results. The pseudo-code allows for multiple iteration on the same block. In practice one to four iterations were

```

for q = 1..num_blocks
  t1 = q * block_size, t2 = t1 + block_size - 1
  compute  $\Delta^q \mathbf{w}_i = \beta \nabla_{w_i} Q(t_1, t_2) + \alpha \Delta^{q-1} \mathbf{w}_i$ 
  update  $\mathbf{w}_i$  using  $\mathbf{w}_i = \mathbf{w}_i + \Delta^q \mathbf{w}_i$ 
  for iter=1 to num_updates(q)
    compute  $\Delta^q \mathbf{w}_i = \beta \nabla_{w_i} Q(t_1, t_2) + \alpha \Delta^q \mathbf{w}_i$ 
    update  $\mathbf{w}_i$  using  $\mathbf{w}_i = \mathbf{w}_i + \Delta^q \mathbf{w}_i$ 
  end
  adjust  $\beta$  to balance speed of convergence with stability
end

```

Figure 1: Pseudo-Code for implementation of Stochastic Gradient Descent

found to give good results. A larger number of iterations were found to be useful with the initial seconds of adaptation data, but once enough data is seen, the number of iterations can be reduced.

3. Improving the rate of convergence

The biggest concern with gradient descent is slow convergence. Short of using second order techniques such as Conjugate Gradients or other Quasi-Newton methods [8], Stochastic Gradient Descent has often proved to achieve faster convergence [7]. Both stochastic and normal gradient descent have been extensively studied in the Machine Learning community. Moller [7] gives a good summary of the pros and cons of the various schemes used to update feed-forward neural networks. For gradient descent, the underlying rate of convergence is directly determined by the condition number of the underlying Hessian of the objective function [7]. Ideally we would like the condition number to be one, in which case the gradient points in the direction of the maximum. With Stochastic Gradient Descent there is the further complication that the Hessian for each block is a noisy version of the true Hessian for all the data, and so some of the small eigenvalues are likely to vary dramatically. However any ‘‘whitening’’ of the eigenvalue spectrum of the expected Hessian for each block is likely to be beneficial. We propose two techniques to improve the conditioning of the expected Hessian for each block of data.

3.1. Scaling the means and variances

The first, consists of setting the average variance across all Gaussians to unity for each dimension. In other words scale the means and variances for element i of each each Gaussian by a factor λ_i so that

$$\sum_m \lambda_i p_m P_{ii}^m = 1$$

where p_m is a prior on the m^{th} Gaussian, with $\sum_m p_m = 1$.

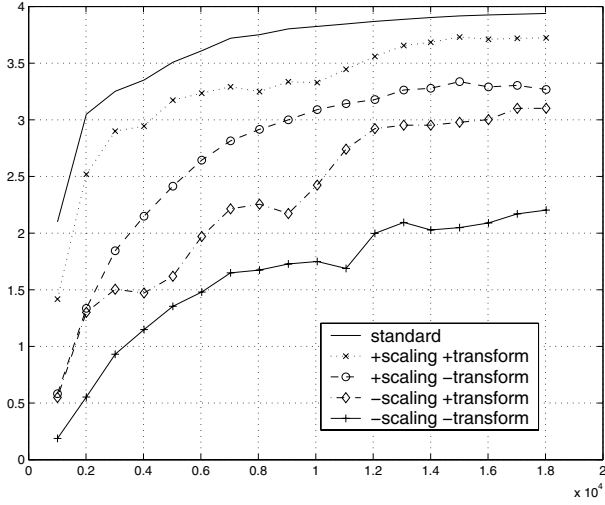
The means and variance of the Gaussians are then adjusted as follows

$$\begin{aligned} \hat{\mu}_i^m &= \frac{\mu_i^m}{\sqrt{\lambda_i}} \\ \hat{P}_{ii}^m &= \lambda_i P_{ii}^m \end{aligned}$$

The replacement of the old mean and variance μ_i^m and P_{ii}^m with $\hat{\mu}_i^m$ and \hat{P}_{ii}^m respectively, does not change the overall recognition results, as λ_i is applied to the feature vectors too:

$$\hat{f}_i(t) = \frac{f_i(t)}{\sqrt{\lambda_i}}$$

Figure 2: $Q(\mathbf{W})$ versus number of frames of adaptation data



3.2. Estimating the gradient in the transformed space

The second technique, is to estimate $\nabla_{w_i} Q(t_1, t_2)$ for block q , after applying the transformation \mathbf{W} computed for block $q-1$ to the feature vectors. Similar results for the standard technique have been found in [2]. $\nabla_{w_i}^q Q(t_1, t_2)$ is now computed as follows:

$$\nabla_{w_i}^q Q(t_1, t_2) = \begin{bmatrix} 0 \\ \mathbf{1}_i \end{bmatrix} - \sum_{t=t_1}^{t_2} (c_i(t)\hat{\mathbf{f}}(t)\hat{f}_i(t) + d_i(t)\hat{\mathbf{f}}(t)) \quad (6)$$

where

$$c_i(t) = \sum_m \gamma_m(t) P_{ii}^m \quad (7)$$

$$d_i(t) = \sum_m \gamma_m(t) \mu_i^m P_{ii}^m \quad (8)$$

$$[\mathbf{1}_i]_j = \delta_{ij} \quad (9)$$

$$\hat{\mathbf{f}}(t) = \hat{\mathbf{W}}^{q-1} \mathbf{f}(t) \quad (10)$$

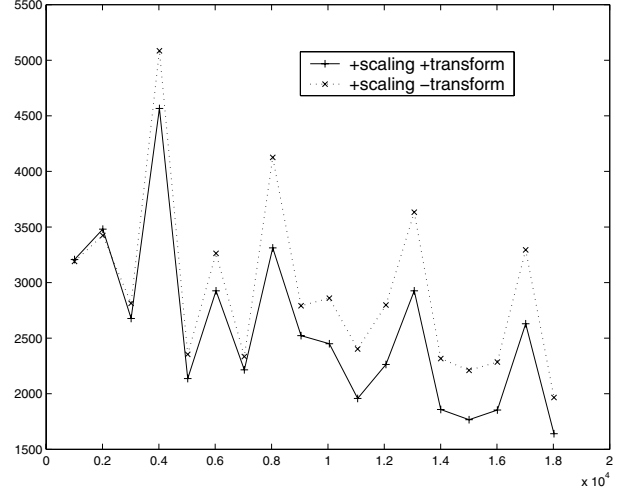
$$\hat{\mathbf{W}}^{q-1} = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{b}^{q-1} & \mathbf{A}^{q-1} \end{bmatrix} \quad (11)$$

In the transformed space, \mathbf{W} becomes $[\mathbf{0} \ \mathbf{I}]$ hence the need to compute the inverse of \mathbf{A} after each update is avoided.

Figure 2 illustrates the effects of the two techniques on the rate of convergence of the stochastic gradient based implementation of fMLLR. Each plot shows the value of $Q(\mathbf{W})$, given the estimate of \mathbf{W} after t frames of adaptation data using different methods of updating \mathbf{W} . The values of Q are calculated using \mathbf{G}_i 's and \mathbf{k}_i 's computed on all the data, whereas the \mathbf{W} 's at times less than T are computed from only part of the adaptation data. The top line shows the the value of Q for \mathbf{W} , estimated using the standard incremental technique with smoothing, based on [2]. The final value of Q at time T corresponds to the value that would be obtained with adaptation on all the data. Ideally, the stochastic implementation should follow this curve even at intermediate points as it represents the best that can be achieved with the standard implementation.

All values were computed for a set of 100 utterances of total length of 18000 frames, with a sampling rate of 67 frames a second. To obtain the curves for the stochastic implementation, the gradient scale β was set as large as it could be without causing the updates to become unstable (which results in a decrease

Figure 3: Condition number of Hessian using scaling versus number of frames of adaptation data



in the value of Q). Plots are shown for all four combinations of using scaling (+/-scaling) and computing the gradients in the transformed space (+/- transform). It can be seen that without using a combination of scaling the means and variances and computing the gradients in the transformed space, it is very hard to match the performance of the standard implementation.

If we ignore the determinant term in $Q(\mathbf{W})$, the Hessian for row i of the transform matrix would be $-\mathbf{G}_i$ where:

$$\mathbf{G}_i = \sum_m \gamma_m(t) P_{ii}^m \mathbf{f}(t) \mathbf{f}(t)^T$$

Hence we can compute an approximation to the condition number by finding the largest eigenvalue out of all the \mathbf{G}_i 's and dividing it by the smallest eigenvalue among all the \mathbf{G}_i 's. Figure 3 shows a plot of this approximate condition number for each block of frames as a function of the number of adaptation frames. The block size is 67 frames or ones second. As t increases, it can be seen that the approximate condition number of the Hessian in the transformed space gets lower, relative to the condition number in the non-transformed space. When no mean and variance scaling is used, the condition numbers are 10^4 times worse on average.

4. Experimental results

All experiments reported on in this paper were conducted on a test database collected in a car [5]. We report word error rates on a test set comprised of small vocabulary grammar based tasks (addresses, digits, command and control) that consists of 73743 words. Data for each task was collected at 3 speeds: idling, 30mph and 60mph. There are a 147 combinations of speaker, task and environment in the test set, and for each combination there are 100 test utterances.

We compare the performance of incremental adaptation using the standard implementation plus smoothing from [2] with the stochastic gradient descent implementation. Full and block diagonal (13x13 blocks for static, delta and delta-delta features) transforms are compared.

To simulate real word conditions where the speaker and environment may change quickly, the transforms and statistics were reset after various fixed intervals of utterances. The maximum interval is 100 since this corresponds to the length of each set of utterances in the test set for which the speaker, environment and task are fixed.

Figure 4: Word errors per block of 5 utterances versus number of adaptation utterances

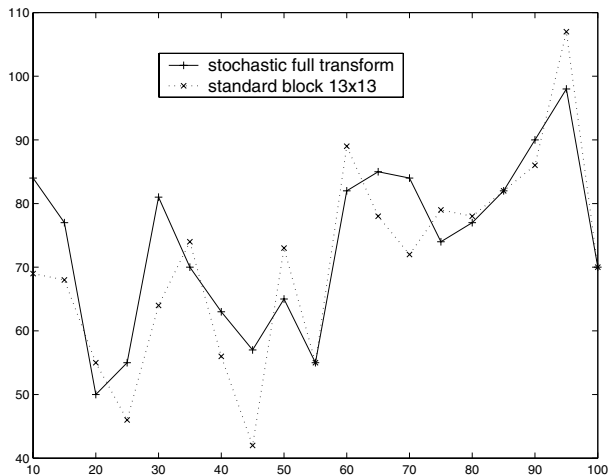


Table 1 shows the results comparing the standard technique with stochastic gradient descent on our car database.

Num Cell Adapts	Standard Full	Stochastic Full	Standard 13x13	Stochastic 13x13
0	2.81	2.81	2.81	2.81
5	2.66	2.66	2.24	2.53
10	2.34	2.48	2.09	2.44
20	2.12	2.21	2.02	2.26
100	1.94	2.00	1.92	2.05

Table 1: WER comparison of Standard fMLLR adaptation versus Stochastic Gradient Descent for Full and Block Diagonal 13x13 transforms

For full matrix transforms, it can be seen that the stochastic implementation compares very favourably in terms of error rate with the smoothed standard implementation. Especially considering it uses $O(n^2)$ computations per frame versus the $O(n^3)$ that are required to collect the statistics for the standard implementation. However, for this particular data set and model, restricting the transform to be block diagonal achieves even better results for both the standard implementation and stochastic implementation, especially with fewer utterances. In addition, using block diagonal transforms results in a reduction in computational load and storage by a factor of 9 for the standard implementation and factor of 3 for the stochastic method. Even with these reductions the stochastic method with full covariance requires less computation than the standard implementation of the block diagonal.

As the number of the cells increases, the full transform outperforms the block transform. Figure 4 plots the word errors for each block of 5 cells as adaptation progresses. The errors are summed over all 147 test sets of 100 cells each. It compares the errors made by the standard implementation of block diagonal, with the stochastic estimation of the full transform. It can be seen that after about 50 cells the full transform starts to outperform the block diagonal one even though it is being estimated using stochastic gradient descent.

5. Conclusions

A new computationally efficient method for estimating the transform in feature space maximum likelihood regression has been presented. It provides an order of magnitude of savings in computation when compared with the standard implementation of incremental fMLLR. Since it employs stochastic gradient descent, it is sensitive to the conditioning of the expected Hessian of the objective function. To achieve acceptable convergence rates, two techniques to improve the conditioning of the Hessian have been proposed. These are prescaling the means and variances and computing the gradients in the transformed space. Both are shown to be highly effective in improving convergence. The overall scheme enables continuous online adaptation using full rank transforms even for highly resource constrained speech recognizers. This is done with minimal impact on the effectiveness of the adaptation compared to using the standard method based on gathering statistics and using an iterative update scheme [1].

6. Acknowledgements

The author wishes to thank Scott Axelrod, Ramesh Gopinath, Peder Olsen and Karthik Visweswariah for numerous tips, helpful discussions and extensive help with the experimental set-up.

7. References

- [1] M.J.F. Gales, "Maximum likelihood linear transformations for hmm-based speech recognition," *CUED Technical Report tr291*, May 1997.
- [2] Y. Li, H. Erdogan, Y. Gao, and E. Marcheret, "Incremental on-line feature space mlr adaptation for telephony speech recognition," *International Conf. on Spoken Language Processing 2002*, Sept 2002.
- [3] V. Digalakis and L. Neumeyer, "Speaker adaptation using combined transformation and bayesian methods," in *Proc. ICASSP '95*, Detroit, MI, 1995, pp. 680–683.
- [4] V. Goel, K. Visweswariah, and R.A. Gopinath, "Rapid adaptation with linear combinations of rank-one matrices," in *Proc. ICASSP 2002*. ICASSP, 2002, vol. I.
- [5] P. Olsen and R. A. Gopinath, "Modeling inverse covariances in gaussian mixture models," in *Proc. ICASSP 2002*. ICASSP, 2002, vol. I.
- [6] L. Bottou and P. Gallinari, "A framework for the cooperation of learning algorithms," in *Advances in Neural Information Processing Systems*, Richard P. Lippmann, John E. Moody, and David S. Touretzky, Eds. 1991, vol. 3, pp. 781–788, Morgan Kaufmann Publishers, Inc.
- [7] M. Moller, *Efficient Training of Feed-Forward Neural Networks*, Ph.D. thesis, Computer Science Dept., Aarhus University, November 1997.
- [8] R. Fletcher, *Practical Methods of Optimization*, Wiley, 1987.