Feature adaptation using projection of Gaussian posteriors

Karthik Visweswariah, Peder Olsen

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

Abstract

In this paper we consider the use of non-linear methods for feature adaptation to reduce the mismatch between test and training conditions. The non-linearity is introduced by using the posteriors of a set of Gaussians to adapt the original features. Parameters are estimated to maximize the likelihood of the test data. The modeling framework used is based on the fMPE models [1]. We observe significant gains (17% relative) on a test data base recorded in a car. We also see significant gains on top of FMLLR (38% relative over the baseline and 8.5% relative over FMLLR).

1. Introduction

State of the art speech recognitions systems typically adapt their features and/or acoustic models to the test speaker to get improved recognition accuracy. In this paper we only consider adapting the features. Popular techniques for feature adaptation/normalization include spectral subtraction, Codeword Dependent Cepstral Normalization (CDCN) [2] and Feature space Maximum Likelihood Linear Regression (FMLLR) [3]. FMLLR is a linear technique where the features are linearly transformed to maximize the likelihood of the test data under a given fixed model. FMLLR differs from most of the other techniques in that no explicit assumptions are made about the type of noise or channel. Although FMLLR has been quite successful, several attempts have been made at generalizing the technique to allow for non-linear transforms of the feature vectors [4], [5]. [4] and [6] consider non-linear transforms at training time. In this paper we present a non-linear method for feature adaptation that is based on the fMPE technique for discriminatively estimating improved features. We borrow the basic feature transformation model from [1], but we estimate the parameters to maximize likelihood. The feature transformation adds to the original features a projection of posteriors calculated from the original features using a given Gaussian Mixture Model (GMM). Although we use the posteriors from a GMM to introduce the non-linearity, the methods used to estimate parameters will be independent of the actual non-linearity used.

The rest of this paper is organized as follows. In Section 2 we describe the feature transformation model and the objective we use to estimate the parameters. In Section 3 we present the technique used to estimate parameters. Section 4.2 describes the databases and the experimental setup used to evaluate our techniques, and presents our results on this database. We present our conclusions and some directions for future work in Section 5.

2. Feature transformation model and objective function

Let us denote the feature vector at time t by $\mathbf{x}_t$. Then the basic model we use to generate the transformed feature $\mathbf{y}_t$ is

$$\mathbf{y}_t = \mathbf{A}\mathbf{x}_t + \mathbf{B}\phi(\mathbf{x}_t),$$

where ϕ is some non-linear function that maps d dimensional vectors into D dimensional vectors and $\mathbf{B}$ is a projection matrix of size $d \times D$. Note that if we fix $\mathbf{A}$ to be identity then we are only using the non-linear part of the transform, and if we fix $\mathbf{B}$ to be zero then we are only applying a linear transform as in FMLLR.

Although the estimation techniques apply to general ϕ, in this paper we only consider the use of Gaussian posteriors [1]. We assume we have a given fixed GMM with N_G Gaussians which we use to calculate the gth component of $\phi(\mathbf{x}_t)$ as:

$$\phi_g(\mathbf{x}_t) = \frac{\pi_g N(\mathbf{x}_t; \mu_g, \mathbf{\Sigma}_g)}{\sum_k \pi_k N(\mathbf{x}_t; \mu_k, \mathbf{\Sigma}_k)},$$

where $N(\mathbf{x}; \mu, \mathbf{\Sigma})$ denotes the likelihood of $\mathbf{x}$ under a Gaussian density with mean μ and covariance $\mathbf{\Sigma}$.

We would like to estimate our parameters $\mathbf{B}$ (and possibly $\mathbf{A}$) to maximize the likelihood of the test data. For this to be valid we need to ensure that the feature transform we are using is invertible and compensate the likelihood with the log determinant of the Jacobian. Let M denote the GMM and G denote a graph which specifies a set of allowed state sequence. Then the objective function we need to maximize is:

$$g(\mathbf{A}, \mathbf{B}) = \log \left| \det \frac{d\mathbf{Y}}{d\mathbf{X}} \right| + \log P(\mathbf{Y}|M, G),$$

where $\mathbf{X}$ denotes all the acoustic features for a particular test speaker and $\mathbf{Y}$ denotes the transformed acoustic features for that speaker. We split this into the Jacobian term and the likelihood term which are handled differently:

$$g_{\mathrm{J}}(\mathbf{A},\mathbf{B}) = \log\left|\det\frac{d\mathbf{Y}}{d\mathbf{X}}\right|$$

and

$$g_{\mathrm{L}}(\mathbf{A},\mathbf{B}) = \log P(\mathbf{Y}|M,G).$$

Note that since $\mathbf{y}_t$ is a function of only $\mathbf{x}_t$,

$$\log\left|\det\frac{d\mathbf{Y}}{d\mathbf{X}}\right| = \sum_t \log\left|\det\frac{d\mathbf{y}_t}{d\mathbf{x}_t}\right|.$$

Ensuring that our transform is invertible is equivalent to ensuring that the Jacobian is full rank for all $\mathbf{X}$. This is hard to do in general, and we do not deal with the issue rigorously. We note that the log determinant term in the objective function goes to negative infinity when the Jacobian becomes singular. We assume this will prevent us, in practice, from making the transform non-invertible.

3. Parameter estimation

We use the limited memory BFGS algorithm [7] with the More-Thuente line search algorithm [8] as implemented in [9] to minimize g. This requires computation of g and its gradient with respect to $\mathbf{A}$ and $\mathbf{B}$. Each computation of g requires a pass through the adaptation data. Note that we do not use an auxiliary function to optimize the objective function. Using an auxiliary function does not give us the usual benefit of being able to go through the data once and collect sufficient statistics, which can be used to perform the optimization. This is because of the Jacobian term g_{J}, for which we need to run through the data each time we want to calculate it and its gradient.

Let us now go into the calculation of g_{L} and its gradient. First we note that if we can calculate gradient g_{L} w.r.t $\mathbf{y}$ then we can propagate this gradient using the chain rule to calculate all required gradients as follows:

$$\frac{dg_{\mathrm{L}}}{dA} = \frac{dg_{\mathrm{L}}}{d\mathbf{Y}}\frac{d\mathbf{Y}}{d\mathbf{A}} \tag{1}$$

and

$$\frac{dg_{\mathrm{L}}}{dB} = \frac{dg_{\mathrm{L}}}{d\mathbf{Y}}\frac{d\mathbf{Y}}{d\mathbf{B}}. \tag{2}$$

Let $\mathcal{G}$ be the set of Gaussian sequences determined by the model M and the graph G. Then we can write:

$$g_{\mathrm{L}} = \log P(\mathbf{Y}|M,G) = \log\sum_{g^n\in\mathcal{G}} P(g^n)P(\mathbf{Y}|g^n).$$

The gradient $g_{\mathrm{L}}(\mathbf{y})$ w.r.t a given frame $\mathbf{y}_t$ is given by:

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

where $\mathcal{G}_t$ is the set of Gaussians that are allowed at time t according to the set of Gaussian sequences $\mathcal{G}$. Plugging this result into Equations 1 and 2 we get

$$\frac{dg_{\mathrm{L}}}{d\mathbf{A}} = \sum_t\sum_{g\in\mathcal{G}_t}\gamma_g(t)\boldsymbol{\Sigma}_g^{-1}(\mu_g-\mathbf{y}_t)\mathbf{x}_t^T, \tag{3}$$

and

$$\frac{dg_{\mathrm{L}}}{d\mathbf{B}} = \sum_t\sum_{g\in\mathcal{G}_t}\gamma_g(t)\boldsymbol{\Sigma}_g^{-1}(\mu_g-\mathbf{y}_t)\phi(\mathbf{x}_t)^T. \tag{4}$$

We now turn to the calculation of g_{J} and it's gradient. The Jacobian of our feature transform is

$$g_{\mathrm{J}} = \sum_t\log\left|\det\frac{d\mathbf{y}_t}{d\mathbf{x}_t}\right| = \sum_t\log\left|\det(\mathbf{A}+\mathbf{B}\frac{d\phi(\mathbf{x}_t)}{d\mathbf{x}_t})\right|.$$

Let us consider $\phi_g(\mathbf{x}_t)$ the gth component of $\phi(\mathbf{x}_t)$. The derivative of this is:

$$\frac{d\phi_g(\mathbf{x}_t)}{d\mathbf{x}_t} = \phi_g(\mathbf{x}_t)\Big(\boldsymbol{\Sigma}_g^{-1}(\mu_g-\mathbf{x}_t) - \sum_k\phi_k(\mathbf{x}_t)\boldsymbol{\Sigma}_k^{-1}(\mu_k-\mathbf{x}_t)\Big).$$

Note that this gradient is zero when $\phi_g(\mathbf{x}_t) = 0$ or $\phi_g(\mathbf{x}_t) = 1$. Thus the gradients of g_{J} are

$$\frac{dg_{\mathrm{J}}}{d\mathbf{A}} = \sum_t\left(\mathbf{A}+\mathbf{B}\frac{d\phi(\mathbf{x}_t)}{d\mathbf{x}_t}\right)^{-T}$$

and

$$\frac{dg_{\mathrm{J}}}{d\mathbf{B}} = \sum_t\left(\mathbf{A}+\mathbf{B}\frac{d\phi(\mathbf{x}_t)}{d\mathbf{x}_t}\right)^{-T}\frac{d\phi(\mathbf{x}_t)}{d\mathbf{x}_t}^T.$$

4. Experimental setup and Results

4.1. Training and test database description

The experiments reported on in this paper were performed on an IBM internal database [10]. The test data consists of utterances recorded in a car at three different speeds: idling, 30 mph and 60 mph. Four tasks are

included in the test set: addresses, digits, commands and radio control. Following are typical utterances from each task:

A: New York City ninety sixth street West.
C: Set track number to seven.
D: Nine three two three three zero zero.
R: Tune to F.M. ninety three point nine.

The test data base has 73743 words and each speaker has on the average 5.2 minutes of data.

Training data was also collected in a car at three speeds. Since most of the data was collected in a stationary car, the training data was augmented by adding noise collected in a car to the data collected in a stationary car. Data was collected with microphones in three different positions: rear-view mirror, visor and seat belt. 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 we use is fairly standard; MFCC (13 dimensional) with mean normalization (max normalization for c0) and delta and double deltas (final feature 39 dimensional).

All of our experiments are unsupervised adaptation experiments. We first decode the test data, and then used the decoded script to generate a forced alignment. This alignment is then used as the graph G used to calculate the likelihood g_{L}. We could use the decoding graph or the decode word script instead but past experience shows that this is usually no better than using an alignment of the decoded script.

4.2. Results

The baseline error rate on our test database is 2.08%. At the outset we note that for all adaptation experiments we ran the limited memory BFGS algorithm for 100 iterations for each speaker. This number of iterations was determined in some preliminary experiments, and is sufficient to achieve convergence of the WER. Table 1 shows the performance when we use the non-linear adaptation technique described above with a varying number of Gaussians in the secondary model used to compute the posteriors. The secondary model is created by using starting with the GMM corresponding to the full acoustic model and clustering (to minimize Kullback-Liebler divergence) to a desired number Gaussians. We see that the best performance is obtained with about 64 Gaussians. As the number of Gaussians is reduced we have very few parameters and this hurts performance. In the extreme that we have only one Gaussian in the secondary model whose posterior is always 1, we are reduced to a simple shift of the features. As we increase the number of Gaussians beyond a certain point we expect to degrade because of over training. Clearly the optimal point will depend on the amount of data for a certain speaker. In our test set each speaker has the same amount of data so we did not experiment with varying the number of Gaussians per speaker.

N_G	WER	Num. parameters
Baseline	2.08%	-
4	2.04%	156
8	1.94%	312
16	1.86%	624
32	1.79%	1248
64	1.72%	2496
128	1.73%	4992

Table 1: Adaptation with different number of Gaussians in secondary model

In calculating the posteriors in ϕ we could use an additional scale factor α as below:

$$\phi_g(\mathbf{x}_t) = \frac{\pi_g N(\mathbf{x}_t; \mu_g, \mathbf{\Sigma}_g)^\alpha}{\sum_k \pi_k N(\mathbf{x}_t; \mu_k, \mathbf{\Sigma}_k)^\alpha}.$$

We considered this option since choosing alpha appropriately can cause the posteriors to be smoother in their variation across time. Note that we could choose α by optimizing over α to maximize likelihood, which we did not do in this paper. The error rates at various α's are shown in Table 2. All of these results use 64 Gaussians in the secondary model. As α goes to zero the performance degradation is expected since the posterior distribution is uniform in the limit. Although the total performance across scales from 0.2-2.0 is pretty much the same and close to optimal, we maybe able to further improve the performance by allowing the scale to be speaker dependent. Picking the best scale out the four scales from 1.0 - 0.2 gives a total error rate of 1.62%. To practically obtain this improvement we could choose the scale to maximize likelihood, in fact we could let the scale be a parameter which is also optimized along with B.

α	WER
Baseline	2.08%
8.0	1.93%
4.0	1.84%
2.0	1.75%
1.0	1.72%
0.8	1.74%
0.4	1.72%
0.2	1.72%
0.1	1.80%
0.05	1.77%
0.01	1.91%
0.001	2.07%

Table 2: Adaptation with different scales in calculating secondary model posteriors

In our final set of experiments we tried to improve upon FMLLR with the non-linear adaptation technique introduced in this paper. All of these experiments used a scale of 0.8 for the non-linear features. Using just FMLLR we get to an error rate of 1.41%. Fixing the FMLLR matrix A and then training the B matrix to maximize likelihood gave us an error rate of 1.37%. In this configuration the features used were:

$$\mathbf{y}_t = \mathbf{A}\mathbf{x}_t + \mathbf{B}\phi(\mathbf{x}_t).$$

Another way of doing this is to let:

$$\mathbf{y}_t = \mathbf{A}\mathbf{x}_t + \mathbf{B}\phi(\mathbf{A}\mathbf{x}_t)$$

where A is trained by the standard FMLLR procedure, and then B is trained with A fixed. This configuration gave us an improved performance of 1.33%, which can be explained because the FMLLR matrix is used to compensate the features input to the non-linear transform. Once we estimate B we could reestimate a linear transform that is applied on top of the transform we have giving us:

$$\mathbf{y}_t = \mathbf{A}_2(\mathbf{A}_1\mathbf{x}_t + \mathbf{B}\phi(\mathbf{A}_1\mathbf{x}_t)).$$

This gives us an error rate of 1.29%, which is significantly better than the WER of 1.41% obtained only using FMLLR.

5. Conclusions and future work

In this paper we introduced a non-linear adaptation technique based on the FMPE feature generation model [1]. We see a 17% relative improvement using this non-linear technique. We also were able to obtain a modest 8.5% relative improvement over FMLLR, which was a 38% relative improvement over the baseline system. In the future we would like to explore the choice of ϕ, use larger secondary GMMs in conjunction with a map to a smaller number of classes to constrain the number of parameters and to allow the parameters of the secondary model to be changed to maximize likelihood of the test data. We would also like to use this maximum likelihood approach during training, as opposed to the original FMPE paper which uses the MPE criterion.

6. References

[1] D. Povey, B. Kingsbury, L. Mangu, G. Saon, H. Soltau, and G. Zweig, "FMPE: discriminatively trained features for speech recognition," in *Proceedings of ICASSP*, 2005.

[2] A. Acero and R. M. Stern, "Environmental robustness in automatic speech recognition," in *Proceedings of ICASSP*, 1990.

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

[4] P. Olsen, S. Axelrod, K. Visweswariah, and R. A. Gopinath, "Gaussian mixture modeling with volume preserving non-linear feature space transforms," in *Proceedings of ASRU*, 2003.

[5] S. Dharanipragada and M. Padmanabhan, "A nonlinear unsupervised technique for unsupervised adaptation," in *Proceedings of ICSLP*, 2000.

[6] M. K. Omar and M. Hasegawa-Johnson, "Non-linear maximum likelihood transformation for speech recognition," in *Proceedings of Eurospeech*, 2003.

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

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

[9] M. S. Gockenbach, W. W. Symes , "The Hilbert Class Library," http://www.trip.caam.rice.edu/txt/hcldoc/html/.

[10] S. Deligne, S. Dharanipragada, R. Gopinath, B. Maison, P. Olsen, and H. Printz, "A robust high accuracy speech recognition system for mobile applications," *IEEE Transactions on Speech and Audio Processing*, pp. 551–561, 2002.