Log-linear Optimization of Second-order Polynomial Features with Subsequent Dimension Reduction for Speech Recognition

Muhammad Ali Tahir.[1]*, Ralf Schlüter* [1]*, Hermann Ney* [1]

[1]Human Language Technology and Pattern Recognition
Computer Science Department, RWTH Aachen University, Aachen, Germany
{tahir,schlueter,ney}@cs.rwth-aachen.de

Abstract

Second order polynomial features are useful for speech recognition because they can be used to model class specific covariance even with a pooled covariance acoustic model. Previous experiments with second order features have shown word error rate improvements. However, the improvement comes at the price of a large increase in the number of parameters. This paper investigates the discriminative training of second order features, with a subsequent dimension reduction transform to limit the increase in number of parameters. The acoustic model parameters and the transformation matrix parameters are modeled log-linearly and optimized using maximum mutual information criterion. The advantage of log-linear optimization lies in its ability to robustly combine different kinds of features. Experiments are performed for second order MFCC features on the EPPS large vocabulary task and have resulted in a decrease in word error rate.

Index Terms: speech recognition, log-linear modeling, polynomial features

1. Introduction

The motivation for using polynomial features for speech recognition stems from support vector machines (SVM) where a polynomial kernel may be used to project feature vectors to a higher space [1]. Classes which are not linearly separable may become separable in that higher dimensional space, by fitting a hyperplane between them. MFCC (mel frequency cepstral coefficients) are a popular way of extracting information from the audio data, and projecting them to a higher dimension could increase separation between different phone classes. In [2] second order features have been used to train a large vocabulary system and has resulted in WER improvement. Another reason for using second order polynomial features is the fact that for a pooled covariance Gaussian HMM model they could be used to approximate a class specific covariance model. If the polynomial projection step is followed by a dimension reduction, then the number of additional parameters is small. This could bring the benefit of using a class specific covariance model while avoiding the overfitting risk associated with it. A similar work using class specific covariance model is [3]

In [4] it has been shown that the posterior form of Gaussian HMM can be represented as an HCRF model. For the case of a pooled covariance HMM this simplifies to a CRF or log-linear model. The usefulness of log-linear model lies in its convexity property according to the maximum entropy principle [5]. For a fixed feature vector to state alignment and a single density per state, the corresponding log-linear model has a global maximum, that can be reached regardless of the intial values of parameters, also shown by experiments in [6]. Another similar work is [7] although it assumes a different structure of Markov model. In addition to the convexity, another useful property of the log-linear models is that it could be used to combine features from different knowledge sources [8].

2. Log-Linear Mixture Models

Let the speech feature vectors x_1^T belong to one of $s = 1, ..., S$ triphone classes, each class with Gaussian parameter set $\theta_s = \{\mu_s, \Sigma_s\}$. Then the posterior probability is

$$
\begin{aligned}
p_\theta(s|x) &= \frac{p(s)p_\theta(x|s)}{\sum_{s'} p(s')p_\theta(x|s')} \\
&= \frac{\exp(x^\top \Lambda_s x + \lambda_s^\top x + \alpha_s)}{\sum_{s'} \exp(x^\top \Lambda_{s'} x + \lambda_{s'}^\top x + \alpha_{s'})}
\end{aligned}
\quad (1)
$$

The conversion is done by expanding $p_\theta(x|s)$ in its Gaussian form and collecting the terms of x. In the final equation, the new parameters $\Lambda_s \in \mathbf{R}^{\mathbf{D}\times\mathbf{D}}$, $\lambda_s \in \mathbf{R}^{\mathbf{D}}$ and $\alpha_s \in \mathbf{R}$ are present in log-quadratic form. Note that here the posterior probability is directly modeled and the exponential term does not conform to the constraints of a probability distribution.

Using a pooled covariance matrix Σ can lead to further simplification.

$$
p_\theta(s|x) = \frac{\exp(\lambda_s^\top x + \alpha_s)}{\sum_{s'} \exp(\lambda_{s'}^\top x + \alpha_{s'})} \quad (2)
$$

the resulting form is log-linear with no squared term in the exponent. However, a squaring of the input features x as in Section 3 leads to a similar behavior as in equation 1.

In case of a mixture densities, the corresponding posterior probability is

$$
p_\theta(s|x) = \frac{\sum_l \exp(\lambda_{s,l}^\top x + \alpha_{s,l})}{\sum_{s',l} \exp(\lambda_{s',l}^\top x + \alpha_{s',l})} \quad (3)
$$

for $l = 1...L_s$ mixture parameters in each class s.

2.1. Log-Linear Discriminative Training

The frame level objective function is

$$
\begin{aligned}
\mathcal{F}^{(frame)}(\Lambda, A) &= -\tau_A ||A||^2 - \tau_\Lambda ||\Lambda||^2 \\
&+ \sum_{r=1}^{R} \sum_{t=1}^{T_r} w_s \log p_{\Lambda,A}(s_t|x_t)
\end{aligned}
\quad (4)
$$

$$p_{\Lambda,A}(s_t|x_t) = \frac{\exp\left(\lambda_{s_t}^\top A x_t + \hat{\alpha}_{s_t}\right)}{\sum_{s'} \exp\left(\lambda_{s'}^\top A x_t + \hat{\alpha}_{s'}\right)} \tag{5}$$

Here the state parameters are $\Lambda_s = \{\lambda_s, \alpha_s\}$ with a transformation matrix A and a fixed alignment s_1^T. τ_A and τ_Λ are regularization parameters. w_s are state weights which could be tuned to give less weight to the accumulations of e.g. noise and silence states. $\hat{\alpha}_s = \alpha_s + \log p(s)$, $p(s)$ is the prior probability of state s and R is the total number of sentences in the training corpus. The state priors are added to α_s for training, and later subtracted at recognition time. The objective function is a frame-level Maximum Mutual Information (MMI), with extra regularization terms. The MMI optimization can also be done at sentence level i.e. the prior probabilities are sentence level language-model probabilities.

2.2. Training of the Feature Transformation Matrix

A dimension reducing transform $A \in \mathbf{R}^{\mathbf{D}' \times \mathbf{D}}$ such that $y = Ax$ can be included into Equation 2

$$\begin{aligned} p_{\Lambda,A}(c|x) &= \frac{\exp(\lambda_c^\top A x + \alpha_c)}{\sum_{c'} \exp(\lambda_{c'}^\top A x + \alpha_{c'})} \\ &= \frac{\exp\left(\sum_{d'd} a_{d'd}(\lambda_{c,d'} x_d) + \alpha_c\right)}{\sum_{c'} \exp\left(\sum_{d'd} a_{d'd}(\lambda_{c',d'} x_d) + \alpha_c\right)} \end{aligned} \tag{6}$$

A is a projective transformation. λ_c and α_c are as in Equation 2. $\lambda_{c,d}$ is the d^{th} scalar component of vector λ_c, x_d is the d $^{\text{th}}$ component of x, and $a_{d'd}$ is the element of matrix A at d'^{th} row and d $^{\text{th}}$ column. The equation is log-linear with respect to either $\lambda_{c,d}$ or $a_{d'd}$, provided the other one is held constant.

2.3. Optimization Procedure

Frame level log-linear optimization can be done using the GIS algorithm [5], but for this case it is found to be slower than the general purpose RPROP algorithm [9]. RPROP is a first order optimization algorithm that takes only the sign of the partial derivatives into account. The weights for parameters are increased if there was no sign change in the partial derivatives in the last iteration, and vice versa. Although the RPROP method is dependent on the initial step size for the partial derivatives, still it performs robustly in case of a poorly chosen initial step size. In all the following experiments in section 4 the RPROP algorithm is used for optimization.

Another issue is the use of Viterbi approximation for the optimization of mixture densities. This means for each $p(x|s)$ using the score of the highest scoring density instead of the sum of all the densities. In practice it was found to be detrimental for the optimization process. When the Viterbi option is enabled, only those feature vectors contribute to the partial derivatives of $\lambda_{s,l}$ which lie closer to it that all other $\lambda_{s,l'}$. Therefore if a particular $\lambda_{s,l}$ strays away from the solution due to a large step size, it will not be brought back towards the solution because there are no feature vectors to contribute towards its partial derivatives. This leads to discontinuities in the partial derivatives. For this reason we calculate the sum of all the densities for the experiments.

3. Second Order Polynomial Features

Let us take the posterior probability in equation 1 as starting point. This is the log-quadratic equation by directly converting a class-specific covariance Gaussian model. By vectorizing the matrices Λ_s and $x^\top x$, we get an equation in log-linear form with respect to $\begin{bmatrix} vec(x^\top x) \\ x \end{bmatrix}$

$$p_\theta(s|x) = \frac{\exp\left(vec^\top(\Lambda_s) \cdot vec(x^\top x) + \lambda_s^\top x + \alpha_s\right)}{\sum_{s'} \exp\left(vec^\top(\Lambda_{s'}) \cdot vec(x^\top x) + \lambda_{s'}^\top x + \alpha_{s'}\right)} \tag{7}$$

here the *vec* operator denotes the vectorization of a matrix, which converts an $m \times n$ matrix into an $mn \times 1$ column vector by stacking the columns of the matrix on top of one another. Since the matrix Λ_s is obtained from the Gaussian covariance as $\Lambda_s = -\frac{1}{2}\Sigma_s^{-1}$, therefore it is symmetric too. Hence to avoid redundancy, we use the half vectorization *vech* operator instead of *vec* in equation 7. The above form of the posterior allows us to implicitly model the class specific covariance model while having the simplicity of optimization and robustness associated with a log-linear model.

The dimension of the input vector $\begin{bmatrix} vech(x^\top x) \\ x \end{bmatrix}$ can be large, therefore a projective transformation matrix A could be used to reduce the number of parameters

$$p_\theta(s|x) = \frac{\exp\left(\hat{\lambda}_s^\top \cdot A \begin{bmatrix} vech(x^\top x) \\ x \end{bmatrix} + \alpha_s\right)}{\sum_{s'} \exp\left(\hat{\lambda}_{s'}^\top \cdot A \begin{bmatrix} vech(x^\top x) \\ x \end{bmatrix} + \alpha_{s'}\right)} \tag{8}$$

where $\hat{\lambda_s}$ are the log-linear parameters to be trained. The resulting form is no longer convex due to the use of a projective transformation. However, in practice the loss in the objective function is not large provided a good initial guess for the parameters $\hat{\lambda_s}$ is given.

4. Experiments and Results

4.1. Speech Corpus and Baseline System

For the performance analysis of second order features, a large vocabulary continuous speech recognition task European Parliament Plenary Sessions (EPPS) is used. It is a part of 2006 TC-STAR ASR evaluation campaign. It is composed of recorded speeches of the European Parliament in British English under clean conditions. The training corpus is 40.8 hours and the evaluation corpus is 3.5 hours. The newer versions of EPPS English corpus contain more than 100 hours of training data. The lexicon size is 54k words.

The acoustic model of the baseline system is cross-word using triphones. A trigram language model is used. The initial features are 16 MFCC features and a voiced feature, and 9 consecutive frames are concatenated together. This vector of size 17× 9 is then projected by a classical LDA matrix to 45 dimensions. The classes are 4501 triphone CART leaves and a pooled covariance is used.

4.2. Experimental Setup

The output vector x of classical LDA matrix is used to create second order features $vec(x \cdot x^\top)$ which is then concatenated to x. This gives a new vector of 1080 dimensions. Another classical LDA is done on this new vector and the dimension is reduced again to 45. A Generative Maximum Likelihood mixture density estimation is used to calculate Gaussian means and a pooled covariance. This gives a starting word error rate of 34.4 %, which is worse than the maximum likelihood error rate of 28 % for first order features. The reason for this is that the classical LDA algorithm does not work so well for this implicit class specific covariance case. The LDA algorithm assumes homoscedasticity. Secondly, the number of parameters for the second LDA transformation matrix is much larger i.e. 1080 × 45, which leads to poorly determined estimates. Therefore we optimize the transformation matrix log-linearly, by a procedure called log-LDA as described in [10].

The single density Gaussian model is converted to log-linear parameters λ_s as in equation 2, and RPROP algorithm is used to optimize the objective function of equation 4. This brings the WER to 26.7 %. This is followed by two alternations of transformation matrix optimization via log-LDA and λ_s log-linear optimization. Surprisingly this causes a significant decrease in the WER and it comes down to 21.3 %. This is 2.2 % absolute better than if the same procedure is repeated with normal first order MFCC features.

4.3. Training of Mixture Densities

The log-linear training is only convex for a single density per state s. For mixture density training this presents challenges as the initial guess in very important and can influence the final objective function and WER. For 2 densities per state, the ML word error for second order features is 31.1 %, and after log-linear density training it drops to 29.1 %. This is even worse than the WER for log-linear training of 1 density per state! (which was 26.7 %). Therefore we need a method to specify a better initial guess to the training of mixture densities, so that the WER is at least as good as the word error rate of a similar but less complex model. To solve this problem we adopt an approach similar to the iterative density splitting algorithm used in a maximum likelihood framework. All the $\lambda_{s,l}$ in state s are duplicated and a small offset is added to both new lambdas to put them away from each other. Since the log-linear model does not have any covariance (it is covariance normalized), therefore the direction of the offset does not matter. Subsequent training of this newly split model causes a decrease in the objective function as the new lambdas discriminatively adapt themselves to the data. In this way, training of the 2 densities per state model using the initial guess from single densities leads to a WER of 18.6 %, which is a 2.7 % absolute improvement after the split.

Table 1: EPPS: WER(%) for first and second order MFCC systems

Mixture Model	**single densities**	**16 dens./state**
1st order ML	28.3	18.6
1st order log-linear	23.5	17.0
2nd order ML	34.4	23.0
2nd order log-linear	21.3	16.5

4.4. Effect of Additional Parameters

If full second order polynomial features without dimension reduction were used, it could be argued that the improvement in the word error rate is coming from the large number of additional parameters. Instead of $(d+1)\times S$ as for the first order MFCC system now we would have $(\frac{d^2+3d}{2}+1)\times S$ parameters $\lambda_{s,d}$ and α_s . Here d is the number of dimensions in λ_s and S is the total number of CART states. However, in our experiments we use a second transformation matrix for dimension reduction, therefore the number of additional parameters is very small. In this case there are $(d+1)\times S$ lambda parameters as previously, plus $\frac{d^2+3d}{2}\times d$ transformation matrix elements. This just means a 23 % increase in the number of parameters for single densities and a mere 1.4 % increase for 16 densities per state. Secondly, this means a negligible overhead at recognition time, as the additional parameters are in the feature extraction phase. During recognition, the feature extraction phase is less computationally intensive as compared to the beam search phase, therefore *moving* parameters from the acoustic model into the feature extraction part will invariably speed up the recognition process.

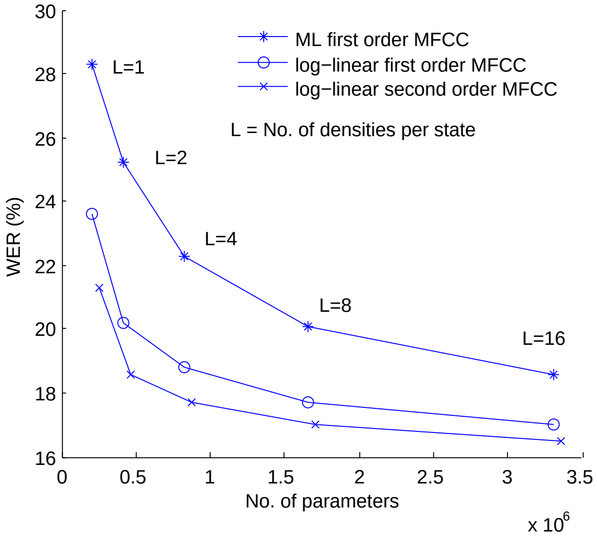

Figure 1: *EPPS: Comparison of discriminatively trained second order and first order MFCC systems*

4.5. Effect of Unconstrained α_s

Another interesting fact about the log-linear mixture models is the unconstrained nature of the constant parameter α_s. It is initialized from the Gaussian model by the following equality

$$\alpha_s = -\frac{1}{2}\mu_s^\top \Sigma_s^{-1}\mu_s - \frac{1}{2}\log\det(2\pi\Sigma_s) + \log p(s) \qquad (9)$$

this equality implies a dependence of α_s on the λ_s parameters. However in case of log-linear training, α_s are optimized as parameters too. This means that they could deviate away from the equality if this means an increase in the value of the objective function. Therefore although theoretically the Gaussian and log-linear posterior models are equivalent, they may not be equal at the recognition time because for recognition the priors are calculated from the language model and not from state priors. To test the effect of this extra degree of freedom, the discriminatively trained log-linear models were converted again to Gaussian models and recognition was done using these models. Experiments show that for single densities, this conversion had a significant effect and the newly converted Gaussian models

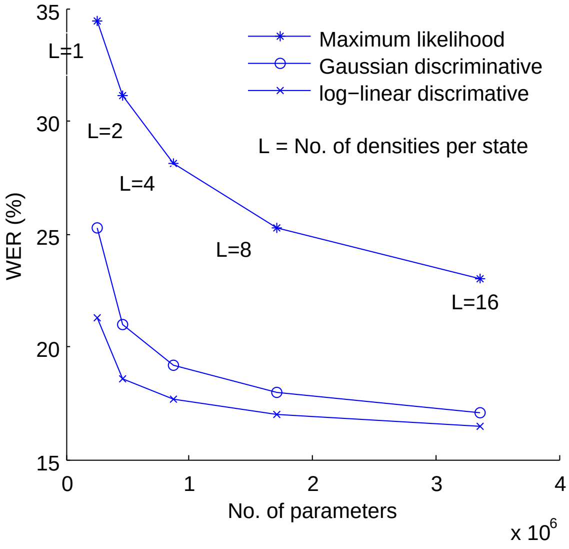

Figure 2: *EPPS with Second order features: Comparison of discriminatively trained Log-linear model versus the Gaussian mixture model obtained from that*

performed about 4 % worse than the corresponding log-linear models. This is because for a small number of densities per state, a free α_s can significantly alter the decision boundaries. On the other hand, for 256 densities per state acoustic model, there was no WER difference between the log-linear model and the converted Gaussian model.

5. Conclusion

In this paper we have presented results for a speech recognition system which uses second order polynomial features with a subsequent dimension reduction transform. This gives us the advantage of second order features while the number of additional parameters is small. We present results for an MFCC system that has been trained discriminatively by log-linear frame level MMI optimization. The mixture density splitting is also done during the process and the acoustic model parameters before each split are used for initial guess of the new parameters. The dimension reduction transformation applied to the second order features has also been trained log-linearly. This setup has been experimentally found to achieve better WER as compared to the baseline MFCC system. One reason for this improvement is the implicit modeling of class specific covariance while undergoing only a slight increase in the number of parameters. Secondly, it has been found that the unconstrained optimization of the priors plays a role in the log-linear optimization; it is one of the reasons why log-linear model parameters can achieve better WER than the corresponding Gaussian mixture model. The envisioned future work in this direction is integrating speaker specific feature and model transformation techniques such as MLLR and CMLLR into such a framework. Secondly, better methods for initializing a mixture density system are needed, since the initial guess is very important in that case. Higher order polynomial features also need to be investigated i.e. making the feature extraction system more complex in order to make the acoustic model simpler; thereby creating more robust models. This can also speed up the recognition process as the feature extraction is generally computationally less expensive than the Viterbi search phase.

6. Acknowledgments

This work was partly realized as part of the Quaero Programme, funded by OSEO, French State agency for innovation, and also partly based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001-06-C-0023. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the DARPA.

7. References

[1] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, pp. 121–167, 1998.

[2] S. Wiesler, M. Nußbaum, G. Heigold, R. Schlüter, and H. Ney, "Investigations on features for log-linear acoustic models in continuous speech recognition," in *IEEE Automatic Speech Recognition and Understanding Workshop*, Merano, Italy, Dec. 2009.

[3] M. K. Omar and M. Hasegawa-Johnson, "Maximum conditional mutual information projection for speech recognition," in *Proc. INTERSPEECH'03*, Geneva, Switzerland, Sep. 2003.

[4] G. Heigold, P. Lehnen, R. Schlüter, and H. Ney, "On the equivalence of Gaussian and log-linear HMMs," in *Proc. INTERSPEECH'08*, Brisbane, Australia, Sep. 2008.

[5] J. Darroch and D. Ratcliff, "Generalized iterative scaling for log-linear models," *Annals of Mathematical Statistics*, vol. 43, pp. 1470–1480, 1972.

[6] G. Heigold, D. Rybach, R. Schlüter, and H. Ney, "Investigations on convex optimization using log-linear HMMs for digit string recognition," in *Proc. INTERSPEECH'09*, Brighton, U.K., Sep. 2009.

[7] H.-K. J. Kuo and Y. Gao, "Maximum entropy direct models for speech recognition," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 14, no. 3, pp. 873 – 881, May 2006.

[8] J. Fayolle, F. Moreau, C. Raymond, G. Gravier, and P. Gros, "Crf-based combination of contextual features to improve a posteriori word-level confidence measures," in *Proc. INTERSPEECH'10*, Makuhari, Japan, September 2010, pp. 1942–1945.

[9] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," in *Proc. ICNN'93*, San Francisco, USA, 1993, pp. 586–591.

[10] M. A. Tahir, G. Heigold, C. Plahl, R. Schlüter, and H. Ney, "Log-linear framework for linear feature transformations in speech recognition," in *IEEE Automatic Speech Recognition and Understanding Workshop*, Merano, Italy, Dec. 2009.