Accelerated Batch Learning of Convex Log-linear Models for LVCSR

Simon Wiesler, Ralf Schlüter, Hermann Ney

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

Abstract

This paper describes a log-linear modeling framework suitable for large-scale speech recognition tasks. We introduce modifications to our training procedure that are required for extending our previous work on log-linear models to larger tasks. We give a detailed description of the training procedure with a focus on aspects that impact computational efficiency. The performance of our approach is evaluated on the English Quaero corpus, a challenging broadcast conversations task. The log-linear model consistently outperforms the maximum likelihood baseline system. Comparable performance to a system with minimum-phone-error training is achieved.

Index Terms: acoustic modeling, discriminative models

1. Introduction

Conventional speech recognition systems rely on Gaussian mixture HMMs (GHMMs). The training of the acoustic model begins with a maximum likelihood (ML) training with the expectation maximization (EM) algorithm. The performance of the generative GHMM can further be improved by subsequently optimizing the parameters according to a discriminative criterion, *e.g.* the minimum phone error (MPE) criterion [1] .

Recently, the interest in discriminative models has greatly increased. They are conceptually advantageous because they do not depend on a maximum likelihood initialization. In particular, log-linear models are promising because they fit into the probabilistic framework of hidden Markov models (HMMs). Another important property of log-linear models is that their training according to the conditional maximum likelihood (CML) criterion is a convex optimization problem. This property guarantees that the optimization can not get stuck in local optima. Algorithms with guaranteed convergence, e.g. steepest descent and other more sophisticated optimization algorithms, converge to the global optimum from any initialization. This is in contrast to discriminative training of GHMMs, where the performance depends on the quality of the initialization, the choice of the optimization algorithm, and internal parameters of the optimization algorithm. Therefore, log-linear models require less engineering work than conventional discriminative training.

Convex log-linear models have been successfully applied to phoneme recognition [2, 3], promising results have also been obtained on continuous speech recognition tasks, *e.g.* in [4, 5].

Since the training time for discriminative models is very high, the efficient training of log-linear models is of great interest. In this paper, we extend our previous work on log-linear models [4, 6] from the small-sized Wall Street Journal task to a challenging large-vocabulary broadcast conversation (BC) task. We provide a detailed description of our modified training procedure and point out critical issues for computational efficiency. An important new aspect in the training procedure is the utilization of linearly transformed features without explicitly applying the transformation.

The paper is organized as follows. Section 2 describes the form of the log-linear model and the training criterion. A detailed description of the training procedure is given in Section 3. Experimental results are given in Section 4. Finally, Section 5 presents the conclusion.

2. Model Definition and Training Criterion

Let $X \subset \mathbb{R}^D$ denote the observation space and $\mathcal{S} = \{1, \ldots, S\}$ a finite set of classes. A *log-linear model* with parameters $\Lambda = (\lambda_1; \ldots; \lambda_S) \in \mathbb{R}^{N \times S}$ and $\alpha = (\alpha_1; \ldots; \alpha_s)^T \in \mathbb{R}^S$ is a model for class posterior probabilities of the form

$$p_{(\Lambda,\alpha)}(s|x) = \frac{\exp\left(\lambda_s^T f(x) + \alpha_s\right)}{\sum_{\bar{s}=1}^{S} \exp\left(\lambda_{\bar{s}}^T f(x) + \alpha_{\bar{s}}\right)} , \qquad (1)$$

where the components of $f : X \to \mathbb{R}^N$ are called *feature functions*. In this work, we consider log-linear models that are defined on frame-level, i.e. $s \in \mathcal{S}$ denotes an HMM state and $x \in X$ an acoustic observation. The posterior probabilities can be used in HMM speech recognizers via the hybrid approach [7].

The regularized CML criterion is often regarded as the natural training criterion for log-linear models. With the

above notation, it minimizes the objective function

$$\mathcal{F} : \mathbb{R}^{(N+1)\times S} \to \mathbb{R}$$
$$(\Lambda, \alpha) \mapsto -\frac{1}{T}\sum_{t=1}^{T} \ln p_{(\Lambda,\alpha)}(s_t|x_t) + C\|(\Lambda,\alpha)\|_2^2 \ . \quad (2)$$

Here, $(x_t, s_t)_{t=1,\ldots,T}$ is the training sample. The objective function is convex for $C \geq 0$ and strictly convex for $C > 0$. The gradient of the unregularized objective function is

$$\nabla_{\lambda_s}\mathcal{F}(\Lambda,\alpha) = \frac{1}{T}\sum_{t=1}^{T}\left(p_{(\Lambda,\alpha)}(s|x_t) - \delta_{s,s_t}\right) f(x_t), \quad (3)$$

where $1 \leq s \leq S$, and δ denotes the Kronecker delta. The gradient with respect to α_s is analogous with f replaced by the constant one.

3. A Training Recipe

In this section, we give a detailed description of our training procedure for log-linear models. Our goal is a system that is independent from a generative system, in particular it should not require an initialization from a generative system. The training should use only standard routines, require few tuning parameters, and give at least as good results as a discriminatively trained generative system. Furthermore, the training needs to be scalable to large datasets.

3.1. Choice of Model

Conventional discriminative training of generative models is typically performed on sequence-level. In contrast, we use a log-linear model defined on frame-level in this work. Conceptually, the approach of working on sequence level is preferable because it directly corresponds to the application of the model in recognition. However, in practice, the competing hypotheses are approximated by word lattices, which are generated with an ML model [8]. In addition, only the Viterbi alignment is considered within the word boundaries.

For training discriminative models *from scratch*, the lattice approach has several drawbacks. First, the lattices have to be generated with an existing generative model, which is in contrast to our goal of designing a system independent from the generative system. Second, generating useful lattices for discriminative training requires a lot of engineering work which we aim to avoid. In particular for spontaneous speech, care has to be taken that word-fragments and non-speech events as coughing, laughter, and noises do not dominate the word lattice.

For frame-wise models, only an HMM-state alignment is required. In principle, the alignment can be derived from scratch, starting from a linear alignment [4]. In our experiments, we observed only a weak dependence on the quality of the alignment. Therefore it is more convenient to work with the ML alignment on large datasets.

3.2. Choice of Features

A crucial aspect in the application of log-linear models is the definition of appropriate feature functions. In our previous work [4], we used a combination of two types of feature functions. First, we used second-order polynomial feature functions, *i.e.,*

$$f : X \to \mathbb{R}^N, \ x \mapsto \begin{pmatrix} x \\ xx^T \end{pmatrix} \quad (4)$$

with $N = D + D(D+1)/2$. Polynomial features are global in the sense that their support is the whole observation space. Therefore, they generalize well to unseen data. However, training models with these high-dimensional dense features is computationally demanding. Furthermore, by increasing the polynomial degree, the number of features increases exponentially.

In [4], we investigated sparse features, similar to the features employed in fMPE [9]. These features mimic the locality of Gaussian mixture models (GMMs). They are derived from a GMM for the marginal probability $p(x)$, which can be trained without an existing GHMM system. Given the probabilities $(p(x|l))_{1\leq l\leq L}$ from a GMM, the features are defined as

$$f_l : X \to \mathbb{R}, \ x \mapsto \frac{p(x|l)}{\sum_{l'} p(x|l')} \quad l = 1, \ldots, L \ . \quad (5)$$

By setting a small threshold, sparse features are obtained, which strongly reduces the computational demands of the gradient accumulation. The number of sparse features can further be increased by including features of neighboring time frames. Using such local features, log-linear models can approximate the training data much better. However, models with sparse features overfit more quickly. In previous experiments on smaller tasks, we found that using a combination of both polynomial features and sparse features gives a good compromise.

3.3. Regularization *vs.* early stopping

In order to control model complexity, usually multiple trainings with different regularization constants are performed. Each training is run until convergence to the global optimum and the best model is chosen on the development set. Since this approach is very time consuming, we use *early stopping* in this work, *i.e.,* the optimization is stopped as soon as the error rate on the development set increases. As regularization, early stopping bounds the norm of the parameters and therefore has a similar effect, see *e.g.* [10].

3.4. Optimization

In literature, the Quasi-Newton algorithm L-BFGS is commonly regarded as the best optimization algorithm for log-linear models [11]. In [6], we found that an improved version of Rprop [12] converges faster than L-BFGS when using very high-dimensional features. An

advantage of Rprop is its insensitivity to internal tuning parameters. The only critical parameter is the initial step size. In our experiments, we used the largest step size that leads to a reduction of the objective function in the first step. We used a backtracking procedure for finding the best initial step size, starting with a value of 0.01.

One possibility for accelerating log-linear training might be the choice of a good initialization. In initial experiments, we compared zero initialization, random initialization, and initialization with a generative model. It turned out that starting from zero led to fastest convergence. In addition, the first iteration can be accelerated strongly because in this case the posterior probabilities are all uniform and do not have to be computed.

3.5. Gradient Accumulation

Most of the computation time for training log-linear models is spent on the accumulation of the gradient, which requires the calculation of the posterior probabilities and updating the gradient for each observation.

The posterior probabilities require the calculation of high-dimensional inner products (*scores*) of the class-specific parameter vector and the feature vector. Calculation of the scores can be performed as a matrix-vector product of the parameter matrix and the feature vector. If multiple features are processed at once, a matrix-matrix product is required. The gradient update is a weighted vector sum. For these computations, it is important to use optimized libraries. In our implementation, we observed a speedup of the whole training by a factor of two, simply by using the optimized linear algebra routines in the AMD Math Core Library[1] instead of a naive implementation.

A simple way for accelerating the gradient update is to prune the posterior probabilities. Then, the parameters of the classes with zero probability do not need to be updated, *cf.* Equation (3).

3.6. Implicit Feature Transformation

In our previous work [6], we found that the optimization problem (2) can be poorly conditioned, in particular when using polynomial features. It turned out to be crucial to improve the conditioning of the optimization problem by normalizing means and variances of the features and decorrelating them.

In principle, the features can be preprocessed once before training and then stored to disk. However, for log-linear models, typically very high-dimensional features are derived from low-dimensional observations. For large datasets, these high-dimensional features can not be stored and have to be computed on-the-fly in every iteration. The decorrelation requires an expensive high-dimensional matrix-vector product. Here, we show that these computations can be avoided by performing the transformation on model side.

In order to simplify notation, we do not distinguish between the observations and their high-dimensional representations, *i.e.,* $f(x) = x$. Let $W \in \mathbb{R}^{N \times \bar{N}}$ and $b \in \mathbb{R}^{\bar{N}}$ denote the parameters of an affine feature transformation. Then the scores are

$$\lambda_s^T (Wx + b) + \alpha_s = (W^T \lambda_s)^T x + \lambda_s^T b + \alpha_s \ . \quad (6)$$

Thus, the posterior probabilities of transformed features can be calculated by transforming the model instead of the features. Let

$$(\bar{\Lambda}, \bar{\alpha}) = (W^T \Lambda, \Lambda^T b + \alpha) \quad (7)$$

denote the transformed model parameters. Then

$$\begin{aligned} \nabla_{\lambda_s} \mathcal{F} &= \frac{1}{T} \sum_{t=1}^{T} \left(p_{(\Lambda,\alpha)}(s|Wx_t + b) - \delta_{s,s_t} \right) (Wx_t + b) \\ &= W \ \frac{1}{T} \sum_{t=1}^{T} \left(p_{(\bar{\Lambda},\bar{\alpha})}(s|x_t) - \delta_{s,s_t} \right) x_t \\ &+ \frac{1}{T} \sum_{t=1}^{T} \left(p_{(\bar{\Lambda},\bar{\alpha})}(s|x_t) - \delta_{s,s_t} \right) b. \end{aligned} \quad (8)$$

An analogous identity holds for the gradient with respect to α_s. Thus, accumulation of the gradient with transformed features can be performed by accumulation with the original features but with the transformed model (7) and transforming the gradient via (8). This procedure reduces the complexity from $\mathcal{O}(TN\bar{N})$ to $\mathcal{O}(SN\bar{N})$ and is therefore beneficial when the number of data points is larger than the number of classes.

4. Experimental Results

In this section, we describe experiments with log-linear models on the English Quaero corpus, a challenging broadcast conversations task, see Table 1 for corpus statistics. Our GHMM baseline system is a simplified version of our evaluation system, which performed best in the Quaero evaluations in 2010 and 2011 [13]. In contrast to our evaluation system, the baseline system is only a single one-pass system without multi-layer perceptron (MLP) features.

The baseline system uses MFCC features with vocal tract length normalization and a voicedness feature. Context is incorporated by concatenating features from a window of nine frames. The dimension of the resulting feature vector is reduced to 45 by means of an LDA. The GMM has a pooled, diagonal covariance matrix and models 4500 generalized triphones. An ML model is trained with the EM algorithm with a splitting procedure. The ML model is used as initialization for an MPE training.

[1] http://developer.amd.com/libraries/acml

	train	dev10	eval10	eval11
Amount of data	103h	3.3h	3.7h	3.3h
WER ML	-	25.5	25.1	32.2
WER MPE	-	24.0	24.0	30.6
WER log-lin	-	24.2	24.0	30.8
WER syscomb		22.2	22.3	28.9

Table 1: Results and corpus statistics for the 2010 English Quaero corpus. Word error rates (WER) of the ML and MPE baseline systems, the log-linear system, and system combination are given.

The number of splits of the GMM is tuned with respect to performance of the MPE model, leading to a model with roughly one million densities. The best MPE model is obtained after 21 Rprop iterations. The recognition lexicon consists of 150k words. The language model is a smoothed 4-gram, trained on roughly three billion words.

The log-linear system uses the same baseline features and the same state tying as the GHMM system. The training follows the description in Section 3. Second-order polynomial features and sparse features with an acoustic context of nine frames are computed from the baseline features. The best log-linear model is obtained with sparse features derived from a GMM with 12 splits. The resulting feature vector has 36864 sparse and 1080 dense components. The best word error rate is achieved after 17 Rprop iterations.

The training time per iteration for the log-linear model is still larger than the training time for MPE by a factor of around five. On the other hand, the frame-wise log-linear approach does not require the creation of lattices, and requires less tuning. Since our training scheme can easily be parallelized, the log-linear approach is feasible for large datasets.

The experimental results are summarized in Table 1. The log-linear system consistently outperforms the ML model. The MPE model and the log-linear model perform equally well. In addition, we performed a confusion network system combination (as in [13]) of the MPE and the log-linear system. Strong improvements are obtained although both systems use the same baseline features and the same state tying.

5. Conclusions

In this paper, we extended our previous work on log-linear models from a small database for read speech (Wall Street Journal) to a challenging broadcast conversation task. The increased amount of training data required improvements to our implementation and several modifications to our training setup.

It is interesting to note that one new aspect in our modified training scheme, the feature transformation on model side, can be applied to other models as well, in particular to neural networks. However, this trick only reduces training time, when the number of accumulated samples is larger than the number of classes. This is typically the case for batch training or mini-batch training with large batch sizes.

Our proposed training scheme for log-linear acoustic models is scalable to large tasks. It requires only standard routines and is therefore implemented relatively easily. It depends on only few tuning parameters, and therefore avoids much engineering work commonly related to discriminative training. Still, we observed consistent improvements over an ML trained generative model and comparable performance to a discriminatively trained generative model. Furthermore, the log-linear system provides valuable complementary information for system combination. It remains to verify that these findings hold for systems with MLP features and in combination with speaker combination.

Acknowledgment– This work was partly realized as part of the Quaero Programme, funded by OSEO, French State agency for innovation.

6. References

[1] D. Povey and P. C. Woodland, "Minimum phone error and I-smoothing for improved discriminative training," in *Proc. ICASSP*, 2002, pp. 105 – 108.

[2] Y. Hifny and S. Renals, "Speech recognition using augmented conditional random fields," *IEEE Trans. Audio Speech Lang. Process.*, vol. 17, no. 2, pp. 354–365, 2009.

[3] E. Fosler Lusier and J. Morris, "Crandem systems: Conditional random field acoustic models for hidden Markov models," in *Proc. ICASSP*, 2008, pp. 4049–4052.

[4] S. Wiesler, M. Nußbaum, G. Heigold, R. Schlüter, and H. Ney, "Investigations on features for log-linear acoustic models in continuous speech recognition," in *Proc. ASRU*, 2009, pp. 52–57.

[5] A. Ragni and M. Gales, "Structured discriminative models for noise robust continuous speech recognition," in *Proc. ICASSP*, 2011, pp. 4788–4791.

[6] S. Wiesler, R. Schlüter, and H. Ney, "A convergence analysis of log-linear training and its application to speech recognition," in *Proc. ASRU*, 2011, pp. 1–6.

[7] H. Bourlard and N. Morgan, *Connectionist speech recognition*. Kluwer Academic Publishers, 1994.

[8] V. Valtchev, J. Odell, P. Woodland, and S. Young, "MMIE training of large vocabulary recognition systems," *Speech Commun.*, vol. 22, no. 4, pp. 303–314, 1997.

[9] D. Povey, B. Kingsbury, L. Mangu, G. Saon, H. Soltau, and G. Zweig, "fMPE: Discriminatively trained features for speech recognition," in *Proc. ICASSP*, 2005.

[10] C. Bishop, *Neural networks for pattern recognition*. Oxford University Press, 1995.

[11] R. Malouf, "A comparison of algorithms for maximum entropy parameter estimation," in *Proceedings of the Sixth Conference on Natural Language Learning*, 2002, pp. 49–55.

[12] C. Igel and M. Hüsken, "Empirical evaluation of the improved rprop learning algorithms," *Neurocomputing*, vol. 50, pp. 105–123, 2003.

[13] M. Sundermeyer, M. Nußbaum-Thom, S. Wiesler, C. Plahl, A. El-Desoky Mousa, S. Hahn, D. Nolden, R. Schlüter, and H. Ney, "The RWTH 2010 Quaero ASR evaluation system for English, French, and German," in *Proc. ICASSP*, 2011, pp. 2212–2215.