JOINT DISCRIMINATIVE FRONT END AND BACK END TRAINING FOR IMPROVED SPEECH RECOGNITION ACCURACY

Jasha Droppo and Alex Acero

Speech Technology Group
Microsoft Research, Redmond, WA, USA
{jdroppo,alexac}@microsoft.com

ABSTRACT

This paper presents a general discriminative training method for both the front end feature extractor and back end acoustic model of an automatic speech recognition system. The front end and back end parameters are jointly trained using the Rprop algorithm against a maximum mutual information (MMI) objective function. Results are presented on the Aurora 2 noisy English digit recognition task. It is shown that discriminative training of the front end or back end alone can improve accuracy, but joint training is considerably better.

1. INTRODUCTION

The acoustic processing of standard automatic speech recognition systems can be roughly divided into two parts: a front end which extracts the acoustic features, and a back end acoustic model which scores transcription hypotheses for sequences of those acoustic features.

This division of the acoustic processing into front end and back end can lead to a suboptimal design. If useful information is lost in the front end, it can not be recovered in the back end. The acoustic model is constrained to work with whatever fixed feature set is chosen.

In theory, a system that could jointly optimize the front end and back end acoustic model parameters would overcome this problem. The feature representation would adapt to provide more useful information to the back end, which would in turn learn to leverage the new information.

One problem with traditional front end design is that the parameters are manually chosen based on a combination of trial and error, and reliance on historical values. Even though much effort has been applied to training the parameters of the back end to improve recognition accuracy, the front end contains fixed filterbanks, cosine transforms, cepstral lifters, and delta/acceleration regression matrices.

Many existing front end designs are also uniform, in the sense that they extract the same features regardless of absolute location in the acoustic space. This is not a desirable quality, as the information needed to discriminate `/f/` from `/s/` is quite different from that needed to tell the difference between `/aa/` and `/ae/`. A front end with uniform feature extraction must make compromises to get good coverage over the entire range of speech sounds.

This paper demonstrates how Rprop[6], a general learning algorithm developed for neural networks, can be used to discriminatively train both the front end and the back end parameters, and begin to overcome all of these outstanding problems.

The general idea of jointly training the feature extractor and classifier is not entirely new. It has, for instance, been shown to improve character recognition[1]. In [2], the idea was applied to hybrid ANN/HMM speech recognition systems, where the neural network was initialized with articulatory features. In [3], minimum classification error (MCE) training used to update the parameters of a hybrid SVM/HMM system.

In this paper, the front-end parameters pass through a SPLICE (stereo piecewise linear compensation for environment) [4] transform. Given enough parameters, SPLICE can approximate any feature transformation to an arbitrary precision. As a result, the system presented here is a generalization of any discriminative feature space transformation.

This paper is organized as follows. Section 2 demonstrates how the Rprop algorithm can be used to maximize a discriminative objective function. Sections 3 and 4 derive the specific update equations for training the back end and front end, respectively. In Section 5, experimental results are presented that show the benefits of joint training.

2. MMI TRAINING WITH RPROP

This section presents a general framework for using Rprop to train the system parameters with respect to a maximum mutual information (MMI) objective function.

2.1. The MMI Objective Function

The MMI objective function is the sum of the log conditional probabilities for all correct transcriptions w_r of utterance numver r, given their corresponding acoustics $\mathcal{Y}_r$.

$$\mathcal{F} = \sum_r \mathcal{F}_r = \sum_r \ln p(w_r|\mathcal{Y}_r) \tag{1}$$

To derive $p(w_r|\mathcal{Y}_r)$, both halves of the acoustic processing need to be considered: the front end transformation, and the back end acoustic score. The front end feature transformation $\mathcal{X}_r = f(\mathcal{Y}_r; \lambda)$ is parameterized by λ and converts the rth input sequence $\mathcal{Y}_r$ into the feature vector sequence $\mathcal{X}_r$. The back end acoustic score $p(\mathcal{X}_r, w; \theta)$ defines a joint probability distribution over feature sequences $\mathcal{X}_r$ and transcriptions w using the parameters θ.

Using these definitions, Bayes' rule, and a change of variables, it is simple to show that the global objective function is equal to

$$\mathcal{F} = \sum_r \ln \frac{p(f(\mathcal{Y}_r;\lambda), w_r; \theta) J_f(\mathcal{Y}_r)}{\sum_w p(f(\mathcal{Y}_r;\lambda), w; \theta) J_f(\mathcal{Y}_r)}. \tag{2}$$

Here, $J_f(\mathcal{Y}_r)$ is the Jacobian of the transformation $f(\mathcal{Y}_r; \lambda)$, evaluated at $\mathcal{Y}_r$. When this Jacobian is nonzero, it disappears entirely from Eq. 2.

Since exact optimization of Eq. 2 can be resource intensive, the probabilities $p(\mathcal{X}_r, w; \theta)$ are approximated on word lattices generated by the baseline maximum likelihood acoustic model. The numerator is calculated over the best path that corresponds with the correct transcription, and the denominator is calculated over all paths in the lattice.

As is commonly done in lattice-based MMI estimation[5], the objective function was also modified to include posterior flattening, the time marks in the lattices were held fixed, and forward-backward was used within each arc to determine arc conditional posterior probabilities.

2.2. The Rprop Algorithm

Rprop[6] is a well known algorithm that was originally developed to train neural networks. For this paper, Rprop is employed to find parameter values that increase the MMI objective function.

By design, Rprop only needs the sign of the gradient of the objective function with respect to each parameter. It divorces the scale of the step size Δ_i from the magnitude of the current gradient $\frac{\partial \mathcal{F}}{\partial \lambda_i}$. As a result, Rprop is quite easy to implement and is robust to non-uniform scaling of the features.

```
For each parameter λ_i {
   d ← ∂F/∂λ_i (t−1) · ∂F/∂λ_i (t)
   if ( d >= 0 ) then {
      if ( d > 0 ) then Δ_i ← min(1.2 Δ_i, Δ_max)
      λ_i(t+1) ← λ_i(t) + sign(∂F/∂λ_i (t)) · Δ_i
   } else if ( d < 0 ) then {
      Δ_i ← max(0.5 Δ_i, Δ_min)
      λ_i(t+1) ← λ_i(t−1)
      ∂F/∂λ_i (t) ← 0
   }
}
```

There are only a handful of parameters to set in the Rprop algorithm. For this paper, we chose $\Delta_{\min} = 10^{-5}$ and $\Delta_{\max} = 0.1$ to bound the step size within a reasonable range. The initial value for the step size was $\Delta_0 = 0.01$ and for the first iteration, the previous gradient was assumed to be zero.

Analysis of the Rprop algorithm is simple. At each iteration, for every parameter, Rprop does one of three things.

If the current and previous gradient are in the same direction ($d > 0$), the step size Δ_i is increased and applied in the same direction as the current gradient.

If the current and previous gradient are in opposite directions ($d < 0$), it means that a local maximum has been overshot. In this case, the step size is reduced and the parameter is reset to its value before the last update. Also, the memory of the current gradient is set to zero. This serves as a flag for the next iteration of the algorithm.

If either the current or previous gradient are zero, then $d = 0$ and the current step size is applied in the direction of the current gradient. This is appropriate whether the current gradient is zero, and Rprop has found a local maximum, or the previous gradient is zero, indicating that the algorithm had overshot and backtracked during the previous iteration.

3. TRAINING THE BACK END ACOUSTIC MODEL PARAMETERS

This section derives the equations necessary to train the back end acoustic model parameters with Rprop. Gradient-based MMI training of HMM parameters was used in [7] and [8], but due to computational difficulties, the method has not been widely adopted. Although it has been shown that efficient gradient based MCE training is feasible [9], this paper demonstrates that gradient based MMI training is also practical.

3.1. The Back End Gradient

To train the back end acoustic model parameters, it is necessary to compute the partial derivative of the objective function $\mathcal{F}_r$ with respect to these parameters.

Every $\mathcal{F}_r$ is a function of many acoustic model state conditional probabilities $p(x_t^r|s_t^r)$, which are in turn, functions of the back-end acoustic model parameters.[1] This structure allows a simple application of the chain rule.

$$\frac{\partial \mathcal{F}_r}{\partial \theta} = \sum_{t,s,i} \frac{\partial \mathcal{F}_r}{\partial \ln p(x_t^r|s_t^r = s)} \frac{\partial \ln p(x_t^r|s_t^r = s)}{\partial \theta} \tag{3}$$

Here, r is an index into the training data. The tth observation vector in utterance r is identified by x_t^r. The back end acoustic model state at time t in utterance r is s_t^r.

The first term in Eq. 3 captures the sensitivity of the objective function to individual acoustic likelihoods in the model. It can be shown to be equal to the difference of the conditional and unconditional posterior, with respect to the correct transcription. These are simply the flattened numerator and denominator terms that occur in standard lattice-based MMI estimation[5].

$$\begin{aligned} \frac{\partial \mathcal{F}_r}{\partial \ln p(x_t^r|s_t^r = s)} &= p(s_t^r = s|\mathcal{X}_r, w_r) - p(s_t^r = s|\mathcal{X}_r) \\ &= \gamma_{rts}^{\text{num}} - \gamma_{rts}^{\text{den}} \end{aligned} \tag{4}$$

The second term in Eq. 3 captures the sensitivity of individual likelihoods in the acoustic model with respect to the back end model parameters.

In this paper, we only consider updating the mean parameters μ_s. For this case, the second term is equal to the following, where the function $1(z)$ equals one, if and only if z is true.

$$\frac{\partial \ln p(x_t^r|s_t^r = s')}{\partial \mu_s} = \Sigma_s^{-1}(x_t^r - \mu_s)1(s = s') \tag{5}$$

The gradient of the objective function $\mathcal{F}$ with respect to the mean parameter μ_s is therefore:

$$\frac{\partial \mathcal{F}}{\partial \mu_s} = \sum_{r,t} \left(\gamma_{rts}^{\text{num}} - \gamma_{rts}^{\text{den}}\right) \Sigma_s^{-1}(x_t^r - \mu_s)$$

3.2. Initial Values for the Back End Parameters

The back-end acoustic model parameters are first trained with a standard maximum likelihood training regime. After convergence, the discriminative training is applied. At that point, only the mean parameters are changed. The variance, mixture weights, and transition probabilities are held constant. This isn't strictly necessary, and the authors suspect further gains are available by properly training these parameters.

[1] This derivation assumes one Gaussian mixture component per state of the acoustic model. For the multiple mixture component case, the variable s indexes not state, but the individual mixture components. Nothing else needs to be changed.

4. TRAINING THE FRONT-END PARAMETERS

This section derives the equations necessary to train the front end transformation parameters with Rprop.

4.1. The SPLICE Transform

The SPLICE transform was first introduced as a method for overcoming noisy speech [4]. It models the relationship between feature vectors y and x as a constrained Gaussian mixture model (GMM), and then uses this relationship to construct estimates of x given observations of y.

In this paper, y is a traditional feature vector based on static cepstra and its derivatives. But, it should be possible to expand y to include more context information, finer frequency detail, or other non-traditional features.

No explicit constraints are placed on x, other than it represents a feature space that improves our objective function. This gives the system more freedom than existing methods that define x as clean speech[4] or phone posteriors[10].

One way of parameterizing the joint GMM on x and y is as a GMM on y, and a conditional expectation of x given y and the model state m.

$$\begin{aligned} p(y,m) &= N(y;\mu_m,\sigma_m)\pi_m \\ E[x|y,m] &= A_m y + b_m \end{aligned}$$

The parameters λ of this transformation are a combination of the means μ_m, variances σ_m, and state priors π_m of the GMM $p(y,m)$, as well as the rotation A_m and offset b_m of the affine transformation.

The SPLICE transform $f(y;\lambda)$ is defined as the minimum mean squared estimate of x, given y and the model parameters λ. In effect, the GMM induces a piecewise linear mapping from y to x.

$$\hat{x} = f(y;\lambda) = E[x|y] = \sum_m (A_m y + b_m) p(m|y) \tag{6}$$

For this paper, a simplified form of the SPLICE transformation is used. All of the rotations A_m are replaced with the identity matrix, and Eq. 6 reduces to

$$f(y;\lambda) = y + \sum_m b_m p(m|y). \tag{7}$$

4.2. The Front End Gradient

Computing the gradient of Eq. 2 with respect to the front-end parameters is also a simple application of the chain rule.

As seen previously, every $\mathcal{F}_r$ is a function of many acoustic model state conditional probabilities $p(x_t^r|s_t^r)$. These are, in turn, functions of the front end transformed features x_{it}^r. And, each transformed feature is a function of the front end parameters λ.

$$\frac{\partial \mathcal{F}_r}{\partial \lambda} = \sum_{t,s,i} \frac{\partial \mathcal{F}_r}{\partial \ln p(x_t^r|s_t^r=s)} \frac{\partial \ln p(x_t^r|s_t^r=s)}{\partial x_{it}^r} \frac{\partial x_{it}^r}{\partial \lambda} \tag{8}$$

Here, r is an index into the training data. The tth observation vector in utterance r is identified by x_t^r. The scalar x_{it}^r is the ith dimension of that vector. The back end acoustic model state at time t in utterance r is s_t^r.

The first term in Eq. 8 is identical to its counterpart in the previous section. The second term in Eq. 8 captures the sensitivity of individual likelihoods in the acoustic model with respect to the front end transformed features. Computing this differential is a simple matter.

$$\frac{\partial \ln p(x_t^r|s_t^r=s)}{\partial x_t^r} = -\Sigma_s^{-1}(x_t^r - \mu_s) \tag{9}$$

Here, μ_s and Σ_s are mean and variance parameters from the Gaussian component associated with state s in the back end acoustic model.

The final term in Eq. 8 captures the relationship between the transformed features and the parameters of the front end. For the simplified SPLICE transform in this paper, only the offset parameters b_m are trained.[2] For the uth element of the vector b_m,

$$\begin{aligned} \frac{\partial x_{it}^r}{\partial b_{um}} &= \frac{\partial}{\partial b_{um}} \left(y_{ut}^r + \sum_{m'} b_{im'} p(m'|y_t^r) \right) \\ &= 1(i=u) p(m|y_t^r) \end{aligned} \tag{10}$$

Combining Eqs. 1, 8, 4, 9 and 10, the complete gradient with respect to the vector b_m is

$$\frac{\partial \mathcal{F}}{\partial b_m} = -\sum_{r,t,s} p(m|y_t^r) \left(\gamma_{rts}^{\text{num}} - \gamma_{rts}^{\text{den}} \right) \Sigma_s^{-1}(x_t^r - \mu_s) \tag{11}$$

4.3. Initial Values for SPLICE Parameters

The SPLICE GMM was trained from scratch using maximum likelihood (ML) re-estimation. To initialize the means, M vectors were uniformly chosen from the training data. The variance structure was diagonal, initialized to unit covariance, and tied across all mixture components. Ten iterations of ML training were then performed to refine the model parameters.

Initial values for the offset parameters of the SPLICE transform are chosen to correspond to an identity transform of the input data ($b_m = 0$). This ensures that, at the start of discriminative training, the front end and back end are well matched.

Although the framework would easily enable updating the GMM parameters μ_m and σ_m at the same time as the offset parameters b_m, they were held fixed for the experiments presented in this paper.

5. RESULTS

To demonstrate the effectiveness of joint front end and back end discriminative training, five separate experiments were conducted against a strong maximum likelihood baseline.

5.1. Aurora 2 Baseline

The experiments presented here were based on the data, code, and training scripts provided within the Aurora 2 task[12]. The task consists of recognizing strings of English digits embedded in a range of artificial noise conditions.

The acoustic model (AM) used for recognition was trained with the standard "complex back end" Aurora 2 scripts on the multi-condition training data. This data consists of 8440 utterances, and includes all of the noise types seen in test set A, at a subset of the SNR levels.

The AM contains eleven whole word models, plus `sil` and `sp`, and consists of a total of 3628 diagonal Gaussian mixture components, each with 39 dimensions.

[2] Training the rotations A_m is also possible, and was derived in [11].

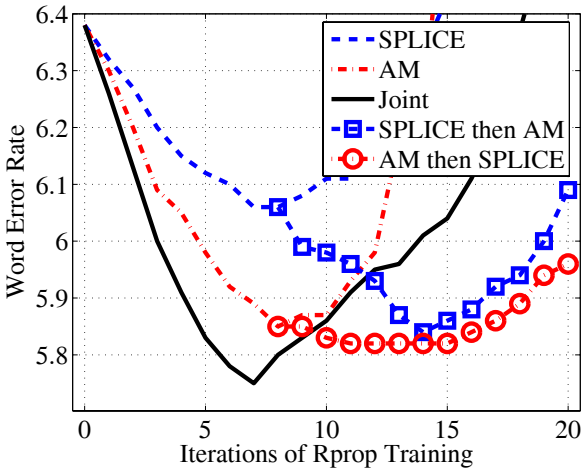

Fig. 1. *Performance on test set A. Joint training outperforms every other combination of front end and back end training.*

Each utterance in the training and testing sets was normalized using whole-utterance Gaussianization[13]. This simple cepstral histogram normalization (CHN) method provides us with a very strong baseline, and effectively normalizes the useful dynamic range of the model parameters.

Word error rate (WER) results are presented on test set A, averaged across 0 dB to 20 dB SNRs. The baseline system in this paper achieves a WER of 6.38%, which is better than most published numbers on this task. Consequently, even small gains represent strong experimental results. For reference, the ETSI Advanced Front-End has a WER of 6.26%[14] on the same test set.

5.2. Experimental Results

It is not surprising that training the back end parameters according to an MMI objective function decreases the WER of the system. The "AM" graph in Figure 1 describes the performance improvement when the mean parameters in the back end acoustic model are trained with Rprop. After eight iterations, the system is well trained and significantly better than the maximum likelihood baseline.

The "SPLICE" graph in Figure 1 indicates that the WER can be improved just by retraining the offset parameters of the SPLICE transformation. In fact, even though the back end has over fourteen times as many parameters, the improvement from SPLICE training alone is well over half the improvement from AM training alone.

The graphs "SPLICE then AM" and "AM then SPLICE" in Figure 1 demonstrate the result of serial training. For the former case, the best SPLICE model is used as a starting point for discriminative training of the back end. For the latter, the order is reversed. Training the back end first (AM then SPLICE) appears to be the more stable approach.

The "Joint" graph in Figure 1 shows that joint discriminative training can be faster and more accurate than any of the other training schedules. Both the SPLICE feature transformation and the mixture components in the acoustic model are cooperating to find a feature space that works well for the task. The best accuracy is achieved after iteration seven, at 5.75% WER.

The improvement from the best AM then SPLICE model to the best joint model is significant at the $p = 0.043$ level according to the Sign-test (216 utterances improve, 175 utterances degrade).

6. SUMMARY

We have presented a framework for jointly training the front end and back end of a speech recognition system against the same discriminative objective function. Experiments indicate that joint training achieves better word error rates with fewer iterations.

To expand upon this initial result, future work should include training state-conditional or tied rotation matrices in addition to the offsets, and training the SPLICE GMM parameters.

7. REFERENCES

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] Y. Bengio, R. De Mori, G. Flammia, and R. Kompe, "Global optimization of a neural network-hidden Markov model hybrid," *IEEE Trans. on Neural Networks*, vol. 3, no. 2, pp. 252–259, 1992.

[3] J. Wu and Q. Huo, "An environment compensated minimum classification error training approach and its evaluation on Aurora2 database," in *Proc. ICSLP*, 2002, pp. 453–456.

[4] J. Droppo, L. Deng, and A. Acero, "Evaluation of SPLICE on the Aurora 2 and 3 tasks," in *Proc. ICSLP*, 2002, pp. 29–32.

[5] D. Povey, *Discriminative Training for Large Vocabulary Speech Recognition*, Ph.D. thesis, Cambridge University, 2003.

[6] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," in *IEEE Int. Conf. on Neural Networks*, 1993, vol. 1, pp. 586–91.

[7] P. F. Brown, *The Acoustic Modeling Problem in Automatic Speech Recognition*, Ph.D. thesis, CMU, 1987.

[8] V. Valtchev, *Discriminative Methods in HMM-based Speech Recognition*, Ph.D. thesis, Cambridge, 1995.

[9] J. Le Roux and E. McDermott, "Optimization methods for discriminative training," in *Proc. Eurospeech*, 2005, pp. 3341–3345.

[10] D. Ellis and M. Gomez, "Investigations into tandem acoustic modeling for the Aurora task," in *Proc. Eurospeech*, 2001, pp. 189–192.

[11] J. Droppo, M. Mahajan, A. Gunawardana, and A. Acero, "How to train a discriminative front end with stochastic gradient descent and maximum mutual information," in *Proc. IEEE ASRU*, 2005.

[12] H. G. Hirsch and D. Pearce, "The Aurora experimental framework for the performance evaluations of speech recognition systems under noisy conditions," in *ISCA ITRW ASR2000*, 2000.

[13] A. de la Torre, J. C. Segura, C. Benítez, A. Peinado, and A. J. Rubio, "Non-linear transformations of the feature space for robust speech recognition," in *Proc. ICASSP*, 2002, vol. 1, pp. 401–404.

[14] D. Macho, L. Mauuary, B. Noé, Y. M. Cheng, D. Ealey, D. Jouvet, H. Kelleher, D. Pearce, and F. Saadoun, "Evaluation of a noise-robust DSR front-end on Aurora databases," in *Proc. ICSLP*, 2002, pp. 17–20.