ADAPTIVE BOOSTING FEATURES FOR AUTOMATIC SPEECH RECOGNITION

Kham Nguyen†, Tim Ng, and Long Nguyen

BBN Technologies, 10 Moulton Street, Cambridge, MA 02138, USA
† Northeastern University, 360 Huntington Ave., Boston, MA 02115, USA
knguyen@ece.neu.edu, {tng,ln}@bbn.com

ABSTRACT

In this paper, we present a method to extract probabilistic acoustic features by using the Adaptive Boosting algorithm (AdaBoost). We build phoneme Gaussian mixture classifiers, and use AdaBoost to enhance the classification performance. The outputs from AdaBoost are the posterior probabilities for each frame given all phonemes. Those posterior features are then used to train a new acoustic model in a similar way as the original features. The gains are obtained when we combine them with the baseline features PLP. Adaboost systems for both Arabic and Mandarin have contributed gains on the final system combination in the GALE evaluation of 2011.

Index Terms— Adaptive Boosting, Posterior features, Gaussian mixture classifiers

1. INTRODUCTION

The traditional acoustic features such as PLP and MFCC widely used in most automatic speech recognition (ASR) systems do not carry phoneme information explicitly. In recent years, Multi-Layer Perceptrons have been used to extract discriminative features. The final posterior probabilities from the output layer, or any intermediate layer, are then used as new features to train acoustic models for ASR systems. Yu and Deng have developed a deep neural-net technique (DNN) [1] using an MLP with higher number of layers. Even though DNN technique produces better performance, it has a scalable problem for large amount of training data. The scalable problem is solved by the deep convex network (DCN) technique, where each DCN includes a number of modules and the modules can be trained in parallel with convex optimzation. Another method to extract discriminative features by using Adaptive Boosting algorithm[2] was proposed by Pei Yin[4]. In their work, they used decision trees as "base" classifiers. The final features were used in various recognition applications with good improvements.

In this work, we build the Gaussian mixture based classifier to convert each acoustic feature vector to a posterior probability vector given all the phonemes. To enhance the performance of the Gaussian mixture classifiers, we apply the adaptive boosting (AdaBoost) algorithm to combine the GMM-based classifiers. We also develop a parallel technique where the whole training process is done in parallel so that the training latency can be feasible even for very large systems. The output phoneme posteriors obtained by this AdaBoost algoritm with GMMs are used in place of or in addition to the traditional PLP features to train the acoustic model.

The rest of the paper is organized as follows. In section 2, we review the AdaBoost algorithm and propose our newly-developed multiclass AdaBoost algorithm for GMMs. Section 3 will describe the experimental results. Section 4 will concludes the paper.

2. ADAPTIVE BOOSTING

Adaptive Boosting (AdaBoost) was introduced by Freund and Shapire in 1997 as a meta-algorithm to combine classifiers and generate outputs that are more accurate than any individual classifier[2]. These individual classifiers, also known as *weak learners* in AdaBoost lingo, need not be of the same form and/or structure. The input to the algorithm is a set of N labeled samples (x_1, y_1), (x_2, y_2), ..., (x_N, y_N) as training data, where x_i is a sample instance and y_i is the sample label or the truth. During iterative training, the total sum of the weights for all training samples stays constant at 1 at each iteration, i.e. $\sum_{i=1}^{N} w_i^t = 1$. At the beginning, all weights are set equally to $1/N$. Misclassified samples caused by classifier h_t at iteration t are assigned bigger weights so that the next classifier h_{t+1} at iteration $t+1$ will focus more on these hard samples. Note that, this meta-algorithm implicitly assumes that classifier h_{t+1} would have been trained/updated using the same training data but with the non-uniform weights derived from the classification performance of classifier h_t at iteration t. In addition to the sample weights that are changed between iterations, each classifier h_t is assigned a weight β_t computed from the total error (or *pseudo loss*) ε_t when h_t was evaluated on all training samples at iteration t. Without loss of generality, assume we are dealing with a 2-class classification problem, i.e. $y_i \in \{0, 1\}$, then the hypothesis $h_t(x)$ is a binary mapping $h_t(x) : x \rightarrow \{0, 1\}$. The final output of the AdaBoost meta-algorithm for a binary case is the weighted summation of all iteration classification hypotheses as shown in the following equation.

$$H(x) = \frac{\sum_{t=1}^{T} (\log \frac{1}{\beta_t}) h_t(x)}{\sum_{t=1}^{T} \log \frac{1}{\beta_t}} \quad (1)$$

where, $\beta_t = \frac{\varepsilon_t}{(1-\varepsilon_t)}$, the classifier's weight, and
$\varepsilon_t = \frac{1}{N} \sum_{i=1}^{N} |h_t(x_i) - y_i|$, the classifier's total error.

2.1. Example of a 2-Class Problem Using AdaBoost

To illustrate the steps and strength of the AdaBoost algorithm, we would like to present Schapire's toy example again here in Figures 1 and 2. In this simple example, we aim to classify all the training samples with two labels, red and blue in Fig. 1. We can see that the data is not separable by a straight-line classifier. By using AdaBoost, we can combine three weak classifiers to classify the data perfectly. At the first iteration in Fig. 2(a), the weak learner h_1 is the vertical line. It classifies 10 samples to two groups, with blue on the left and red on the right. Three out of the 10 samples (samples 5, 6, and 8) are misclassified. With uniform weights, the error of the iteration is $\varepsilon_1 = 0.3$, and the weight β_1 is computed accordingly. The weights

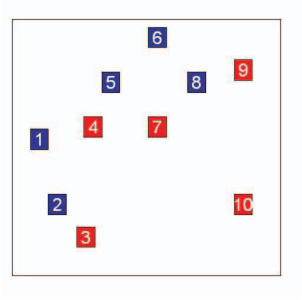

Fig. 1. 2-D data points.

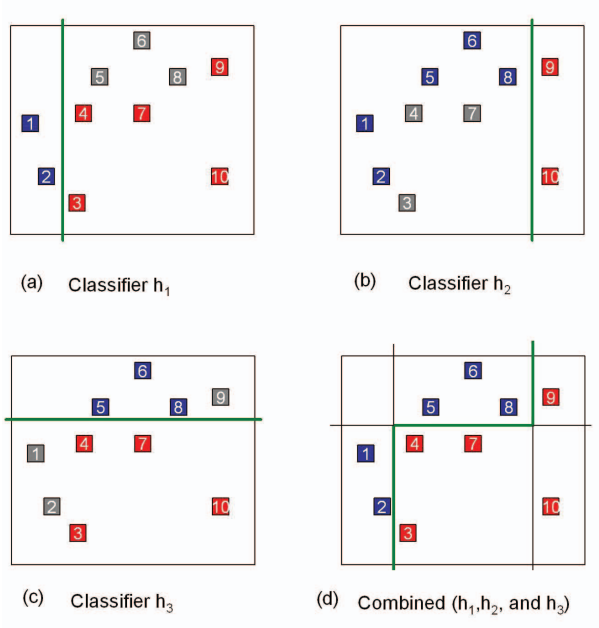

Fig. 2. h_1, h_2, h_3 classifiers and combined (h_1, h_2 and h_3)

of training samples are also changed properly in which the weights of the three misclassified samples are increased, while the weights of the rest are decreased. The weak learner is called at the second iteration in Fig. 2(b) with new sample weights. This time, the misclassified samples are 3, 4, and 7. The error ε_2 is 0.21, instead of 0.3, due to the weighted samples. Similarly, the third classifier is the horizontal line in Fig. 2(c). The outputs of the three weak learners are then combined with weights β_t, and we can see that the data is classified correctly for all samples in Fig. 2(d) using the combined green zig-zag line segments. [A thorough analysis of this illustration can be found on the web by searching for Schapire's boosting tutorial.]

2.2. Multiclass AdaBoost for Automatic Speech Recognition

As mentioned before, instead of using MLP to derive phoneme posteriors, we explore using multi-class AdaBoost to derive phoneme posteriors as probabilistic acoustic features for automatic speech recognition. Assume the phoneme set in our speech recognition system has k phonemes. Given each speech frame, we want to derive a k-dimensional vector of phoneme posteriors, one for each phoneme. The simplest approach could be using a Gaussian mixture model (GMM) for each phoneme. Each GMM is estimated using all speech frames aligned to that phoneme. The resulting model is then just **one** set of k GMMs. However, we believe that we could do better by having T sets of GMMs using adaptive boosting with T iterations such that the GMM set obtained in iteration t is capable of circumventing the misclassifications in iteraton $t-1$.

In this multiple phonetic class AdaBoost using GMM formulation, we can denote the phoneme posterior probability of sample x for phoneme y as

$$h_t(x|y) = \frac{\sum_j m_y^j \mathcal{N}_y^j(x)}{\sum_c \sum_j m_c^j \mathcal{N}_c^j(x)} \tag{2}$$

where $\mathcal{N}_y^j(x)$ is the mixture component Gaussian density value for x of component j of the GMM for phoneme y, and m_y^j is the mixture weight of component j of the GMM for phoneme y.

The final output of this multi-class AdaBoost using GMMs is a series of k $H(x|y)$, one for each phoneme y of the set of k phonemes, of the form:

$$H(x|y) = \frac{\sum_{t=1}^{T} (\log \frac{1}{\beta_t}) h_t(x|y)}{\sum_{t=1}^{T} \log \frac{1}{\beta_t}} \tag{3}$$

2.2.1. *Multiclass AdaBoost Parallel Algorithm*

We can partition all of the training data into k disjoint subsets, one for each phoneme, i.e. each data subset consists of only speech frames having the same phoneme label. We can then divide the multiclass AdaBoost training procedure into two stages. The first stage will be done in k parallel processes, one for each phoneme, in which the algorithm goes over only the data subset of that phoneme to estimate the phoneme-specific GMM and saves other relevant data for the calculation of the total pseudoloss in the second stage. The two stages of this algorithm are explained in the pseudo code below:

For each iteration t:

- Stage 1: For each class c in parallel, weight and train:

  $t = 0$: $D_t(x, y) = 1/(n-1)$, $y \neq c$, else 0, where n is the size of the data subset.

  $t > 0$: $D_t(x, y) = D_{t-1}(x, y)/Z_{t-1} * \beta_{t-1}^{(1+h(x|c)-h(x|y))/2}$ (4)

  - weight sample x with weight $w_t(x) = \sum_y D_t(x, y)$

  - train a new GMM for class c with weighted samples

  - compute un-normalized pseudoloss $\varepsilon_t^c = 1/2 \sum_j \sum_y D_t(x, y)(1 - h(x|c) + h(x|y))$

  - store the class normalizer $Z_c = \sum_j \sum_y D_t(x_j, y)$

- Stage 2: Synchronize (after all proceses in Stage 1 finished)

  - compute total normalizer $Z_t = \sum_c Z_c$

  - compute total psuedoloss $\varepsilon_t = \sum_c \varepsilon_t^c / Z_t$

  - compute classifier weight for the iteration $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$ (5)

During the training for each class, the un-normalized pseudoloss ε_t^c is computed and is normalized afterward, and that's why the procedure can be done in parallel. The total pseudoloss will be normalized after the value of the total normalizer is computed. One of the termination conditions is that the total psuedoloss is less than 0.5, and by using (5) we have the iteration weight β_t less than 1. In formula

(4) of distribution $D_t(x, y)$, we can see that when the likelihood value of $h(x|y)$ is "large" relatively compared to the likelihood of the truth $h(x|c)$, or the sample is not classified correctly, the distribution $D_t(x, y)$ will increase relatively compared with other samples in the class since β_t is less than 1. That means the total weight of the sample will increase relatively compared to the other samples, which makes the sample to be emphasized more in the next iteration.

3. EXPERIMENTAL RESULTS

3.1. Experimental Data

To study the behaviour of the multiclass AdaBoost algorithm using GMMs proposed above, we set up three data sets derived from the big English broadcast news speech corpus used at BBN for the EARS and GALE Programs. The first data set to be used for training comprises 50 hours of randomly selected broadcast news shows. The second data set, also randomly selected, consists of 3 hours to be used for cross-validation purpose. The third data set is the development test set that consists of 3 hours of CNN's news shows aired in 2004. The training data set was phonetically aligned to obtain the *truth* labels for training the AdaBoost model. The cross-validation data set was also phonetically aligned to obtain the *truth* labels to measure the frame classification accuracy of the AdaBoost model. The forced alignments were obtained using one of the best English BN systems developed in the past at BBN. However, we expected there are errors in these alignment since it was done automatically.

3.2. AdaBoost with GMM Training

The labeled sample (x_i, y_i) input to the proposed multiclass AdaBoost algorithm is a pair of a speech frame and its phoneme label. Each speech frame is the standard 60-dimensional PLP feature vector (14 base cepstral and a normalized energy, together with their first, second, and third derivatives). In Stage 1 of the algorithm, we have 50 parallel processes for the 50 phonemes in our English Phoneme set. The 512-component GMM for each phoneme class are initialized by running a few iterations of the K-Means algorithm, starting from the randomly-selected 512 samples as the initial 512 centroids. The GMM was then refined by running a few more iterations of the Expectation-Maximization (EM) algorithm. Note again that the samples used to train the GMM always use iteration-specific sameple weights derived based on the performance of the *classifiers* in the previous iteration.

3.3. Frame Classification Accuracy

Figure 3 shows the frame classification accuracy of the 512-component GMM AdaBoost model on the cross-validation data set as a function of the number of iterations of the training of the AdaBoost model. We can see from the frame accuracy curve that the multiclass AdaBoost algorithm using GMMs improves *smoothly* and converges at around after 20 iterations. However, we cannot explain the *discontinuity* occured at the second iteration.

To measure the effect of the number of GMM components on the classification results, we also tried 1024- and 2048-component GMMs. The comparison of accuracy rates using these 3 different model sizes are shown in Table 1. On the training set, the accuracy rate improved by 5% absolute when doubling the number of GMM components. However, the rate improved less than 1% absolute on the cross-validation set. The reason could be overfitting when the classifiers try to memorize the training data when the model has more parameters.

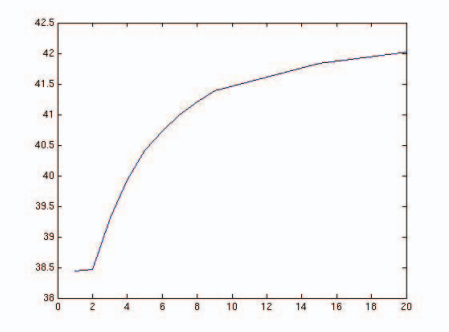

Fig. 3. Classification Results: Frame Accuracy Rates Over Number of Iterations.

#GMM-bins	Training	Cross Validation
512	44.1	42.0
1024	49.3	42.6
2048	54.2	42.8

Table 1. Frame Accuracy Rates Over Number of GMM Components.

3.4. ASR Experiments on English Data

We carried out three automatic speech recognition (ASR) experiments using three different ASR models trained on three different sets of input features. The first baseline ASR system use the standard PLP feature vectors. This is only a speaker-independent configuration without speaker and/or model adaptation. In this system, the standard 60-dimensional feature vectors comprising the 14 base cepstra and the normalized energy together with their first, second, and third derivatives, are dimensionally reduced to 46 dimensions using Linear Discriminant analysis (LDA) and then are decorrelated using maximum likelihood linear transform (MLLT). Recognition word error rate (WER) on the CNN test set (described above) using this first baseline standard ASR system is showed in the first row of Table 2.

The second baseline ASR system uses phoneme posterior probabilities features derived by a Multi-Layer Perceptron (MLP) neural network. The WER of this system is shown in the second row of Table 2. This is the typical performance (with some degradation) relative to standard PLP features when the MLP features are used *in place* of the PLP features as reported by all ASR research sites.

The ASR experimental result produced by the third ASR system that uses the AdaBoost features in place of the PLP features is shown in the last row of Table 2. The AdaBoost model uses 512-component GMMs trained with 20 iterations. To have a direct comparison to the baseline PLP system, the 50-dimensional AdaBoost features (representing the 50 phoneme poseriors) are also dimensionally reduced to 46 dimensions using LDA (as in the case of the PLP features) and then decorrelated using MLLT. Similar to the situation of the MLP features, the AdaBoost features produced higher WER than the baseline PLP system (24.6% versus 21.1%). However, the AdaBoost features are 0.8% absolute better than the MLP features (24.6% versus 25.4%). Again, it has been shown that probabilistic acoustic features (such as MLP) helped only if they were used in addition to PLP features within sophitistacated concatenation or combination framework[5].

We also experimented with AdaBoost features using different

System	WER
PLP	21.1
MLP	25.4
AdaBoost	24.6

Table 2. Comparison of ASR WERs for ASR systems using PLP, MLP, and AdaBoost Features.

numbers of phonetic classes. Instead of using 50 phoneme classes as in the first set of experiments above, we increased the number of phonetic classes to 99 and 300. We replaced the phoneme classes by using clustered allophones that are phonemes in specific phonetic contexts. When increasing the number of classes from 50 to 99, the WER of the new AdaBoost features is reduced by 0.3% absolute. The ASR performance was improved further (by 0.9% absolute) when the number of phonetic classes is increased to 300. Comparison of these three results are in Table 3.

No. classes	WER
50	24.6
99	24.3
300	23.4

Table 3. Comparison of ASR WERs using different numbers of phonetic classes.

3.5. ASR Experiments on Arabic and Mandarin Data

Even though this was on-going work with only preliminary results during the Phase 5 evaluation of the GALE program, we attempted using AdaBoost features in our official GALE evaluation systems for Arabic and Mandarin languages where we have more than a thousand of hours of training data. For both systems, we combined AdaBoost features with PLP features and used region dependent tranform (RDT) technique to reduce the merged features for HMM training. The detail of using RDT on the combined features can be found in [5]. Note that the system that used the AdaBoost as additional features was only as **one** additional system to be combined with many other systems using ROVER combination.

As can be seen in Table 4, the comparison of the two ROVER results, ROVER4 of 4 systems without AdaBoost features and ROVER5 of 4 previous systems and the additional system that used AdaBoost features, shows that AdaBoost features contributed to the reduction of 0.1% absolute in character error rates (CER) for test sets cd10c and cd10d for the Mandarin language while there are no degradations in other test sets.

System	cd9s	ce9s	cd10c	cd10r	cd10d
ROVER4	7.8	7.2	12.3	7.9	24.6
ROVER5	7.8	7.2	12.2	7.9	24.5

Table 4. ROVER results for Mandarin data without and with AdaBoost system.

Similarly for the Arabic language, we constructed the 12^{th} system in which we used the AdaBoost features to combine with the other 11 systems also produced a modest 0.1% absolute reduction in word error rate (WER) for the three test sets ad9s, ae9s, and ad10c, while there were no degradations in the other test sets. The comparison of the WERs of the two ROVER combinations are shown in Table 5.

System	ad9s	ae9s	ad10c	ad10d	ad10r
ROVER11	13.2	8.5	12.1	22.2	9.9
ROVER12	13.1	8.4	12.0	22.2	9.9

Table 5. ROVER results for Arabic data without and with AdaBoost system.

4. CONCLUSION

In this paper, we presented the multiclass AdaBoost algorithm for GMMs to extract phoneme posterior features and its usage for ASR. Firstly, compared to the "base" classifier of GMM, [AdaBoost with only the first iteration with equal sample's weights], the AdaBoost for GMM has shown improvement in classification results. When we increase the number of GMM components as well as the number of training iterations, the classification results also improve. Secondly, the output features have been used to train ASR systems with some reasonable results. The gains were obtained when we tried combining AdaBoost with PLP features using RDT training. The AdaBoost systems for both Arabic and Mandarin have been used for GALE evaluation in 2011.

5. ACKNOWLEDGMENTS

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-06-C-0022. 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 DARPA or its Contracting Agent, the U.S. Department of the Interior.

6. REFERENCES

[1] Li Deng and Dong Yu, "Deep Convex Network: A Scalable Architecture for Speech Pattern Classification," in *Proceedings of Interspeech*, Florence, Italy, August 2011

[2] Yoav Freund and Robert Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting", Journal of Computer and System Sciences, 1997.

[3] G. Zweig and M. Padmanabhan, "Boosting Gaussian Mixtures in an LVCSR System," in *Proceedings of ICASSP*, Istanbul, Turkey, June 2000.

[4] Pei Yin, Irfan Essa, Thad Starner, James M. Rehg "Discriminative Feature Selection For Hidden Markov Models Using Segmental Boosting," in *Proceedings of ICASSP*, Las Vegas, Nevada, May 2008.

[5] T. Ng, B. Zhang, and L. Nguyen, "Jointly optimized discriminative features for speech recognition," in *Proceedings of InterSpeech*, Makuhari, Chiba, Japan, September 2010.

[6] B. Zhang, S. Matsoukas, and R. Schwartz, "Discriminatively trained region-dependent transform for speech recognition," in *Proceedings of ICASSP*, Toulouse, France, May 2006.