Boosting Systems for LVCSR

George Saon and Hagen Soltau

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

Abstract
We employ a variant of the popular Adaboost algorithm to train multiple acoustic models such that the aggregate system exhibits improved performance over the individual recognizers. Each model is trained sequentially on re-weighted versions of the training data. At each iteration, the weights are decreased for the frames that are correctly decoded by the current system. These weights are then multiplied with the frame-level statistics for the decision trees and Gaussian mixture components of the next iteration system. The composite system uses a log-linear combination of HMM state observation likelihoods. We report experimental results on several broadcast news transcription setups which differ in the language being spoken (English and Arabic) and amounts of training data. Additionally, we study the impact of boosting on ML and discriminatively trained acoustic models. Our findings suggest that significant gains can be obtained for small amounts of training data even after feature and model-space discriminative training.

Index Terms: speech recognition, boosting

1. Introduction
Current ASR systems can reach high levels of performance for particular domains as attested by various government-sponsored speech recognition evaluations. This comes at the expense of an ever increasing complexity in the recognition architecture. Typically, LVCSR systems employ multiple decoding and rescoring passes with several speaker adaptation passes in-between. Improved system performance is obtained by “cross-pollinating” diverse acoustic models through cross-adaptation and system combination. These models differ in one or more design parameters such as input features, acoustic modeling paradigm, phonetic context, discriminative training criterion to name a few. Unfortunately, a lot of human intervention is required in choosing which systems are good for combination, knowledge which is often task-dependent and cannot be easily transferred to other domains. Ideally, one would want an automated procedure for training accurate systems/models which make complementary recognition errors.

Boosting is a popular machine learning technique for incrementally building linear combinations of “weak” models to generate an arbitrarily “strong” predictive model. This is achieved by focusing the weak learners on examples that are hard to classify. Focusing is done by training the learners on weighted data where the weights of wrongly classified examples are increased at each iteration. Adaboost, a popular boosting algorithm introduced in [1], has several desirable properties. One of them is that the training error is guaranteed to drop to zero exponentially fast in the number of boosting rounds regardless of what weak learners are used as long as their individual weighted error rate is less than 50%. Boosted models have also a small generalization error which is attributed to their “separating” or margin maximizing behavior. Lastly, a not so well-known property is that the weight vector for the next iteration is orthogonal to the error vector of the current iteration forcing the next weak learner to make more uncorrelated errors.

Despite these appealing properties, there has been relatively little research on boosting for ASR. Indeed, except for the works of [2, 3, 4] and [5], the authors are not aware of other applications of boosting in the context of GMM-based speech recognition. One explanation could be that the GMM classifiers prevalent in ASR are neither weak nor unstable1.

In [2], the authors boost GMM’s for frame classification using a multi-class generalization of Adaboost called Adaboost.M2. The same algorithm is applied in [4] with the added refinement that the pseudo-loss uses a discriminative measure based on either utterance or word posterior difference between correct and competing hypothesers. In [5], the authors perform a functional gradient boosting of the observation log-likelihoods by adding one Gaussian mixture component per state at each round of boosting. All works have in common that they try to boost the performance of GMM’s which is already a strong and stable classifier. A more promising approach is based on the observation that modern ASR systems do include an unstable learner which is the phonetic decision tree used to define the context-dependent HMM states.

In [6], the authors introduce randomness in the decision tree growing process and combine several systems with randomized trees using ROVER. In [7], the authors build directed decision trees to generate complementary systems: frame-level statistics for the frames that are correctly decoded by the current system up to that point.

What we propose to do in this paper is to boost entire ASR systems including the phonetic decision trees. Based on the findings of [6, 7], changing the decision trees should result in more diverse ASR systems which should be beneficial for system combination. Also, similar to [3], we opt for the conceptually simpler Adaboost.M1 variant with some modifications that will be explained later on.

The paper is organized as follows: in section 2, we outline the boosting methodology. In section 3, we present some experimental evidence of its utility followed by some concluding remarks in section 4.

2. Boosting framework

2.1. General setting

Given training data \(z_1, \ldots, z_m\), a dictionary of weak learners \(\mathcal{H}\) and a loss function \(L\), a boosting algorithm sequentially finds \(h_1, h_2, \ldots, h_T \in \mathcal{H}\) and the constants \(\alpha_1, \alpha_2, \ldots, \alpha_T \in \mathbb{R}^+\)

1Unstable classifiers such as decision trees have the property that small changes in parametrization or training data lead to large changes in classification performance.
which minimize the loss \( \sum_{i=1}^{m} L(z_i, H(z_i)) \) where

\[ H(z) = \sum_{t=1}^{T} \alpha_t h_t(z). \]

The Adaboost algorithm introduced in [1] was specifically developed for binary classification \( z = (x, y) \) with \( y \in \{-1, 1\} \) and the exponential loss function

\[ L(y, H(x)) = e^{-yH(x)}. \]

The argument of the exponential is called the margin of the classifier and the loss is sometimes referred to as the potential of the margin. The algorithm works by iteratively fitting a weak learner to the re-weighted training data. The weights depend on the performance of the models so far and are updated so as to emphasize the more challenging examples. It has been shown that Adaboost implements a coordinate-wise gradient descent on the exponential loss function. In this paper we focus on the slightly more general Adaboost algorithm which is given below.

**Algorithm 1** Generalized Adaboost for multiple classes

**Input:** Training data \( \{(x_i, y_i)\}_{i=1}^{m}, x_i \in \mathcal{X}, y_i \in \{1, \ldots, C\} \)

Dictionary of weak classifiers \( H \)

1: **Initialize** \( D_1(i) = 1/m \)
2: for \( t = 1 \ldots T \) do
3: Train weak classifier \( h_t \) using distribution \( D_t \)
4: Choose \( \alpha_t > 0 \)
5: Update distribution:

\[
D_{t+1}(i) = \frac{D_t(i)}{Z_t}, \quad \left\{\begin{array}{ll}
e^{-\alpha_t}, & \text{if } h_t(x_i) = y_i \\
e^{\alpha_t}, & \text{otherwise}
\end{array}\right.
\]

where \( Z_t \) is a normalization factor such that \( D_{t+1} \) is a distribution.
6: end for

**Output:** \( H(x) = \arg \max_{y \in \{1, \ldots, C\}} \sum_{t=1}^{T} \alpha_t I(h_t(x) = y) \)

\( I(\cdot) \) is the indicator function (1 if the argument is true, 0 otherwise). One noteworthy property of this algorithm is that the training error is bounded from above by the product of normalizing constants for any \( \alpha_t \) i.e.:

\[
\frac{\sum_{i=1}^{m} I(H(x_i) \neq y_i)}{m} \leq \prod_{t=1}^{T} Z_t
\]

The proof of (2) follows from bounding the empirical error by the average exponential loss for multi-class classification

\[
\frac{1}{m} \sum_{i=1}^{m} e^{-\sum_{t=1}^{T} \alpha_t [I(h_t(x_i) = y_i) - I(h_t(x_i) \neq y_i)]}
\]

and from showing that the latter is the RHS of (2). This is accomplished by unraveling the update rule for \( D_{T+1}(i) \) and from observing that \( D_{T+1} \) is a distribution.

In standard Adaboost, \( \alpha_t \) is chosen to minimize \( Z_t \) at each iteration. Define

\[
\epsilon_t = \frac{1}{m} \sum_{i=1}^{m} D_t(i) I(h_t(x_i) \neq y_i)
\]

the weighted error of classifier \( h_t \). A simple calculation shows that the minimizer of \( Z_t \) is

\[
\alpha_t = \frac{1}{2} \log \left( \frac{1 - \epsilon_t}{\epsilon_t} \right)
\]
and an $N$-state HMM $\lambda$, the auxiliary function for weighted maximum likelihood training can be written as$^2$

$$Q(\lambda, \vec{x}) = \sum_{t=1}^{T} \sum_{i=1}^{N} p(q_t = i|x_1 \ldots x_T, \lambda) \log p(x_t|q_t = i, \vec{x}) \, w_t$$

$$= \sum_{t=1}^{T} \sum_{i=1}^{N} \gamma_t(i) w_t \log p(x_t|q_t = i, \vec{x})$$

(9)

where $\gamma_t(i)$ represents the posterior probability of being in state $i$ at time $t$ given the observation sequence. This means that state posterior occupancies are simply multiplied with the frame weights when computing the sufficient statistics for Gaussian estimation. The resulting algorithm is called boosted Baum-Welch in [5] with the difference that the weights are also state-dependent which is not the case here. Similarly, for discriminative training, the frame-level statistics for transform and model parameter estimation are also multiplied with the weights.

### 2.3. Model combination

Once the component acoustic models are trained, there are several ways of combining them into an aggregate system. One way is to do separate decodings and to combine the outputs using ROVER as was done in [6]. We follow the example of [3] where the authors combine the likelihoods at the HMM state level, a technique referred to as state-locked, multi-stream decoding. This has the advantage that only one decoding pass is required for the combined models. It is not necessary for the component acoustic models to have the same phonetic decision tree. Instead, we perform model fusion using decision tree arrays as described in a companion paper [9]. The state likelihoods are combined in a log-linear fashion with uniform weights.

### 3. Experiments

We report results here on three experimental setups for broadcast news transcription which differ in the language being spoken (English and Arabic) and amount of training data (50 hours and 2000 hours). In all setups, the input speech is represented by VTL-warped PLP cepstral frames which are mean and variance normalized on a per-speaker basis. Every 9 consecutive frames are concatenated and projected down to 40 dimensions by an LDA transform followed by a decorrelating semi-tied covariance (STC) transform. Words in the recognition lexicon are represented as sequences of phones, and phones are modeled with 3-state left-to-right HMMs that do not permit state skipping. All acoustic models have pentaphone cross-word context and are SAT trained with feature-space MLLR (FMLLR). At test time, speaker adaptation is performed with VTLN, FMLLR and multiple regression tree-based MLLR transforms. We discuss results for two sets of acoustic models: models trained with maximum likelihood and discriminatively trained models and feature-space transforms. The criterion used for discriminative training is boosted or margin-based MBI described in [10, 11].

Each boosting iteration consists of three steps: (1) train acoustic model $t$ (including decision trees) with weight distribution $D_t$; (2) decode training data with model $t$ and a unigram LM and mark correct and incorrect frames; (3) construct distribution $D_{t+1}$ by multiplying the $D_t$'s of the correct frames by a factor $\beta < 1$ and by re-normalizing. In order to avoid having to change various thresholds for decision tree growing and Gaussian mixture splitting, we normalize the weights so that they sum up to the number of frames in the training data.

#### 3.1. English BN 50 hours

The size of the acoustic model trained at each boosting iteration is roughly 2200 context-dependent HMM states and 50K Gaussians. The recognition vocabulary has 90K words and decoding is done with a small 4M ngrams 4-gram language model.

In Figure 1 we show the word error rates for the boosted systems for both maximum likelihood and discriminatively trained (DT) models on the DEV04f test set which has 2 hours of speech and 22.6K words.

![Figure 1: Word error rates of boosted systems on DEV04f for English BN with 50 hours of training data.](image)

As can be seen, the improvements for the ML and DT systems are 1.2% and 0.9% absolute, respectively. We tuned the weight decaying factor $\beta$ on a held-out set and found that its optimal value varies between the ML models ($\beta = 0.5$) and the DT models ($\beta = 0.7$). Similar values are found to be best on the Arabic BN 50 hours setup. We conjecture that higher values (i.e., less aggressive steps) are required when boosting more accurate base acoustic models. The influence of $\beta$ is also illustrated in Figure 2 where we plot the weighted and unweighted frame error rates on the training data for the individual ML models for two values of $\beta$. We observe that the $\epsilon_t$'s increase faster for a smaller $\beta$. We also note that standard Adaboost with the stepsize from equation (5) (which corresponds to $\beta = 0.16$ at iteration 1) will terminate after only one iteration.

#### 3.2. Arabic BN 50 and 2000 hours

The size of the acoustic models that are trained on 50 hours are comparable to the English setup: 2250 context-dependent states and 40K Gaussians. In contrast, the models trained on 2000 hours are much larger with 5000 states and 800K Gaussians. This size has been optimized for best performance during the Darpa GALE phase 4 evaluation. For both scenarios, the recognition vocabulary contains 795K words and decoding is done with a 78M ngrams 4-gram LM. More details about the Arabic system used during the GALE phase 3 evaluation can be found in [12].
The authors acknowledge the support of DARPA under Grant HR0011-06-2-0001 for funding part of this work. The views, opinions, and/or findings contained in this article/presentation are those of the author/presenter and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense.

6. References


