USING BOOSTING TO IMPROVE A HYBRID HMM/NEURAL NETWORK SPEECH RECOGNIZER

Holger Schwenk[*]

International Computer Science Institute
1947 Center Street, Suite 600
Berkeley, CA 94704-1198

ABSTRACT

"Boosting" is a general method for improving the performance of almost any learning algorithm. A recently proposed and very promising boosting algorithm is *AdaBoost* [7]. In this paper we investigate if AdaBoost can be used to improve a hybrid HMM/ neural network continuous speech recognizer. Boosting significantly improves the word error rate from 6.3% to 5.3% on a test set of the OGI Numbers95 corpus, a medium size continuous numbers recognition task. These results compare favorably with other combining techniques using several different feature representations or additional information from longer time spans.

Ensemble methods or committees of learning machines can often improve the performance of a system in comparison to a single learning machine. A recently proposed and very promising boosting algorithm is *AdaBoost* [7]. It constructs a composite classifier by sequentially training classifiers while more and more emphasis on certain patterns. Several authors have reported important improvements with respect to one classifier on several machine learning benchmark problems of the UCI repository, e.g. [2, 6]. These experiments displayed rather intriguing generalization properties, such as continued decrease in generalization error after training error reaches zero. However, most of these data bases are very small (only several hundreds of training examples) and contain no significant amount of noise. There is also recent evidence that AdaBoost may very well overfit if we combine several hundred thousands classifiers [8] and [5] reports severe performance degradations of AdaBoost when adding 20% noise on the class-labels. In summary, we can say that the reasons for the impressive success of AdaBoost are still not completely understood. To the best of our knowledge, an application of AdaBoost to a real world problem has not yet been reported in the literature either. In this paper we investigate if AdaBoost can be applied to boost the performance of a continuous speech recognition system. In this domain we have to deal with large amounts of data (often more than 1 million training examples) and inherently noisy phoneme labels.

The paper is organized as follows. In the next two sections we summarize the AdaBoost algorithm and our baseline speech recognizer. In the third section we shown how AdaBoost can be applied to this task and we report results on the Numbers95 corpus and compare them with other classifier combination techniques. The paper finishes with a conclusion and perspectives for future work.

[*]new address: LIMSI-CNRS, bat 508, BP 133, 91403 Orsay cedex, FRANCE, email: schwenk@limsi.fr

1. ADABOOST

AdaBoost, constructs a composite classifier by sequentially training classifiers while putting more and more emphasis on certain patterns. For this, AdaBoost maintains a probability distribution $D_t(i)$ over the original training set. In each round t the classifier is trained with respect to this distribution. Some learning algorithms don't allow training with respect to a weighted cost function. In this case sampling with replacement (using the distribution D_t) can be used to approximate a weighted cost function. Examples with high probability would then occur more often than those with low probability, while some examples may not occur in the sample at all although their probability is not zero. Previous experiments have shown that best results in terms of training time and generalization error can be obtained when resampling a new training set from the original training set after each epoch [10].

After each round, the probability of incorrectly labeled examples is increased and the probability of correctly labeled examples is decreased. The result of training the t^{th} classifier is a *hypothesis* $h_t : X \to Y$ where $Y = \{1, ..., k\}$ is the space of labels, and X is the space of input features. After the t^{th} round the weighted error ϵ_t of the resulting classifier is calculated and the distribution D_{t+1} is computed from D_t, by increasing the probability of incorrectly labeled examples. The probabilities are changed so that the error of the t^{th} classifier using these new "weights" D_{t+1} on the errors would be 0.5. In this way the classifiers are optimally decoupled. The global decision f is obtained by weighted voting. Figure 2 left summarizes the basic AdaBoost algorithm.

In general, neural network classifiers provide more information than just a class label: it can be shown that the network outputs approximate the a-posteriori probabilities of classes, and it should be reasonable to use this information rather than performing a hard decision for one recognized class. This issue is addressed by another version of AdaBoost, called *AdaBoost.M2* [7]. It can be used when the classifier computes confidence scores[1] for each class. The result of training the t^{th} classifier is now a hypothesis[2] $h_t : X \times Y \to [0, 1]$. Furthermore we use a distribution over the set of all *miss-labels*: $B = \{(i, y) : i \in \{1, ..., N\}, y \neq y_i\}$, where N is the number of training examples. Therefore $|B| = N(k - 1)$. AdaBoost modifies this distribution so that the next learner focuses not only on the examples that are hard to classify, but more specifically on the incorrect labels against which it is hardest to discriminate. Note that the miss-label distribution D_t induces a distribution over the examples: $P_t(i) = W_i^t / \sum_i W_i^t$

[1]The scores do not need to sum to one.

[2]neural networks are usually interpreted as vector functions $\vec{h}(x) = (h(x, y_1), ..., h(x, y_k))^T$.

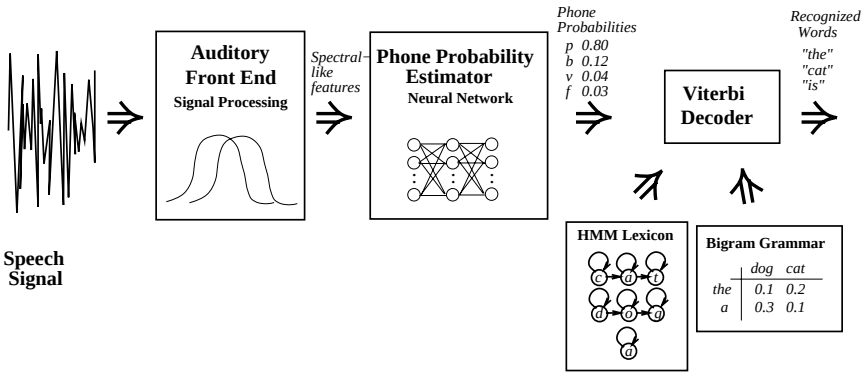

Figure 1: Block-diagram of the base-line recognizer.

where $W_i^t = \sum_{y \neq y_i} D_t(i,y)$. $P_t(i)$ may be used for resampling. The final decision f is obtained by adding together the weighted confidence scores of all classifiers (see Figure 2).

Input: sequence of N examples $(x_1, y_1), \ldots, (x_N, y_N)$ with labels $y_i \in Y = \{1, \ldots, k\}$

Init: let $B = \{(i,y) : i \in \{1, ..., N\}, y \neq y_i\}$
$D_1(i,y) = 1/|B|$ for all $(i,y) \in B$

Repeat:

1. Train neural network with respect to distribution D_t and obtain hypothesis $h_t : X \times Y \rightarrow [0,1]$
2. calculate the pseudo-loss of h_t:
$$\epsilon_t = \frac{1}{2} \sum_{(i,y) \in B} D_t(i,y)(1 - h_t(x_i, y_i) + h_t(x_i, y))$$
3. set $\beta_t = \epsilon_t / (1 - \epsilon_t)$
4. update distribution D_t
$$D_{t+1}(i,y) = \frac{D_t(i,y)}{Z_t} \beta_t^{\frac{1}{2}((1 + h_t(x_i,y_i) - h_t(x_i,y))}$$
where Z_t is a normalization constant

Output: final hypothesis:
$f(x,y) = \sum_t \left(\log \frac{1}{\beta_t}\right) h_t(x,y)$

Figure 2: AdaBoost algorithm (version using confidence scores).

AdaBoost has very interesting theoretical properties. In particular it can be shown that the error of the composite classifier on the training data decreases exponentially fast to zero as the number of combined classifiers is increased [7]. More importantly, however, bounds on the *generalization error* of such a system have been formulated [9]. These are based on a notion of *margin* of classification, defined as the difference between the score of the correct class and the strongest score of a wrong class. Obviously, the classification is correct if the margin is positive. This bound on the generalization error depends only on the complexity of one MLP and on the distribution of the margins, but not on the number of combined classifiers. It can be shown that the AdaBoost algorithm seems to be well suited to the task of maximizing the number of training examples with large margin, but recent results [8] suggest that maximization of the minimum margin may not be enough to explain the good generalization behavior of AdaBoost.

2. THE BASELINE SYSTEM

Our baseline speech recognition system is a hybrid connectionist system using a large fully connected MLP to estimate the phoneme a-posteriori probabilities and a hidden Markov model (HMM) for the time alignment. The basic processing flow is outlined in figure 1 (see [1] for more details). The raw speech signal is first converted to a more compact feature representation. We use eighth-order log-RASTA-PLP features computed over 25-ms windows with a 10-ms window step, supplemented with the log energy and delta features. Then, a fully-connected neural network with one hidden layer is used to estimate the phoneme a-posteriori probabilities. The net input consists of 9 consecutive frames, and the softmax net outputs are interpreted as the a-posteriori probabilities of the 56 possible phonemes. In this paper we use only context-independent phones. Training is done using stochastic backpropagation with the cross-entropy error function. The HMM emission probabilities are obtained by dividing the network outputs by the class prior probabilities. The decoder uses a set of per-word HMMs for multiple pronunciations and a bigram language model.

In general, the neural network is first trained alone using the hand-described phoneme labels of the training set. Then the whole recognizer is run on the training set and we perform a forced alignment knowing the whole utterances. This normally changes some of the phoneme labels, in particular at word boundaries, and the neural network is retrained on these new labels. This process, called *embedded training*, can be repeated iteratively until the labels and/or word error stabilize (typically 2-3 runs). For some data bases hand-described phoneme labels are not available. In this case the neural network is "pre-trained" using another training corpus, usually TIMIT.

This hybrid NN/HMM speech recognizer usually achieves state-of-the-art results, but there is some practical evidence that it is difficult to take advantage of very large speech corpora, e.g. more than 50h of training data. In contrast to pure HMM-based systems, adding more training data often does not improve the word error of the system, even when the number of parameters of the neural network are substantly increased. In this paper we investigate if AdaBoost's emphasizing algorithm can be used to improve the estimation of the a-posteriori class probabilities by the neural network by focusing training on the difficult and/or informative examples. For this, we replace the single neural network in figure 1 by an ensemble obtained by boosting (using the AdaBoost.M2 algorithm). Note, that in contrast to previous applications of AdaBoost, we are interested in the exact numerical output values, and not only in a hard decision for one class. However, we are not

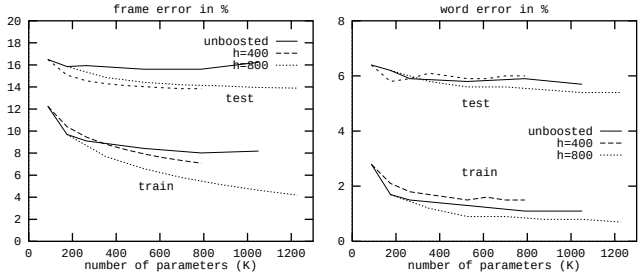

Figure 3: Frame (left) and word (right) errors using one unboosted neural network with varying number of hidden units (solid line), using several boosted 162-400-56 MLPs (dashed line) and using several boosted 162-800-56 MLPs (dotted line).

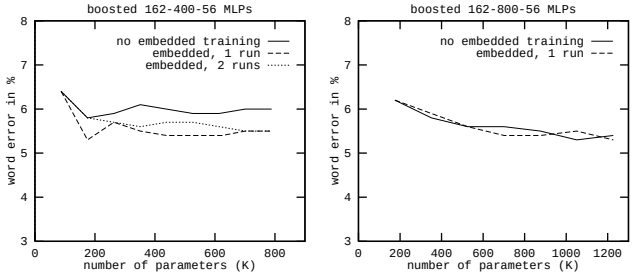

Figure 4: Effect of using embedded training on the word error rates (test set).

aware of a proof that the outputs of a *weighted ensemble of neural networks* can be interpreted as a-posteriori class probabilities. In fact, the AdaBoost algorithm tries to increase the margin between the correct class and the best second one, which over-emphasizes the posterior probability of the correct class and under-estimates the ones of the other classes. However this certainly decreases the search space of the decoder and it potentially can exclude wrong word solutions that may have be found due to too high posteriors of wrong phonemes. From the point of view of speech recognition technology this approach is worth pursuing since it may help to train hybrid NN/HMM speech recognizers with many more parameters, and to close by these means the gap between hybrid and pure HMM-based systems. It is also interesting to see if improvements at the frame level actually decrease the word error. This may indicate if current research should rather focus on better acoustic modeling or on better dictionaries and language modeling.

3. EXPERIMENTAL RESULTS

All our experiments have been done with a subset of the Numbers corpus that was collected by the Oregon Graduate Institute in 1995 [3]. It contains continuous number utterances that were cut out of naturally spoken responses from many different speakers over telephone lines. The 32-word vocabulary is restricted to numbers including such confusable sets as “four”, “fourteen” and “forty”. The training set contains almost 2 hours of speech (610,000 frames). Generalization performance was measured on a independent test set of about 40 min of speech. This speech corpus is small enough to allow numerous experiments and it has enough data to allow statistically significant conclusions. There is also a small validation set that was used to stop training of the neural network of the baseline classifier (typically 6-8 epochs). Previous experiments suggest that early stopping is not crucial for boosted networks [10], and they were trained for 15 epochs.

The solid lines in figure 3 show the frame and word errors using one unboosted neural network of varying capacity. Up to 3 runs of embedded training have been performed. One can clearly see that increasing the size of the hidden layer, and therefore the number of parameters, does increase the performance on the training data, but not on the test set. With more than about 500K parameters (corresponds to about 2400 hidden units) the network starts over-fitting and the test set frame and word error increase.

We first tried to combine 162-400-56 MLPs[3] using AdaBoost. This gives a much lower train and test set frame error (dashed lines in figure 3 left), but there is no improvement in the word error (3 right). For this reason we combined only nine MLPs (about 800K parameters in total). We hypothesize that either the 162-400-56 MLPs have not enough capacity to learn the resampled data sets that get more and more complicated with increasing number of combined classifiers, or the emphasizing algorithm of AdaBoost is partially focusing on noisy or wrongly-labeled frames. Using embedded training on top of the whole boosted ensemble allowed the adaption of the frame labels to the capacity of this architecture and we were able to achieve a word error of 5.4% on the test set using about 500K parameters (see figure 4 left).

When boosting 162-800-56 MLPs without embedded training the test set word error improves from 6.2% to 5.3% (dotted lines in figure 3). As shown in figure 4 right, embedded training of this architecture didn’t change the results further. Note that, in contrast to other applications of AdaBoost, we still do not reach zero frame training error, even after combining seven 162-800-56 MLPs. We did not try to combine more neural networks due to the long training time.

We also believe that it is not advantageous to combine too many networks in our case since AdaBoost would overfit by focusing on the noisy and incorrectly labeled examples. In fact, we are currently working on techniques to make AdaBoost give up on noisy or “hopelessly difficult” examples. These examples are mainly at word boundaries where the estimation of frame probabilities is very difficult. During training, these word boundaries are known and the corresponding examples could be excluded from the training if they have very small or even negative margins (indicating that they are too difficult to learn). We are also working

[3]the notation 162-h-56 describes a fully connected MLP with a 162 dimensional input layer (9 frames with 18 features each), h hidden units and 56 outputs corresponding to the phoneme classes.

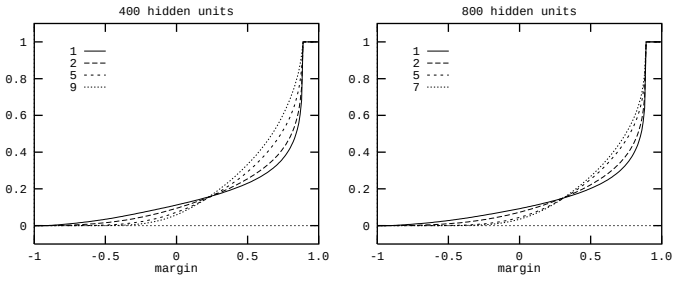

Figure 5: Example of margins distributions (1 run of embedded training)

Table 1: Comparison of different combining techniques (test set word errors).

system:	1 MLP (RASTA baseline)	1 MLP (modspec)	1 MLP (modspec + syllables)	4 MLP at frame level	2 MLP syllable level	6 MLP (RASTA + AdaBoost
number of params:	87K	99K	253K	372K	330K	1050K
word error:	6.4%	8.5%	10.6%	5.5%	5.4%	**5.3%**

on dealing with noisy examples in a similar way to support vector machines. Figure 5 shows some examples of the margins distributions, i.e. the fraction of examples whose margin is at most x as a function of $x \in [-1, 1]$. There seems to be some kind of a fix-point for which the percentage of examples with this margin does not change (about 0.2 when boosting 162-400-56 MLPs and about 0.35 when boosting 162-800-56 MLPs). Note in particular, that even after combining seven 162-800-56 MLPs there are still examples with a margin of -1.0. This means that the ensemble gets these examples completely wrong by predicting an a-posteriori probability of 1.0 for a wrong phoneme and of 0.0 for the correct phoneme (based on the labels of 1 run of embedded training). AdaBoost should certainly give up on these hopelessly difficult examples.

Table 1 compares the boosted MLPs with other combining technique developed at the International Computer Science Institute (see [11] for more details). The baseline recognizer using RASTA features achieves 6.4% word error, while using another feature representation, the modulation-spectrogram, gives 8.5% word error. These two feature representations result in quite different error patterns and combining them by multiplying the frame probabilities gives 5.5% word error. Another approach uses half-syllables instead of phonemes as output units of the neural network. This system alone has a 10.6% word error rate, which can be reduced to 5.4% by combining it at the syllable level with the RASTA base-line system. Boosting achieves the same low error rates as these systems using only one feature representation. In addition, there is hope that we could reduce the error rate further by dealing more explicitly with noise or by using also information from longer time spans.

An older boosting algorithm has been already used to improve a hybrid HMM/recurrent neural network speech recognizer [4]. The authors report a relative word error improvement of 10–20% on subsets of the Wall Street Journal corpus. This boosting algorithm doesn't use example emphasizing and resampling, but sequentially trains neural networks with examples on which the previous neural networks have an error rate of 50%.

4. CONCLUSION AND PERSPECTIVES

In conclusion, we have shown that the AdaBoost algorithm can be applied to a very difficult and noisy learning problem: the estimation of a-posteriori probabilities in a hybrid HMM/neural network continuous speech recognizer. This system performs at least as well as other combining techniques using several different feature representations or additional information from longer time spans.

Our experiments suggest that AdaBoost should not be run too long with noisy training examples. As outlined in the results section, we are currently working on different extensions of AdaBoost for noisy learning tasks. We are also applying this approach to bigger speech corpora, in particular "Switchboard." I would like to thank Nelson Morgan, Eric Fosler-Lussier, Brian E.D. Kingsbury and Su-Lin Wu for fruitful discussions and helpful comments.

5. REFERENCES

[1] Bourlard, H. and Morgan, N. (1994). *Connectionist Speech Recognition- A Hybrid Approach*. Kluwer Academic Press.

[2] Breiman, L. (1996). Bias, variance, and arcing classifiers. Technical Report 460, Statistics Department, University of California at Berkeley.

[3] Cole, R. A., Noel, M., Lander, T., and Durham, T. (1995). New telephone speech corpora at CSLU. In *Eurospeech*, volume 1, pages 821–824.

[4] Cook, G., Waterhouse, S., and Robinson, A. (1997). Ensemble methods for connectionist acoustic modelling. In *Eurospeech*, volume 3, pages 1959–1962, ESCA.

[5] Dietterich, T. G. (1998). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *submitted to Machine Learning*

[6] Freund, Y. and Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of Thirteenth International Conference*, pages 148–156.

[7] Freund, Y. and Schapire, R. E. (1997). A decision theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Science*, 55(1):119–139.

[8] Grove, A. J. and Schuurmans, D. (1998). Boosting in the limit: Maximizing the margin of learned ensembles. In *Proc. of the Fifteenth National Conference on Artificial Intelligence*. to appear.

[9] Schapire, R. E., Freund, Y., Bartlett, P., and Lee, W. S. (1997). Boosting the margin: A new explanation for the effectiveness of voting methods. In *Machine Learning: Proceedings of Fourteenth International Conference*, pages 322–330.

[10] Schwenk, H. and Bengio, Y. (1998). Training methods for adaptive boosting neural networks. *Advances in Neural Information Processing Systems*, The MIT Press. in press.

[11] Wu, S.-L., Kingsbury, B. E. D., Morgan, N., and Greenberg, S. (1998). Incorporating information from syllable-length time scales into automatic speech recognition. In *ICASSP*, in press.