A Boosting Approach for Confidence Scoring

Pedro J. Moreno, Beth Logan and Bhiksha Raj

Compaq Cambridge Research Laboratory,
One Cambridge Center,
Cambridge, MA 02142 USA

Abstract

In this paper we present the application of a boosting classification algorithm to confidence scoring. We derive feature vectors from speech recognition lattices and feed them into a boosting classifier. This classifier combines hundreds of very simple 'weak learners' and derives classification rules that can reduce the confidence error rate by up to 34%. We compare our results to those obtained using two other standard classification techniques, Support Vector Machines (SVMs) and Classification and Regression Trees (CART), and show significant improvements. Furthermore, the nature of the boosting algorithm allows us to combine the best single classifier and improve its performance.

We present experimental results on real world corpora derived from our *SpeechBot* Web index *http://www.speechbot.com* and from the HUB4 DARPA evaluation sets. We believe these results have wide applicability to audio indexing and to acoustic and language modeling adaptation where word confidence scores can be used in iterative adaptation schemes.

1. Introduction

Speech recognition technology has advanced to the stage where real-world applications are feasible. However, due to the current imperfect nature of speech recognition, confidence scoring has emerged as an important component of current systems. Confidence scoring attempts to assign 'trust' to the hypotheses produced by speech recognition systems.

We are interested in audio indexing systems for the Web. Confidence scores can be very useful for such systems where an enormous amount of data is indexed and the ground truth is not known. For example, our speech indexing system *SpeechBot* [1] indexes close to 9000 hours of untranscribed audio content. A good confidence scorer could enable us to make use of such data, either for acoustic and language model adaptation or even for retraining [2]. We could also use confidence scores to improve our indexing function.

The literature contains many examples of techniques for word confidence scoring. Typical approaches form a feature vector by concatenating or otherwise combining one or more basic features correlated with word confidence, including basic features of adjacent words. One of a variety of classifiers is then applied to this vector to determine confidence for the word. Features based on the acoustic model (*e.g.* see [3]), the language model (*e.g.* [4]), the decoding process (*e.g.* [5, 6, 7, 8, 9]) and word semantics [10, 11]) have been proposed. Classifiers investigated include simple thresholding [7], linear discriminant analysis followed by a linear thresholds [3, 11], Bayes classifiers [8], neural networks [5, 3, 6, 12], generalized linear models [9, 13] and decision trees [6, 11].

In this paper we explore the use of boosting techniques to classify confidence feature vectors. Boosting combines hundreds or even thousands of very simple classifiers (called 'weak learners' in the Machine Learning literature) by a weighted sum. Each classifier focuses its attention on those vectors on which the previous classifier fails.

The use of boosting classifiers with the choice of weak learners proposed in [14] offers us the unique advantage of being less sensitive to spurious features. That is, components of the confidence feature vector that do not add any advantage are ignored at the expense of more promising features. Additionally, we are able to analyze the relative importance of each feature in a principled way. A simple inspection of the weak learners highlights those features that contribute most to classification.

2. Confidence Features

We use a fairly standard set of confidence features augmented with one novel feature to form a feature vector for each hypothesized word. Since our boosting classifier will ignore components that supply spurious information, there is no harm in including as many features as possible (other than wasted processing time). Our basic set of features is listed in Table 1.

Component	Basic Feature
0	word graph probability *e.g.* [7]
1	hypothesis density at word beginning
2	hypothesis density at word end
3	average hypothesis density over the word
4	hypothesis density at preceding frame
5	hypothesis density at following frame
6	acoustic score
7	unigram score
8	word length in frames
9	word length in phones
10-12	3D point representing the first phone of the word (explained in the text)

Table 1: Core feature set used to construct the feature vector for each hypothesized word. This vector is augmented by left and right context as described in the text.

In addition to this basic set, we include context information for each word. We form the final confidence feature vector for each hypothesized word as the concatenation of the feature set in Table 1 for that word, and the corresponding sets for the most likely (in the Viterbi search sense) preceding and following words. Our final confidence feature vector thus has dimension 39.

Our one novel feature is a 3D representation of the first

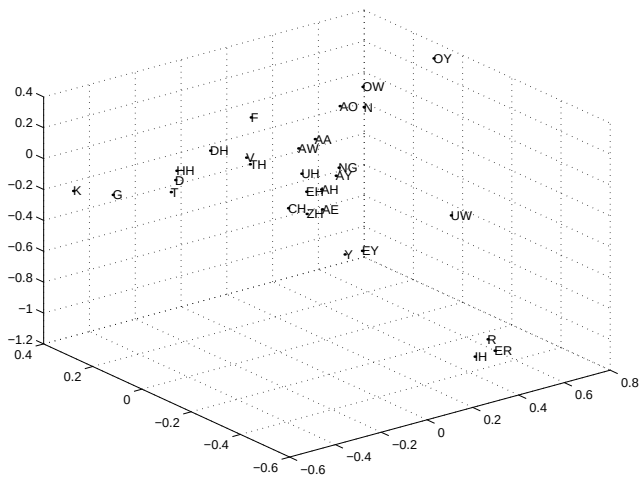

Figure 1: 3D Euclidean representation of TIMIT phones derived using MDS on their confusion matrix. For clarity, only points close to the origin are shown.

phone of each word. Our motivation is that we wish to include more information about the intrinsic confusability of words in confidence scoring metrics. However, since there is no simple low-dimensional monothetic representation of word confusability, we approximate it by the confusability of the first phone of the word. This is reasonable since an error at the beginning of the word will impact the whole word. Indeed, many words begin with easily confusible consonants.

We represent the confusability of the first phone in the word by transforming the phone label to a real three-dimensional point using Multi-dimensional scaling (MDS). This transformation from a label to the real space allows us to treat this feature numerically, similar to all other features. MDS (*e.g.* [15]) is a standard technique which transforms a series of objects, about which only relative distance information is available, to a series of N-dimensional points. The mapping attempts to preserve the relative distances between objects such that objects which are known to be 'close' to each other are 'close' in the N dimensional space. To transform phone labels using MDS, we use a phone confusion matrix as a measure of the relative distance among them. Figure 1 shows our 3D representation of TIMIT phones derived using MDS on their confusion matrix. We see that linguistic categories are well preserved in this Euclidean space. We use this mapping to obtain a 3D point for the first phone of each word.

3. Boosting Classifier

Boosting is a novel approach to classification which has lately received much attention due to its simplicity, elegance, power and ease of implementation. The basic ideas and algorithms were introduced by Schapire [16] and Freund [17].

Boosting applies a classification procedure iteratively to a set of weighted data vectors. At first each vector is assigned an equal weight (or a weight depending on its prior probability). On each iteration, a classifier is learnt and the vectors that are classified incorrectly have their weights increased while those that are correctly classified have their weights decreased. The intuition is that vectors which are difficult to classify receive more attention on subsequent iterations.

The classifier learnt at each iteration is called a 'weak' classifier. It is called weak because it is not expected to classify the training data very well, only better than 50%. Typically a very simple weak classifier is used. The final classifier, the so-called 'strong' classifier, is formed as a weighted sum of the weak classifiers learnt at each step. Table 2 gives a algorithmic description of the boosting classification procedure.

- Begin with N training vectors x_i and their associated labels y_i where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:
  1. Normalize the weights,
     $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
     so that w_t is a probability distribution and adds up to 1.0.
  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_t.
  4. Update the weights:
     $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$
     where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- The final strong classifier is:
  $$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$
  where $\alpha_t = \log \frac{1}{\beta_t}$

Table 2: The boosting algorithm for learning a classifier. T weak classifiers are constructed. The final strong classifier is a weighted linear combination of the T weak classifiers where the weights are inversely proportional to the training errors.

The formal guarantees provided by boosting classification theory are quite strong. Freund and Schapire prove that the training error of the strong classifier approaches zero exponentially in the number of iterations.

3.1. Choice of Weak Learner

The boosting algorithm does not impose any restriction on the nature of the weak learner. Any classifier that does a better job than pure chance is acceptable. In this paper we have experimented with a rather simple weak learner. We use a variant of AdaBoost [17] proposed by Tieu and Viola [14] in which the weak learner is a simple threshold that depends on a single component of the feature vector. This weak learner examines the feature vector and finds the component and threshold that best separates the two classes. Therefore each weak learner $h_j(x)$ is identified by a feature component f_j, a threshold θ_j, and a direction d_j indicating the direction of the inequality sign.

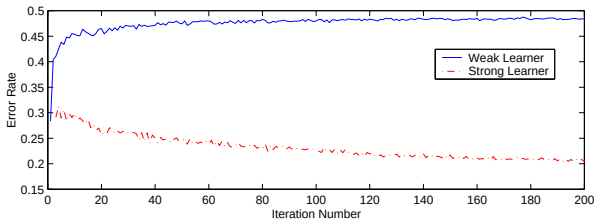

Figure 2: Strong and weak error rates as a function of the number of iterations in the boosting algorithm. The dataset was a subset of the HUB4 confidence set.

$$h_j(x) = \begin{cases} 1 & \text{if } d_j f_j(x) < d_j \theta_j \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

In practice no single feature component can perform the classification task with low error. Typically the first weak learner has an error rate of about 0.3 and the final learners closer to 0.5. Figure 2 shows weak and strong learner error rates for the HUB496 data set as a function of the number of iterations. We see that the strong error rate converges to 0.2 while the weak learner converges to around 0.5.

4. Alternative Classifiers

In addition to boosting, we also experiment with alternative classifiers on our feature set. We use standard implementations of Support Vector Machines (SVM) [18] and Classification and Regression trees (CART) [19].

5. Experimental Results

We test our algorithm on confidence features obtained from two data sets. The first set is the 1996 HUB4 test set [20], a total of about 3 hours of speech. The second set is sampled from around 9 hours of transcribed Web Broadcast News from our internal *SpeechBot* test set [1].

To obtain lattices from which our confidence features are extracted, we run a standard HMM-based decoder built on the HUB496 and HUB497 training sets. For the *SpeechBot* test set, the training data is Real-Audio encoded and decoded to account for the streamed nature of the test set. The decoder for the HUB4 data uses 16 Gaussian mixture components per state. For the *Speechbot* data, 8 mixture components are used. The word error rates for the data sets are 32.9% and 55.0% respectively.

Using the decoded word lattices, We construct confidence feature vectors as described in Section 2 for each word in the top hypothesis. Each feature is labeled with '1' or '0', reflecting whether or not the word is correct. Table 3 gives further details of the feature sets, including the baseline error or prior probability of Class 0. Notice that the error rates for the confidence vectors are not the same as the recognizer error rates. This is because deleted words, which count as errors for word error rate scores, do not appear in confidence feature sets (since there is no word to obtain features for).

Data Set	Nr. Vectors	Baseline Error
HUB496	43k	29.0%
SpeechBot	43k	41.7%

Table 3: Details of the HUB4 and *SpeechBot* confidence feature sets

For all the experiments reported in this paper we perform cross validation. The data sets were randomized and split into 10 different sets. Training was performed on 9 sets and testing on the remaining set. This experiment was repeated 10 times by testing on all 10 sets. Our error rates are therefore averages over all 10 sets. This experimental method provides more accurate and valid results.

We also report the confidence error rates for both classes. Any classifier can be tuned to minimize global error rate or to minimize false positives or false negatives. In this paper we tune our classifiers to operate close to the equal error rate point where both false positives and false negatives are similar. Otherwise, our results will be biased by the prior probabilities of each class.

5.1. HUB496 results

Table 4 shows the results of tests on the HUB4 dataset. We show error rates for boosting systems with up to 200 weak learners. We did not observe significant improvements beyond this number. The results show that we can reduce the error rate to 25.5%, an improvement of 12.1% relative to the baseline of 29.0%.

Number of weak learners	Class 1 Error Rate	Class 0 Error Rate	Total Error Rate
1	30.4%	28.9%	28.1%
50	27.6%	27.4%	26.1%
100	27.4%	27.1%	25.9%
200	28.0%	26.3%	25.5%

Table 4: Error rates for the HUB4 96 data set and their relationship to the number of weak learners.

On this set, the CART classifier produces an error rate of 28.1%, almost no improvement over the baseline. The SVM classifier yields an error rate of 31.2%, again no improvement.

5.2. *SpeechBot* results

Table 5 presents results for the *Speechbot* dataset. Again, we show error rates for up to 200 weak learners. A substantial improvement over the baseline result is observed. We improve the error rate from 41.7% to 27.6%, a relative improvement of 33.8%. On this dataset, the CART classifier produces an error rate of 28.4% and the SVM classifier an error rate of 32.6%.

Number of weak learners	Class 1 Error Rate	Class 0 Error Rate	Total Error Rate
1	28.6%	34.4%	31.9%
50	26.1%	29.7%	28.2%
100	25.5%	29.2%	27.7%
200	25.7%	28.9%	27.6%

Table 5: Error rates for the *SpeechBot* data set and their relationship to the number of weak learners.

6. Discussion

We observe that our boosting classifier outperforms both SVMs and CART classifiers. Even on the HUB96 dataset where the CART and SVM classifiers failed to yield any improvement boosting gave a 12.1% relative improvement.

Because our choice of weak learner is a dimension specific

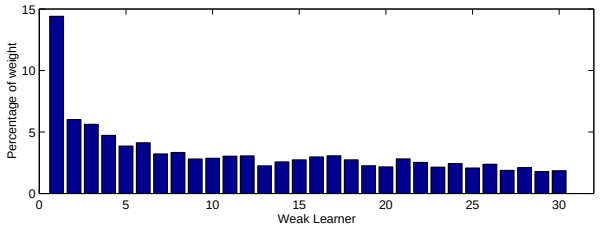

Figure 3: Weights applied to each of the weak learners. The first five learners contribute close to 25% to the decision.

classifier, it is interesting to examine when each feature component is chosen by the boosting iterative procedure. Intuitively, confidence vector features that are chosen early are more *informative* than those chosen later on. Using this simple analysis we observe that features 3, 0, 3, 1, 7 and 11 are the first six features chosen by the strong classifier. These features correspond to the average hypothesis density over the word, the word graph probability, the hypothesis density at the word beginning, the unigram score and the middle component of our 3D representation of the first phone in the word. This order of feature choice is relatively consistent across experiments and datasets. Interestingly, our 3D phone representation is more informative than many of the other lattice-derived features.

Figure 3 displays typical weights applied to the first thirty weak learners. We observe that the features for context words (from components 14 to 39) that provide another 26 components to our 39 dimensional vector only appear after position 10 or so. In fact, after learning 100 weak learners only 26 out of the possible 39 are chosen. This is due to the fact that our boosting implementation plays a dual role of learning classifiers and picking those features that are more promising in classifying the data correctly. This characteristic of our boosting implementation could be used as a preprocessor to extract informative features from an arbitrarily large set to aid dimensionality reduction in other tasks.

7. Conclusion

In this paper we have explored the use of boosting techniques for confidence scoring. We have compared them with two other classification schemes, CART and SVMs, and consistently outperformed them. Our choice of boosting algorithm offers several advantages. It is simple to implement, fast in its learning time, and very flexible in the choice of weak learner. In this paper we have used a very simple learner that picks individual features and classifies them with a threshold and a flag indicating the direction of the inequality sign. Remarkably, such a simple classifier is able to provide up to a 34% improvement in performance on the *SpeechBot* dataset. More sophisticated weak learners such as CART should be able to improve this performance at the cost of longer training time.

In the future we will explore how confidence scores can be used to improve our public audio indexing system, both to refine the retrieval function as well as for language and acoustic model adaptation. Confidence scores will allow us to effectively mine more than 9000 hours of unlabeled audio currently indexed by the *SpeechBot* system.

8. Acknowledgments

We thank Michael Jones and Paul Viola for their help and support in this research. We also thank Jean-Manuel Van Thong for providing us with phonetic confusion matrices.

9. References

[1] B. Logan, P. Moreno, J.-M. Van Thong, and E. Whittaker, "An experimental study of an audio indexing system for the Web," in *Proc. ICSLP*, 1996.

[2] T. Kemp and A. Waibel, "Unsupervised training of a speech recognizer: recent experiments," in *Proc. EUROSPEECH*, 1999.

[3] T. Schaaf and T. Kemp, "Confidence measure for spontaneous speech recognition," in *Proc. ICASSP*, 1997.

[4] C. Uhrik and W. Ward, "Confidence metrics based on n-gram language model backoff behaviors," in *Proc. EUROSPEECH*, 1997.

[5] M. Weintraub, F. Beaufays, Z. Rivlin, Y. Konig, and A. Stolcke, "Neural - network based measures of confidence for word recognition," in *Proc. ICASSP*, 1997.

[6] T. Kemp and T. Schaaf, "Estimating confidences using word lattices," in *Proc. EUROSPEECH*, 1997.

[7] F. Wessel, K. Macherey, and R. Schlueter, "Using word probabilities as confidence measures," in *Proc. ICASSP*, 1998.

[8] S. Cox and R. Rose, "Confidence measures for the switchboard database," in *Proc. ICASSP*, 1996.

[9] L. Gillick, Y. Ito, and J. Young, "A probabilistic approach to confidence estimation and evaluation," in *Proc. ICASSP*, 1997.

[10] S. Cox and S. Dasmahapatra, "A semantically-based confidence measure for speech recognition," in *Proc. ICSLP*, 2000.

[11] C. Pai, P. Schmid, and J. Glass, "Confidence scoring for speech understanding systems," in *Proc. ICSLP*, 2000.

[12] A. Wendemuth, G. Rose, and J. G. A. Dolfing, "Advances in confidence measures for large vocabulary," in *Proc. ICSLP*, 2000.

[13] M.-H. Siu, H. Gish, and F. Richardson, "Improved estimation, evaluation and applications of confidence measures for speech recognition," in *Proc. EUROSPEECH*, 1997.

[14] K. Tieu and P. Viola, "Boosting image retrieval," in *International Conference on Computer Vision*, 2000.

[15] F. W. Young and R. M. Hamer, *Multidimensional Scaling: History, Theory and Applications*, Erlbaum, 1987.

[16] Robert E. Schapire, "The strength of weak learnability," *Machine Learning*, vol. 5, no. 2, pp. 197–227, 1990.

[17] Yoav Freund and Robert E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *Computational Learning Theory: Eurocolt '95*. 1995, pp. 23–37, Springer-Verlag.

[18] Christopher Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, 1998.

[19] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J.Stone, *Classification and Regression Trees*, Chapman & Hall, New York, 1984.

[20] Linguistic Data Consortium, *1996 English Broadcast News Dev and Eval*, http://www.ldc.upenn.edu, 1996.