Is AdaBoost Competitive for Phoneme Classification?

Gábor Gosztolya
MTA-SZTE Research Group on Artificial Intelligence
Szeged, Hungary
Email: ggabor@inf.u-szeged.hu

***Abstract*—In the phoneme classification task of speech recognition, usually Gaussian Mixture Models and Artificial Neural Networks are used. For other machine learning tasks, however, several other classification algorithms are also applied. One of them is AdaBoost.MH, reported to have high accuracy, which we tested for phoneme recognition on the well-known TIMIT dataset. We found that it can achieve an accuracy comparable to standard ANNs in this task, but lags behind recently-proposed Deep Neural Networks. Based on our experimental results, we list a number of possible reasons why this might be so.**

I. Introduction

In machine learning, a wide variety of classification methods have been introduced, ranging from the simpler, earlier algorithms of decision trees [1] and Gaussian Mixture Models (GMMs) [2] through Artificial Neural Networks (ANN) [3] to more recent methods like Support Vector Machines (SVM) [4] and AdaBoost.MH [5]. In the phoneme classification task of speech recognition, however, GMMs have been the dominant method for decades, which was only recently overtaken by ANN.

In this task the utterance to be recognized is first divided into small, equal-sized parts (*frames*) of typically 10 milliseconds long, which are then classified independently as one of the possible phonemes in the given language. However, evaluation is usually not done on this level, but in the next step a search is performed (typically via a Hidden Markov Model (HMM)), based on the frame-level likelihood values provided by the classification method. This search gives the most probable path over the phonemes through time, from which a phoneme sequence for the whole utterance is constructed. So the performance of the frame-level phoneme classification method is rated based on the accuracy of this utterance-level phoneme sequence.

Therefore it is clear that phoneme recognition is not a pure classification task, but the goal is essentially that of *probability estimation*: in the search step those methods that provide poor probability estimates have a disadvantage. On the other hand, misidentifying individual frames is usually not a mistake at all, as long as the given phoneme finally appears in the utterance-level result (probably thanks to the high appropriate likelihood scores in the neighbouring frames).

This publication is supported by the European Union and co-funded by the European Social Fund. Project title: Telemedicine-oriented research activities in the fields of mathematics, informatics and medical sciences. Project number: TÁMOP-4.2.2.A-11/1/KONV-2012-0073

Furthermore, from a machine learning perspective, speech recognition databases usually have somewhat specific characteristics. The number of features is quite low, as well as the number of classes: the very popular MFCC feature set consists of only 39 features for a frame, and the number of classes is at most two hundred (at least for context-independent states). Another special characteristic is that the classes are usually not independent of each other: it is common to define different states of the same phoneme as separate classes (usually the beginning, middle and last parts (*tri-state setup*)), which are naturally closely related, but to the machine learning method is completely hidden. Another special property of the phoneme recognition task is the number of examples: training on several hours of audio recordings (nowadays hundreds of hours is usual) leads to literally millions of examples.

This set-up is quite uncommon in machine learning, and most algorithms simply cannot handle such a huge amount of data. It is more usual to have fewer training examples, which are described by a large feature vector, and which belong to one of the few classes. Even speech technology has such tasks like emotion detection [6], [7] and speaker recognition [8].

As ANNs can be readily trained on huge amounts of data by processing one chunk of data at a time, and are able to produce reliable posterior scores, it is not surprising that they have become the dominant machine learning method used for phoneme classification. (We regard Deep Neural Networks (DNN) [9] as a variety of ANNs.) Although other algorithms like SVM and AdaBoost value each individual training example more, in this given problem it is not so important due to the huge number of training examples available. These methods, due to their greater complexity, have other drawbacks as well like higher training and evaluation times [10], high memory requirements owing to the need to train on all the examples at the same time, and necessity of tuning more meta-parameters to achieve optimal performance.

In spite of the above points, AdaBoost was applied earlier in speech technology. It is often applied when the number of examples is small (typically one for each utterance), as in emotion detection [11] and speaker verification [12]. Via the increase in computational capacity and memory requirements have become less critical issues, tasks that have several training examples like that of voice activity detection [13], [14], feature selection [15] (also for laughter detection [16]) and hypothesis rescoring [17] were studied recently.

978-1-4799-5338-7/14/$31.00 ©2014 IEEE

Some of these examples were frame-based (e.g. voice activity detection [14]). Recently, we successfully applied AdaBoost.MH to frame-level emotion detection [7], but with specific features and only three classes. However, it is quite rare to try to apply AdaBoost directly in the phoneme classification task. Klautau [18]) trained one-against-all-type classifiers for phoneme classification, but did not attempt to construct an utterance-level phoneme-sequence of the results. Dimitrakakis [19], Du [20] and Tang [21] boosted entire HMMs, and Saon [22] also provided a training algorithm for the whole speech recognizer framework. In most of these studies, they boosted GMMs as weak learners, adding one mixture in each iteration, which can be reasoned by the need for good likelihood estimations. Yet, GMMs are already stable on their own, which perhaps explains why using them as base learners is practically unknown outside speech recognition.

In this work, however, we seek to apply AdaBoost as it is typically used as a classification technique, and apply it directly in the standard phoneme classification task. We will also use a standard feature set and the the well-known TIMIT database (frequently used as a benchmark). For comparison, we also trained some other standard methods.

II. AdaBoost.MH

Next, we briefly describe the AdaBoost.MH algorithm. First, let $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ be the *observation matrix*, where $x_i^{(j)}$ are the elements of the d-dimensional observation vectors $\mathbf{x}_i \in \mathbb{R}^d$. We are also given a *label matrix* $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)$ of dimension $n \times K$ where $\mathbf{y}_i \in \{+1, -1\}^K$. In a *multi-class* classification one and only one of the elements of $\mathbf{y}_i$ is $+1$; we will denote the index of the correct class by $\ell(\mathbf{x}_i)$.

A. AdaBoost.MH

The goal of the AdaBoost.MH algorithm [5] is to return a classifier $\mathbf{f} : \mathcal{X} \to \mathbb{R}^K$ with a small *Hamming loss*

$$R_{\mathrm{H}}\big(\mathbf{f}^{(T)}, \mathbf{W}^{(1)}\big) = \sum_{i=1}^{n} \sum_{\ell=1}^{K} w_{i,\ell}^{(1)} \mathbb{I}\{\mathrm{sign}\big(f_\ell^{(T)}(\mathbf{x}_i)\big) \neq y_{i,\ell}\}$$

by minimizing its upper bound (the exponential margin loss)

$$R_{\mathrm{e}}\big(\mathbf{f}^{(T)}, \mathbf{W}^{(1)}\big) = \sum_{i=1}^{n} \sum_{\ell=1}^{K} w_{i,\ell}^{(1)} \exp\big(-f_\ell^{(T)}(\mathbf{x}_i) y_{i,\ell}\big), \quad (1)$$

where $f_\ell(\mathbf{x}_i)$ is the ℓth element of $\mathbf{f}(\mathbf{x}_i)$, and the indicator function $\mathbb{I}\{A\}$ is 1 if A is true and 0 otherwise. The user-defined weights $\mathbf{W}^{(1)} = \big[w_{i,\ell}^{(1)}\big]$ are usually set either uniformly to $w_{i,\ell}^{(1)} = 1/(nK)$, or, in multi-class classification, to

$$w_{i,\ell}^{(1)} = \begin{cases} \frac{1}{2n} & \text{if } \ell = \ell(\mathbf{x}_i) \text{ (i.e., if } y_{i,\ell} = 1), \\ \frac{1}{2n(K-1)} & \text{otherwise (i.e., if } y_{i,\ell} = -1) \end{cases} \quad (2)$$

to create K well-balanced one-against-all classification problems. AdaBoost.MH builds the final classifier $\mathbf{f}$ as a weighted sum of *base classifiers* $\mathbf{h}^{(t)} : \mathcal{X} \to \mathbb{R}^K$ returned by a *base learner* algorithm $\textsc{Base}(\mathbf{X}, \mathbf{Y}, \mathbf{W}^{(t)})$ for each iteration t. The base learner should seek to minimize the *base objective*

$$E\big(\mathbf{h}, \mathbf{W}^{(t)}\big) = \sum_{i=1}^{n} \sum_{\ell=1}^{K} w_{i,\ell}^{(t)} \exp\big(-h_\ell(\mathbf{x}_i) y_{i,\ell}\big). \quad (3)$$

In our tests we used *discrete* AdaBoost.MH, in which the vector-valued base classifier $\mathbf{h}(\mathbf{x})$ is represented as

$$\mathbf{h}(\mathbf{x}) = \alpha \mathbf{v} \varphi(\mathbf{x}), \quad (4)$$

where $\alpha \in \mathbb{R}^+$ is the *base coefficient*, $\mathbf{v} \in \{+1, -1\}^K$ is the *vote vector*, and $\varphi(\mathbf{x}) : \mathbb{R}^d \to \{+1, -1\}$ is a *scalar* base classifier. The simplest scalar base learner used in practice is the *decision stump*, a two-leaf decision tree having the form

$$\varphi_{j,b}(\mathbf{x}) = \begin{cases} 1 & \text{if } x^{(j)} \geq b, \\ -1 & \text{otherwise,} \end{cases} \quad (5)$$

where j is the index of a feature and b is the decision threshold. Although boosting decision stumps often yields satisfactory results, state-of-the-art performance is usually achieved by using decision trees as base learners, parametrized by their number of leaves.

B. Accelerated AdaBoost.MH

Running a full search for each boosting iteration step of AdaBoost.MH is prohibitively expensive (especially for huge datasets), so we turned to an accelerated version using a multi-armed bandit (MAB) setup [23]. The main idea is to partition the base classifier space $\mathcal{H}$ into (not necessarily disjunct) subsets $\mathcal{G} = \big\{\mathcal{H}_1, \ldots, \mathcal{H}_M\big\}$ and use MABs to learn the usefulness of the subsets. Each arm represents a subset; so, for each iteration, the bandit algorithm selects a subset. The base learner then finds the best base classifier *in the subset* (instead of searching in the *whole* $\mathcal{H}$ *space*), and returns a reward based on the output of this optimal base learner. From the reward values, the MAB algorithm learns the usefulness of the subsets. We applied the bandit algorithm EXP3.P [24].

For simple decision stumps, the natural partitioning is to assign a subset to each feature: $\mathcal{H}_j = \big\{\varphi_{j,b}(\mathbf{x}) : b \in \mathbb{R}\big\}$. But the idea behind AdaBoost.MH.BA cannot be applied directly to decision trees, since it is hard to find a suitable partitioning of the space of trees, due to the large size of this space. So we followed the setup of [25] where trees are treated as sequences of decisions over the smaller partitioning used for stumps.

III. Experiments

In our tests we used the TIMIT dataset. The standard training set was split into a larger training and a small development set: 50,693 frames belonged to the latter taken from 176 utterances, leaving 1,074,130 frames for training. To aid the phoneme recognizer, the training frame labels were re-aligned by an ANN to more precisely position phoneme state boundaries. All the 61 phonemes were used in a tri-state setup, resulting in 183 classes overall. To improve the performance, we used another task-specific modification: we also included the feature vectors of 8 neighbouring frames on both sides in

the training vector (so it consisted of 663 features accumulated from a 170ms-wide long window), while keeping the original class label. We used the MFCC+Δ+$\Delta\Delta$ feature set, and used the HTK toolkit [26] for preprocessing.

A. Classification Methods

We tested a variety of machine learning methods for comparison. We utilized the LibSVM [27] library for SVM and the multiboost tool [28] for AdaBoost. For ANNs, we used our custom implementation, which can also be trained as a Deep Neural Network using rectifier activation function [29], [30] in the hidden layers, which can be regarded as a state-of-the-art technique [30]. We trained a neural network with one hidden layer using sigmoids, and a DNN having 5 1024-sized hidden layers with rectifier functions for comparison. For AdaBoost.MH, we used decision stumps and decision trees (with 8 leaves) as base learners, and employed the bandit-based approach.

Including the feature vectors of the neighbouring frames is a task-specific modification of the general machine learning task. Our custom ANN implementation handled this easily by training on a sliding window over the input feature vectors, but as we used off-the-shelf libraries for SVM and AdaBoost, we had to insert these extra feature vectors into the training data. As this increased the memory requirements to 17 times the original case, we had to sample the training data by keeping only every third element (more for rarer classes), resulting in $375,673$ actual training examples. For comparison, we trained ANNs on this subset as well.

To avoid overfitting, we employed the straightforward technique of early stopping (meaning we just stop training after a certain number of iterations). We trained AdaBoost for 100k iterations, evaluated at every 10k, and chose the one for testing on the core test set when the phoneme-level accuracy score was the highest on the development set.

B. Frame-level evaluation

We measured the frame-level performance of the machine learning methods in terms of accuracy, i.e. the percentage of correctly determined class-labels over the number of examples. We calculated this score for the development set only, where we had re-aligned frame-level class labels. To determine how much of the error came from intra-phoneme state mistakes, and how much might be due to confusing similar phonemes, we also calculated the accuracy scores without tri-states (61 classes), and for the reduced 39-long phoneme set with and without tri-states (117 and 39 classes, respectively), along with the original 183-class values. To calculate these scores, we did not train any new models, but got these values via simple class fusing. As fusing classes clearly reduces error, we calculated and compared the relevant Relative Error Reduction (RER) scores, and estimated the amount of confusing similar classes and/or states of the same phoneme based on these values.

C. Phoneme-level evaluation

Computing frame-level accuracy values is not the most adequate way of measuring the performance of a phoneme recognition method for several reasons. First, examples are treated independently, which does not really fit in with frame-level classification (although including the feature vectors of neighbouring frames may be viewed as a solution for this). The second, more important reason is that it is virtually impossible to determine the exact border between successive phonemes within a 10ms accuracy, and it is even harder in a tri-state setup. Therefore an utterance-level phoneme sequence is computed from the posterior probability values using a HMM, which is then rated via edit distance-based accuracy. (We used the HTK toolkit [26] for these steps.) As we just focused on the acoustic model, we did not employ any language model.

If we have no posterior probability values (e.g. when using sparse representation [31]), or they are not too precise (which is the case for AdaBoost.MH, requiring post-calibration [32]), we can also construct a phoneme sequence from the frame-level class labels via a simple dynamic search method. Of course, due to information loss, the resulting accuracy score is expected to be somewhat lower. We also constructed posterior estimates simply by assuming that the output scores of AdaBoost were log-likelihoods; hence we only had to normalize them frame-wise by subtracting a constant from them to get their exponential function to add up to one. As using a HMM instead of the dynamic search method reduces the recognition error, and the amount of this reduction depends on the accurateness of posterior probability values, we again calculated RER scores to judge the posterior quality of the classification methods.

IV. Results

A. Frame-level results

Table I lists the frame-level accuracy scores we obtained. It can be seen that database sampling reduced the accuracy of neural networks by about 1-2%, depending on the number of classes; we can assume that restricting the dataset to its one-third reduced the accuracy scores of SVM and the AdaBoost.MH trainings by this amount as well. The scores for AdaBoost.MH using stumps as base learners were quite low, but with decision trees the results are comparable with those of ANNs, although lower than those of DNNs. This was expected, though, DNNs being the current state-of-the-art machine learning method for phoneme classification. SVM training achieved slightly better scores than AdaBoost.MH.

Fusing the three states (reducing 183 classes to 61) led to relative error reduction (RER) scores of 16.20% to 18.66% for the neural networks; the score of 18.02% for AdaBoost.MH with tree base learners lay in this range, showing that identifying the three states was no harder than that for neural networks. But fusing phonemes (reducing the 183 classes to 117) led to a RER score of 11.30% to 12.76% for ANNs, whereas for AdaBoost it was 9.51% and 10.46%, suggesting that even though boosting was not sensitive to inter-state label mismatch, it could not tolerate similar phonemes as well as neural networks could.

TABLE I
FRAME-LEVEL RECOGNITION ACCURACY SCORES FOR THE MACHINE LEARNING METHODS FOR 39 AND 61 PHONEMES, WITH AND WITHOUT TRI-STATES, ON THE DEVELOPMENT SET.

DB size	Method	No. of classes			
		39	117	61	183
1/1	ANN	77.74%	69.39%	71.89%	65.44%
	DNN	80.12%	73.33%	74.55%	69.43%
1/3	ANN	76.53%	68.21%	70.60%	64.16%
	DNN	78.25%	71.23%	72.54%	67.23%
	SVM	76.49%	68.56%	70.70%	64.48%
	Stumps	67.93%	58.04%	61.25%	53.63%
	Trees	76.01%	67.80%	70.52%	64.04%

TABLE II
UTTERANCE-LEVEL PHONEME ACCURACY SCORES FOR THE MACHINE LEARNING METHODS APPLIED, FOR THE DEVELOPMENT AND TEST SETS, USING A DYNAMIC SEARCH METHOD AND A HMM.

DB size	Method	Development Set		Test Set	
		Dynamic	HMM	Dynamic	HMM
1/1	ANN	75.23%	77.44%	71.99%	74.76%
	DNN	76.80%	79.37%	73.90%	76.27%
1/3	ANN	73.67%	76.31%	70.99%	73.60%
	DNN	74.77%	77.39%	72.34%	75.19%
	SVM	72.79%	74.85%	70.53%	72.73%
	Stumps	65.91%	68.48%	63.68%	66.94%
	Trees	74.09%	75.29%	70.76%	72.85%

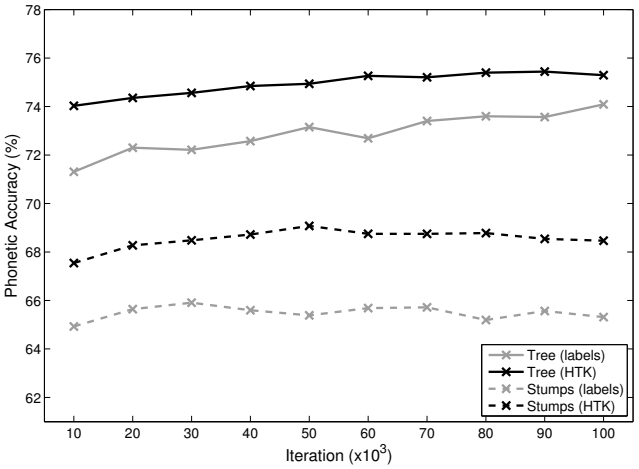

Fig. 1. Phonetic-level accuracy as a function of iterations for AdaBoost.MH with both base learners (stumps and trees).

B. Phoneme-level results

Phonetic accuracy scores (see Table II), regardless of the search method applied, mirror the frame-level tendencies. Values got by training on the whole database were somewhat better than ones got by training on the sampled subset. AdaBoost with stumps again proved to be the worst method, while SVM and AdaBoost with trees as base learners again practically matched the performance of standard-architecture ANNs. SVM slightly outperformed AdaBoost in terms of frame-level accuracy, but its phonetic-level results are clearly worse than those of AdaBoost using trees as base learners.

The quality of the probability values may be judged from the extent of the difference in scores got via dynamic and HMM search methods. This difference is greatest with the neural networks: switching to the latter search method brought an improvement of between 9.00% and 10.38%, whereas for AdaBoost this value lay between 7.15% and 8.98%, and for SVM we got an RER score of 7.47%. This is in accordance with the literature, that although AdaBoost can produce high accuracy scores, its posterior scores are quite imprecise. It may be worth applying a more sophisticated post-calibration method (e.g. [32]), but this is beyond the scope of our study.

C. Overall Performance

Overall, we can say that the performance of AdaBoost, using stumps as base learners, was quite low for a complex task like frame-level phoneme classification. However, using decision trees as base learners the results were practically the same as those for standard ANNs, making it precise enough for actual use. The 72.85% accuracy score on the TIMIT core test set may not seem too high at first glance, but it was achieved without using any language model. Another point was that, for technical reasons (namely using an off-the-shelf implementation) we had to reduce the size of the training set, which may be the reason for an additional 1-2% loss in accuracy. If we recall that we used a very simple posterior construction method, what we get is a fair acoustic performance. Note however, that the execution times tend to be high, especially for training, but a feasible solution is to parallelize the algorithm [33].

Figure 1 shows the phonetic-level accuracy scores of both base learners. Stumps attained their optimum around 30k iterations (50k with HMM), whereas with decision trees the accuracy improved even after 100k iterations; this also suggests that this task is just too complex for decision stumps.

But if AdaBoost is well-known for its high precision, why did it rank slightly below a standard ANN in this particular task, even in terms of frame-level accuracy? Besides the factors already mentioned like the huge dataset size, there may be other factors as well; one of them, which is usually ignored, is that of training labeling. As phoneme boundaries cannot be positioned objectively with one-frame precision, frame labels are not obvious. ANNs, probably due to their good posterior estimation capabilities, are robust to this phenomenon; other methods, however, may be less tolerant to imprecise labeling, making labeling quality quite important. Vinyals achieved a 51.1% frame-level accuracy with sparse representations [31], from which they got a 75.1% phonetic accuracy with their simple dynamic method. Although these scores, measured on the test set, cannot be directly compared to our 64.04% frame-level and 74.09% phonetic accuracy scores, the contrast is quite apparent, and in our opinion it is due to their carefully-constructed frame labels.

Another factor may be the way phoneme classification is employed in a speech recognition framework. Tri-state modeling is a technique developed originally for the HMM/GMM architecture, and it is well suited to the generative nature of GMMs. But if we shift to a discriminative classification

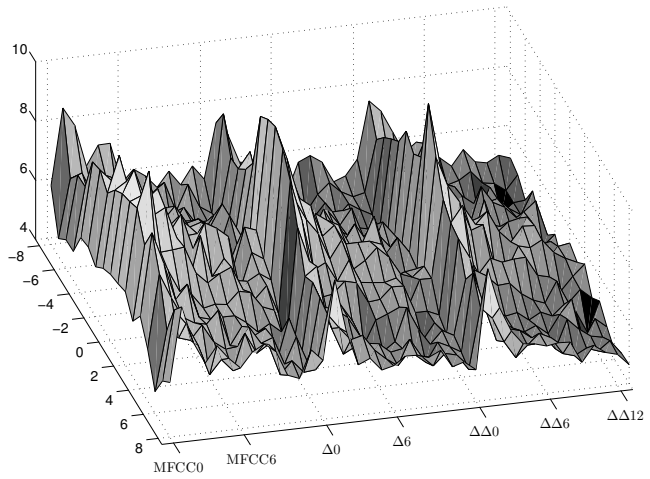

Fig. 2. Total weights for the 39 MFCC features and 17 neighbouring rows when using boosted decision trees.

technique, which seeks to find the *only* correct class label, the interpretation is not that plausible any more (although a HMM/ANN hybrid set-up still performs well). The states of a phoneme are strongly correlated, and confusing them is not a serious error (sometimes not even an error), but this connection among the classes remains hidden to the classifier, which could be quite far from ideal for a discriminative machine learning method. For example, AdaBoost.MH has an one-against-all set-up, which explains why it is quite sensitive to labeling noise. All this suggests that for optimal performance, besides applying another phoneme classifier method, other modifications in the architecture like avoiding tri-state set-up and/or redundant phonemes might be required as well.

V. Relation to Feature Importance

AdaBoost.MH.BA trains base classifiers on feature subsets, using weights that reflect their learned usefulness. This information can be extracted from the model by taking the sum of the corresponding α values, this being the sum of the weights of base classifiers using the given features, which could serve as valuable feedback for assessing relative feature importance.

Fig. 2 shows the sum of the corresponding α values for the 39 MFCC coefficients and 17 frames, using trees as base learners. As it can be seen, the low-level coefficients (especially the energy, treated as the 0th) are more important than the high-level ones. This suggests that for phoneme recognition we mostly need gross information about the spectral content of the current portion of speech, whereas the fine spectral details are somewhat less important. The original coefficients are used more often than the Δ values, $\Delta\Delta$ vectors being the least frequent vectors used. It is also quite apparent that the values associated with the central frames are more important than those of more distant ones. The exceptions are the first and last frames, where the feature values (especially the derivatives) contain information about subsequent speech regions. These observations mirror our earlier findings [7], which seems a bit surprising. Then we sought to detect filler events and laughter, for which it is logical to prefer lower-level features, but this is not obvious when performing phoneme classification.

VI. Conclusions

In this work we applied a high-precision meta-learner algorithm, the AdaBoost.MH method in the phoneme classification task of speech recognition, where traditionally Gaussian Mixture Models and Artificial Neural Networks are used. We found that AdaBoost is able to produce a performance comparable to that of standard neural networks, although lags behind recently-proposed Deep Neural Networks. These results can be explained by the specifics of task from a machine learning perspective, and the simplicity of our method used to construct class-wise posterior scores from the raw likelihood outputs of AdaBoost. This, however, can be helped by using some more sophisticated posterior construction method or even posterior calibration, but these are subjects of future studies.

References

[1] R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
[2] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. Wiley & Sons, New York, 1973.
[3] C. M. Bishop, *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.
[4] B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.
[5] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine Learning*, vol. 37, no. 3, pp. 297–336, 1999.
[6] L. Vidrascu and L. Devillers, "Detection of real-life emotions in call centers," in *Proceedings of Interspeech*, Lisboa, Portugal, 2005, pp. 1841–1844.
[7] G. Gosztolya, R. Busa-Fekete, and L. Tóth, "Detecting autism, emotions and social signals using AdaBoost," in *Proceedings of Interspeech*, Paris, France, 2013, pp. 220–224.
[8] D. A. van Leeuwen, A. F. Martin, M. A. Przybocki, and J. S. Bouten, "NIST and NFI-TNO evaluations of automatic speaker recognition," *Computer Speech & Language*, vol. 20, no. 23, pp. 128–158, 2006.
[9] A. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Trans. ASLP*, vol. 20, no. 1, pp. 14–22, 2012.
[10] S. Shalev-Shwartz and N. Srebro, "SVM optimization: Inverse dependence on training set size," in *Proceedings of ICML*, 2008, pp. 928–935.
[11] M. Borchert and A. Dusterhoft, "Emotions in speech-experiments with prosody and quality features in speech for use in categorical and dimensional emotion recognition environments," in *Proceedings of NLP-KE*, 2005, pp. 147–151.
[12] H. Wu, Y. Lu, and Z. Wu, "Improved AdaBoost algorithm using VQMAP for speaker identification," in *Proceedings of ICECE*, 2010, pp. 1176–1179.
[13] O.-W. Kwon and T.-W. Lee, "Optimizing speech/non-speech classifier design using AdaBoost," in *Proceedings of ICASSP*, vol. 1. IEEE, 2003, pp. 436–439.
[14] P. C. Khoa, "Noise robust voice activity detection," Ph.D. dissertation, Nanyang Technological University, 2012.
[15] A. Roy, M. Mathew Magimai-Doss, and S. Marcel, "Boosting localized binary features for speech recognition," in *Proceedings of MLSLP*, 2012.
[16] S. Petridis and M. Pantic, "Audiovisual discrimination between laughter and speech," in *Proceedings of ICASSP*. IEEE, 2008, pp. 5117–5120.
[17] H. Fujimura, Y. Shinohara, and T. Masuko, "N-best rescoring by phoneme classifiers using subclass AdaBoost algorithm," in *Proceedings of Interspeech*, 2013, pp. 3327–3331.
[18] A. Klautau, N. Jevtic, and A. Orlitsky, "Combined binary classifiers with applications to speech recognition," in *Proceedings of Interspeech*, 2002, pp. 2469–2472.

[19] C. Dimitrakakis and S. Bengio, "Boosting HMMs with an application to speech recognition," in *Proceedings of ICASSP*, vol. 5. IEEE, 2004, pp. 621–624.
[20] J. Du, Y. Hu, and H. Jiang, "Boosted mixture learning of Gaussian mixture HMMs for speech recognition," in *Proceedings of Interspeech*, 2010, pp. 2942–2945.
[21] H. Tang, M. Hasegawa-Johnson, and T. Huang, "Toward robust learning of the Gaussian mixture state emission densities for hidden Markov models," in *Proceedings of ICASSP*, 2010, pp. 5242–5245.
[22] G. Saon and H. Soltau, "Boosting systems for large vocabulary continuous speech recognition," *Speech Communication*, vol. 54, no. 2, pp. 212–218, 2012.
[23] R. Busa-Fekete and B. Kégl, "Fast boosting using adversarial bandits," in *Proceedings of ICML*, vol. 27, 2010, pp. 143–150.
[24] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "The non-stochastic multi-armed bandit problem," *SIAM Journal on Computing*, vol. 32, no. 1, pp. 48–77, 2002.
[25] R. Busa-Fekete and B. Kégl, "Accelerating AdaBoost using UCB," in *KDDCup 2009 (JMLR W&CP)*, vol. 7, Paris, France, 2009, pp. 111–122.
[26] S. Young, G. Evermann, M. J. F. Gales, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, *The HTK Book*. Cambridge, UK: Cambridge University Engineering Department, 2006.
[27] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 1–27, 2011.
[28] D. Benbouzid, R. Busa-Fekete, N. Casagrande, F.-D. Collin, and B. Kégl, "MultiBoost: a multi-purpose boosting package," *Journal of Machine Learning Research*, vol. 13, pp. 549–553, 2012.
[29] A. Maas, A. Hannun, and A. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proceedings of ICML*, 2013.
[30] L. Tóth, "Phone recognition with deep sparse rectifier neural networks," in *Proceedings of ICASSP*, 2013, pp. 6985–6989.
[31] O. Vinyals and L. Deng, "Are Sparse Representations rich enough for acoustic modeling?" in *Proceedings of Interspeech*, 2012.
[32] R. Busa-Fekete, B. Kégl, T. Éltető, and G. Szarvas, "Ranking by calibrated AdaBoost," in *Yahoo! Learning to Rank Challenge*, 2011, pp. 37–48.
[33] Z. Huang and X. Shi, "A distributed parallel AdaBoost algorithm for face detection," in *Proceedings of ICIS*, 2010, pp. 147–150.