<table>
<thead>
<tr>
<th><strong>Title</strong></th>
<th>Speech recognition using adaptively boosted classifier (Accepted version)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Author(s)</strong></td>
<td>Foo, Say Wei; Lim, Eng Guan</td>
</tr>
<tr>
<td><strong>Citation</strong></td>
<td>Foo, S. W., &amp; Lim, E. G. (2003). Speech recognition using adaptively boosted classifier. Proceedings of IEEE Region 10 International Conference on TENCON, 442-446.</td>
</tr>
<tr>
<td><strong>Date</strong></td>
<td>2003</td>
</tr>
<tr>
<td><strong>URL</strong></td>
<td><a href="http://hdl.handle.net/10220/4612">http://hdl.handle.net/10220/4612</a></td>
</tr>
</tbody>
</table>

© IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE. This material is presented to ensure timely dissemination of scholarly and technical work. Copyright and all rights therein are retained by authors or by other copyright holders. All persons copying this information are expected to adhere to the terms and constraints invoked by each author’s copyright. In most cases, these works may not be reposted without the explicit permission of the copyright holder. This material is presented to ensure timely dissemination of scholarly and technical work. Copyright and all rights therein are retained by authors or by other copyright holders. All persons copying this information are expected to adhere to the terms and constraints invoked by each author’s copyright. In most cases, these works may not be reposted without the explicit permission of the copyright holder.
Speaker Recognition Using Adaptively Boosted Classifier

Say Wei FOO, senior member IEEE and Eng Guan LIM

Abstract --- In this paper, a novel approach for speaker recognition is proposed. The system makes use of adaptive boosting (AdaBoost) and multilayer perceptrons (MLP) as classifier for closed set, text-dependent speaker recognition. The performance of the systems is assessed using a subset of 20 speakers, 10 male and 10 female, drawn from the YOHO speaker verification corpus. Results show that improvement in accuracy of recognition can be achieved through adaptive boosting of the classifier.

Index terms: Adaptive boosting, Neural network, Speaker identification, Speaker recognition, Speaker verification.

I. INTRODUCTION

The problem of speaker recognition can be divided into two main areas, speaker verification and speaker identification [1,2]. The aim of speaker verification is to decide if a speaker is who he or she claims to be while the objective of speaker identification is to pinpoint the identity of the speaker without any a priori claims on his or her identity.

The speakers to be recognized can be classified as closed-set or open-set. For a closed-set system, the speakers to be recognized are confined to those who have enrolled with the system. For an open-set system, no restriction on speakers is imposed. The problem with the close-set system is more complex and difficult as no reference information of speakers who have not enrolled is available.

To further reduce the complexity, speaker recognition system may be confined to recognize chosen text; such a system is called a text-dependent system. Text-dependent systems require a high degree of speaker cooperation by dictating the spoken sentence with predetermined text.

A good set of discriminating features that is capable of capturing speaker dependent characteristics and a competent classifier are critical to the overall performance of any speaker recognition system. A collection of feature extraction algorithms and pattern recognition techniques may be found in [3]-[10].

In this paper, a novel approach to speaker recognition is proposed. The system involves the application of AdaBoost to multilayer perceptrons (MLP).

The remainder of the paper is organized as follows. In the next two sections, the architecture and implementation details of the systems are described. The experimental findings are detailed in Section IV and the concluding remarks presented in Section V.

II. THE SYSTEMS

A. System Architecture

The system architecture is illustrated in Figure 1. A speech detection algorithm is applied to the signal to remove silence. Each speech utterance is segmented into frame size of 256 (32ms) with an overlap of 128 (16ms) samples between adjacent frames. Linear Predictive Cepstral Coefficients (LPCC) are computed and used as features of speech segments. Coefficients of order 10 and their respective delta cepstral coefficients form a composite feature vector of 20 components.

An adaptively boosted binary classifier (this classifier itself is composed of multiple classifiers formed by boosting for 20 cycles) is created for each speaker. Each classifier is trained with the true speaker’s samples against samples of the rest of the speakers (false speakers). This structure is based on “one speaker against the rest” dichotomy. The structure has the advantage of breaking down a complex multi-classification problem into N simpler binary classification problems that often facilitates the learning process of the classifier being used. The decision strategy used in this Model depends on the function of the system, that is whether it is performing speaker verification or identification. The details are given in the subsections that follow.
B. Adaptive Boosting

AdaBoost [11]-[15] belongs to the class of boosting algorithms implemented using sample re-weighting where a single learning algorithm is called each time a new weighted distribution is applied to the training samples.

Let \( S \) be a set of \( N \) training examples defined as follows.

\[
S = \{(x_i, y_i)\}_{i=1}^{N}, \quad x_i \in X, \ y_i \in Y = \{0,1\}
\]

where \( X \) is the set of input vectors \( x_i \) and \( Y \) is the set of desired response vector \( y_i \). In the case of binary classification, elements of \( Y \) take on a value of either 0 or 1. A value of 1 indicates that it is a positive example and a value of 0 indicates otherwise.

The AdaBoost algorithm applies a weighted distribution to its set of training examples before calling upon the weak learning algorithm.

Denote the weight vector as \( W(t) \) where

\[
W(t) = \{w_i(t)\}_{i=1}^{N}
\]

in which \( w_i(t) \) is the weight given to the \( t \)th example. Let the weighted distribution at the \( t \)th boosting cycle be denoted by \( D(t) \). This distribution can be expressed mathematically as

\[
D(t) = \{p_i(t)\}_{i=1}^{N}
\]

where

\[
p_i(t) = \frac{w_i(t)}{\sum w_i(t)}
\]

Other important inputs for the AdaBoost algorithm include the choice of the weak learning algorithm and the number of boosting cycles \( T \).

Prior to the start of the algorithm, the elements of the weight vector \( W(t) \) are initialized with a value of 1.0 such that \( D(t) \) follows a uniform distribution. The weak learning algorithm \( C \) is then called upon and given the set of weighted training examples to construct a classifier with the aim of producing a decision/hypothesis that gives a low error rate with respect to the given distribution \( D(t) \) over the training examples.

We define the hypothesis (binary classification) provided by the classifier in boosting round \( t \) as

\[
h_t(x) \in \{0,1\}
\]

where \( x \) represents a training example.

Using the generated hypothesis, the error rate of the classifier is then computed over the given weighted training set. This error rate is basically the sum of the distribution weights of the misclassified examples and can be calculated as follows

\[
\varepsilon_t = \sum_{i=1}^{N} p_i(t) |h_t(x_i) - y_i|
\]

The error rate may be reduced if the classifier focuses on examples achieving the correct response that bears a greater weight at the expense of the not so “important” examples. Instead of minimizing \( \varepsilon_t \), an intermediate parameter \( \beta_t \) defined in (7) is used.

\[
\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}
\]

The following observations can be obtained from (7)

\[
\beta_t = \begin{cases} 
[0,1] & \text{if } 0 \leq \varepsilon_t \leq 0.5 \\
>1 & \text{if } \varepsilon_t > 0.5
\end{cases}
\]

The weight assigned to the classifier for boosting at round \( t \), denoted by \( \alpha_t \), is related to \( \beta_t \) as follows.

\[
\alpha_t = \log \left( \frac{1}{\sqrt{\beta_t}} \right)
\]

The AdaBoost process terminates when \( \alpha_t \leq 1 \), or equivalently \( \beta_t \geq 1 \) or \( \varepsilon_t \geq 0.5 \). From (7), it can be seen that the smaller the value of \( \beta_t \), hence the smaller the value of \( \varepsilon_t \), will give a larger classifier weight \( \alpha_t \). Thus the AdaBoost algorithm places more importance or weight on decisions by more accurate classifiers.

The weight vectors \( W(t) \) and \( D(t) \) are then adjusted by first applying the following adjustments to the elements of \( W(t) \)

\[
w_i(t+1) = w_i(t) \beta_t^{\left[1-y_i\left|h_t(x_i)-y_i\right|\right]}
\]

Note that for a correctly classified example, the weight is reduced by a factor of \( \beta_t \) (which is less than 1) and for an incorrectly classified example, the weight remains the same. It is also now apparent from (10) why AdaBoost terminates if \( \varepsilon_t = 0.5 \) as this will lead to \( \beta_t = 1.0 \) and \( W(t+1) = W(t) \). That is, there is no change to the weight distribution and the process has gone back to Square one.

This procedure is repeated until either premature termination occurs or the specified number of boosting cycles is met. The latter is assumed here. The remaining task is to combine all the generated hypotheses \( h_t(x) \) into a single final hypothesis \( h_f(x) \) that is a function of the classifier weights derived in (9). \( h_f(x) \) can be expressed as

\[
h_f(x) = \begin{cases} 
1 & \text{if } \frac{T}{2} \sum_{i=1}^{T} \alpha_i h_t(x) \geq \frac{T}{2} \sum_{i=1}^{T} \alpha_i \\
0 & \text{otherwise}
\end{cases}
\]

(11) can be interpreted as a weighted majority voting rule. A positive result or a value of 1.0 is produced if more than half of the total votes are in favor of \( x \).

C. Multilayer perceptrons (MLP)

The proposed system uses multilayer perceptrons (MLP) as the base classifier. Multilayer perceptrons belong to the class of neural classifiers, which are popular in solving pattern recognition problems. It can be visualized as a multiple-layer (usually three layers: input, hidden and output) network of fully interlayer-connected neurons with each connection bearing a certain weight. When the
network is exposed to a problem and taught the “right” and “wrong” examples, it starts to learn by altering these connection weights to produce the desired output. Adaptive boosting is applied to the MLP learning algorithm to construct the ensemble classifier. In order to implement adaptive boosting, some slight modifications have to be made to the backpropagation algorithm [16] used to train the MLP network. The details of the backpropagation algorithm are presented in the following paragraphs.

Neurons are the basic building units of the multilayer perceptrons. Referring to Fig.2, let \( y_j^{(l)}(n) \) denote the output or function signal of Neuron \( j \) in Layer \( l \). The synaptic weights of Neuron \( j \) are denoted by \( w_{ji}^{(l)}(n) \) which is the weight connecting Neuron \( i \) to Neuron \( j \). \( v_j^{(l)}(n) \) is the weighted sum of all the synaptic inputs and is also called the induced local field. \( \psi(v_j^{(l)}(n)) \) is the activation function (usually a non-linear function such as hyperbolic tangent function or sigmoidal function) of the neuron, which is also the part of the neuron that performs a non-linear transformation of the inputs. \( d_j(n) \) is the desired response of the system and \( e_j(n) \) is the error between the desired response and the actual response from the system.

\[
y_j^{(l-1)}(n) \quad w_{ji}^{(l)}(n) \quad v_j^{(l)}(n) \quad y_j^{(l)}(n) \quad e_j(n)
\]

Fig.2. Model of a neuron in the last layer

The online mode (sequential example) training is used. The training algorithm consists of a feed-forward process to determine the output of the MLP given a particular input. An error is computed by comparing the output with the desired response and this information is propagated backwards to update the synaptic weights accordingly. This algorithm iterates itself until a stopping criterion is met or until the specified number of epochs is reached.

At the start, the synaptic weights of the MLP are randomly initialized before the training begins. The order of training examples is also randomized at the start of each epoch before being presented to the MLP. Let \((x(n), d(n))\) be the training pattern presented to network.

The local induced field of Neuron \( j \) in Layer \( l \) can be computed as follows

\[
v_j^{(l)}(n) = \sum_{i=0}^{m} w_{ji}^{(l)}(n)y_i^{(l-1)}(n)
\]

where \( y_i^{(l-1)}(n) \) is the output signal from the neuron \( i \) in the previous layer \( l-1 \) at iteration \( n \). Given the value of \( v_j^{(l)}(n) \), the output of the neuron can be calculated by applying the activation function

\[
y_j^{(l)}(n) = \psi(v_j^{(l)}(n))
\]

This process is repeated for each neuron within the layer and the corresponding outputs form the inputs for neurons in the next layer. In the final layer \( L \), the error signal is obtained from the network output and the desired response

\[
e_j(n) = d_j(n) - y_j^{(L)}(n)
\]

where \( d_j(n) \) is the \( j \)th element of the desired response vector \( d(n) \). The size of this error determines the magnitude of the local gradient \( \delta_j^{(l)}(n) \) used to adjust the synaptic weights during the backward propagation part of the algorithm. The local gradient of the network can be computed using two different equations depending on whether the neuron is in the hidden layer (intermediate layers) or in the output layer (final layer). For a neuron in the output layer, the local gradient is defined by

\[
\delta_j^{(l)}(n) = e_j^{(L)}(n)\psi'(v_j^{(L)}(n))
\]

where \( \psi'(v_j^{(L)}(n)) \) is the first derivative with respect to the argument. If the neuron lies in the hidden layers, the local gradient is a function of the weighted contribution of all the local gradients from the following layer and is determined using

\[
\delta_j^{(l)}(n) = \psi'(v_j^{(l)}(n))\sum_k \delta_k^{(l+1)}(n)w_{kj}^{(l+1)}(n)
\]

The last step is to update the synaptic weights of the network using the generalized delta rule given by

\[
w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \alpha \delta_j^{(l)}(n-1)\eta \delta_j^{(l)}(n) y_j^{(l-1)}(n)
\]

where \( \alpha \) is the momentum constant and \( \eta \) is the learning rate of the network. Both parameters are variable and the optimum values may be found by trial and error. In general, a large momentum and learning rate result in faster convergence rate while smaller values of these two parameters allow a more refined search over the weight space for an optimal solution.

The algorithm is iterated until a stopping criterion is met such as the error rate falling below the preset threshold or the number of iterations reaching the specified value.

III. ADAPTIVELY BOOSTED MLP

The main objective of adaptively boosting MLP is to make the error of each example sensitive to the weight
assigned by AdaBoost. To achieve this aim, Equation (14) of the error signal is modified as follows

$$e_j(n) = w_n(t) [d_j(n) - y_j^{(L)}(n)]$$

where $w_n(t)$ is the weight of the $n$th example at boosting round $t$. This will place more emphasis on examples with greater weight (i.e., examples that are misclassified) since a misclassification of such an example will give rise to a larger error. This in turn implies a bigger correction to the synaptic weights in the backpropagation algorithm to obtain the desired response.

**Decision Strategy**

The system may be used for speaker identification or verification. For speaker identification, a winner-takes-all approach is adopted. The score of the test utterance against each speaker model is obtained and the speaker that corresponds to the highest score is selected. Mathematically, this is expressed as:

$$S = \arg \max_{1 \leq j \leq M} W_j$$

(19)

The score $W_j$ for the $j$th speaker model is calculated using

$$W_j = \frac{1}{N} \sum_{i=1}^{N} y_i$$

(20)

where $N$ is the number of feature vectors in the test utterance and $y_i$ is the output corresponding to the $i$th feature vector.

For qualifier-based approach speaker verification, the following assignment rule is adopted

$$V = \begin{cases} 1 & \text{if } W \geq T_d \\ 0 & \text{if } W < T_d \end{cases}$$

(21)

where $V$ denotes the verification status (1 implies a success, 0 implies failed), $W$ is the score of the test utterance against the claimed speaker model and $T_d$ is a preset threshold value, which can be adjusted for optimum performance, usually at the expense of a tradeoff between true speaker rejection errors and impostor acceptance errors.

An alternative decision strategy for speaker verification is to use a competitive-based approach where the speaker’s voice is matched against not only the claimed model but other models as well. A decision is then made based on how well the output of the claimed model fare against the other models.

A measure of how well the claimed model does is expressed in terms of the ratio between the score produced by the claimed model and the score produced by the next most competitive model. Mathematically,

$$R = \frac{S_c}{S_v}$$

(22)

where $S_c$ is the score of the claimed model and $S_v$ is the score of the next most competitive model. The ratio $R$ is then matched against a predefined threshold $T_r$ (e.g. a value of 1.0). For the system proposed in this paper, a speaker independent threshold is adopted. The decision logic is defined as

$$V = \begin{cases} 1 & \text{if } R \geq T_r \\ 0 & \text{if } R < T_r \end{cases}$$

(23)

**IV. EXPERIMENTS AND RESULTS**

The identification and verification performance of the proposed system is assessed using data from the YOHO speaker verification corpus.

**A. YOHO Database**

The YOHO database is a corpus designed specifically for speaker recognition purposes by the US government. The entire database consists of speech samples from 138 speakers, 108 males and 30 females, taken under fairly quiet office conditions. Each training or test utterance is a combination lock phrase e.g. 38_57_62. The training samples are enrolled in four different sessions with 24 utterances of different combination lock phrases per session. The test samples are recorded in ten different sessions, each consisting of four different combination lock phrases per session. The test samples are recorded in ten different sessions, each consisting of four different combination lock phrases.

The proposed system is tested using a subset of 20 speakers, 10 males and 10 females, from the database. For the construction of the training set, 76 utterances from the true speaker and 76 utterances from the remaining speaker (4 from each of the remaining 19 speakers) are used. The verification test set consists of 20 utterances from the true speaker and 10 utterances from each false speaker. The test set for identification is the same 20 utterances from each true speaker used for verification. This gives a total of 400 true speaker tests (also 400 identification tests) and 3800 impostor tests for verification.

**B. Performance Measures**

The accuracy for speaker identification tests is evaluated using

$$\text{Accuracy} = \frac{\text{No. of correctly identified utterances}}{\text{Total number of utterances}}$$

(24)

For evaluation of verification performance, two types of errors are assessed. They are the true speaker rejection rate (type I error) and impostor acceptance rate (type II error). The former corresponds to an error where the correct speaker is rejected by the system and the latter
corresponds to an error where a false speaker is accepted by the system. The two error rates, denoted by \( e_T \) and \( e_F \) respectively, are defined mathematically as follows

\[
e_T = \frac{\text{No. of true speaker rejection errors}}{\text{Total no. of true speaker tests}} \quad (25)
\]

\[
e_F = \frac{\text{No. of impostor acceptance errors}}{\text{Total no. of impostor tests}} \quad (26)
\]

C. System Performance

The performance of the system and the performance of a standalone MLP classifier system without boosting are both evaluated. The results of verification performance and identification performance are presented in Tables I and II respectively.

<table>
<thead>
<tr>
<th>Without Competitive based Scoring</th>
<th>With Competitive based Scoring</th>
</tr>
</thead>
<tbody>
<tr>
<td>( e_T )</td>
<td>( e_T )</td>
</tr>
<tr>
<td>Boosted MLP</td>
<td>6.25%</td>
</tr>
<tr>
<td>Best MLP</td>
<td>7.5%</td>
</tr>
</tbody>
</table>

Table II. Identification performance

<table>
<thead>
<tr>
<th>Identification Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boosted MLP</td>
</tr>
<tr>
<td>Best MLP</td>
</tr>
</tbody>
</table>

From the results, it is evident that boosting results in significantly better performance for both identification and verification. In the absence of competitive-based scoring for speaker verification, boosting achieves a reduction of 1.25% in true speaker rejection errors and 1.77% in impostor or false speaker acceptance errors. The errors are also reduced when using competitive based scoring. Boosted MLP system gives a 4% and 0.42% reduction in true speaker rejection errors and impostor acceptance errors respectively. The favorable results from verification also translate to better performance in identification. The identification errors are in fact the same as the true speaker rejection errors of competitive based scoring systems.

The results also indicate a significantly higher error rate without the use of competitive-based scoring in verification. The absence of additional information in systems without competitive-based scoring forces the decision to be made based on matching the output against a simple threshold. In competitive-based scoring, not only are the true speaker rejection errors reduced, the impostor acceptance errors are also reduced. This is an advantage over the conventional qualifier-based approach where the same impostor utterance can cause the occurrence of multiple errors when tested against all the speaker models.

V. CONCLUSION

An adaptively boosted multipler perceptrons system is proposed for speaker recognition. Experimental results show that significant improvement in the accuracy of recognition is achieved using the boosted MLP system as compared with the standalone MLP system. Results also show that by adopting competitive-based scoring in the treatment of the outputs of the MLP system, further gain in accuracy is obtained.

REFERENCES