Probabilistic scoring using decision trees for fast and scalable speaker recognition

Gilles Gonon, Frédéric Bimbot, Rémi Gribonval

Probabilistic scoring using decision trees for fast and scalable speaker recognition

Gilles Gonon Frédéric Bimbot * Rémi Gribonval

IRISA/METISS (CNRS & INRIA), Campus Universitaire de Beaulieu, 35042 Rennes cedex, France.

Abstract

In the context of fast and low cost speaker recognition, this article investigates several techniques based on decision trees. A new approach is introduced where the trees are used to estimate a score function rather than returning a decision among classes. This technique is developed to approximate the GMM log-likelihood ratio (LLR) score function. On top of this approach, different solutions are derived to improve the accuracy of the proposed trees. The first one studies the quantization of the LLR function to create classification trees on the LLR values. The second one makes use of knowledge on the GMM distribution of the acoustic features in order to build oblique trees. A third extension consists in using a low-complexity score function in each of the tree leaves. Series of comparative experiments are performed on the NIST 2005 speaker recognition evaluation data in order to evaluate the impact of the proposed improvements in terms of efficiency, execution time and algorithmic complexity. Considering a baseline system with an equal error rate (EER) of 9.6% on the NIST 2005 evaluation, the best tree-based configuration achieves an EER of 12.9%, with a computational cost adapted to embedded devices and an execution time suitable for real-time speaker identification.

Key words: speaker recognition, decision trees, embedded systems, resource constraint, biometric authentication

* Corresponding author.
Email addresses: Gilles.Gonon@irisa.fr (Gilles Gonon), Frederic.Bimbot@irisa.fr (Frédéric Bimbot), Remi.Gribonval@irisa.fr (Rémi Gribonval).
URL: http://www.irisa.fr/metiss (Frédéric Bimbot).

1 Introduction

1.1 Presentation and context

The objective of speaker verification is to authentify a person's claimed identity based on a spoken utterance. The task of speaker recognition can be formulated in a probabilistic framework as a hypothesis testing problem : Given a speech segment $Y = \{y_1, ..., y_T\}$ (represented as a sequence of acoustic features y_t) and a claimed identity X, the speaker verification task consists in deciding between two hypotheses H_X and $H_{\bar{X}}$:

- H_X: the observed spoken utterance Y has been pronounced by X
- $H_{\bar{X}}$: the observed spoken utterance Y has been pronounced by another speaker

In practical applications, H_X corresponds to a match between the person's voice and the claimed authorized user enrolled in the system (also called client). Conversely, $H_{\bar{X}}$ corresponds to a mismatch between the current speaker's identity and the observed voice, which happens in the case of an impostor access.

In the Bayesian framework, the optimal test to decide between these two hypotheses is a (log) likelihood ratio (LLR) test:

$$S_X(Y) = \log \frac{p_{H_X}(Y)}{p_{H_{\bar{X}}}(Y)} \begin{cases} \geq \Theta \text{ accept } H_X \\ < \Theta \text{ reject } H_{\bar{X}} \end{cases} \quad (1)$$

where Θ is a decision threshold. This approach relies on the existence of both probability density functions p_{H_X} and $p_{H_{\bar{X}}}$ on the whole acoustic feature space.

Most state of the art text-independent speaker recognition systems rely on probabilistic models of the acoustic features. The dominant approach consists in modelling these features with Gaussian mixture models (GMM) (1) and then using these models in the Bayesian framework. More recently, new approaches using Support Vector Machines (SVM) techniques have been proposed to directly classify either a whole sequence of acoustic features (2; 3) or to learn the decision function on top of GMM systems (4). Performance can be further improved using a combination and fusion of modelling and discriminative approaches (5).

For both GMM and SVM techniques, the size of the speaker templates and/or the complexity of the decision process constitute a bottleneck for the development of real large-scale applications. In particular:

- The score computation time is a crucial point in applications like speaker identification when the number of users is large.
- The size and complexity of the speaker templates can be a limiting factor for implementing voice authentication algorithms on embedded devices (e.g. smart cards, robots).

Several methods have been developed in the field of speech processing to speed up the computation of the likelihood function and reduce the complexity of the model description in the case of GMM models (this aspect has been much less studied for SVM, as it is a more recently emerging technique in the speech area).

Many studies have focused on how to rapidly select the most significant Gaussians in a GMM at the observation level, without having to compute all Gaussians probabilities. Various approaches have been investigated to create "shortlists" directly from the GMM models using vector quantization techniques (6; 7) or decision trees (8); see (9) for a brief review of shortlists techniques and (10) for more information.

Specifically in the field of speaker recognition, two factors dominate the computation process for a GMM system, as studied in (11): the complexity of the mixture and the number of observations to score. Speed-ups are typically obtained by:

- reducing the dimensionality of the GMM
- down-sampling/pre-selecting the number of observations to score
- quantizing the scoring module output for specific devices

With these approaches, typical speed-up factors lie between 15 and 50.

Gaussian selection methods applied to speaker verification are presented for instance in (12). In (9), a similar approach is derived based on structural GMMs to hierarchically cluster all Gaussian mixtures. When the whole test utterance is known, e.g. for processing files rather than speech streams, an observation reordering pruning method is proposed in (13) to rapidly discard unlikely speakers. This method is coupled with Gaussian selection in (14) to achieve a better computational speed-up. In (15), an alternative method is presented which consists in re-quantizing the whole scoring algorithm to implement it on a PDA (Personal Digital Assistant).

While these approaches focus on the log-likelihood ratio function for GMM models, only few generic low-cost alternatives have been proposed for fast probabilistic scoring (16; 17). The initial motivation for the present work has been to seek solutions for biometric authentication with drastically low computational resources and minimal arithmetics, i.e. additions and multiplications.

The targeted embedded devices are smart cards or integrated circuits [1] .

1.2 Decision trees

Decision trees have been extensively used in many applications and offer one of the most economical solutions for classifying data in terms of constrained algorithmic resource needs. When used for speaker recognition, decision trees are usually trained to learn a decision boundary between speaker and non-speaker speech patterns, using for instance the CART (Classification And Regression Tree) approach, as in (16).

Tree based classification schemes have several drawbacks:

- They usually perform less well than conventional probabilistic approaches,
- Decision trees built with the Classification And Regression Tree algorithm (CART) are considered as weak classifiers (18; 19),
- They model a decision boundary which is only valid for a particular operating condition.

In (17; 20), Blouet and Bimbot introduced a new paradigm using decision trees in speaker recognition for directly modelling a score function, for instance a likelihood ratio. This alternative viewpoint has several favorable properties:

- Trees are trained to estimate a score rather than making a decision, which interfaces well with Bayesian decision theory (independently of the operating condition), when the score can be interpreted as a likelihood ratio.
- Trees offer an extremely fast way to assign a score rather than computing it with conventional GMM-based likelihood ratios.
- Trees can be used to approximate any kind of score stemming from any type of classifier.

From the functional point of view, conventional GMM-based systems output a log-likelihood score calculated as the ratio of two probability density functions. In the approach considered in this article, the tree-based system outputs a leaf-dependent score value which is indexed by the nodes of the decision tree.

After the initial work by Blouet et al. (17), additional improvements of this method have been proposed (21). This article further investigates on this paradigm and provides a detailed study and experimental results concerning the accuracy, the speed and the required resources of several variants derived from decision trees. In particular, we present extensions dealing with:

[1] This work was carried out by the IRISA METISS project team in the context of the Inspired IST-2003-507894 European project,http://www.inspiredproject.com.

- score quantization,
- oblique splits in relation with discriminative directions,
- non-uniform leaf-dependent scoring.

Score quantization is used as a way to turn continuous-valued scores into discrete-valued ones, so as to enable the use of classification trees instead of regression trees. Oblique splits are considered so as to rely on region borders which are more suited to the acoustic feature distribution. Finally, the replacement of a constant leaf-dependent score by a non-uniform low-complexity function is intended to refine the ability of the tree-based approach to model score functions with local gradients.

The outline of the paper is as follows. In Section 2, we present the common evaluation framework for all experiments of this article. Section 3 recalls the background and notations for Gaussian Mixture Models applied to speaker recognition. Section 4 presents how decision trees can be used for fast speaker recognition. In section 5, we present the aforementioned extensions of the baseline decision trees. In section 6, the corresponding variants are extensively compared on the common evaluation framework. The results are further discussed in section 7, before the conclusion.

2 Evaluation framework

As indicated in the introduction, we present in detail the assessment protocol used throughout this article to evaluate the performance of the proposed approaches.

2.1 The NIST'05 speaker recognition evaluation

The different systems presented in this article are evaluated under the core test conditions of the NIST 2005 speaker recognition evaluation plan (22). This evaluation uses conversational telephone multilingual speech data of the Mixer Corpus by the Linguistic Data Consortium using the "Fishboard" platform. The core test conditions for training and test segments consist in one side of a conversation excerpt of approximately 5 minutes, which yields about 2 minutes of speech per speaker. The training corpus is composed of 372 female and 274 male speakers and the system evaluation is performed on 31243 tests, composed of 28472 impostor accesses and 2771 client accesses. We denote as N_T the total number of tests, N_C the number of client accesses and N_I the number of impostor accesses.

2.2 Evaluation of system accuracy

The accuracy of a biometric system can be measured in terms of False Acceptance Rate (FAR) and False Reject Rate (FRR). The false acceptance rate measures the percentage of unauthorized impostor attempts accepted by the system and the false reject rate indicates the percentage of genuine client accesses rejected by the system.

When changing the decision threshold of the system, the FAR and the FRR evolve in opposite directions and the point where FAR = FRR is called the Equal Error Rate (EER). For all the systems presented in this article, we focus on the Equal Error Rate (EER) which we consider relevant to the global system performance in terms of hypothesis separability, even though it does not reflect any practical operational situation.

In the context of NIST 05 core test evaluation, the accuracy of the different methods is assessed using the methodology for statistical tests defined in (23): the confidence intervals (CI) around the FAR and FRR are respectively FAR$\pm$ $\sigma_{\text{FAR}}.Z_{\alpha/2}$ and FRR $\pm\, \sigma_{\text{FRR}}.Z_{\alpha/2}$ with

$$\sigma_{\text{FAR}} = \sqrt{\frac{FAR(1-FAR)}{N_I}} \quad , \quad \sigma_{\text{FRR}} = \sqrt{\frac{FRR(1-FRR)}{N_C}} \tag{2}$$

$$Z_{\alpha/2} = \begin{cases} 1.645 \text{ for a 90 \% CI } (CI_{90\%}) \\ 1.960 \text{ for a 95 \% CI } (CI_{95\%}) \\ 2.576 \text{ for a 99 \% CI } (CI_{99\%}) \end{cases} \tag{3}$$

Applying this definition for FAR=FRR, it is reasonable to consider that the confidence interval around the EER is the maximum of the two CIs around FAR and FRR. For the NIST 05 core test, $N_C < N_I$ and therefore σ_{EER} is set equal to $\sqrt{\frac{EER(1-EER)}{N_C}}$. Table 1 gives the CI semi-width for typical EER values in the range of performance of the different systems presented in this article.

Table 1
Confidence intervals semi-width around the EER for typical values of the EER.

EER	σ_{EER}	$CI_{90\%}$	$CI_{95\%}$	$CI_{99\%}$
8%	5.15e-3	0.85%	1.01%	1.33%
10%	5.70e-3	0.93%	1.12%	1.47%
12%	6.17e-3	1.02%	1.21%	1.59%
15%	6.78e-3	1.12%	1.33%	1.75%
20%	7.60e-3	1.25%	1.49%	1.96%

2.3 Evaluation of system complexity

For each system presented in this paper, we present the raw complexity of each algorithm in terms of the number of basic operations, i.e.:

- additions
- multiplications
- logarithm / exponential
- tests (relational operators)

This complexity is also evaluated by measuring the global CPU time and memory used to perform the whole NIST05 core task (31243 tests). All experiments are run on a Linux system with a Pentium 4 @3.20GHz and the CPU times reported are measured using the Unix command *time*.

3 Gaussian Mixtures Models for speaker recognition

3.1 GMM system principles and notations

The most common approach to text-independent speaker recognition is to use Gaussian Mixtures Models (GMM) to model the speech data. A speaker independent model, the Universal Background Model (UBM) is trained over a large amount of speech data from multiple speakers. This model is then adapted to each specific speaker using Bayesian adaptation techniques, e.g. with an Expectation-Maximization (EM) algorithm (24), and a Maximum A Posteriori (MAP) criterion (25).

In the context of the NIST evaluation core task, Reynolds *et al.* showed in (1) that, for state of the art GMM configurations (i.e. several hundreds of Gaussian components) and limited amounts of training data (typically a few minutes), it is experimentally relevant to adapt only the mean of each Gaussian distribution in the mixture. The result of the adaptation process is therefore a speaker-dependent GMM for which there is a one-to-one correspondence between each Gaussian component and its unadapted version in the UBM (which in fact consists in a simple mean shift).

The speaker GMM and the UBM are respectively denoted X and $\bar{X}$. A mixture of N_g Gaussian distributions in a feature space of dimension D is denoted GMM(N_g,D). The mean, the diagonal covariance matrix and the weight of the i^{th} Gaussian distribution in the mixture are respectively denoted μ_i, Σ_i and w_i. By construction, Σ_i and w_i are equal for X and $\bar{X}$, while the means

μ_i^X are adapted from the means of the UBM $\mu_i^{\bar{X}}$. The estimated likelihood that an observation y_t belongs to a GMM(N_g,D) of class $Z \in \{X, \bar{X}\}$ is then:

$$\widehat{\mathcal{P}}(y_t|Z) = \sum_{i=1}^{N_g} w_i . \frac{1}{(2\pi)^{\frac{D}{2}} . |\Sigma_i|^{\frac{1}{2}}} . e^{-\frac{1}{2}(y_t - \mu_i^Z)^T . (\Sigma_i)^{-1} . (y_t - \mu_i^Z)} \tag{4}$$

According to Bayesian decision theory, the authentication of a test data segment Y composed of T observations is done by comparing Y with both a speaker model and the UBM, i.e. X and $\bar{X}$. For each observation y_t composing Y, the score is locally computed as the log-likelihood ratio (LLR):

$$\begin{aligned} S_X(y_t) &= \log \frac{\widehat{\mathcal{P}}(y_t|X)}{\widehat{\mathcal{P}}(y_t|\bar{X})} \\ &= \log(\widehat{\mathcal{P}}(y_t|X)) - \log(\widehat{\mathcal{P}}(y_t|\bar{X})) \end{aligned} \tag{5}$$

This score indicates whether the test observation is better explained by the speaker model or by the UBM. Under the hypothesis of independence of the observations $\{y_t\}$, the score for the verification of Y with respect to the claimed identity X is obtained by computing the mean of the LLR scores over all the observations:

$$S_X(Y) = \frac{1}{T} \sum_{y_t} S_X(y_t), \tag{6}$$

The final binary decision (accept/reject) is obtained by thresholding this score.

A frequent optimization described in (1) consists in searching N_b (typically 5) most likely Gaussians in the UBM for each observation and then compute only those Gaussians likelihood for the target GMM. By removing the computation of less relevant contributions to the likelihood values, this optimization allows to reduce the computation time by a factor $2 * N_g/(N_g + N_b)$ and is bounded by a factor 2.

Several normalization techniques have been developed both at the feature (26; 27) and score (28) levels to enhance the robustness of speaker recognition regarding the acoustic conditions. In the framework of this article, we do not focus on score normalization techniques, although they could improve the system for a specific decision cost function as defined for instance in the NIST05 SRE (22).

Figure 1 illustrates the learning process of a client model from a UBM and the resulting LLR score function in the case of a GMM(128,2). The MAP adaptation is only represented for the 8 most adapted Gaussians (as a complete plot with 256 Gaussians would be unreadable).

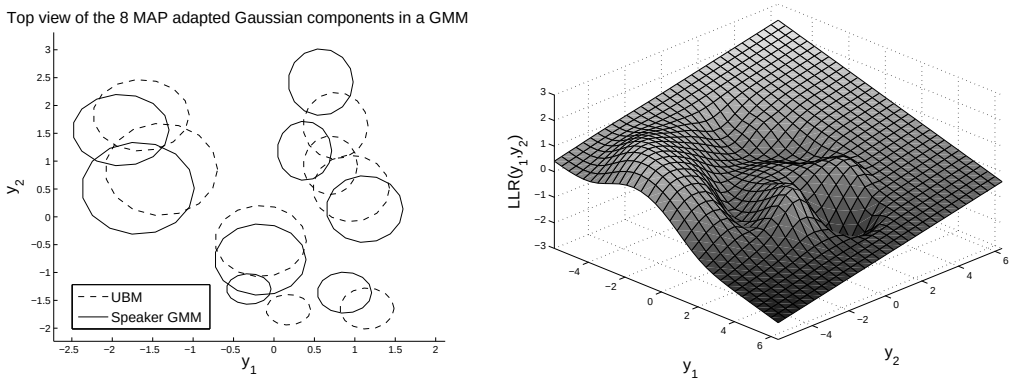

Fig. 1. Examples of a UBM and a client GMM(128,2). Left: Top view of the 8 most adapted Gaussians with MAP mean adaptation in the client GMM and their corresponding Gaussian in the UBM. Right: Log likelihood ratio score function in the feature space (y_1, y_2).

3.2 GMM system parametrization

In this article, the acoustic observations are obtained from 20 ms speech frames at a 10 ms frame rate, each frame being parametrized as a vector y_t of 25 or 33 features: 12 or 16 linear prediction cepstral coefficients (LPCC) + 12 or 16 Δ LPCC + log energy. Cepstral Mean Subtraction (CMS) (29), variance normalization and short-term Gaussianization (26) are applied to the feature vectors. Two gender-dependent UBM with diagonal covariance matrices are trained with approximately 25 hours of speech.

GMM with different complexity are compared in this article in order to evaluate the relationship between system accuracy and model complexity. We consider GMM with 32 to 2048 Gaussians in feature spaces of dimension 25 or 33 features. The computation of the score function given in eq.(6) using the LLR and the GMM probability density function is computationally expensive and cannot be implemented in most embedded devices or for real-time processing.

3.3 GMM system complexity

The number of required operations for computing the LLR in the case of GMM with diagonal covariance matrices is given in table 2. This table is used to compare the complexity of the GMM systems based on LLR computation with the different decision tree based systems.

The performance obtained for GMM systems with different number of Gaussians are presented in section 6.1.

Table 2
LLR score computation complexity of a GMM(N_g,D) with diagonal covariance matrices for scoring a test signal with T frames using the $N_b = 5$ most likely Gaussians in the UBM for each frame.

	addition	multiplication	log/exp	tests
	$(N_g + N_b)*$ $(D+1)*T$	$2.(N_g + N_b)*$ $(D+1)*T$	$(N_g + N_b + 1)$ $*T$	$N_b * N_g$
GMM(128,25)	$3458 * T$	$6916 * T$	$134 * T$	$640 * T$
GMM(2048,33)	$69802 * T$	$139604 * T$	$2054 * T$	$10240 * T$

4 CART for speaker verification

This section first recalls basic concepts about decision trees and then reviews previous investigations on using decision trees for speaker recognition.

4.1 Generalities

Functionally speaking, a decision tree R provides a partition of the acoustic feature space $\mathbf{B}$, defined as:

$$R = \{R_k\}_{1<k\leq K}, \text{ with } \bigcup R_k = \mathbf{B} \text{ and } R_i \cap R_j = \emptyset \tag{7}$$

where each partition corresponds to a particular terminal leaf of the tree. Each leaf can be reached by a sequence of tests which govern the path along the branches of the tree. In the most basic version of decision trees, these tests are binary comparisons of one of the variables with a given value, thus corresponding to hyperplanes perpendicular to that variable. In classification tasks, decision trees are generally used to partition the space in order to define regions in which the data can be assumed homogeneous (i.e., belonging to a same class).

Training a tree to perform a good separation between classes requires the definition of a criterion $C(R)$ which measures the "purity" of the partition. Once this criterion defined, optimizing the tree can be stated as finding the best partition $R^\star$ with respect to C as:

$$R^\star = \arg\max_R C(R) \tag{8}$$

In practice, finding $R^\star$ is generally intractable and several algorithms have been proposed to find sub-optimal solutions, such as CART (Classification

and regression trees) (18). Two commonly used criteria are the the entropy and the Gini dispersion index, which locally measure a region purity as:

$$\text{Entropy criterion} : c_k = \sum_{j=1}^{J} p_{jk} \log p_{jk} \tag{9}$$

$$\text{Gini criterion:} \; c_k = \sum_{j=1}^{J} p_{jk}^2 - 1 \tag{10}$$

In both equations, c_k denotes the value of the criterion on a region R_k and p_{jk} is the probability of the class j in region k (J denoting the total number of classes). Altogether, values of c_k are summed over all partitions to obtain the overall value of criterion $C(R)$:

$$C(R) = \sum_{k=1}^{K} c_k \tag{11}$$

From the algorithmic point of view, univariate trees are grown incrementally, following a greedy procedure by finding the best split over one variable, adding the corresponding branch, and searching again for a new optimal split. Oblique trees consider a linear combination of all the variables, which leads to a more elaborate optimization procedure.

Breiman *et al.* explain in (18) that no stopping rule can be efficiently established to discover the optimal tree and, as a consequence, they introduce the notion of over-growing trees and then pruning them back. This approach prevents from stopping too soon the tree creation process and thus missing detailed clusters of data.

In (16), Farrell *et al.* first evaluated the use of CART decision trees for speaker recognition along with other conventional classifiers. Decision trees are not addressed in terms of computational cost and the authors report poor performance.

In former work, within an applicative context of very low computational resources, Blouet and Bimbot further investigated the use of classification trees for speaker recognition (20; 17). In their approach, a decision tree is used to model locally the density ratio of speaker and non-speaker observations. The results are encouraging but the trees still lack efficiency regarding state of the art results.

The approaches reported in this paper are extensions of Blouet and Bimbot's approach (20; 17), which are summarized in the next paragraph.

4.2 Classification trees on X and $\bar{X}$

Considering the speaker verification application, the most natural approach to use CART consists of learning a speaker specific classification tree for labelling each observation y_t as belonging to X (the genuine speaker) or $\bar{X}$ (any other speaker). Once the partition obtained, it is possible to determine for any new observation y_t the region $R^X(y_t)$ to which it belongs.

The initial approach in (17) consists of assigning a constant score (estimated in the maximum likelihood sense) to each leaf of the tree. This score can be used as the individual contribution of observation y_t when computing the global score of the entire utterance Y :

$$S_X(Y) = \frac{1}{T}\sum_t S_X(y_t) = \frac{1}{T}\sum_{y_t} S_{R^X(y_t)} \tag{12}$$

Therefore, the global score is obtained as the simple accumulation of predetermined values, the sequence of which is governed by the successive regions in which the observations fall.

For each leaf the score can be set in various ways :

- Binary score: $S_{R^X} = +1$ if $N_{R^X}(X) > N_{R^X}(\bar{X})$, -1 otherwise, i.e. a binary value depending on the dominant class in region R^X.
- Log probability ratio score: $S_{R^X} = \log P(X|R^X) - logP(\bar{X}|R^X)$, the local ratio between speaker and non-speaker (training) observation probabilities in the region R^X.

Figures 2 and 3 respectively illustrate the resulting score function in the 2D case for a binary and a log probability score assignment. The score function corresponds to a piecewise constant function in the feature space.

Table 3 gives complexity figures in terms of basic operations for the score computation of a test signal with T observations. Compared to a GMM based system (cf table 2), the decision tree based method reduces drastically the complexity of the verification algorithm since it allows to classify each frame using only a sequence of at most N_Q binary questions, where N_Q is the maximal depth of the tree.

These systems, corresponding to the state of the art decision tree based systems, are largely presented in Blouet's PhD thesis (20). In section 6, they are applied to the NIST 2005 core test evaluation in order to be compared with the improved proposed systems.

Next section details the proposed solutions to improve these tree based techniques.

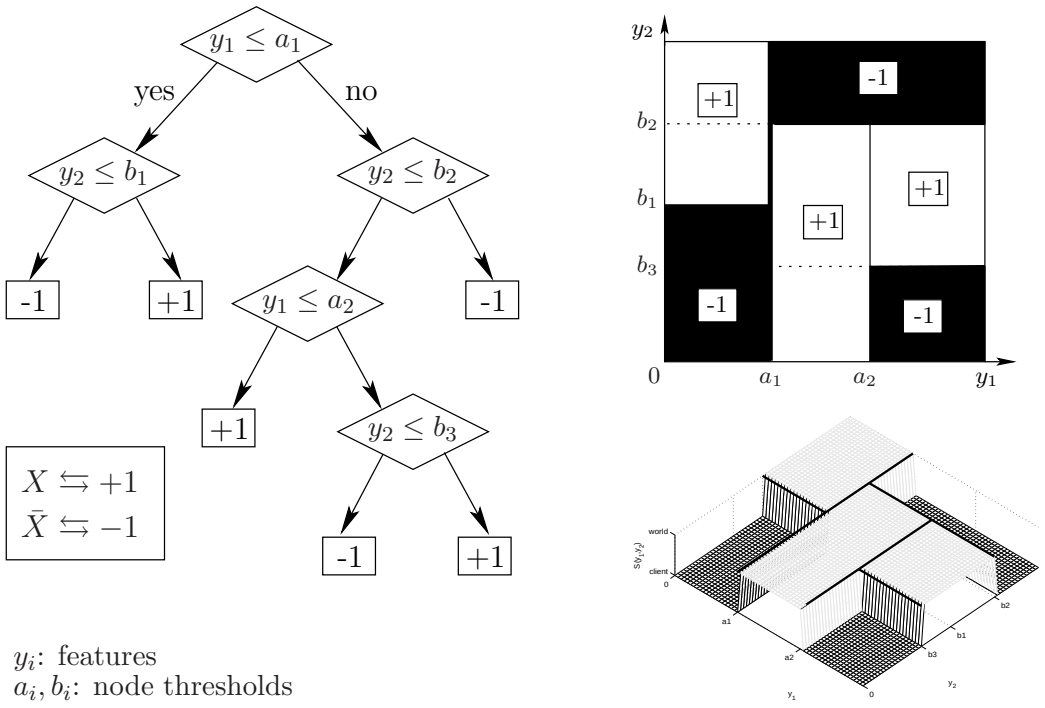

Fig. 2. Example of a univariate decision tree with a binary score assigned to the leaves for a 2-class problem $(X, \bar{X})$. Left: decision tree; right: resulting score function in the 2D feature space.

Table 3
Score computation complexity of a decision tree for scoring a test signal with T observations.

	test	addition	multiplication	log/exp
	$< N_Q \cdot T$	T	1	0
$N_Q = 16$	$< 16 * T$	T	1	0

5 Improved CART-based speaker verification

In this section we further develop and discuss the contributions of this article on decision trees for speaker recognition. The aim is to use decision trees not as classifiers but as a way to perform fast score assignment.

We first point out some limitations of these approaches in the light of the GMM/UBM approach. According to these consideration, a first contribution consists in building classification trees on quantized values of the LLR rather than on the true underlying classes X *vs* $\bar{X}$.

The second contribution takes advantage of the GMM models used to compute the LLR function for creating optimized oblique trees with a lower complexity than standard methods.

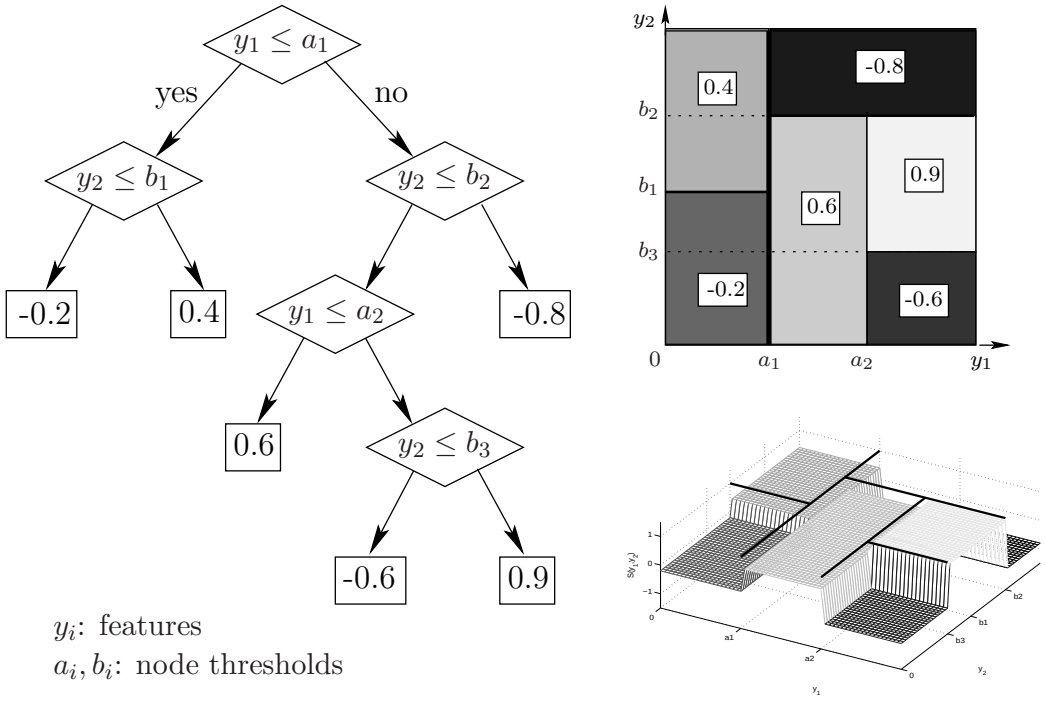

Fig. 3. Example of a univariate decision tree with a log probability ratio score assigned to the leaves for a 2-class problem. Left: decision tree; right: resulting score function in the 2D feature space.

The last contribution presented in this section is a rather general improvement of the problem of designing accurate regression trees : we propose to use a low-complexity score function (instead of a constant score) within each leaf of the tree, in order to potentially reduce the discontinuities between adjacent regions of the trees.

5.1 Initial CART method limitation

The performance of the CART method in its initial implementation is limited by the fact that observations for classes X and $\bar{X}$ tend to overlap considerably in the feature space, while the tree building process operates clustering into regions which are assumed to be composed of pure data of one class. But in the feature space some regions are not discriminative. The CART method spends a significant number of nodes trying to model irrelevant data clusters in such non-discriminative regions.

Alike the GMM approach using the LLR function, we propose to build trees that attempt to approximate a likelihood score function in the feature space rather than trying to roughly cluster feature frames into class X or $\bar{X}$.

Fig. 4 shows the LLR values for a speech excerpt scored with the correspond-

ing client GMM (fig. 4 left) and an impostor GMM (fig. 4 right). This figure illustrates the fact that for both hypotheses, the LLR values take both positive and negative values. In other words, the LLR function is continuous and relatively smooth in the feature space.

In the initial work (17), each training frame was labeled binarily with respect to the speaker who had pronounced the utterance (+1 if it is the genuine speaker, -1 otherwise). This meant not only that the label was binary, but also that the label was assigned at the level of the whole utterance, irrespectively of the local values of the LLR. We will refer to this variant as "Tree output type : $T_{(-1;+1)}$".

In the work reported here, we assume that labelling training data with their respective LLR values is more appropriate and shall lead to more accurate decision trees. A direct consequence of this assumption is that the resulting decision trees will be an approximation of the GMM used for scoring and therefore its accuracy is bounded by the GMM efficiency. This approach will be referred to as "Tree output type : T_{llr}".

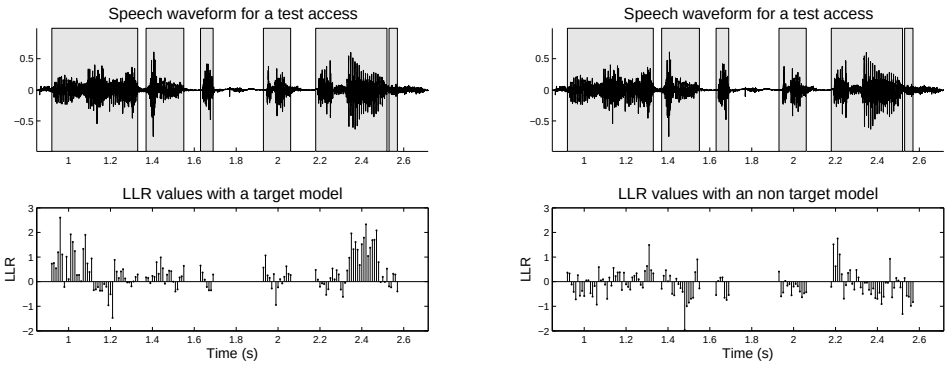

Fig. 4. Top: a test access speech excerpt waveform; gray boxes outline scored segments where speech activity is detected. Bottom: corresponding LLR values for each speech frames scored with a target model (left) and a non target GMM (right).

In practice, a first step consists in training GMM models which are then used to generate sets of observations which follow the GMM density function. These synthetic data then serve as training data for learning the decision tree (21).

This idea is developed in the next section, and we introduce additional proposals for improving the performance of tree-based speaker verification along these lines.

5.2 Classification trees on quantized values of the LLR

The approach investigated in this section is to create trees which approximate the LLR score of acoustic observations, i.e. regression trees on the LLR score.

Regression trees (18), are built using a binary recursive partitioning. The splits are commonly chosen so as to minimize the sum of the squared deviations from the mean in each element of the partition. The idea is to cluster the regression variable into homogeneous regions of the feature space.

In the case of LLR values, figure 5 shows the global histogram of LLR values for all scored frames in the NIST05 evaluation. It can be seen on this figure that the distribution of the LLR values is globally Gaussian. As a consequence, several attempts at building regression trees on the LLR values failed to create trees with more than a few regions. We assume that this is due to the purity splitting criterion, which is not appropriate to the global Gaussian distribution of the LLR values.

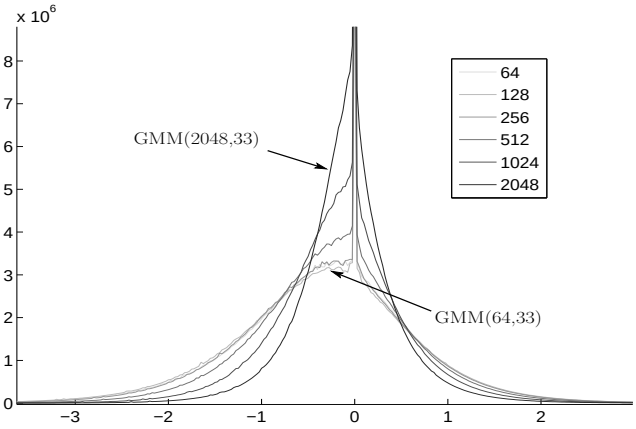

Fig. 5. Histograms of LLR values for all scored frames in the NIST05 evaluation for GMM(64 → 2048,33) systems. The peak values around 0 for all systems corresponds to unadapted regions of the feature space.

Instead of actually trying to approximate the real values of the LLR using regression trees, we have investigated the simpler possibility of coarsely quantizing these values so that the CART method can use classification trees (one class per quantization level). In order to get some insight on the influence of the number of quantization levels of the LLR, we made experiments on the different baseline GMM systems. For each system shown in figure 5, a non-uniform quantization has been applied according to the global LLR score distribution, so as to divide the x-axis in boxes with equal counts.

The EER results for all GMM systems with a number of quantization levels ranging from 2 to 16 are given in table 4. The results show that the quanti-

zation has only little influence on the resulting performance. Only the 2-level quantization of the LLR significantly alters the performance of the GMM system [2].

Table 4
EER results for NIST05 core task. Influence of LLR quantization. ∞ (right column) stands for the baseline results with no quantization. For each line, the best result is in bold.

	Number of quantization level					
	2	3	4	8	16	∞
GMM(64,33)	**11.9**	12.8	12.1	12.4	12.2	12
GMM(128,33)	12.5	11.7	11.3	**11.2**	11.6	11.2
GMM(256,33)	10.8	10.5	10.4	10.6	**10.2**	9.9
GMM(512,33)	10.9	**10.1**	10.2	10.3	10.5	9.8
GMM(1024,33)	10.5	10.6	10.7	**10.1**	10.1	9.8
GMM(2048,33)	10.7	12.1	**10.5**	10.5	10.6	9.6

The slight variability of the results in table 4, which do not behave in a completely monotonous way, can be explained by the width of the confidence interval, i.e. 1% (see table 1).

The main conclusion on this first series of experiments is that they confirm the assumption that it is possible to turn the regression problem on the LLR into a classification problem over a few number of discrete values of the LLR. Regarding table 4, we choose for the rest of the article a 8 level quantization of the LLR, as it gives a fair trade-off for the different GMM complexities and a reasonably low number of quantization levels. These experiments also justify the assumption that accurate classification trees of the LLR quantized values are likely to perform the score assignment task defined in the introduction with a highly reduced complexity.

5.3 Oblique splits based on GMM distribution

The trees considered so far are built using univariate splitting criteria of the type $y_t^{(d)} \leq \theta$ at each node, where $y_t^{(d)}$ is the d-th coordinate of the D-dimensional feature vector y_t, as shown in Fig. 2 and Fig. 3. Oblique trees,

[2] It is important to note that this 2-level LLR quantization problem differs from the binary $X/\bar{X}$ problem in the sense that ± 1 values are assigned for each frame and not fixed for the whole utterance ($+1$ for all frames of a client and -1 for all frames of an impostor in the case of the $X/\bar{X}$ classification problem)

presented in (30) can be more efficient at building discriminant regions, using splitting criteria of the type:

$$\langle y_t, a \rangle := \sum_{d=1}^{D} a^{(d)}.y_t^{(d)} \leq \theta \tag{13}$$

However, the greedy search for the best oblique discriminating hyperplane in the construction of oblique tree is intractable as the complexity grows exponentially with the size and dimension of the training dataset.

We propose to build oblique trees using a specific set of oblique directions determined by the baseline GMM for X and $\bar{X}$, thus limiting the complexity of the training phase. To start with, we illustrate the proposed concept in the simple case of a GMM with $N_g = 1$ Gaussian. In Fig. 6, two Gaussian distributions with the same covariance matrix and different means are represented for the speaker and UBM models. Between the two distributions is represented the set of points where $\widehat{\mathcal{P}}(y_t|X) = \widehat{\mathcal{P}}(y_t|\bar{X})$, which is a hyperplane. On this hyperplane, the score is zero, and more generally the score is a function of:

$$\sum_{d=1}^{D} a^{(d)}.y_t^{(d)} = \langle y_t, \vec{\delta\mu} \rangle, \text{ with } \vec{\delta\mu} := \Sigma^{-1}(\mu^X - \mu^{\bar{X}}), \tag{14}$$

where $\vec{\delta\mu}$ is defined as the "most discriminative direction", Σ being the common covariance matrix of the two Gaussian distributions and μ^X, $\mu^{\bar{X}}$ their means.

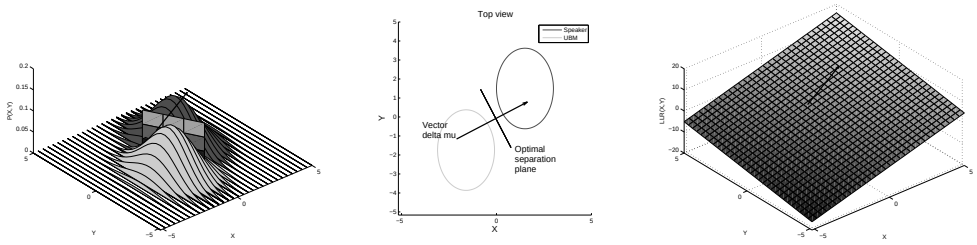

Fig. 6. Left: optimal separation hyperplane and discriminative direction $\vec{\delta\mu}$ for 2 Gaussian distributions with different means in the 2D case. Center: top view. Right: log-likelihood ratio and optimal discriminative direction $\vec{\delta\mu}$.

Extending this notion to the case of a GMM with $N_g > 1$ Gaussians, a set of locally "discriminative directions" is defined as:

$$\vec{\delta\mu}_i = \Sigma_i^{-1}.(\mu_i^X - \mu_i^{\bar{X}}) \tag{15}$$

When the observation y_t lies in a region where the density $\widehat{\mathcal{P}}(y_t|X)$ depends mostly on the i-th Gaussian of the mixture, we expect the direction $\vec{\delta\mu}_i$ to be the most discriminant one. Thus, we propose to build oblique trees by simply

extending the original observation $y_t = (y_t^{(1)}, \ ... \ , \ y_t^{(d)} \ , \ ... \ , \ y_t^{(D)})$ as

$$y_t^{ext} := (y_t^{(1)}, \ ... \ , \ y_t^{(d)} \ , \ ... \ , \ y_t^{(D)} \ , \ \langle y_t, \vec{\delta\mu}_1 \rangle, ..., \langle y_t, \vec{\delta\mu}_{N_g} \rangle). \tag{16}$$

This allows the univariate tree construction on the extended observations to implicitly choose locally optimal hyperplane splits as some nodes of the trees, and not only the canonical directions.

Extending the feature vector is equivalent to augmenting the total dimension of the problem. To avoid an over-extension of the problem when the GMM are composed of hundreds of Gaussians, we sort the influence of the vectors $\vec{\delta\mu}$ according to their $\mathcal{L}^2$-norms and keep only a limited number of them, denoted N_{best}. This sorting is justified by the fact that the more the Gaussians are adapted from the UBM, the more influence they will have on the LLR. Fig.7 shows an example of such an oblique tree that uses $N_{best} = 6$. The resulting partition fits much better the LLR variations than would do an univariate tree with parallel hyperplanes. In all our experiments, we have used $N_{best} = 128$, except when $N_g < 128$, in which case, we have used $N_{best} = N_g$.

Regarding the system resources required for handling oblique trees with extended feature data at the test level, both the algorithmic complexity and memory requirements for the trees are increased compared to basic univariate trees. Experiments in section 6.2.2 show that the use of oblique splits makes the trees much more accurate. The complexity reported in table 5 remains several hundreds time lower than the GMM systems, table 2. Memory aspects are further detailed in section 6.2.6.

Table 5
Score computation complexity for a tree with extended feature data, assuming $N_Q \leq N_g$. Numerical applications stand for a maximal depth of the tree $N_Q = 16$.

	test	addition	multiplication	log/exp
	$< N_Q \cdot T$	$< N_Q \cdot D \cdot T$	$< N_Q \cdot D \cdot T$	0
$N_Q = 16, D = 25$	$< 16 * T$	$< 400 * T$	$< 400 * T$	0

5.4 Assigning a function to the tree leaves

A major drawback of CART trees is known to be the discontinuities between adjacent regions corresponding to different leaves, which may lead to a high misclassification error near the region frontiers. This is due to the fact that the score in each leaf of a tree is constant, either binary in a two class problem, or corresponding to the mean of the target variable value on the leaf region in a regression problem. A classical solution to smooth the piecewise

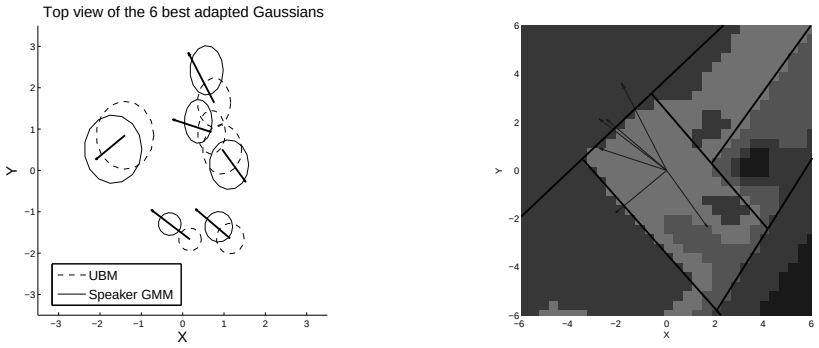

Fig. 7. Oblique splits based on the GMM of fig. 1. Left: the 6 best adapted Gaussians in the speaker model and their corresponding $\vec{\delta\mu}$. Right: example of an oblique tree built with the 6 best $\vec{\delta\mu}_i$ on the quantized LLR values (LLR shown in fig. 1).

Table 6
Score computation complexity for a tree with extended feature data and linear or quadratic score function.

	test	addition	multiplication	log/exp
Linear function	$N_Q \cdot T$	$(N_Q+1) \cdot D \cdot T$	$(N_Q+1) \cdot D \cdot T$	0
$N_Q = 16, D = 25$	$< 16 * T$	$< 425 * T$	$< 425 * T$	0
Quadratic function	$N_Q \cdot T$	$(N_Q+2) \cdot D \cdot T$	$(N_Q+2) \cdot D \cdot T$	0
$N_Q = 16, D = 25$	$< 16 * T$	$< 450 * T$	$< 450 * T$	0

approximation of the tree is to use the boosting technique, see (31), which consists in linearly combining multiple weak classifiers (here multiple trees).

We propose an alternative technique that consists in replacing the constant score with a linear or quadratic score function on each leaf of the tree: for any observation y_t falling into a leaf i of the tree, the score is expressed as $S(y_t) = f_i(y_t^{(1)}, ..., y_t^{(D)})$. In the case of a linear score function $f_i(y_t^{(1)}, ..., y_t^{(D)}) = \sum_{d=1}^{D} \alpha_{di} * y_t^{(d)} + C$, where C is a constant and the coefficients α_{di} depend on the leaf. This approach can be extended in a straightforward manner to a quadratic function of the variables $y_t^{(d)}$.

We apply a conventional least squares algorithm on the collection of training observations falling into the leaf in order to compute the coefficients that fit best the true LLR in the leaf region, which we substitute to the quantized LLR used to build the tree. In terms of computational complexity, see Table 6, this modification has little influence, adding only a linear combination on the feature coefficients for the score computation compared to tables 3 and 5, and remains much lower than the GMM system complexity (table 2). It has much more impact on the memory aspects, as discussed in section 6.2.6.

5.5 Summary of contributions for the tree creation

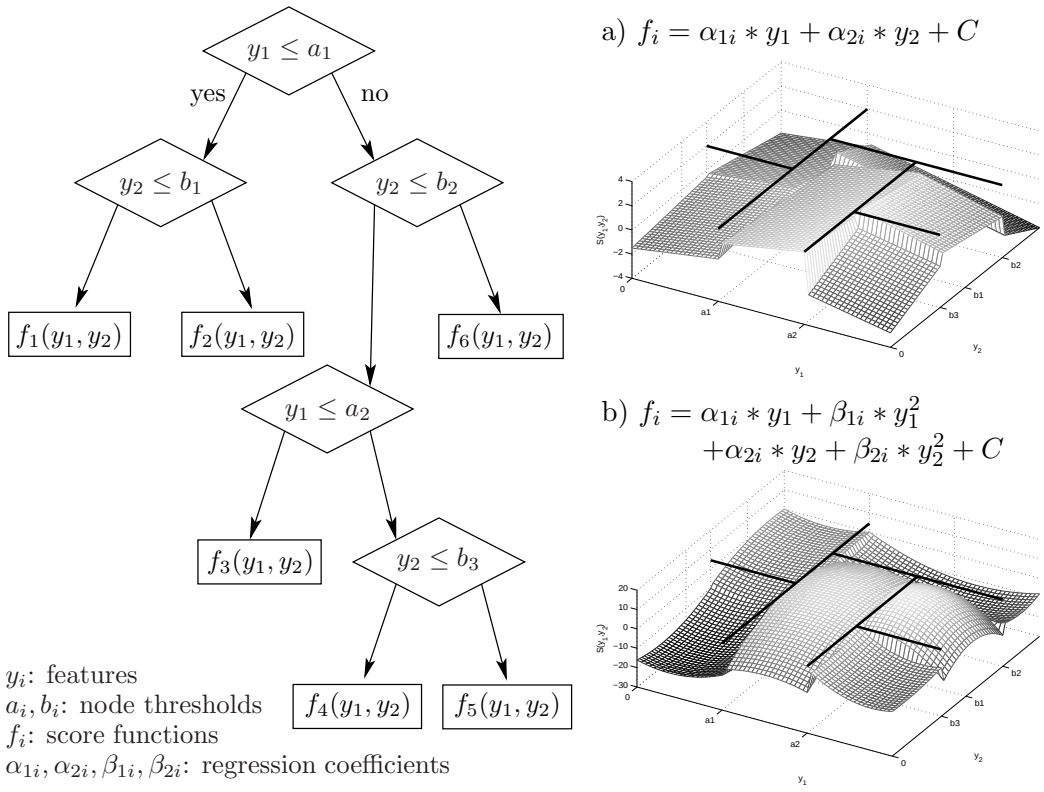

Fig. 8. Example of a univariate decision tree with different score functions as LLR estimation : a) piecewise linear function, b) piecewise quadratic function.

Three improvements have been introduced in this section for the construction of more accurate decision trees. The first and second ones are closely related to the GMM approach of speaker recognition. They could be then easily transposed to the simplification of any classification problem based on GMM models and likelihood scores. The third improvement, which consists in assigning a score function to the tree leaves is an extension of the regression tree method that can be applied on top of any tree creation method. The different steps of the tree creation algorithm are schematically summarized in table 7, and fig. 8 shows an example of the scores assigned by decision trees with linear and quadratic functions assigned to the leaves of the trees.

6 Experiments

The aim of the experiments reported in this section is twofold: we both assess the accuracy of the proposed decision trees approaches and we evaluate the resources required by these approaches. To achieve these goals, we present

Table 7
Summary of the algorithmic steps for the tree construction including our contributions.

Step 0	**Build a GMM for the target speaker**
	- Learn the means of the speaker dependent GMM from the UBM
Step 1	**Construction of the training dataset**
	- Draw synthetic data from UBM and speaker GMM
	- Compute extended features $\vec{\delta\mu}$ and append them to the dataset
	- Label the dataset with quantized values of the LLR
Step 2	**Creation of the decision tree**
	- CART method with the extended features dataset
Step 3	**Assignment of a score function**
	- Store the leaves where the features frames fall
	- For each leaf, estimate a score function using a regression on the LLR values for the training observations

several series of experiments which focus on specific aspects. A summary, with a general discussion of these results is given in section 7.

Before studying the decision trees, a first series of results for baseline GMM systems are presented in section 6.1. They are used as reference results for the comparison of the accuracy and speed of the tree based systems. Experiments on decision trees are grouped in section 6.2. The influence of the proposed improvements and the different tunable parameters are isolated as much as possible in separate sections, from 6.2.1 to 6.2.5. Considerations on computation speed and resource usage are presented in section 6.2.6.

All the experiments are done in the framework of the NIST05 speaker recognition evaluation plan, as described in section 2, and the CART method introduced by (18) is used to grow the trees, using the *wagon* algorithm of the Edinburgh Speech Tools Library, developed by the Centre for Speech Technology Research, *http://www.cstr.ed.ac.uk/projects/speech_tools/*. All the proposed improvements have been developed on top of this algorithm. We do not address the comparison between different tree creation algorithms but rather focus on the evaluation of the influence of each proposed improvements.

6.1 GMM baseline results

6.1.1 GMM accuracy

Table 8 presents the results for the NIST 2005 core test for the reference GMM systems presented in section 3. Results are given for GMM systems with different numbers of Gaussians ranging from 32 to 2048, functioning with 33 features (16 LPCC coefficients, 16 Δ coefficients and the Δ log-energy). These results are referred to as the baseline results in the rest of this article. The corresponding DET (Detection Error Tradeoff) curves for these systems are plotted in Fig.9.

Table 8
NIST05 EER results for GMM systems trained on 33 features.

Number of Gaussians	32	64	128	256	512	1024	2048
EER with LLR scores	13.5	12.0	11.2	9.9	9.8	9.8	9.6

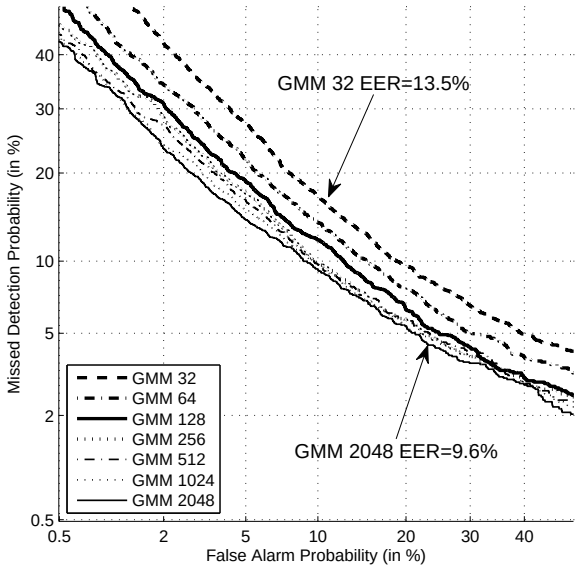

Fig. 9. NIST05 core test DET curves for all GMM systems

6.1.2 GMM LLR computation speed-up

Table 9 summarizes the computation time required for the different GMM systems considered in the previous experiment. The LLR scores are computed in a "one test" versus "one target model" way using a $N_{\text{best}} = 5$ scoring technique as described in section 3. Using this setup, we are focusing on the

raw score computation time, without assumptions on the input test file (it can be scored as an audio stream) or on the target model [3] .

Table 9
Computation times in seconds for the NIST05 core task for GMM systems trained on 33 features with different number of Gaussians (N_g) and fast scoring using $N_{\text{best}} = 5$.

N_g	32	64	128	256	512	1024	2048
CPU time (s)	4073	8068	18546	39951	87916	191239	405160

6.2 Decision trees results

In this section we compare the different approaches based on decision trees which have been presented in sections 4 and 5 in terms of accuracy and computation speed.

The first three series of experiments investigate on the influence of the improvements proposed in section 5, i.e. the type of tree output (6.2.1), the benefit of oblique splits types (6.2.2), and the use of non-constant score function to the tree leaves (6.2.3). For this series of experiments, the common parameters are:

- the acoustic observations are 25-dimensional feature vectors (12 Cepstrum Coefficients, 12 Δs and the Δ log-energy),
- a $GMM(128, 25)$ is used to compute the LLR and create the trees,
- the number of synthetic training data is set to $N_T = 30000$ (50 % for the UBM and 50 % for the target speaker GMM)

The resulting average size of the trees is $N_{leaves} \approx 180$ after the pruning phase.

The other series of experiments focus on the tree size, 6.2.4, on the complexity of the LLR function, 6.2.5 and on the computation speed for the tree-based systems, 6.2.6.

[3] In the context of the NIST evaluations, it is also possible to concatenate all the models tested with the same test in a list. This allows to apply the N_{best} scoring technique for all models in the list, and to reduce these CPU times by a factor of approximately 10 as for the NIST evaluation, each test is typically compared with 10 models. Nevertheless, we consider that this technique falls out of the scope of our comparisons, as we want to compare the algorithms in the case of a single comparison.

6.2.1 Influence of tree output type

In this section we compare the two different ways of labelling the training data for the tree creation process as presented in sections 4 and 5. We recall that classification trees with Speaker (X) *vs* UBM ($\bar{X}$) labelling of data at the utterance level are denoted $T_{(-1;+1)}$ (see section 4). Classification trees on the LLR between the target speaker GMM and UBM are denoted T_{llr} (see section 5.2). Results are shown in table 10.

Table 10
Comparison of EER results as a function of the tree output type $T_{(-1;+1)}$ and T_{llr}. Experiment parameters: LLR computed with $GMM(128, 25)$ quantized on 2 values, number of synthetic training data $N_T = 30000$.

Tree output type	$T_{(-1;+1)}$	T_{llr}
EER	22.3	19.5

The trees based on T_{llr} perform better than trees based on $T_{(-1;+1)}$. The EER for T_{llr} is 3% lower than for $T_{(-1;+1)}$. This confirms the assumption that the problem is better addressed as a classification problem over quantized values of the LLR than over the global speaker class of a speech utterance.

6.2.2 Influence of oblique splits

This section presents results about the use of extended features based on the GMM means adaptation of a speaker model as presented in section 5.3. Table 11 shows EER results of trees created with three different types of datasets: cepstral coefficients alone (CC), $\delta\vec{\mu}$ features alone (Dmu) and both types of data combined (CC+Dmu). Both score types presented in section 4.2 are considered, i.e. a binary score (for T_{llr} trees, the LLR is quantized on 2 values $(+1; -1)$) or a log probability ratio score. As for the previous experiments, results are presented for tree output type $T_{(-1;+1)}$, and for classification trees on the LLR quantized values (tree output type T_{llr}).

For both tree output types, the use of vector $\delta\vec{\mu}$ has a great impact on the results. For the raw classification problem with a binary score, it decreases the EER by more than 2.5%. The use of both types of features (LPCC and $\delta\vec{\mu}$) tends to improve further the results. This may be caused by additional splitting opportunities at the creation process. It will also be seen in section 6.2.6 that this approach slightly speeds up the scoring process.

A drawback of using both types of features is that it increases the dimension of the problem and thus the tree creation time. As a consequence of this, the number of training data cannot be grown as much as when using only the extended features. These experiments show that, though the optimality of the

Table 11
Comparison of EER results for different type of extended features: cepstral coefficients (CC), $\vec{\delta\mu}$ features (Dmu), both types of data (CC+Dmu). Experiment parameters: LLR computed with $GMM(128, 25)$, number of synthetic training data $N_T = 30000$.

		Score type	
Tree output type	Feature type	Binary score	Log prob ratio score
$T_{(-1;+1)}$	CC	22.3	19.2
	Dmu	19.0	16.6
	CC+Dmu	17.6	**16.1**
T_{llr}	CC	19.5	18.1
	Dmu	17.0	15.7
	CC+Dmu	16.2	**15.6**

oblique splits can not be proven theoretically, the resulting partitioning of the feature space is relevant to the variations of the LLR function as it provides a significantly better classification performance for the data labelled with the LLR quantized value.

6.2.3 Influence of the scoring function

These experiments compare various low-complexity scoring functions at the leaves of the trees. To simplify the interpretation of the results, we consider only the classification performance on quantized LLR values (T_{llr}). Results for three types of functions are presented in table 12: the mean LLR value in a leaf, a linear regression on the LLR values and a quadratic regression. As the assignment of the scoring function is done after the tree creation, each horizontal line compares the raw gain provided by the use of different scoring functions, for a given tree structure.

The results obtained with the mean LLR function are significantly the same than those obtained with the log probability ratio score presented in table 11, thus confirming that both methods are qualitatively equivalent. The use of a linear or quadratic regression score systematically improves the EER, between 1.5% and 4%. This confirms the assumption that assigning a score function to the tree leaves in the regression case increases the efficiency of the tree and offers an alternative to boosting techniques.

Experiments in sections 6.2.1 to 6.2.3 confirm and quantify the influence of each proposed improvements for the tree creation. Compared to Blouet's classification tree approach, the EER decreases by 5.5%, from 19.2% to 13.7% and

Table 12
Influence of feature types and of the scoring function complexity on EER results. Experiment parameters: LLR computed with $GMM(128, 25)$, number of synthetic training data $N_T = 30000$.

		Score type		
Tree output type	Feature type	Mean	Linear regression	Quadratic regression
T_{llr}	CC	18.0	15.3	14.2
	Dmu	15.4	13.9	13.8
	CC+Dmu	15.6	14.0	**13.7**

is now only 2.5% away from the baseline GMM(128,25) at 11.2%.

6.2.4 Influence of the number of training data

The experiments presented in this subsection study the influence of the number of training data used to learn the tree structure. The number of training data and the size of the resulting trees are directly related to each other, as the stopping criterion for the tree creation is based on the minimal number of data in each leaf (see section 4.2). In order to make the results clearer, we only consider a subset of the previous parameter combinations. We compare the two classification approaches (tree output types $T_{(-1;+1)}$ and T_{llr}), but we only consider the linear and quadratic score functions.

For this series of experiments, the common parameters are:

- the acoustic observations are 25-dimensional feature vectors (12 Cepstrum Coefficients, 12 Δs and the Δ log-energy) (larger feature vectors does not allow to increase the number of training data over 30000 due to the greediness of the CART algorithm),
- a $GMM(128, 25)$ is used to compute the LLR and create the trees,
- All synthetic training data are equally distributed between the UBM the target speaker GMM.

Table 13 presents the average size of the trees created with different number of training data, and table 14 reports the EER for the corresponding trees and their variants.

Table 13
Average tree size for different number of training data.

Number of training data N_T	30000	60000	90000	120000
Average tree size (number of leaves)	160	311	461	612

Table 14
Influence of number of training data, tree output type and score type on EER results. Lreg and Qreg refer to linear and quadratic regression score functions. Experiment parameters: LLR computed with $GMM(128, 25)$.

		Number of training data N_T			
Tree output type	Score type	30000	60000	90000	120000
$T_{(-1;+1)}$	Lreg	15.5	14.8	15.2	15.8
	Qreg	14.7	**14.2**	14.4	15.8
T_{llr}	Lreg	15.3	14.5	14.1	14.1
	Qreg	14.2	13.9	13.8	**13.6**

Like in previous experiments, T_{llr} trees with quadratic score functions perform better than classification $T_{(-1;+1)}$ trees. The accuracy of the trees is linked to the tree size, and thus to the number of training data. In the case of T_{llr} trees, the EER keeps decreasing when the tree size increases, whereas it passes through a minimum in the case of $T_{(-1;+1)}$ trees. A possible explanation for this is that the $T_{(-1;+1)}$ trees accuracy is bounded because the partitions of the feature space obtained with classified data ($\{X, \bar{X}\}$) are not homogeneous with the LLR score values and so the regression on the LLR in the corresponding regions is not optimal.

It can also be seen from these experiments that additional (though probably marginal) improvement could be expected with more than 120000 training data, with T_{llr} trees. However, in the context of this work, we were limited by the memory constraints of the CART software.

6.2.5 Influence of the underlying GMM complexity

Under our approach, the tree tends to approximate a target score function. Thus its accuracy is bounded by the baseline system performances, in our case a UBM/GMM system. In this section, we focus on the efficiency of the trees to approximate UBM/GMM systems with different number of Gaussians, i.e. when the LLR function is more complex.

For instance, a GMM(64,33) system yields a relatively smooth LLR function in the feature space, whereas a GMM(2048,33) can be expected to yield a more complex-shaped one. Both LLR functions cover approximately the same range in the feature space but the LLR function for a GMM(2048,33) has potentially many more local variations with small amplitudes.

For this comparison, only classification trees on the LLR quantized values (T_{llr}) are considered with regular or extended features and three score functions,

(mean, linear and quadratic regression) are taken into account. The results are presented synthetically on figure 10 and the corresponding EER values are reported in table 15.

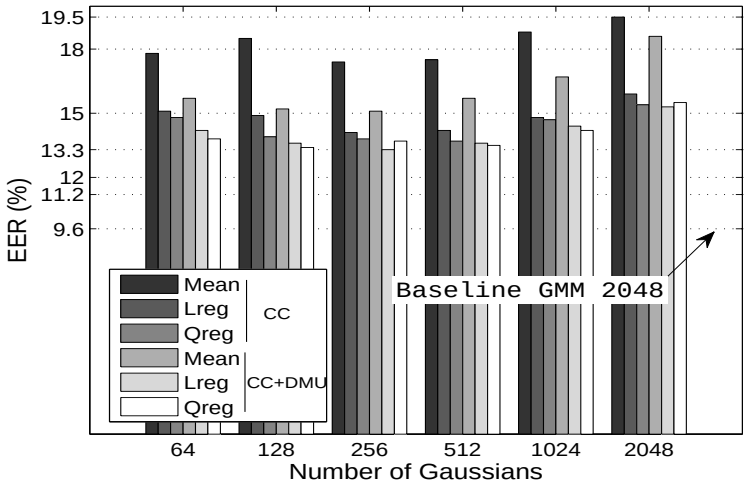

Fig. 10. Influence of GMM complexity and score type on EER results. The vertical axis starts at GMM 2048 baseline EER (9.6%).

Table 15
Influence of GMM complexity and score type on EER results. Results are illustrated in Fig.10.

		Number of Gaussians					
Feature type	Score type	64	128	256	512	1024	2048
CC	Mean	17.8	18.5	17.4	17.5	18.8	19.5
	Lreg	15.1	14.9	14.1	14.2	14.8	15.9
	Qreg	14.8	13.9	13.8	**13.7**	14.7	15.4
CC+Dmu	Mean	15.7	15.2	15.1	15.7	16.7	18.6
	Lreg	14.2	13.6	**13.3**	13.6	14.3	15.3
	Qreg	13.8	13.4	13.7	13.5	14.2	15.5

As in previous experiments, extended features (incorporating oblique projections) outperform standard ones (canonical directions only) and quadratic regression scores yield better results than linear regression scores, which are themselves better that log probability ratio-based scores. Regarding the number of Gaussians in the GMM, the EERs decrease until $N_g = 512$ and then increases for $N_g \geq 1024$. This observation can be connected to experimental results of 6.2.3 and 6.2.4 and can yield to the following hypothesis: all the trees created for these experiments approximately have the same size, $N_{leaves} \approx 190$,

which corresponds to a given capability in terms of being able to model the complexity of a score function. When the number of Gaussians in the GMM model exceeds a given threshold, the tree may become unable to render the many local variations in the LLR score, and it may loose its efficiency with respect to the underlying GMM model. Alternatively (or additionally), a cause for this behavior [4] could also be the limited amount of synthetic training data, that could become insufficient when the complexity of the GMM increases, and would benefit from being scaled to the number of GMM components.

6.2.6 Resources usage

In this section we focus on the resources needed at execution time by the different algorithms proposed. We do not focus on resources for storing the whole tree information. The reason for this is that non-volatile flash memory in embedded device is not anymore a bottleneck to the development of embedded applications (with sizes of at least 4MB, up to several GB).

The execution time for the tree-based systems is reported in table 16 and can be compared to those of the GMM-based systems, presented in table 9. Note that execution time doesn't depend on the tree output type and is the same for classification and regression trees.

Maximal memory requirements during code execution for scoring is inferior to 1kB. During scoring the code only need to load in cache memory the coefficients of the current feature frame (25 or 33 coefficients) and then for each node it needs to load the threshold value and the coefficient for oblique splits comparison or score computation. We have typically 52 floating point values to load in cache memory (approx. 0.2kB) at each step of the iteration.

Table 16
Execution time in seconds for the whole NIST05 common task.

	Score type		
Feature type	Mean	Linear regression	Quadratic regression
CC	127	145	172
Dmu	275	291	321
CC+Dmu	248	266	294

Table 16 shows that decision trees perform 1300 times faster than our implementation of a 2048-Gaussian GMM system (at the expense of a performance loss), and still about 15 times faster than a GMM-system yielding comparable

[4] as relevantly suggested by one of our reviewers.

performance (32-Gaussian GMM). Note also that trees operate without using any log/exp operation.

The costs of the improvements proposed in this article can also be evaluated in terms of execution time for basic operations: the use of extended features doubles the execution time and the use of a quadratic score function multiplies the execution time by a factor 1.25. For the most complex trees, with extended features and a quadratic scoring function, the execution time increases by a factor 2.5. This increase remains relatively low compared to GMM system and can be judged as acceptable when considering the performance gain. In general, the execution time grows logarithmically with the tree size, so that it increases by a factor $1/\log 2 \approx 1.45$ when the tree size doubles.

Regarding memory aspects, the typical size for an unoptimized compiled tree scoring code that includes the template of a speaker is around 100KB, which would be acceptable for an embedded device. For information, this size decreases by a factor 3-4 using a zip compression scheme, and the tree storage can be organized in a more convenient compressed way. It is also important to note that memory is no longer a bottleneck parameter on embedded devices, as flash memories with several MB of non volatile memory are nowadays typically used.

7 Summary of experiments and discussion

The approaches and results presented in this article show a number of trends:

- For all experiments, it turns out to be more efficient to learn trees for estimating the score in the feature space rather than classifying the data. This translates into a typical absolute gain of 1% of EER. As a consequence, the proposed solutions to estimate the LLR score with a tree based approach seems relevant.
- The use of oblique splits perpendicular to the local LLR variation improves systematically the performance compared to the simple use of canonical splits on the standard features, with typical absolute gains of 3% EER.
- Using a regression function (rather than a constant score) in the tree leaves is a way to reduce the discontinuities between neighbouring regions of the trees and also yields a clear benefit, with typical absolute gains of 3% of EER.
- finally, the appropriate combination of these improvements leads to better global results. Compared with classification trees on $\{X, \bar{X}\}$ using a log probability ratio score, the typical absolute gain when combining both improvements is 5%.

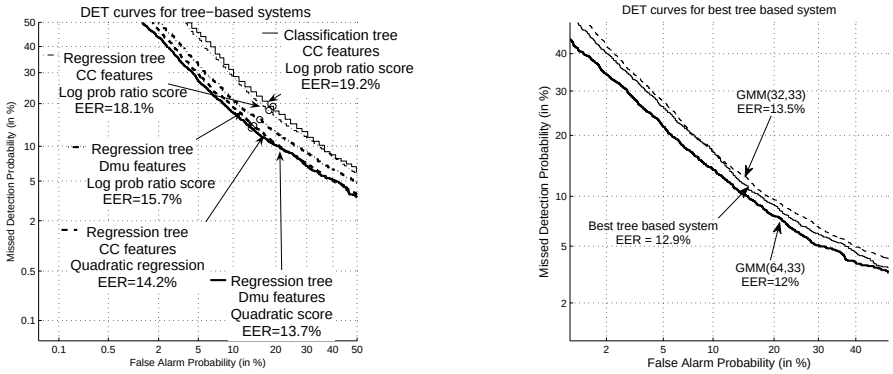

Fig. 11. Comparison of DET curves for the NIST05 SRE. Left: different tree-based systems. Right: Best tree-based system and closest GMM systems.

A limitation of the proposed approach is that it becomes harder to approximate the LLR function when the GMM complexity increases, i.e. when $N_g > 512$. The main reason for this is that the regression function applied to the tree leaves is not relevant of the complex score variations within the leaves. This could be partly overcome by creating larger trees, but there seems to be intrinsic limitations anyway. In our experiments, the best trade-off between system accuracy and function complexity is provided by a GMM(256,33) system trained on 60000 data, which results in a EER of 12.9%.

Figure 11 shows a comparison of the DET curves for the main GMM and tree-based systems presented in this article. The left figure shows the relative influence of each proposed improvement for the tree-based systems. The right figure compares the best tree-based system with the GMM (32,33) (slightly worse) and the GMM(64,33) (slightly better). The proposed improved tree-based approach shows a level of performance somewhere in between a GMM(64,33) and a GMM(32,33).

In the context of this article, we only focus on the EER as an appropriate indicator of the system accuracy. Nevertheless, in the framework of the NIST evaluations, systems are optimized for a specific *cost function* of the FAR and the FRR (the decision cost function, see (22)). To achieve this, score normalization are often used. Regarding the proposed decision trees, as we are estimating a score and not a decision, it remains possible to use score normalization techniques, for instance T-Norm, in order to improve the targeted cost function.

8 Conclusion

This paper has introduced and developed alternative ways of using decision trees, namely for approximating complex score functions rather than for taking

a class decision. The idea behind this proposal is to use decision trees as a low cost replacement system for classifiers such as Gaussian Mixture Models, and apply them in speaker verification (or any type of comparable decision making process).

The decision tree performs fast score assignment instead of heavy score computation, e.g. log-likelihood ratio computation for GMM. To achieve this, various ways of improving the classic CART method have been presented. The first improvement makes use of a priori knowledge on the score function to be estimated for incorporating oblique boundaries to the partitioning process. The second improvement uses simple functions, e.g. linear or quadratic regression, within each leaf of the tree to potentially reduce the discontinuity between adjacent regions of the trees. The resulting trees and their associated score functions can be seen as oblique piecewise quadratic approximations of the true score functions.

This approach has been applied in the context of speaker recognition, where decision trees are used as an estimator of the LLR function deriving from GMM models. As compared to the basic CART approach, the relative error is highly reduced by the proposed improvements while the execution time and complexity of the tree based system makes it suitable for embedded device. For the NIST05 core test condition, the baseline GMM-2048 system has an EER of 9.6% and the best proposed tree-based system reaches an EER of 12.9%, while previous attempts at using CART trees had typical EERs in the range of 19%. The execution time is tenth of times faster than a GMM-64 system (which provides an EER of 12%).

Finally, it is worth noting that the framework of the approach and the variants described in the article can be readily reused for a wide variety of decision making processes, in particular to those related to biometry applications on embedded devices.

References

[1] D. Reynolds, T. Quatieri, R. Dunn, Speaker verification using adapted gaussian mixture models, Digital Signal Processing 10 (1-3).

[2] W. Campbell, Generalized linear discriminant sequence kernels for speaker recognition, in: IEEE ICASSP'02, 2002, pp. 161–164.

[3] V. Wan, S. Renals, Svmsvm: support vector machine speaker verification methodology, in: IEEE ICASSP'03, Vol. 2, 2003, pp. 221–224.

[4] S. Bengio, J. Mariethoz, Learning the decision function for speaker verification, in: IEEE ICASSP '01, Vol. 1, 2001, pp. 425–428.

[5] W. M. Campbell, D. A. Reynolds, J. P. Campbell, Fusing discriminative and generative methods for speaker recognition: Experiments on switch-

board and nri/nto field data, in: Odyssey: The Speaker and Language Recognition Workshop, ISCA, 2004, pp. 41–44.

[6] E. Bocchieri, Vector quantization for the efficient computation of continuous density likelihoods, in: IEEE ICASSP '93, Vol. 2, 1993, pp. 692–695.

[7] H. Murveit, P. Monaco, V. Digalakis, J. Butzberger, Techniques to achieve an accurate real-time large-vocabulary speech recognition system, in: HLT '94: workshop on Human Language Technology, 1994, pp. 393–398.

[8] J. Fritsch, I. Rogina, The bucket box intersection (bbi) algorithm for fast approximative evaluation of diagonal mixture gaussians., in: IEEE ICASSP '96, Vol. 2, 1996, pp. 837–840.

[9] B. Xiang, T. Berger, Efficient text-independent speaker verification with structural gaussian mixture models and neural network, IEEE Trans. on Speech and Audio proc. 11 (5) (2003) 447–456.

[10] M. Gales, K. Knill, S. Young, State-based gaussian selection in large vocabulary continuous speech recognition using hmms, Speech and Audio Processing, IEEE Transactions on 7 (2) (1999) 152–161.

[11] J. McLaughlin, D. A.Reynolds, T. Gleason, A study of computation speed-ups of the gmm-ubm speaker recognition system, in: EUROSPEECH '99, Vol. 3, 1999, pp. 1215–1218.

[12] R. Auckenthaler, J. S. Mason, Gaussian selection applied to text-independent speaker verification, in: Odyssey: The Speaker and Language Recognition Workshop, 2001, pp. 83–88.

[13] B. L. Pellom, H. L. Hansen, An efficient scoring algorithm for gaussian mixture model based speaker identification, IEEE Signal Processing Letters 5 (11) (1998) 281–285.

[14] Z. Xiong, F. T. Zheng, Z. Song, W. Wu, Combining selection tree with observation reordering pruning for efficient speaker identification using gmm-ubm, in: IEEE ICASSP '05, Vol. 1, 2005, pp. 625–628.

[15] Y. S. Moon, C. C. Leung, K. H. Pun, Fixed-point gmm-based speaker verification over mobile embedded system, in: WBMA '03: Workshop on Biometrics methods and applications, 2003, pp. 53–57.

[16] K. R. Farrell, R. J. Mammone, K. T. Assaleh, Speaker recognition using neural networks and conventional classifiers, Speech and Audio Processing, IEEE Transactions on 2 (11) (1994) 194–205.

[17] R. Blouet, F. Bimbot, A tree-based approach for score computation in speaker verification, in: Workshop Speaker Odyssey, 2001, pp. 223–227.

[18] L. Breiman, J. Friedman, R. Olshen, C. Stone, Classification and Regression Trees, Wasdworth Int. Group, 1984.

[19] T.-S. Lim, W.-Y. Loh, Y.-S. Shih, A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms, Machine Learning 40 (3) (2000) 203–228.
URL `citeseer.ist.psu.edu/lim00comparison.html`

[20] R. Blouet, Approche probabiliste par arbres de décisions pour la vérification du locuteur sur architectures embarquées, Ph.D. thesis, U-

niversité de Rennes I (2002).

[21] G. Gonon, R. Gribonval, F. Bimbot, Decision trees with improved efficiency for fast speaker verification., in: 9^{th} European conference on speech communication and technology, EUROSPEECH 05, Vol. 4, 2005, pp. 2661–2664.

[22] NIST, The nist year 2005 speaker recognition evaluation plan (dec 2004). URL `www.nist.gov/speech/tests/spk/2005/sre-05_evalplan-v6.pdf`

[23] S. Bengio, J. Mariéthoz, A statistical significance test for person authentication, in: Odyssey: The Speaker and Language Recognition Workshop, ISCA, 2004, pp. 237–244.

[24] A. Dempster, N. Laird, D. Rubin, Maximum-likelihood from incomplete data via em algorithm, Journal of Statistical Society of America 39 (1977) 1–38.

[25] J. Gauvain, C. Lee, Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains, IEEE Trans. on Speech and Audio proc. 2 (2).

[26] B. Xiang, U. V. Chaudhari, J. Navratil, G. N.Ramaswamy, A. R. Gopinath, Short-time gaussianization for robust speaker verification, in: IEEE ICASSP '02, Vol. 1, 2002, pp. 681–684.

[27] Y. Xie, B. Dai, J. Sun, Kurtosis normalization after short-time gaussianization for robust speaker verification, in: World Congress on Intelligent Control and Automation,WCICA 2006, Vol. 2, 2006, pp. 9463–9467.

[28] C. Barras, J.-L. Gauvain, Feature and score normalization for speaker verification of cellular data, in: IEEE ICASSP '03, Vol. 2, 2003, pp. 49–52.

[29] S. Furui, Cepstral analysis technique for automatic speaker verification, IEEE Trans. on Acoustic, Speech and Signal proc. 19 (2) (1981) 254–272.

[30] S. K. Murthy, S. Kasif, S. Salzberg, A system for induction of oblique decision trees, Journal of Artificial Intelligence Research 2 (1994) 1–32. URL `citeseer.ist.psu.edu/murthy94system.html`

[31] R. E. Schapire, Nonlinear Estimation and Classification, Vol. 171 of Lecture Notes in Statistics, Springer, 2003, Ch. 8 - The boosting approach to machine learning: An overview, pp. 149–172.