Decision Trees with Improved Efficiency for Fast Speaker Verification

Gilles Gonon, Rémi Gribonval, Frédéric Bimbot[*]

IRISA (CNRS & INRIA) / METISS,
Campus de Beaulieu,
35042 Rennes Cedex, FRANCE
firstname.lastname@irisa.fr

Abstract

Classification and regression trees (CART) are convenient for low complexity speaker recognition on embedded devices. However, former attempts at using trees performed quite poorly compared to state of the art results with Gaussian Mixture Models (GMM). In this article, we introduce some solutions to improve the efficiency of the tree-based approach. First, we propose to use at the tree construction level different types of information from the GMM used in state of the art techniques. Then, we model the score function within each leaf of the tree by a linear score function. Considering a baseline state of the art system with an equal error rate (EER) of 8.6% on the NIST 2003 evaluation, a previous CART method provides typical EER ranging between 16% and 18% while the proposed improvements decrease the EER to 11.5%, with a computational cost suitable for embedded devices.

1. Introduction

Most state of the art speaker recognition systems rely on probabilistic models of the dynamics of cepstral coefficients. These discriminant features are modeled with either Gaussian mixture models (GMM) or hidden Markov models. A speaker independent model (world model) is trained over a large amount of speech data from multiple speakers. This model is then adapted to each specific speaker. A verification test consists in comparing test data Y with both client and world models, denoted respectively X and $\bar{X}$. For each frame Y_t of the test data, the score is locally computed as the log-likelihood ratio (LLR) of the estimated probabilities $\widehat{\mathcal{P}}$ for X and $\bar{X}$.

$$S_X(Y_t) = \log \frac{\widehat{\mathcal{P}}(Y_t|X)}{\widehat{\mathcal{P}}(Y_t|\bar{X})} \tag{1}$$

This score indicates whether the test frame is closer to the client model or to the world model. The score for the verification of Y with the claimed identity X is obtained as the mean of all frame scores

$$S_X(Y) = \frac{1}{N} \sum_{Y_t} S_X(Y_t) \tag{2}$$

and the binary decision is obtained by thresholding this score.

In the case of text independent speaker recognition, most state of the art techniques use GMM to model the speaker data. The models are trained with EM algorithms and maximum a posteriori techniques. To have the best trade-off between model training complexity and efficiency, Reynolds [5] has shown that it is only worth adapting the mean of each Gaussian distribution in the mixture using the GMM. In the models for $\bar{X}$ and X with N_g Gaussians in a D-dimensional space, the weights w_i and diagonal covariance matrices Σ_i are equal for X and $\bar{X}$ while means $\mu_i^{\bar{X}}$ and μ_i^X are different. The estimated probability of a feature frame Y_t to belong to a GMM $Z \in \{X, \bar{X}\}$ is then:

$$\widehat{\mathcal{P}}(Y_t|Z) = \sum_{i=1}^{N_g} \frac{w_i . e^{-\frac{1}{2}(Y_t-\mu_i^Z)^T . (\Sigma_i)^{-1} . (Y_t-\mu_i^Z)}}{(2\pi)^{\frac{D}{2}} . |\Sigma_i|^{\frac{1}{2}}} \tag{3}$$

Since typical GMM are built with hundreds of Gaussians (e.g. 128, 256 or 512) in a high dimensional space (e.g. 25 or 33 features), the computation of the score function with Eq.(2) using the LLR and the above probability density function is computationally expensive and cannot be implemented in most embedded devices or for real-time processing.

In the framework of the Inspired project IST-2003-507894 one objective is to assess the possibility of designing match-on-card biometric authentication algorithms for the next generation of smart cards. This topic induces heavy constraints on the complexity of such algorithms. In former work within a similar applicative context of very low computational resources, Blouet [1] investigated the use of classification trees for speaker recognition. In his approach, a decision tree is used as a *speaker/world* classifier. The results were encouraging but the trees were lacking efficiency regarding the state of the art results. In this article, we introduce a different approach for the use of trees in speaker verification and propose some solutions which improve the efficiency of the resulting decision trees.

Section 2 provides the experimental framework and baseline results with GMM for comparison. Section 3 summarizes previous experiments on decision trees for speaker recognition and introduces our new approach for using classification trees. Section 4 details the proposed improvements for reducing the difference of efficiency with state of the art techniques. Comparative experiments are discussed in section 5 before concluding and giving perspectives in section 6.

2. Baseline experiment

All experiments reported in this article are done under the condition of the NIST03 evaluation plan [4]. This evaluation makes use of the second release of the cellular switchboard corpus of the Linguistic Data Consortium (LDC, refer to *www.ldc.upenn.edu/Projects/*). We use the part of the women corpus composed of the speech of one speaker excerpted from cellular conversations. Training data for 207 speakers, are excerpted from two minutes of speech from a single conversation and the 2086 test segments are excerpted from one minute conservation, i.e. from few seconds data length to one minute. The

[*]This work is supported by the European Community, in the context of the Inspired project, FP6 IST-2003-507894.

equal error rate (EER) is used as a global performance indicator of the considered system, as the scope of this article is to focus on relative improvements without considering a particular cost function.

GMM were trained on the NIST03 corpus with a mixture of 128 Gaussian distributions in a space of 25 coefficients (12 cepstral + 12 Δ cepstral + log energy). The EER of this recognition system is 8.6% estimated through the NIST 2003 evaluation plan, and can be considered as a state of the art result.

This experiment is used all along this article as the baseline result to which reduced complexity tree techniques will be compared. Three aspects are compared: the algorithmic complexity, the memory required to store the templates and the EER of the system.

3. CART for speaker verification

Classification and regression trees, aka CART [2], provide a method for classifying raw data without any assumption on the data distribution. The general method consists in clustering data by minimizing a purity or dispersion criteria within each cluster, e.g. the Gini criteria. Univariate trees are grown iteratively by finding the best split over one variable, while oblique trees consider a linear combination of all the variables.

3.1. Classification trees on X and $\bar{X}$

Considering the speaker verification application, the most natural approach to use CART consists in labeling each data frame Y_t from the training corpus with X or $\bar{X}$ and building a classification tree trying to match these classes. For each leaf of the resulting tree, denoting N_X and $N_{\bar{X}}$ the number of training data for X and $\bar{X}$ falling in that leaf, the score is set according to the majority ($S(Y_t) = +1$ if $N_X > N_{\bar{X}}$,-1 otherwise). The decision for a test signal Y is the mean of all frame decisions, as in Eq.(2).

Such a method reduces drastically the complexity of the verification algorithm compared to the use of a GMM, since it allows to classify each frame using only a series of at most N_Q binary questions, where N_Q is the maximal depth of the trees. Table 1 (resp. Table 2) gives the complexity in terms of basic operations for the score computation of a test signal with T frames with a GMM (resp. CART) test:

addition	multiplication	log/exp
$2.N_g.(D+1).T$	$4.N_g.(D+1).T$	$(2.N_g+1).T$
$6656 * T$	$13312 * T$	$257 * T$

Table 1: Score computation complexity for a LLR with a N_g GMM in a D-dimensional space. Numerical applications stand for $N_g = 128$ and $D = 25$.

test	addition	multiplication	log/exp
$< N_Q \cdot T$	T	1	0
$\approx 20 * T$	T	1	0

Table 2: Score computation complexity for a tree. Numerical applications stand for a maximal depth of the tree $N_Q = 20$.

With the CART algorithm applied on data frames composed of 25 cepstral coefficients, typical EER lay between 16% and 18.1%, depending on the size of the trees.These results are further discussed in section 5.

3.2. Limitations

The performance of the CART method applied to raw training data is limited by the fact that some feature frames Y_t for X and $\bar{X}$ may be close in the feature space, making it impossible to cluster them into different regions of the feature space. This is inherent to the training method of a speaker model, where some regions of the feature space are not discriminative. Typically, in the GMM approach, the means of the Gaussians that model these regions are not adapted, but CART may try to spend a significant number of nodes trying to model unrelevant data clusters in such non-discriminant regions.

Since averaged local scores computed with the LLR provided by GMM yield better performance, we propose to build trees that, instead of trying to classify each feature frame into X or $\bar{X}$, rather attempts to estimate the LLR. This is the path we follow in the next section, where we introduce some further proposals for improving the performance of tree-based speaker verification.

4. Proposed improvements over CART

This section gathers contributions for the use of trees for speaker verification. The first contribution consists in building regression trees on quantized values of the LLR rather than on the true underlying classes X / $\bar{X}$. The second one consists in creating optimized oblique trees with a lower complexity than standard methods, by exploiting the structure of GMM models. Finally, we propose to use a linear score function instead of a constant score within each leaf of the tree.

4.1. Quantization of the LLR

Given the limitations of classification trees based on the labels X / $\bar{X}$, we have seen that it may be reasonable to rather build trees which try to approximate the LLR of feature frames. Instead of actually trying to approximate the values of the LLR using regression trees, we have investigated the simpler possibility of coarsely quantizing these values so as to use classification trees. In order to get some insight on the influence such a quantization of the LLR, we have made experiments where we actually used the LLR computed with the baseline GMM system but we computed the global score using the average of the quantized LLR.

In a first experiment, we estimated the useful ranges of quantization by applying simultaneously a gate (LLR is set to zero when it is below an absolute threshold) and a limiter (LLR is set to a maximum value when it is over an absolute threshold), as shown in Figure 1. For [gate;limiter] values ranging from $[\pm0; \pm10]$ to $[\pm0.7; \pm1.0]$, experiments showed that the dynamics of the LLR has minimal effect on the global verification performance.

In a next experiment, we studied the influence of the number of quantization levels on system performance. Table 3 shows that the quantization has only little influence on the resulting performance. In the case of quantization on three levels, system performance was insignificantly improved compared to the baseline system.

	Number of classes				
	∞	16	8	3	2
EER	8.60%	8.84%	8.91%	8.56%	8.99%

Table 3: Influence of LLR quantization on system performance.

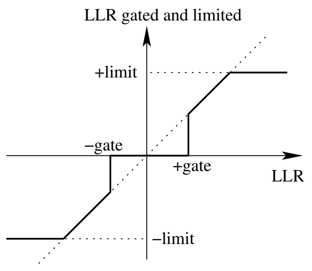

Figure 1: Effect of gate and limiter on LLR.

These two experiments confirmed that it is reasonable to build regression trees on a few number of discrete values of the LLR. Results with such trees are discussed in section 5.

4.2. Oblique trees with extended features

The trees considered so far are built using univariate splitting criteria of the type $y_d \leq \theta$ at each node, where y_d is the d-th coordinate of the D-dimensional feature vector Y_t. Oblique trees [3] can be more efficient at building discriminant regions, using splitting criteria of the type $\langle Y_t, A\rangle = \sum_d a_d.y_d \leq \theta$ with A some vector. However, the greedy search for the best oblique discriminating hyperplane in the construction of oblique tree has a high computational complexity which grows exponentially with the size of the training dataset. Our second proposal is 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 get an idea of how we choose these specific oblique directions, it is worth looking at the simple case of a GMM with $N_g = 1$ Gaussian. On Figure 2, the Gaussian distributions under the client and world models are represented as the two hyper-ellipsoids where the density of the corresponding Gaussian distributions exceed a given threshold, say $\widehat{\mathcal{P}}(Y_t|Z) > \lambda$, for $Z \in \{X, \bar{X}\}$. Between the two ellipsoids 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 actually a function of $\sum_{d=1}^{D} A_d.y_d = \langle Y_y, A\rangle$ which is defined in terms of a "most discriminative direction" $A = \Sigma^{-1}(\mu^X - \mu^{\bar{X}})$, Σ being the common covariance matrix of the two Gaussian distributions and μ^X, $\mu^{\bar{X}}$ their means.

In the case of a GMM with $N_g > 1$ Gaussians, we define for each pair of Gaussians sharing the same variance and weight a "discriminative direction" $\overrightarrow{\Delta\mu_i} = \Sigma_i^{-1}.(\mu_i^X - \mu_i^{\bar{X}})$. When the data Y_t lies in a region where the density $\widehat{\mathcal{P}}(Y_t|X)$ depends mostly on a i-th Gaussian of the mixture, it seems reasonable to expect the direction $\overrightarrow{\Delta\mu_i}$ to be the most discriminant one. Thus, we propose to build oblique trees by simply completing the original data $Y_t = (y_1, \ldots, y_D)$ as

$$Y_t^{ext} := (y_1, \ldots, y_D, \langle Y_t, \overrightarrow{\Delta\mu_1}\rangle, \ldots, \langle Y_t, \overrightarrow{\Delta\mu_{N_g}}\rangle). \quad (4)$$

This allows the univariate tree construction to implicitly choose locally optimal hyperplane splits for some nodes of the trees.

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. Table 4 should be compared with Tables 1-2 for comparison of the complexity of the score computation with other methods. Memory aspects are further detailed in section 5.1.

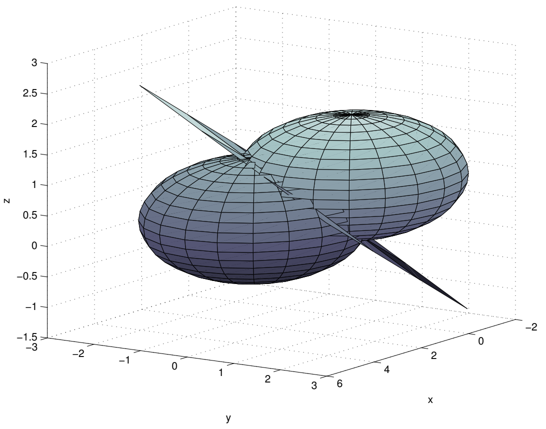

Figure 2: Optimal separation hyperplane for 2 Gaussian distributions with different means in the 3D case.

test	addition	multiplication	log/exp
$< N_Q \cdot T$	$< N_Q \cdot D \cdot T$	$< N_Q \cdot D \cdot T$	0
$< 20 * T$	$< 500 * T$	$< 500 * T$	0

Table 4: 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 $N_Q = 20$.

4.3. Linear score function in the tree leaves

A major drawback of CART trees is known to be the discontinuities between adjacent regions corresponding to different leaves, which lead to a high misclassification error near to the region frontiers. This is due to the fact that the score in each leaf of a tree is generally 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 common solution to smooth the piecewise approximation of the LLR function given by the tree is to use the boosting technique [6], which consists in linearly combining multiple weak classifiers (here multiple trees). Another possibility that we have investigated is to replace the constant score with a linear score function on each leaf of the tree: for any data frame Y_t falling into a leaf, the score will be $S(Y_t) = \sum_{d=1}^{D} a_d.y_d + b$ where the coefficients a_d and b depend on the leaf. We applied the ordinary least squares algorithm on the training data falling into the leaf in order to compute coefficients that rather fit the true LLR in the leaf region than the quantized LLR used to build the tree. In terms of computational complexity, this modification has little influence, adding only a linear combination on the feature coefficients, see Table 5. It has much more impact on the memory aspects, as discussed in section 5.1.

5. Results and discussion

5.1. Considerations on memory

In the actual smart cards context and with the emergence of compact and inexpensive mass storage device (USB-key), many

test	addition	multiplication	log/exp
$N_Q \cdot T$	$(N_Q+1)\cdot D \cdot T$	$(N_Q+1)\cdot D \cdot T$	0
$< 20 * T$	$< 525 * T$	$< 525 * T$	0

Table 5: Score computation complexity for a tree with extended feature data and linear regression function. Numerical applications stand for a maximal depth of the tree $N_Q = 20$.

efforts have focused on increasing memory of embedded devices. The assumption can be made that it is preferable in terms of cost to raise the amount of memory than the processing - unit capability, as long as the required amount of memory does not exceeds a few MB. Smart cards should typically come with 1MB to 8MB non volatile memory.

In the biometric authentication framework, it is better to increase the size of the template rather than to require expensive computation units to perform logarithms and exponentials. A direct consequence for the decision trees is related to the fact that CART trees performance increase to some extent with the size of the trees. Table 6 provides the estimated memory sizes for the different cases presented in the previous sections. These estimates are not optimized on the basis that coefficients for threshold decisions and score functions are stored as floats (4Bytes) and that each node of the tree is indexed with a long int, along with a bit telling whether it is a leaf or a question.

CART type	Size (in Bytes)	$N_t = 500$
Basic	$\approx 18.N_t$	9kB
Extended data	$\approx 4.N_t.(D+3) + \frac{N_t}{4}$	57kB
Linear score	$\approx 8.N_t.(D+1)$	104kB

Table 6: Raw estimate of memory size for different trees with N_t leaves with D-dimensional features.

5.2. Experiments

In order to compare the proposed improvements to the classical CART method, the trees are built with the same amount of training data. A training corpus of 100 000 synthetic data equally drawn from the baseline world and speaker GMM was used. The tree size can only be approximately fixed with the stopping criteria, i.e. the minimal number of training samples falling in a leaf, N_{stop}. The tree size can slightly vary from one speaker to another due to the pruning phase. Regarding the different approaches to compare, having the same stopping values is not relevant. To ensure some consistency to linear regression, it is important to fix a higher value of N_{stop}, whereas in the case of constant score region, it is better to have N_{stop} small in order to have homogeneous regions in terms of fluctutation of the LLR function. This implies a trade-off between the tree size and the performance. The best experiments for each method are reported in Table 7.

5.3. Discussion

Using the CART method on the world and speaker data gives a weak performance with an EER more than twice larger than that of the baseline state of the art technique. In other experiments (not shown here) where we increased the amount of training data to create larger trees, the resulting EER did not reach values below about 16% (for tree sizes of > 2000 leaves). Turning the problem into a pseudo regression problem – i.e. with the quantized LLR as an objective rather than the true class X / $\bar{X}$ – decreases this bound and allows to reach an EER near 14%. The addition of extended data has great influence on both the performance and the number of leaves in the tree. While it reduces the EER to 13%, it also decreases the number of leaves by a factor 3, though this is not reflected in the template size because of the linear combination coefficient to be stored. Finally, the use of a linear score function on each leaf also yields a fair improvement, and with an EER of 11.3%, the loss of performance compared to the baseline system is down to only 30%.

Method	EER	CPU-time (relative)	Memory size
GMM baseline	8.6%	1	42kB
CART $\{X; \bar{X}\}$	18.1%	0.007	10kB
CART LLR 3 classes	14%	0.02	100kB
CART extended data	13.1%	0.05	50kB
CART extended data linear rescoring	11.4%	0.05	100kB

Table 7: Comparison of performances.

Apart from the application of speaker verification, these results also tend to show that the CART method and the proposed improvement can be used in similar situations to provide a piecewise linear approximation of a complex multidimensional likelihood ratio.

6. Conclusions

In this article three improvements have been proposed for the use of decision trees in the rather general context of estimating the LLR for two GMM. Applied to the classical CART method in a speaker verification sytem, they allow to reduce more than 10 times the complexity of the LLR function computation with a relative error reduction of 80%. Moreover, the proposed method is particularly suitable for embedded devices as it works without resorting to a log/exp calculus, but it can also be directly applied in real-time implementations on a PC or a mobile device.

In further work, we intend to investigate on the use of a criteria homogeneous with linear regression during the tree creation process. Another perspective of this work is to overcome the reduction of discontinuity at the tree region boundary due to the use of a linear score function in each tree leaf.

7. References

[1] R. Blouet. *Approche probabiliste par arbres de décisions pour la vérification du locuteur sur architectures embarquées*. PhD thesis, Université de Rennes I, 2002.

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

[3] S. K. Murthy, S. Kasif, and S. Salzberg. A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2:1–32, 1994.

[4] NIST. *The NIST Year 2003 Speaker Recognition Evaluation Plan*, feb 2003.

[5] D. Reynolds, T. Quatieri, and R. Dunn. Speaker verification using adapted gaussian mixture models. *Digital Signal Processing*, 10(1-3), 2000.

[6] R. E. Schapire. The boosting approach to machine learning: An overview. *MSRI Workshop on Nonlinear Estimation and Classification*, 2002.