A DP Algorithm for Speaker Change Detection

Michele Vescovi[1], *Mauro Cettolo*[2], *Romeo Rizzi*[1]

[1]Università degli Studi di Trento
Facoltà di Scienze MM.FF.NN.
I-38010 Povo di Trento - Italy

[2]ITC-irst, Centro per la
Ricerca Scientifica e Tecnologica
I-38010 Povo di Trento - Italy

cettolo@itc.it

Abstract

The Bayesian Information Criterion (BIC) is a widely adopted method for audio segmentation; typically, it is applied within a sliding variable-size analysis window where single changes in the nature of the audio are locally searched.

In this work, a dynamic programming algorithm which uses the BIC method for globally segmenting the input audio stream is described, analyzed, and experimentally evaluated.

On the 2000 NIST Speaker Recognition Evaluation test set, the DP algorithm outperforms the local one by 2.4% (relative) F-score in the detection of changes, at the cost of being 38 times slower.

1. Introduction

In the last years, many efforts have been devoted to the problem of audio segmentation by the research community. This is due to the number of applications of this procedure, that range from the information extraction from audio data (e.g. broadcast news, meeting recordings), to the automatic indexing of multimedia data, to the improvement of accuracy of recognition systems.

A widely used approach to audio segmentation is based on the Bayesian Information Criterion (BIC) [1, 2, 3, 4, 5, 6, 7]. Typically, the BIC method is used locally, within a shift variable-size window algorithm, where single changes in the nature of the audio are searched.

In this work, a Dynamic Programming (DP) algorithm which uses the BIC method to find globally the best segmentation of the input audio, is described, analyzed in detail, from the viewpoint of computational cost, and experimentally compared with the efficient implementation of the local algorithm discussed in [7].

On the 2000 NIST Speaker Recognition Evaluation test set, the global algorithm outperforms the local one by 2.4% (relative) F-score in the detection of changes, and is 38 times slower.

2. BIC-based Segmentation

Segmenting an audio stream means to detect the time indexes corresponding to changes in the nature of audio, in order to isolate segments that are acoustically homogeneous.

Briefly, the BIC method allows to compare different models, from a set $\{M\}$, of the input sequence $\mathcal{O} = \{o_1 \dots o_N\}$ of observation vectors in the $\mathbb{R}^d$ space, on the basis of the values:

$$BIC(M|\mathcal{O}) \;=\; \log L(M|\mathcal{O}) - \lambda\frac{1}{2}\#(M)\log(N) \qquad (1)$$

This work was partially financed by the European Commission under the project FAME (IST-2000-29323).

for each model M, where $L(M|\mathcal{O})$ is the likelihood of M on $\mathcal{O}$, $\#(M)$ is the number of free parameters of M and $\lambda \in \mathbb{R}$ is a weight.

It is supposed that each model induces a different segmentation of the input stream. The procedure outputs the segmentation associated to the model that maximizes the BIC value. The sensitivity of the method can be tuned by adjusting the value λ to the particular task under consideration.

3. Multiple Spectral Changes Detection

In order to segment a stream with an arbitrary large number of changes, a widely adopted algorithm uses a sliding variable-size analysis window, where the above described method is used to detect single changes. The algorithm and its efficient implementation are described in [7]. Its main limitation is that the search is done locally, that is, each change is detected independently from each other, assuming no dependency between changes. In general, this hypothesis could be not true, especially if the audio stream includes many changes close one to each other, as in the case of human-human conversations.

3.1. Global Search: a DP algorithm

In order to relax the approximation of the local search, we developed a dynamic programming algorithm able to globally search the optimum segmentation of the input audio stream according to the BIC. Between the $O\{2^N\}$ possible segmentations, it allows to select the best one in $O\{N^3\}$ steps, as described in the following.

The BIC value in eq. (1) can be seen as the difference between the log-likelihood $V(M) = \log L(M \mid \mathcal{O})$ and the penalty term $P(M) = \lambda\frac{1}{2}\#(M)\log(N)$ that takes into account the complexity (size) of the model. Assuming a Gaussian process, the models $\{M\}$ among which the algorithm has to select the best one, that is the one with the highest BIC value, differ in either the number of Gaussians or the way the input stream is partitioned by those Gaussians.

Let $\mathcal{G}_k$ be the set of models with k Gaussians. Since the number of free parameters of a d-dimensional Gaussian is $D = d + \frac{d(d+1)}{2}$, the penalty term, for each $M \in \mathcal{G}_k$, is:

$$P(M) \;=\; P(k,N) \;=\; \lambda\frac{1}{2}Dk\log(N) \qquad M \in \mathcal{G}_k$$

This means that all $M \in \mathcal{G}_k$ have the same penalty term. Then, the best way of segmenting the input stream in k homogeneous segments is given by the model M_k such that:

$$M_k \;=\; \arg\max_{M\in\mathcal{G}_k} BIC(M) \;=\; \arg\max_{M\in\mathcal{G}_k} V(M)$$

The algorithm for searching the optimum BIC segmentation has to be able to build the set $\mathcal{M} = \{M_k\colon\ k = 1, \ldots, K\}$, with $K \leq N$, of the optimum ways of segmenting the input stream in k segments, for all possible k's.
Let $V_{k,t}$ be the following matrix for the dynamic programming:

$$V_{k,t} := \max\{V(M|o_1, \ldots, o_t) : M \in \mathcal{G}_k\}$$

$$k = 1 \ldots K \qquad t = 1 \ldots N$$

The maximum number K of segments for each t can be defined as $K = \lfloor t/S_{min} \rfloor$, where S_{min} is the minimum allowed duration (size) of a segment, constraint that must be introduced for numerical reasons. The matrix $V_{k,t}$ is filled column by column by means of the following equations:

$$V_{1,t} = \quad A(o_1, \ldots, o_t) \tag{2}$$
$$V_{k,t} = \max_{(k-1)S_{min} \leq t' \leq t-S_{min}} (V_{k-1,t'} + A(o_{t'+1}, \ldots, o_t)) \tag{3}$$
$$k = 2, \ldots, K$$

where $A(o_a, \ldots, o_b)$ indicates the auto-consistency of $o_a, \ldots, o_b$ (Section 3.2), that is the log-likelihood of G_a^b on $o_a, \ldots, o_b$, where G_a^b is the Gaussian estimated on $o_a, \ldots, o_b$. Note that the range of t' is a function of S_{min}.

The best segmentation with k segments of the input up to time t is given by searching, through the eq. (3), the best segmentation in $k-1$ segments up to $t' < t$, and adding to it the k^{th} segment containing the observations from $t'+1$ to t.

In addition to $V_{k,t}$, another matrix $M_{k,t}$ is needed for recording the time indexes of changes:

$$M_{1,t} = \quad 0$$
$$M_{k,t} = \arg\max_{(k-1)S_{min} \leq t' \leq t-S_{min}} (V_{k-1,t'} + A(o_{t'+1}, \ldots, o_t))$$
$$k = 2, \ldots, K$$

The two matrices $V_{k,t}$ and $M_{k,t}$ have the same structure, and their entries store, respectively, the log-likelihood of the best segmentation in k segments of the input up to the index t, and the time indexes of segment boundaries.

When the end of the input is reached, each entry of the last column of $V_{k,t}$ contains the log-likelihood of the best segmentation of the whole input stream in k segments. By subtracting the penalty term, it is possible to obtain k_{opt}, the number of segments of the optimum segmentation, by:

$$k_{opt} = \arg\max_{k=1,\ldots,K} (V_{k,N} - P(k, N))$$
$$\text{with } K = \lfloor N/S_{min} \rfloor$$

The optimum segmentation is finally re-built by backtracking over the matrix $M_{k,t}$, starting from the entry $M_{k_{opt},N}$ and going back to the previous rows, at the columns (time indexes of changes) specified in the entries.

3.2. Efficient Computation of the Auto-consistency

The algorithm requires the computation of the auto-consistency of $o_{t'+1}, \ldots, o_t$, which by definition can be computed by:

$$A(o_{t'+1}, \ldots, o_t) = -\frac{t-t'}{2} \left(\log\left|\Sigma_{t'+1}^t\right| + d\log 2\pi + d\right) \tag{4}$$

An efficient way of computing the covariance matrix $\Sigma_{t'+1}^t$ for all possible indexes t and t' is described in [7], and relies on the encoding of the input signal with cumulative statistics. In particular, for each incoming t the triple (SQ_1^t, SV_1^t, t) is computed and stored, where:

$$SV_1^t = \sum_{i=1}^t o_i \qquad SQ_1^t = \sum_{i=1}^t o_i \cdot o_i^{tr}$$

from which, given $t' < t$, the mean vector and the covariance matrix are easily computed:

$$\mu_{t'+1}^t = \frac{1}{t-t'} \cdot (SV_1^t - SV_1^{t'})$$
$$\Sigma_{t'+1}^t = \frac{1}{t-t'} \cdot (SQ_1^t - SQ_1^{t'}) - \mu_{t'+1}^t \cdot (\mu_{t'+1}^t)^{tr}$$

with a number of operations equals to $2d(d+1) + 2d$.

3.3. Bounding the Auto-consistency

Computing $A(o_a, \ldots, o_b)$ is costly since requires the estimation of a covariance matrix and the computation of its determinant (see eq. (4)). However, in some cases it is possible to avoid those computations. Let $B(o_a, \ldots, o_b)$ be a bound for $A(o_a, \ldots, o_b)$, that is:

$$A(o_a, \ldots, o_b) \leq B(o_a, \ldots, o_b) \ \forall a, b;\ a \leq b \tag{5}$$

and let assume that the computation of $B()$ is cheaper than that of $A()$. If during the computation of $V_{k,t}$, it happens that for a given t':

$$V_{k,t} \geq V_{k-1,t'} + B(o_{t'+1}, ..., o_t) \ \forall k$$

then, given the eq. (5), we have:

$$V_{k,t} \geq V_{k-1,t'} + A(o_{t'+1}, \ldots, o_t) \ \forall k$$

This means that it is not convenient to hypothesize a change in t', whatever the number of segments k; therefore, for that t', the computation of $A(o_{t'+1}, \ldots, o_t)$ can be avoided.

Now the problem is to define such a bound $B()$. Let μ_a^b and Σ_a^b be the parameters of the Gaussian G_a^b estimated on the d-dimensional observations $o_a, \ldots, o_b$, with $a \leq b$ and $n = b - a + 1$, and let assume that $A(o_a, \ldots, o_b)$ has already been computed. Let us suppose that our goal is to obtain the bound $B(o_c, \ldots, o_{a-1}, o_a, \ldots, o_b)$ for $A(o_c, \ldots, o_{a-1}, o_a, \ldots, o_b)$. It can be shown that $A(\mu_a^b, \ldots, \mu_a^b, o_a, \ldots, o_b) \leq A(o_c, \ldots, o_{a-1}, o_a, \ldots, o_b)$; then we can set the bound $B()$ equals to the auto-consistency of the sequence, where the new observations $o_c, \ldots, o_{a-1}$ have been substituted with the mean of the old sub-sequence. Moreover, such a bound can be efficiently computed by:

$$B(\underbrace{o_c, \ldots, o_{a-1}}_{m}, \underbrace{o_a, \ldots, o_b}_{n}) = A(\mu_a^b, \ldots, \mu_a^b, o_a, \ldots, o_b) =$$
$$= \frac{n+m}{n} A(o_a, \ldots, o_b) - \frac{d(n+m)}{2} \log\left(\frac{n}{n+m}\right)$$

on the basis of the value of the previous auto-consistency and the number $m = a - c$ of the new included observations.

3.4. Further Reduction of the Algorithm Cost

The complexity of the DP algorithm can be further reduced by introducing some reasonable and quite obvious approximations listed in the following.

$\mathbf{K_{max}}$, the maximum number of searched segments. It bounds the number of rows of the DP matrices $V_{k,t}$ and $M_{k,t}$, with a number that is independent from the input audio size N.

$\mathbf{S_{max}}$, the maximum size of a segment. Actually, this approximation allows to greatly reduce the number of operations of the algorithm, but it can be too dangerous, unless S_{max} is set to a very high value, thus losing the benefit of its use. Then, to keep the advantage without losing accuracy, the possibility of hypothesizing a segment larger than S_{max} is kept in few specific circumstances.

Resolution δ, for encoding the audio stream. Like in the local algorithm described in [7], δ establishes the resolution of the computation of triples (SQ_1^t, SV_1^t, t) and of the entries of matrices $V_{k,t}, M_{k,t}$; that is, those values are computed only for $t = \delta, 2\delta, 3\delta \ldots N$. The resolution δ reduces the total cost of the algorithm of a factor which is a function of the square of δ.

3.5. Computational Costs

The main stages of the DP algorithm without any approximation have the following costs, including the $d^3/6$ operations for the determinant evaluation of the covariance matrices:

- input stream encoding: $N(d(d+1)+d)$
- matrices initialization: $O\{N \cdot (d^3/6 + 2d(d+1) + 2d)\}$
- matrices filling: $O\{N^2/2 \cdot (N/S_{min} + d^3/6 + 2d(d+1) + 2d)\}$
- selection of the winner model: $O\{N/S_{min}\}$

where of course the most expensive step is that of filling the DP matrices. Then, the overall complexity of the algorithm is $O\{N^3/S_{min} + N^2d^3\}$.

By introducing the approximations described in Subsection 3.4, the step costs become:

- input stream encoding: $N(d(d+1)+d)$
- matrices init.: $O\{(N/\delta) \cdot (d^3/6 + 2d(d+1) + 2d)\}$
- matrices filling: $O\{(NS_{max}/\delta^2) \cdot (K_{max} + d^3/6 + 2d(d+1) + 2d)\}$
- selection of the winner model: $O\{K_{max}\}$.

Also in this case, the DP step is the most expensive. The complexity of the algorithm is $O\{(NS_{max}/\delta^2) \cdot (K_{max} + d^3)\}$.

The benefit given by the bounding $B()$ introduced in Subsection 3.3 has been experimentally quantified and, depending on the value of S_{max}, it allowed 25% to 36% reduction of the execution time.

4. Experimental Evaluation

4.1. Database

The 2000 NIST Speaker Recognition Evaluation included a segmentation task, where systems were required to identify speech segments corresponding to each of two unknown speakers. The test set consists of 1000 telephone conversations, lasting about 1 minute each, included in the Disc `r65_6_1`. 777 speakers, of both genders, pronounced a total of 46K turns/segments; Fig. 1 shows the distribution of segment length: actually, the longest segment is 26.5 seconds; the mean length is 1.3 seconds, while the median is less than 1 second; it is worth noticing that 81% of segments has length lower than 2 seconds.

Other information about the evaluation data can be found at www.nist.gov/speech/tests/spk/2000/.

The proposed algorithm has been evaluated on the above described test set, after having merged the two reference segmentations (one for each speaker) provided for each conversation. Segments resulting shorter than 0.5 seconds have been removed, since they are assumed to be spurious (due to the imprecision of the manual annotation).

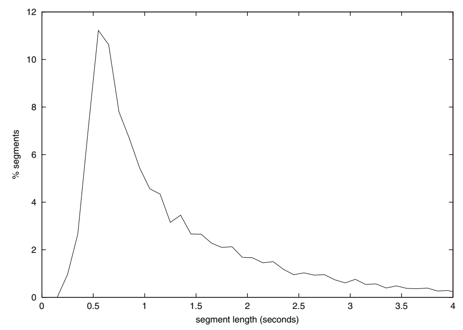

Figure 1: *Distribution of length of test set segments.*

4.2. Performance Measurements

After the official evaluation, NIST made available both reference segmentations of the test files as well as a scoring script. The scoring script computes the segmentation error rate, i.e. the percentage of speech from one speaker wrongly assigned to the other speaker. The measure is suitable for the NIST task which, by definition, is a classification task. On the contrary, the algorithm presented here detects speaker/spectral changes, and no attempt is done to classify segments in terms of speakers; then the NIST scoring script does not fit our evaluation requirements.

In fact, precision and recall (or the corresponding false alarm and miss detection rates) are metrics which better assess performance of the change detection algorithm, and those are the metrics we have adopted, together with their geometric mean (F-score).

4.3. Results

First of all, the global algorithm has been compared with the local one presented in [7]. Table 1 shows both performance and execution times[1] measured for the two algorithms. The improvement in terms of F-score given by the global algorithm is 2.4% relative, at a time cost 38 times higher than the local algorithm. It is worth noticing the benefit (36%) given by the use of the bound $B()$ (Subsection 3.3).

Table 1: *Performance of the local and global algorithms. (0.3 seconds is the tolerance admitted in the change detection.)*

	performance			execution time	
	prec	*rec*	*F-score*	sec	sec without $B()$
local	54.40	75.71	63.31	367.7	—
global	54.67	79.67	64.84	13930.5	21796.8

In the following, we report on the experiments performed in order to measure the impact of the approximations introduced into the exact DP algorithm.

Figures 2, 3, 4 and 5 plot F-score and execution time as functions of S_{min}, K_{max}, S_{max} and δ, respectively, each computed keeping fixed all the other parameters of the algorithm.

$\mathbf{S_{min}}$: It results that the value of the minimum window is critical, and the best one is not equal to the minimum size of test segments as expected (half a second, after the merging stage mentioned in Subsection 4.1), but is a bit larger (0.75 sec). Perhaps, in some way that value could be related to the tolerance used in the automatic evaluation procedure (0.3 sec).

[1]Experiments were performed on a Pentium III 1GHz, 1Gb RAM.

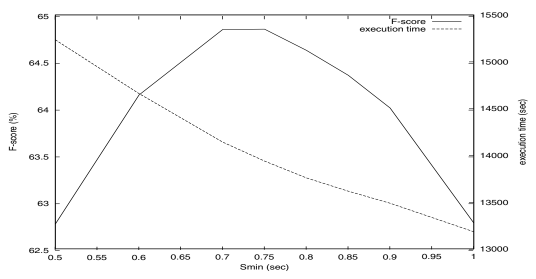

Figure 2: F-score and exec. time as functions of S_{min}.

$\mathbf{K_{max}}$: Also the limit on the number of searched segments starts to affect the accuracy when its value goes under a threshold (about 50). Since the average length of segments in the test set is 1.3 sec, and the audio files lasted about 1 min, the expected number of segments in each file is about 45, which is compatible with the threshold observed in the experiments. This means that K_{max} must be carefully chosen, taking into account the characteristics of audio under processing.

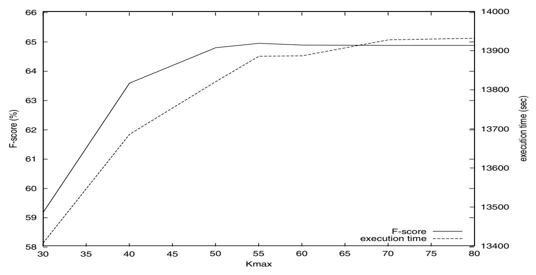

Figure 3: F-score and exec. time as functions of K_{max}.

$\mathbf{S_{max}}$: It can be noted that accuracy does not change if S_{max} is reduced from 60 sec up to 10 sec, halving the execution time. Under 10 sec, the number of insertions increases too much, affecting the overall accuracy. Of course, the nature of the test set (only 0.1% of segments is longer than 10 sec) allows to reduce S_{max} up to 10 sec, but experiments not reported here on a different test set (news programs) show how it is important to introduce some tricks in order to relax, at least in some circumstances, the S_{max} constraint.

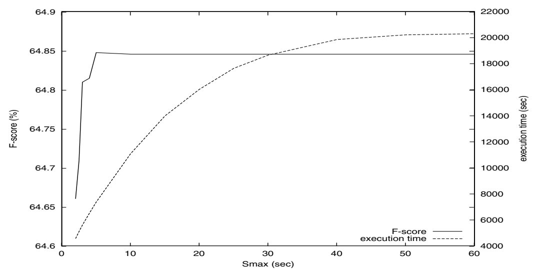

Figure 4: F-score and exec. time as functions of S_{max}.

Resolution δ: The use of a resolution higher (but not too much) than 1 for the processing allows to reduce a lot the amount of time required by the algorithm. Even better, $\delta = 5$ permits a better performance than the maximum resolution, which causes more false alarms.

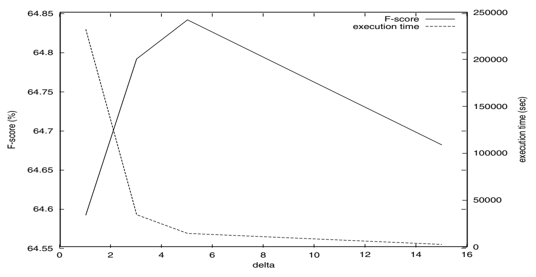

Figure 5: F-score and exec. time as functions of δ.

5. Conclusions

A dynamic programming algorithm which uses the BIC method for globally segmenting the input audio stream has been described, analyzed, and experimentally compared with the widely adopted BIC based algorithm, which locally searches single changes, within a sliding variable-size analysis window.

The slight performance improvement obtained by the global algorithm is lower than expected, while its time cost is definitely higher. Its time requirement was reduced without affecting accuracy, by introducing some reasonable approximations, obtaining a less than real time cost.

Nevertheless, the local algorithm is able to segment an audio stream almost as well as the global algorithm does. Since the DP algorithm can be considered an upper bound for the class of the BIC based segmentation methods, experiments described here confirm that the local algorithm can be the right choice in most cases.

6. References

[1] S. S. Chen and P. S. Gopalakrishnan, "Speaker, environment and channel change detection and clustering via the Bayesian Information Criterion," in *Proc. of the DARPA Broadcast News Transcr. & Underst. Workshop*, Lansdowne, VA, 1998.

[2] M. Harris, X. Aubert, R. Haeb-Umbach, and P. Beyerlein, "A study of broadcast news audio stream segmentation and segment clustering," in *Proc. EUROSPEECH*, Budapest, Hungary, 1999, vol. 3, pp. 1027–1030.

[3] A. Tritschler and R. Gopinath, "Improved speaker segmentation and segments clustering using the Bayesian Information Criterion," in *Proc. EUROSPEECH*, Budapest, Hungary, 1999, vol. 2, pp. 679–682.

[4] M. Cettolo, "Segmentation, classification and clustering of an Italian broadcast news corpus," in *Proc. of the 6th RIAO - Content-Based Multimedia Information Access - conference*, Paris, France, 2000.

[5] P. Sivakumaran, J. Fortuna, and A. M. Ariyaeeinia, "On the use of the Bayesian Information Criterion in multiple speaker detection," in *Proc. EUROSPEECH*, Aalborg, Denmark, 2001, vol. 2, pp. 795–798.

[6] P. Delacourt, D. Kryze, and C.J. Wellekens, "Speaker-based segmentation for audio data indexing," in *Proc. of the ESCA ETRW workshop Accessing Information in Spoken Audio*, Cambridge, UK, 1999.

[7] M. Cettolo and M. Vescovi, "Efficient audio segmentation algorithms based on the BIC," in *Proc. ICASSP*, Hong Kong, 2003.