Classification of Audio Signals

Thomas Kemp
Sony Deutschland GmbH, Hedelfinger Str. 61, 70327 Stuttgart; kemp@sony.de

Introduction

In audio classification and segmentation, unknown audio data is segmented and classified into one of a limited set of predefined classes. Class definitions vary from application to application, however, some typical examples include "music", "male speaker", "female speaker", "laughing", "wind noise", etc. Typical applications include searching in audiovisual material, and aiding the user in editing (cutting) taped material.

Database

Our development database comprised 7 Japanese TV recordings of various types, including three newscasts, two game shows, and two music shows. Because of the limited size and large variability of the material, the split between training and testing was done by cutting each file 10 minutes prior to the end, and using the last segment as test set. Later silence cutting further removed parts of the test material.

A total of 13 predefined classes is used. The entire database is manually labelled, with segment lengths down to 0.5 seconds and shorter in length. There is a total of 9557 segments. The length distribution of the labels is shown in figure 1.

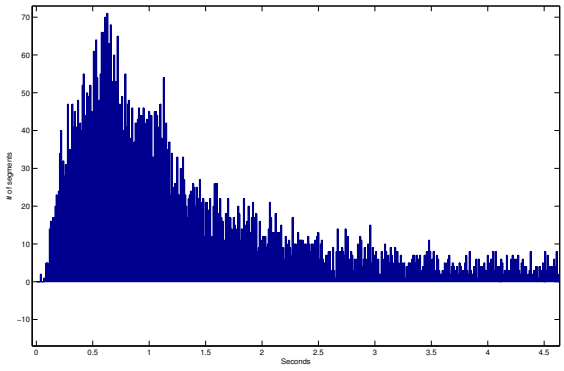

Figure 1: Segment length distribution (transcripts)

Evaluation metrics

The objective of an audio classification system is twofold: classification and segmentation. Therefore, two different metrics have to be used to judge the performance of the system. For classification performance, frame error rate is used; for segmentation, precision (PRC) and recall (RCL) and their combination into F-Measure (F) are computed based on segment boundaries. RCL measures how many segment boundaries are found by the system, and PRC measures how many of the hypothesized segment boundaries were indeed true segment boundaries.

The ideal system achieves a frame error rate of zero and PRC and RCL of both 100%.

Classification

The input data is first cut into overlapping frames of 32 ms length and 10 ms frame shift, hamming windowed, and preemphasized by first order FIR high pass filter $z = 1 - 0.97z^{-1}$. From this, 30 melscale coefficients are computed and concatenated with their first and second order time derivatives. C0 is removed, and the resulting feature vector is transformed with a LDA matrix which was estimated on the 13 training set classes. The classification is achieved by mixture of Gaussian models. The model parameter estimation is done by k-means (in our case, k=64) clustering of the data for every class followed by 20 iterations of E-M (estimation / maximisation) algorithm. Using the Gaussian Mixture Models (GMMs), a probability $p(\mathbf{o}(\mathbf{t})|\omega)$ of the data frames given the classes can be computed for each of the input data frames of the test set. From this probability, the desired probability $p(\omega|\mathbf{o}(\mathbf{t}))$ can be computed using Bayes rule:

$$p(\omega|o(t)) = \frac{p(o(t)|\omega) * p(\omega)}{p(o(t))} \quad (1)$$

We assume no prior knowledge about $p(\omega)$, and the decision for a class ω is not influenced by the probability of the data frames $p(o(t))$. Therefore, the straightforward way to classify the incoming data is to simply compute the probabilities using the 13 GMM models, and chose the most likely class for every frame. Doing so results in a frame error rate of 48.68%, a recall (RCL) of 100% of all segment boundaries, and a precision (PRC) of only 1.42% of the hypothesised segment boundaries being actually correct.

Experiments

Clearly, the result of 48.68% frame error rate is not satisfying. As can be seen from the low value for PRC, the classification result 'jumps' rapidly from frame to frame, producing a huge number of very short segments. We assume that this is because the information content in just 32 ms of audio is not high enough to allow a good classification performance and several frames' worth of information need to be combined to achieve a better accuracy. The easiest way to achieve this is to chop the signal into evaluation windows of predefined length, and compute a classification for each window by combining the individual frame results of that window.

Combining frame results can be done both on the probability level and on the classifier output level. On the probability level, a joint probability for multiple frames can be estimated by multiplying the probabilities for every frame and a given class to get a joint probability for the entire window, and then comparing the joint probabilities for the different classes. On the classifier output level, the combination can be done by majority-voting the outputs of the individual frame classification results inside the classification window. Several experiments with both methods and different lengths of the classification window were performed. The result of this experiments is summarized in table 1.

Table 1: Result of window based evaluation for both majority voting and probability combination

Method	Error	RCL	PRC	F
frame by frame	48.68%	100%	1.42%	2.44%
0.1-s-window, prob.	32.9%	30.2%	73.2%	42.7%
0.1-s-window, maj.	36.9%	36.6%	70.0%	**48.1%**
0.5-s-window, prob.	32.4%	24.7%	83.4%	38.1%
0.5-s-window, maj.	36.4%	28.9%	82.8%	42.8%
1-s-window, prob.	31.9%	22.0%	85.4%	35.0%
1-s-window, maj.	36.1%	24.6%	84.5%	38.1%
2-s-window, prob.	32.1%	17.1%	86.0%	28.5%
2-s-window, maj.	34.9%	18.4%	85.5%	30.2%
3-s-window, prob.	32.1%	15.1%	89.3%	25.8%
3-s-window, maj.	34.9%	14.9%	86.7%	25.5%
4-s-window, prob.	**30.8%**	11.9%	86.8%	20.9%
4-s-window, maj.	33.0%	13.4%	88.6%	23.2%
5-s-window, prob.	32.3%	9.8%	83.2%	17.5%
5-s-window, maj.	34.5%	11.6%	88.1%	20.5%
9-s-window, prob.	33.9%	7.3%	85.6%	13.5%
9-s-window, maj.	35.4%	7.7%	87.0%	14.2%

As can be seen, frame combination helps greatly, reduing error rate from nearly 50% to slightly above 30%. Frame combination by majority voting performs worse in terms of error rate (33% error as compared to 30.8%), but slightly better in segmentation. Segmentation performance, however, is modest for both approaches, with F-measures below 50% regardless of configuration.

Viterbi evaluation

One disadvantage of the frame combination by fixed-sized windows is, that boundaries are imposed by chopping the input into windows of predefined length. This will often lead to analysis windows containing more than one class, causing higher error rates. Therefore, we propose to use the Viterbi algorithm to combine frame results into windows of variable length, governed by the data and not by predefined thresholds.

The Viterbi algorithm [1] can find an optimal state sequence, where states can be associated with classes, given a sequence of observations and a set of state-dependent probability density functions for the observations. In addition to the state dependent probability density functions, it allows to introduce penalties to switch from one state to the other, and there can also be minimum state length constraints (i.e. that a state must be used for at least N frames). A Viterbi evaluation using no minimum state length constraint and no penalty for changing from one state to another, is equivalent to the 'frame by frame' evaluation. A Viterbi evaluation using a minimum state length of 100 states (1 second) forces segment lengths to be at least one second long - i.e. they could be 1.42 s song. In contrast to the 1-second long window evaluation described above, a segment can thus have an arbitrary length (as long as it is longer than one second). This additional degree of freedom should allow to find better segmentations of the input data.
The result of a set of experiments with different values for the minimum state length is summarized in table 2. From this results, it can be seen that Viterbi based evaluation outperforms fixed window based evaluation by a significant margin, particularly in terms of segmentation.

Table 2: Viterbi with different minimum state lengths

Method	Error	RCL	PRC	F
0.01 s min length	48.7%	100%	1.4%	2.44%
0.1 s min length	42.5%	100%	10.3%	18.6%
0.5 s min length.	32.3%	97.0%	48.6%	64.7%
1 s min length	28.8%	76.1%	77.3%	**76.7%**
2 s min length	28.3%	47.6%	87.8%	61.7%
3 s min length	28.7%	32.2%	91.2%	47.5%
4 s min length	**26.5%**	23.1%	90.4%	36.7%
5 s min length	27.2%	20.6%	93.3%	33.7%
9 s min length	35.4%	7.7%	87.0%	14.2%

We ran another set of experiments, where there was no restriction in terms of minimum state duration, but a rather a penalty to switch states. The results are summarized in table 3. This method significantly outperforms the method of using a minimum state length constraint and achieves the lowest error rates on this task (22.2%), less than half of the original unsmoothed frame error of 48.7% .

Table 3: Viterbi with different class transition penalties

Method	Error	RCL	PRC	F
no penalty	48.7%	100%	1.4%	2.44%
penalty 30	28.3%	91.3%	53.4%	67.4%
penalty 60	25.2%	69.2%	79.5%	**74.0%**
penalty 90	24.5%	53.2%	88.1%	66.4%
penalty 120	24.2%	43.7%	92.7%	59.4%
penalty 200	23.8%	28.0%	95.0%	43.2%
penalty 300	**22.2%**	20.6%	93.0%	33.7%
penalty 400	22.6%	17.6%	93.5%	29.7%
penalty 500	23.3%	15.6%	97.4%	26.9%

References

[1] A. J. Viterbi, *Error bounds for convolutional codes and an asymptotically optimal decoding algorithm*, IEEE Transactions on Information Theory, vol. IT-13, S. 260-269, April 1967