CLASS-SPECIFIC MODEL MIXTURES FOR THE CLASSIFICATION OF TIME-SERIES

Paul M Baggenstoss

Naval Undersea Warfare Center
Newport RI, 02841
and Fraunhofer FKIE, Fraunhofer Str 20,
53343 Wachtberg, Germany

ABSTRACT

We present a new classifier for acoustic time-series that involves a mixture of generative models. Each model operates on a feature stream extracted from the time-series using overlapped Hanning-weighted segments and has a probability density function (PDF) modeled with a hidden Markov model (HMM). The models use a variety of segmentation sizes and feature extraction methods, yet can be combined at a higher level using a mixture PDF thanks to the PDF projection theorem (PPT) that converts the feature PDF to raw time-series PDFs. The effectiveness of the method is shown using an open data set of short-duration acoustic signals.

***Index Terms*—** Classification, PDF projection, generative models, kernel methods

1. BACKGROUND

It has been recognized that audio scene classification represents a challenge for feature design [1–4]. If there are a large variety of signal types, it requires a large variety of feature extractors. Existing methods continue to apply a single feature set at a time, a practice that leads to the the dimensionality curse or *feature bottleneck*. We have previously proposed a theoretical approach to use several feature sets simultaneously in a Bayesian framework, without incurring the dimensionality curse [5–7]. These methods are all based on the PDF projection theorem (PPT), which we review below. In this paper, we propose additional innovations including *Hanning-3 segmentation*, class-specific model mixtures with *annealing factor* to arrive at a general purpose classifier for acoustic time-series.

1.1. PDF Projection

PDF projection can circumvent the feature bottleneck to bring more information to bear on the problem without increasing feature dimension [6, 8]. Consider any feature transformation

$$\mathbf{z} = T(\mathbf{x}), \quad \mathbf{x} \in \mathcal{R}^N, \;\; \mathbf{z} \in \mathcal{R}^D, \quad D < N. \tag{1}$$

Consider a canonical statistical reference hypothesis H_0 under which the exact PDF of both $\mathbf{x}$ and $\mathbf{z}$ are known. In this paper, we use the hypothesis $H_0:$ $\mathbf{x}$ consists of independent Gaussian zero mean variance 1 noise (IGN). Define

$$G(\mathbf{x}; T, H_0) = \frac{p(\mathbf{x}|H_0)}{p(\mathbf{z}|H_0)}\; g(\mathbf{z}) = J(\mathbf{x}; T, H_0)\; g(\mathbf{z}), \tag{2}$$

where $g(\mathbf{z})$ is any feature PDF, and $J(\mathbf{x}; T, H_0) = \frac{p(\mathbf{x}|H_0)}{p(\mathbf{z}|H_0)}$ is called the "J-function" because of the analog with the Jacobian for an invertible transformation. The PPT [6] proves that $G(\mathbf{x}; T, H_0)$ is a PDF (integrates to 1 over $\mathbf{x}$) and is a member of the class of PDFs that generate $g(\mathbf{z})$ [1].

The main difficulty in applying the PPT is the need to compute $J(\mathbf{x}; T, H_0)$ precicely. The most difficult part is the evaluation of $p(\mathbf{z}|H_0)$, even in the tails. Most if not all input data samples will be in the far tails of the reference hypothesis, where $p(\mathbf{x}|H_0)$ and $p(\mathbf{z}|H_0)$ are both essentially zero. Luckily, the ratio $p(\mathbf{x}|H_0)/p(\mathbf{z}|H_0)$ can be computed with high precision in the log-domain, often without computing the individual numerator and denominators. For canonical $p(\mathbf{x}|H_0)$ (Gaussian and exponential), $p(\mathbf{z}|H_0)$ has been solved for many important feature transformations [9]. We will see later in Section 1.3 that if the feature transformation can be broken into simple stages, the J-function for complex feature transformations can be computed easily.

In addition to making possible classifiers with mixed feature sets, the PPT can be used to solve two different optimality problems. First, it provides a quantitative means of feature optimization if we have a large set of data samples. Given K samples of data $\mathbf{x}$, the total log-likelihood measure $L(T, H_0) = \sum_{k=1}^{K} \log G(\mathbf{x}_k, T, H_0)$ can be used to select feature transformations T and reference hypotheses H_0 by maximum likelihood. Care must be exercised to separate training and testing data for meaningful results.

Secondly, it has been recently shown that (2) is the maximum entropy PDF that generates $g(\mathbf{z})$. This requires that the feature contain information about the size (norm) of $\mathbf{x}$ [10]. This means that $G(\mathbf{x})$ represents the best expression of the knowledge of $\mathbf{x}$ given the feature density $g(\mathbf{z})$.

[1] The PDF $G(\mathbf{x})$ is said to *generate* feature PDF $g(\mathbf{z})$ if random samples of $G(\mathbf{x})$ passed through the feature transformation $\mathbf{z} = T(\mathbf{x})$ have exactly distribution $g(\mathbf{z})$.

978-0-9928626-3-3/15/$31.00 ©2015 IEEE

1.2. Using the PPT in a classifier

Let $\mathbf{z}_k = T_k(\mathbf{x})$ be class-specific feature transformations. If we use (2) $g(\mathbf{z}) = p(\mathbf{z}_k|H_k)$, we may form the Bayesian classifier

$$\hat{k} = \arg\max_k \left\{ J_k(\mathbf{x}; T_k, H_{0,k})\ p(\mathbf{z}_k|H_k)\ p(H_k) \right\}, \tag{3}$$

where $H_{0,k}$ are class-specific reference hypotheses. This Bayesian classifier is defined on the *raw data* using different feature sets for each class hypothesis.

The main problem with the class-specific features formulation is the one-class/one-feature assumption. Real data from a given class exhibits diversity that can span several feature sets. This problem is solved by the class-specific feature mixture (CSFM) classifier

$$\hat{k} = \arg\max_k \left\{ \sum_{l=1}^{L} \alpha_{k,l}\ J_l(\mathbf{x}; T_l, H_{0,l})\ p(\mathbf{z}_l|H_{k,l}) \right\}\ p(H_k). \tag{4}$$

The PDF for each class is a kernel mixture made from the projected PDFs of a fixed library of L feature transformations.

1.3. The Chain Rule

We already discussed the potential difficulty in computing $J(\mathbf{x})$. The Chain-rule [6] makes this task for multi-stage feature extraction much easier. Consider the three-stage transformation $\mathbf{y} = T_y(\mathbf{x}), \quad \mathbf{w} = T_w(\mathbf{y}), \quad \mathbf{z}_k = T_z(\mathbf{w})$. This suggests the chain-rule form of (2),

$$G(\mathbf{x}) = \left[\frac{p(\mathbf{x}|H_{0x})}{p(\mathbf{y}|H_{0x})}\right] \left[\frac{p(\mathbf{y}|H_{0y})}{p(\mathbf{w}|H_{0y})}\right] \left[\frac{p(\mathbf{w}|H_{0w})}{p(\mathbf{z}|H_{0w})}\right] g(\mathbf{z}), \tag{5}$$

where H_{0x}, H_{0y}, H_{0w} are stage-dependent hypotheses.

The use of stage-dependent hypotheses is extremely useful. To see why, consider what would happen if we used a common reference hypothesis throughout the chain. As we go down the chain, the assumption that $\mathbf{x}$ has PDF $p(\mathbf{x}|H_0)$ would result in feature distributions that were progressively more difficult to derive. At some point, it could be impossible to come up with an exact expression for the distribution of the final feature $p(\mathbf{z}|H_0)$. With stage-specific reference hypotheses, we can get a "fresh start" at the input of each stage, assuming a simple canonical form for the distribution at the input of the stage such as Gaussian or exponential distributions, then needing only to derive (or look up) the feature distribution at the output of each stage.

2. TECHNICAL APPROACH

2.1. Classifier Architecture

We base our classifier on CSFM. Figure 1 shows the means of implementing (4) for a single class k. In each of the L branches, we used hanning-3 segmentation (see Sec. 2.2) followed by feature extraction (see Sec. 2.3). The L feature likelihood functions in branch l, $\log p(\mathbf{Z}^l|H_{k,l})$, are computed using an HMM. Each branch uses a potentially different data segmentation size, feature type and dimension. The HMM mixture at the output implements CSFM using (4) or (6). We used flat model weights ($\alpha_{k,l} = 1/L$).

2.2. Data Segmentation

Let the input data $\mathbf{x}$ be broken into a sequence of time segments, which are individually passed through a feature extraction processor, producing a stream of features $\mathbf{Z}$. The feature transformation $\mathbf{Z} = T(\mathbf{x})$ encompasses both the data segmentation and feature extraction. In the past, we have used only block segmentation with class-specific models because it results in independent segments and simplifies the calculation of $p(\mathbf{x}|H_0)$ and $p(\mathbf{z}|H_0)$ [8]. We have recently discovered a means to circumvent the requirement of independent segments [11]. Consider overlapped segments with segment size K and window time shift S, overlapping by $O = K - S$ samples. If we circularly-index the data such that $x_{N+i} = x_i$, we will obtain exactly $T = N/S$ segments. Let $\mathbf{x}_i = [x_{1+Si}w_1, x_{2+Si}w_2, \ldots x_{K+Si}w_K]$, be the i-th segment where w_i are the Hanning weights[2]. Let the complete *hanning-3* segmentation be denoted by $\mathbf{X}^{2/3} = [\mathbf{x}_1, \mathbf{x}_2 \ldots \mathbf{x}_T]$. Note that $\mathbf{X}^{2/3}$ is $K \times T = (3S) \times (N/S)$, so has a total dimension of $3N$.

To use various *hanning-3* segmentations together in a class-specific classifier, we need to apply the concept of *virtual input data.* Consider two hanning-3 segmentations with different segment sizes K_l and K_m, denoted by $\mathbf{X}^{l,2/3}$ and $\mathbf{X}^{m,2/3}$. It has been shown [11] that with weights w_i as defined in [11], that $\mathbf{X}^{l,2/3}$ and $\mathbf{X}^{m,2/3}$ are related by an orthogonal linear transformation. Specifically, there exists an ortho-normal matrix $\mathbf{U}$ such that $\mathbf{X}^{l,2/3} = \mathbf{U}\ \mathbf{X}^{m,2/3}$. In Figure 1, the output of each segmentation operation is considered as the "virtual input data" of each branch. Each branch has a different virtual input data, but they are considered "equivalent". Therefore, the projected likelihood function for $\mathbf{X}^{l,2/3}$ may be compared to the projected the likelihood function for $\mathbf{X}^{m,2/3}$.

2.3. Features and J-function

We used both auto-regressive (AR) and cepstrum features in the experiments. AR features, widely used in speech and time-series analysis [12], start with discrete Fourier transform (DFT), then magnitude squared, inverse DFT to compute the auto-correlation function (ACF), Levinson algorithm to compute the reflection coefficients (RC) and innovation variance, and log-bilinear transformation to provide an approximate

[2] these must be periodic with a period of N, not $N-1$ as is typically used to avoid any zero weights

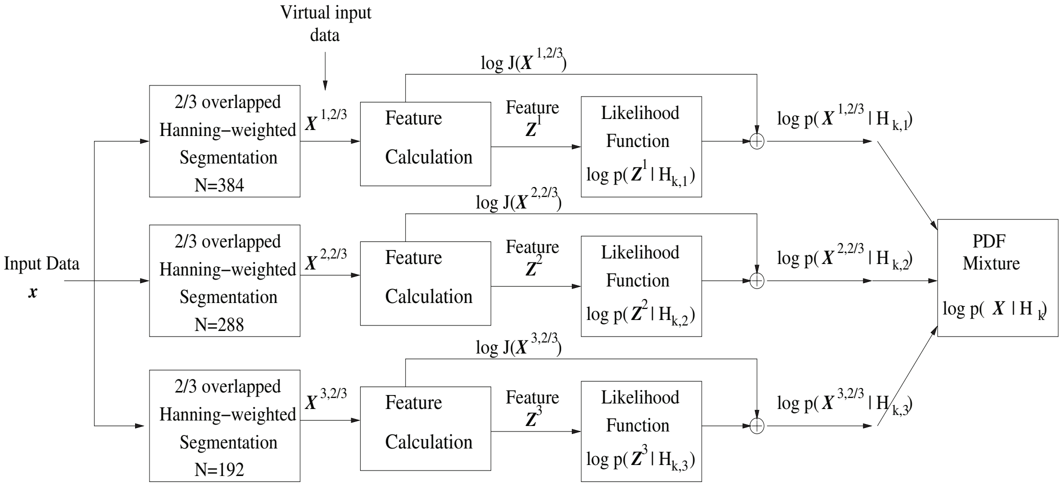

Fig. 1. Architecture of HMM model mixture for class k (mixture of 3 models).

Gaussian distribution for the RC. This process including computation of the J-function is described in ([8] Section VIII).

The Cepstral features were inspired by the MEL frequency cepstral coefficients (MFCC) widely used in speech analysis [13]. In addition to MEL band spacing, we used linear band spacing (LFCC). For LFCC and MFCC features, we compute the inner product of the raw spectral vector and each band function. The discrete cosine transform (DCT) of the log of the band energies is then computed. In contrast to the usual approach in speech analysis, the DCT output is not truncated, so the final feature dimension is equal to the number of band functions. We used zero and Nyquist bands in order that the total spectral energy is computed. The J-function calculation for LFCC/MFCC is the same as the AR features ([8] Section VIII) , except the inner product of the raw spectrum with the MEL band functions replaces the cosine functions needed to compute the ACF. The log and DCT are 1:1 transformations, so require only Jacobian analysis.

3. RESULTS - OFFICE SOUNDS DATA

3.1. Data description

The Office Sounds database [14] contains twenty-four signal classes of 102 samples each, a total of 2448 example sounds, mostly created by dropping common objects or operating office tools such as scissors or staplers. All time-series are 16128 samples long (1/2 second in duration at 32000 Hz). Due to the fixed data length, the data base is ideal for comparing generative classifiers with general-purpose discriminative classifiers. The sounds are consistently generated, thus separable, but having many similar characteristics.

3.2. Benchmark Performance: SVM

For a performance benchmark, we applied the support vector machine (SVM) classifier “SVM-Light” toolkit [15]. Several feature extraction methods were tried including spectrogram and MFCC features, but the best SVM performance was obtained with a straight DFT and principal component analysis (PCA). The full time-series was transformed directly by the DFT, then the log of the magnitude of each bin was calculated. All the training set features were gathered and PCA analysis was done, keeping the top 128 principal vectors. For classification, the features were projected onto the 128-dimensional basis, producing a 128-dimensional feature that was provided to the SVM using linear kernel. Twenty-four one-against-all SVM classifiers were trained, then samples were classified by choosing the model with the highest output. Classification error was 3.15% (77/2448) at 2:1 holdout.

3.3. Individual Models

We used a total of 23 models consisting of 16 Cepstrum (8 LFCC and 8 MFCC) and 7 AR models (See section 2.3). The parameters for Cepstrum features and the model order for AR features as a function of segment size is shown in Table 1.

It is instructive to compare the various models individually, both in terms of classification performance, and in terms of total projected likelihood $L(T, H_0)$. Total projected likelihood was computed individually for each class using 2:1 holdout, then averaged over the 24 classes. We plotted this value as a function of the model index for 23 models. We also measured individual classification performance on the 23 models. Figure 2 shows the result using 2:1 holdout. There is an approximate inverse relationship between classification error and total projected log-likelihood, demonstrating the feasibility of feature selection based on projected likeli-

	Cepstrum			AR	
		LFCC	MFCC		
K	Bands	model	model	order	model
72	6	1	9	3	17
96	8	2	10	4	18
144	12	3	11	6	19
192	16	4	12	7	20
288	24	5	13	11	21
384	32	6	14	16	22
576	48	7	15	20	23
768	64	8	16		

Table 1. Model parameters for Cepstrum and AR models. Model numbers correspond with model numbers used in Figure 2 and 3

hood. This method is unique and has no comparable known technique in the classification and machine learning literature where it is generally accepted that likelihood comparison between different features is meaningless.

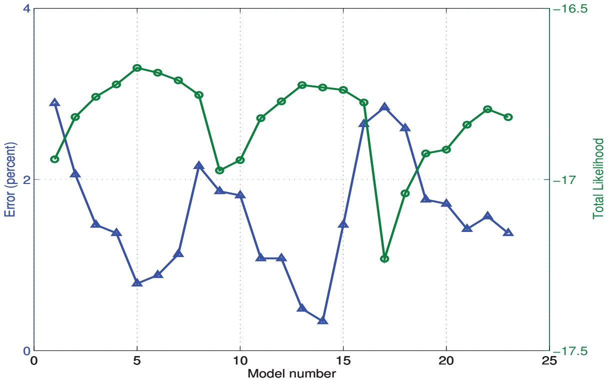

Fig. 2. Classification error probability (triangles) and total likelihood (circles) for 23 models: 8 LFCC models, 8 MFCC models, and 7 AR models, 2:1 holdout. Minimum error is 0.34% for model 14 which is MFCC at $K = 384$.

3.4. Classification results: model mixture

Up to now we have tested feature sets individually for classification error, using individual branches in Figure 1. The CEPSTRUM features with Hanning-3 processing achieves an impressive 0.34% error at 2:1 holdout. We would like to see, however, if combining multiple features achieves even lower error.

To promote better model mixing, we used a method related to simulated annealing (SA), which is used in statistical modeling and optimization [16]. The annealing factor C applied to (4) produces the classifier

$$\hat{k} = \arg\max_{k=1}^{M} \left\{ \sum_{l=1}^{L} \alpha_{k,l} \; \left[J_l(\mathbf{X}) \; p(\mathbf{Z}_l | H_{k,l}) \right]^{1/C} \right\} \; p(H_k), \tag{6}$$

In SA, the factor C is gradually reduced as the model is trained, however, we use a fixed C. Equation (6) is also a form of alpha-integration [17]. Large C equates to a geometric PDF mixture, while small C selects the largest model (infinity norm), and $C = 1$ gives linear mixing. We determined the error performance of (6) as a function of the log-annealing factor l, where $C = Ne^l$, where N is the number of samples in the time-series. This was done so that the optimal value of l would be roughly independent of N.

The error rate as a function of annealing factor l is shown in Figure 3 for 2:1 holdout. The figure shows the performance for a mixture of all 23 models and a mixture with a selected 4 models. Both curves show best perfoamance around $l = -3$. Clearly, selecting a subset of the models can work better than using all of them. Although using all models was not as good as the single best model (0.34%), it does suggest a very good general-purpose classifier if the best model is not known. The performance of the selected 4 models was better than the best individual model, achieving 0.15% error (99.85% correct), or just 3 errors in 2448 opportunities. The fact that selected models works better suggests that unequal mixing weights may also achieve better performance (we used flat mixing weights in the experiments, $\alpha_{k,l} = 1/L$).

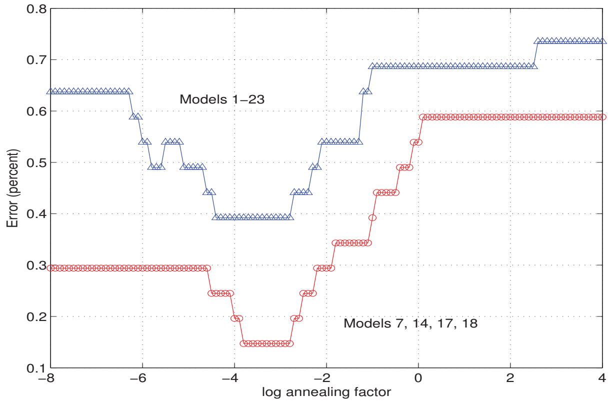

Fig. 3. Classification error as a function of annealing factor l for 16 Cepstrum models (top) and selected models 7,14,17,18 (bottom).

3.5. Discussion of Results

In Figures 2 and 3, we see some important results. First, because the highest likelihood generally coincides with minimum error, figure 2 implies that using the PPT, the model order (number of band functions) may be selected based on individual models, not by classification performance, as is typically done. This makes a classifier more generalizable since its design is less dependent on other classes. Figure 3 shows that the PPT allows combining generative classifiers in a new way. On the right side of the graph, (high l, and therefore high C), the performance approaches that of log-likelihood addition, or a geometric mixture of the models. This is the equivalent to adding or *stacking* models, as is often done in practice. The problem with stacking is that all models have equal influence regardless of how well a model represents a given data sample. This can be thought of as over-mixing. On the left (low C), the performance approaches the classifier that selects the model with the highest likelihood, what be thought of as under-mixing because only the one model has an influence on the likelihood function. The center of the graph is the region of good model mixing, showing a resultant better performance.

4. CONCLUSIONS AND FUTURE WORK

We presented a classification scheme using a mixture of feature models based on PDF projection. Each feature model can have a different feature type, model order, and/or segmentation size. The mixture relies on the PDF projection theorem (PPT) to convert feature PDFs to raw-data PDFs. Likelihood annealing was used to promote better mixing of the models, with optimal performance at medium annealing factor. We demonstrated the method using autoregressive (AR) and cepstrum features having both MEL (MFCC) and linear (LFCC) band spacing. On the Office Sounds Data, using a mixture of 23 different models, the method had a minimum error at moderate annealing factor that was comparable but not better than the best individual model. With a set of four selected models, the mixture performed significantly better than the best single model. The performance of our classifier was significantly better (a factor of between 7 and 20 times better) than the best performance obtained using a support vector machine (SVM).

REFERENCES

[1] Z. Liu, J. Huang, W. Yao, and C. Tsuhan, "Audio feature extraction and analysis for scene classification," in *First IEEE Workshop on Multimedia Signal Processing*, Princeton, NJ, June 1997, pp. 343–348.

[2] Z. Liu, W. Yao, and C. Tsuhan, "Audio feature extraction and analysis for scene segmentation and classification," in *Journal of VLSI Signal Processing Systems, special issue on multimedia signal processing*, 1998, pp. 61–79.

[3] H. Asefi, B. Ghoraani, A. Ye, and S. Krishnan, "Audio scene analysis using parametric signal features," in *Proceedings of the 24th Canadian Conference on Electrical and Computer Engineering (CCECE)*, Niagra Falls, ON, May 2011, pp. 922–925.

[4] J. Geiger, B. Schuller, and G. Rigoll, "Large-scale audio feature extraction and SVM for acoustic scene classification," in *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, Oct. 2013.

[5] P. M. Baggenstoss, "A modified Baum-Welch algorithm for hidden Markov models with multiple observation spaces.," *IEEE Transactions on Speech and Audio*, pp. 411–416, May 2001.

[6] Paul M. Baggenstoss, "The PDF projection theorem and the class-specific method," *IEEE Trans Signal Processing*, pp. 672–685, March 2003.

[7] P. M. Baggenstoss, "A multi-resolution hidden markov model using class-specific features," *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5165–5177, Oct 2010.

[8] P. M. Baggenstoss, "The class-specific classifier: Avoiding the curse of dimensionality (tutorial)," *IEEE Aerospace and Electronic Systems Magazine, special Tutorial addendum*, vol. 19, no. 1, pp. 37–52, January 2004.

[9] Steven M. Kay, Albert H. Nuttall, and Paul M. Baggenstoss, "Multidimensional probability density function approximation for detection, classification and model order selection," *IEEE Transactions on Signal Processing*, pp. 2240–2252, Oct 2001.

[10] P. M. Baggenstoss, "Maximum entropy PDF design using feature density constraints: Applications in signal processing (in review)," *IEEE Transactions on Signal Processing*, 2015.

[11] P. M. Baggenstoss, "On the equivalence of hanning-weighted and overlapped analysis windows using different window sizes," *IEEE Signal Processing Letters*, vol. 19, no. 1, pp. 27–30, Jan 2012.

[12] S. Kay, *Modern Spectral Estimation: Theory and Applications*, Prentice Hall, 1988.

[13] P. Mermelstein, "Distance measures for speech recognition, psychological and instrumental," *Pattern Recognition and Artificial Intelligence*, p. 374388, 1976.

[14] P. Baggenstoss, "Office sounds database," *http://class-specific.com/os*.

[15] Thorsten Joachims, "Svm-light toolkit," 2014.

[16] L. Ingber, "Simulated annealing: Practice versus theory," *Mathematical and Computer Modeling*, vol. 16, no. 11, pp. 29–57, 1993.

[17] S. Amari, "Integration of stochastic models by minimizing alpha-divergence," *Neural computing*, vol. 19, no. 10, pp. 1585–1604, Oct 2007.