AUTOMATIC STATE DISCOVERY FOR UNSTRUCTURED AUDIO SCENE CLASSIFICATION

Julian Ramos, Sajid Siddiqi, Artur Dubrawski, Geoffrey Gordon

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213 USA

Abhishek Sharma

MobileFusion, Inc.
Pittsburgh, PA 15203 USA

ABSTRACT

In this paper we present a novel scheme for unstructured audio scene classification that possesses three highly desirable and powerful features: *autonomy*, *scalability*, and *robustness*. Our scheme is based on our recently introduced machine learning algorithm called Simultaneous Temporal And Contextual Splitting (STACS) that discovers the appropriate number of states and efficiently learns accurate Hidden Markov Model (HMM) parameters for the given data. STACS-based algorithms train HMMs up to five times faster than Baum-Welch, avoid the overfitting problem commonly encountered in learning large state-space HMMs using Expectation Maximization (EM) methods such as Baum-Welch, and achieve superior classification results on a very diverse dataset with minimal pre-processing. Furthermore, our scheme has proven to be highly effective for building real-world applications and has been integrated into a commercial surveillance system as an event detection component.

***Index Terms*—** Hidden Markov Models, audio classification, topology learning

1. INTRODUCTION

Classification of unstructured audio scene data has a variety of applications such as: *ethnomusicology*, i.e., music classification based on cultural style [1]; *audio diarization*, i.e., extraction of speech segments in long audio signals from background sounds [2]; *audio event detection* [3] for audio mining; *acoustic surveillance* [4], especially for military and public safety applications, e.g, in urban search and rescue scenarios; and *human robot interaction* [5], for voice activated robot actuation and control.

For all these applications the fundamental task is to accurately classify temporal audio signals of interest despite background noise. One very popular and effective classification scheme [6] is based on Hidden Markov Models (HMMs). These algorithms can capture the underlying *hidden* distributions or *states* in the time-series audio data and also discover the transitions in these states to learn dynamical models of the complex underlying phenomenon and can be learned by using iterative parameter learning approaches such as Baum-Welch [7].

While EM-based HMM learning approaches have had some success in learning HMMs of predetermined size and topology, the *discovery* of an appropriate number of states (and their parameters) has largely been an unsolved problem. This problem is typically overcome heuristically in practice, e.g. by choosing the number of states manually and then performing EM with random restarts.

To overcome these fundamental challenges with HMM learning, we improved upon previous HMM topology learning efforts such as [8, 9, 10] to create an *automatic* state discovery algorithm called *Simultaneous Temporal and Contextual Splitting* (STACS) [11] and a more efficient variant called Viterbi-STACS (V-STACS). Traditional HMM learning methodology decouples topology selection from parameter search by requiring the number of states to be specified beforehand. Even if topology search is addressed by the learning algorithm, it is done without fully considering temporal correlations in the data. In contrast, STACS state discovery algorithms reformulate the search space of HMMs by interlacing *parameter-optimization steps* to increase observed data likelihood with *state-space-expansion steps* that consider both temporal and geometric properties of the training data to change the HMM topology in order to escape local minima and fit the data better. Overfitting of the state space size is avoided by using the *Bayesian Information Criterion* (BIC) [12] at each state-space-expansion step. Since state discovery, transition topology design and parameter optimization are all carried out in a data-driven manner, we find that our algorithm requires minimal data preprocessing or pre-partitioning, and can be run on simple Mel-cepstral coefficients extracted from audio scene data with good results. The overall algorithm as well as the state-addition method are described in more detail in Section 2. The biggest difference between STACS-based algorithms and previous

SS is currently at Google Pittsburgh. GG was supported by the ONR MURI grant N00014-09-1-1052. JR, SS and AD were partially supported by funding from The Technology Collaborative (TTC) via a project with MobileFusion, Inc.

state-splitting methods is that they jointly model transitions as well as observations when performing a split, whereas previous methods were unable to do both simultaneously. This makes our algorithms much more accurate. We also consider fewer datapoints per split to greatly increase efficiency, while ensuring that the most relevant datapoints are considered.

The key contributions of this paper are the successful application of our learning method to audio classification from unstructured data, as well as the introduction of STACS to audio-processing applications. We applied our scheme to automatically discover states and learn separate HMMs for six different classes of data consisting of audio from human speech, animal sounds, ground vehicle operation, aerial vehicle flight, explosions and miscellaneous background noise. We also conducted experiments with these models by integrating them with a real-time surveillance system (http://www.mobilefusioninc.com).

2. PARAMETER AND TOPOLOGY LEARNING

Topology selection and parameter optimization are typically decoupled in audio processing applications. The state space size is selected heuristically or with cross-validation, and transition topology is hand-designed based on domain knowledge. For domains where the data is of a predictable form (e.g. human speech in a particular language obtained using a telephone or microphone from close range in a quiet environment), this can often work. If the diversity in the data is from known sources (e.g. voice differences due to speaker, gender and age variations) it can be compensated by gathering more training data. However, this approach is unsuitable for domains such as audio scene classification, where the data source and its environment are unstructured and uncontrolled.

STACS is based on the insight that EM for sequential data is more robust to local minima for small state spaces (e.g. two states) but less so for large state spaces. STACS uses this insight to use a constrained two-state EM algorithm to break out of local minima and increase state space size. STACS algorithms perform parameter learning for Continuous-Density HMMs (CD-HMMs) with Gaussian observation models, while simultaneously optimizing state space size and transition topology in a data-driven fashion. Comparing to previous related work: Unlike [10], our method accounts for both temporal and contextual variations in the training observations while splitting states (i.e. increasing the state space size). Unlike [9], our method evaluates every existing state as a possible candidate for splitting. Since naively evaluating all possible ways to increase state space size would be very expensive computationally, STACS algorithms make selective use of *Viterbi approximations* [8] for efficient approximation of the data log-likelihood, as well as some other assumptions detailed below. These approximations keep the complexity of each iteration of STACS and V-STACS to be $\mathcal{O}(TN^2)$, where T is the length of the training sequence and N is the number of states currently in the HMM. [11] showed that STACS outperforms alternative topology learning methods [10, 9] in terms of running time, test-set likelihood and BIC score on a variety of real-world data, and outperforms regular EM even on learning models of predetermined size. This top-down approach proved to be highly effective at avoiding local minima as well, allowing STACS to discover the true underlying HMM in difficult synthetic data where EM failed to find the correct answer even with 50 restarts. This highlights another problem with using cross-validation for determining the number of states: due to the local minima problems rife in EM-based HMM learning, there is no way to ensure during cross-validation that the best possible HMMs of different sizes are being compared. STACS avoids this problem by guaranteeing a consistent increase in data likelihood as it searches the space of HMMs of varying state space sizes.

We summarize details of STACS and V-STACS below. V-STACS is a less precise, more efficient variant of STACS that uses *Viterbi Training* [7] instead of soft-updates Baum-Welch for the parameter optimization step. Details can be found in [11].

For HMMs we follow the notation of Rabiner [7]. $O = [o_1 \dots o_T]$ denotes the training sequence of length T, $Q = [q_1 \dots q_T]$ denotes the corresponding hidden state sequence, $\lambda = [A, B, \pi]$ denotes the parameters of an N-state Gaussian HMM where $A = \{a_{ij}\}_{i,j=1}^{N}$ is the transition matrix, B is a set of N mean vectors μ and covariance matrices Σ, and π is the prior. Let λ_s denote the HMM parameters related to state s (i.e. $\mu_s, \Sigma_s, \pi_s, a_{\cdot s}, a_{s \cdot}$). Let $T^{(s)}$ denote the timesteps assigned to state s by the Viterbi path. Let Q_U denote the subset of the Viterbi path belonging to timesteps in set U, and O_U denote observations in U. $\backslash U$ denotes the complement of U.

2.1. The Algorithm

STACS and V-STACS both have the following overall procedure. For each step that is not constant-time, we list the asymptotic complexities as $[\cdot]$ or as $[\cdot, \cdot]$ respectively if they differ. Details on candidate generation and candidate selection are given below.

1. Initialization: Initialize λ to a single-state HMM ($N = 1$) based on O. $[\mathcal{O}(T)]$
2. Learning: Use Baum-Welch or Viterbi Training until convergence of $P(O|\lambda)$. $[\mathcal{O}(TN^2)]$
3. Candidate Generation: Split each state, to generate N candidate HMMs each with $N + 1$ states. $[\mathcal{O}(TN^2), \mathcal{O}(TN)]$
4. Candidate Selection: Score the original HMM and each candidate, and pick the highest scoring one as λ'. $[\mathcal{O}(TN^2), \mathcal{O}(TN)]$
5. Repeat or Terminate: If a split candidate was picked, $\lambda \leftarrow \lambda', N \leftarrow N + 1$ and go to step 2. Else if original HMM was picked, terminate and return λ'.

2.2. Generating Candidates

To generate a candidate based on state s resulting in new states s_1 and s_2, we use two variants of EM to efficiently

compute optimal means, variances and transition parameters of the resulting 2 states. We first perform Viterbi to find the most probable state sequence Q^* by maximizing $P(O, Q|\lambda)$. We then constrain the parameters for all other states ($\lambda_{\setminus s}$) and assume all timesteps belonging to other states in Q^* are associated with those states exclusively. Then, we perform *Split-State Baum-Welch* (for STACS) or *Split-State Viterbi Training* (for V-STACS) to optimize λ_{s_1,s_2} on the timesteps associated with state s i.e. $O_{T^{(s)}}$. This optimizes a *partially observed likelihood* $P(O, Q^*_{\setminus T^{(s)}}|\lambda)$. The update equations for Split-State Viterbi Training are as follows, for each iteration i:

1. E-step:
   $Q^i_{T^{(s)}} \leftarrow \arg\max_Q P(O_{T^{(s)}}, Q^*_{\setminus T^{(s)}}, Q \mid \lambda_{\setminus s}, \lambda^i_{s_1,s_2})$
2. M-step:
   $\lambda^{i+1}_{s_1,s_2} \leftarrow \arg\max_\lambda P(O_{T^{(s)}}, Q^*_{\setminus T^{(s)}}, Q^i_{T^{(s)}} \mid \lambda_{\setminus s}, \lambda)$

The M-step is carried out using simple closed-form updates as in the M-step of Viterbi Training. The E-step is carried out using *Split-State Viterbi*, a variant of the Viterbi algorithm whose pseudocode is given in [11]. Note that candidate generation is highly efficient: Split-State Viterbi Training is $\mathcal{O}(|T^{(s)}|)$ and Split-State Baum-Welch is $\mathcal{O}(|T^{(s)}|N)$ where $|T^{(s)}|$ is $\mathcal{O}(T/N)$ on average. There are N candidates to be generated, totaling $\mathcal{O}(TN)$ and $\mathcal{O}(TN^2)$ respectively.

2.3. Candidate Selection Criteria

The candidates are compared amongst each other using the fast-to-compute Viterbi likelihood after optimizing the split parameters. This best candidate is then compared to the original model using *Bayesian Information Criterion* (BIC) [12] which is a unimodal approximation of the true posterior probability, which is itself intractable to compute and where the approximation is assumed to be a single gaussian. The BIC score effectively punishes complexity by penalizing the number of free parameters and rewards goodness-of-fit via the data loglikelihood, thus safeguarding against overfitting. Let $\#\lambda$ denote the number of free parameters in HMM λ. Then,

$$\mathrm{BIC}(\lambda, O) = \log P(O|\lambda) - (\log T/2)\#\lambda$$

For V-STACS, the likelihood is approximated using the *Viterbi approximation* [8] to keep the complexity of V-STACS' candidate generation and splitting at $\mathcal{O}(TN)$.

3. AUDIO SCENE CLASSIFICATION DATA

We applied minimal preprocessing of the available raw sound recordings in order to prepare data for audio scene experiments. It involved extraction of a set of 13 Mel-frequency cepstral coefficients followed by manually assigning ground-truth class labels to each of the examples. The cepstral features were then averaged every 5 subsequent datapoints to reduce noise. Each timestep in the resulting data represents 50ms. The resulting values were scaled 10-fold to avoid numerical underflow issues. The target platform for this algorithm in practice is the portable sensor device shown in Figure 1(A). The audio examples were collected using the device

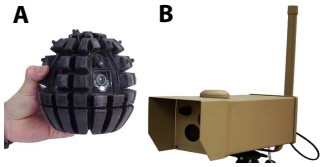

Fig. 1. (A) A mobile tactical device and (B) a fixed device on which our algorithms were deployed

in Figure 1(B), and were selected with the intent to cover a wide variety of sounds representative of the respective classes of audio events. For instance, the examples produced to support Human class models included instances of male, female and children voices speaking in various languages and environments including offices, coffee shops and urban outdoors. The Animal class examples included sounds of dogs, owls, wolves, coyotes, roosters and birds. The class of Ground Vehicles included sample sounds of motorcycles, cars and trucks passing by. The Aerial Vehicles included sounds of jets and helicopters. The Explosions class included sounds of machine-gun fire, grenade blasts and gunshots. The Background class covers other sounds such as ocean waves, city traffic, rain, thunder, crowds plus some silence samples. Details of the data are given in Table 1.

Table 1. Data Specifications. The last row shows the total number of samples, overall average number of timesteps per sample, and total duration in minutes and seconds.

Class	%	# samples	Av. len. (T)	Duration
Human	27	128	209	40:11
Animal	17	78	37	4:58
Ground V.	17	79	47	6:57
Aerial V.	4	20	54	2:16
Explosion	18	87	11	1:34
Backgrnd.	17	83	21	12:58
	100%	475	78	68:54

4. RESULTS

The classifier was tested using 4-fold cross-validation on the dataset described above. Figure 2 and Table 2 summarize the results. We trained HMMs on the data using STACS, V-STACS and EM (Baum-Welch). We used the number of states discovered by V-STACS for EM, which were 56, 26, 26, 18, 14 and 34 on average for the 6 classes respectively. To help EM escape local minima, we re-initialized it 5 times from different random starting points. Training was carried out on a 1.8GHz CPU with 4GB RAM. In Figure 2(A), the training times are plotted *in log scale* of minutes, since the disparity between EM and our algorithms was so large. For example, for the Background class, V-STACS took 166 minutes, STACS took 243 minutes, and EM took 1023 minutes, despite having to only learn parameters for a fixed HMM size.

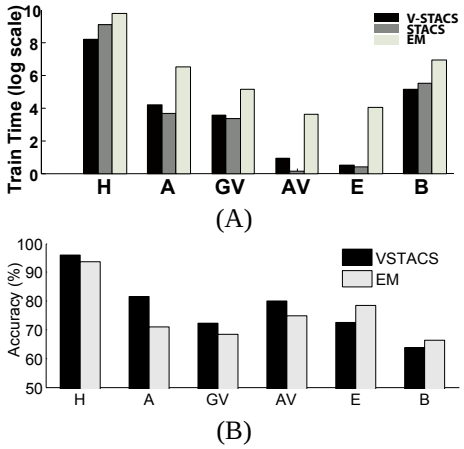

Fig. 2. (A) Training times *in log(minutes)*. V-STACS and STACS are at least an order of magnitude faster than 5-restart EM. (B) Classification accuracies. VSTACS is better than EM in nearly all cases.

We now look at classification accuracy. For brevity, we focus on the classification accuracy results of V-STACS versus EM. STACS results were similar to V-STACS though slightly poorer in some cases (a phenomenon noted and discussed earlier in [11]). Figure 2(B) shows these results for each of the 6 classes, which shows that V-STACS is consistently more accurate than EM, except for labels such as Explosions and Background. We believe this is partly due to lack of sufficient training data (note in Table 1 that these classes had the shortest average sample lengths). Table 2 shows the confusion matrices for models trained using EM (top) and V-STACS (bottom) respectively. Despite the highly unstructured, diverse and in some cases insufficient data, V-STACS is able to learn fairly accurate models.

Table 2. Average Confusion Matrix for EM (top) and V-STACS (bottom). Actual (rows) vs Predicted (columns). Each entry is a percentage of test data averaged over the cross-validation runs, and each row sums to 100. For each entry, the better entry of the two tables is in **bold**. Ties are in *italics*.

	H	A	G.V.	A.V.	E	B
H.	92.69	4.68	**0**	*0*	**0**	2.34
A.	*17.10*	73.68	*0*	1.31	**0**	7.89
G.V.	*9.21*	*3.94*	69.73	2.63	**2.63**	11.84
A.V.	*5*	*0*	15.00	75	**5**	*0*
E.	**9.52**	**1.19**	8.33	**0**	**75**	5.95
B.	15	**8.75**	**5**	**0**	*3.75*	**67.5**
H.	**96.09**	**2.43**	0.78	*0*	0.78	**0**
A.	*17.10*	**81.57**	*0*	**0**	1.31	**0**
G.V.	*9.21*	*3.94*	**72.36**	**0**	3.94	**10.52**
A.V.	*5*	*0*	**5.00**	**80**	10	*0*
E.	14.28	2.38	**4.76**	2.38	72.61	**3.57**
B.	**13.75**	11.25	7.5	**0**	*3.75*	63.75

5. CONCLUSION

We presented a novel, scalable approach to sound classification for interpretation of unstructured audio scenes. It is able to autonomously learn the optimal topology and parameters of the HMM-based classification models from minimally preprocessed training data consisting of small number of diverse sound examples. The learning algorithm converges up to five times as fast as popular alternatives, retaining practical utility even if the composition of target audio scenes is subjected to changes. The results of empirical tests reveal good accuracy when trained on small collection of field data, and low requirements on processing power and working memory. Those characteristics make the proposed approach an attractive solution for practical applications which require compactness and portability. As a result it has been successfully integrated on a commercial surveillance system.

6. REFERENCES

[1] Y. Liu, Q. Xiang, Y. Wang, and L. Cai, "Cultural style based music classification of audio signals," in *ICASSP*, 2009.

[2] D. Rybach, C. Gollam, R. Schluter, and H. Ney, "Audio segmentation for speech recognition using segment features," in *ICASSP*, 2009.

[3] J. Portelo, M. Bugalho, I. Trancoso, J. Neto, A. Abad, and A. Serralheiro, "Non-speech audio event detection," in *ICASSP*, 2009.

[4] S. Ntalampiras, I. Potamitis, and N. Fakotakis, "On acoustic surveillance of hazardous situations," in *ICASSP*, 2009.

[5] H. Okuno, T. Ogata, and K. Komatani, "Computational auditory scene analysis and its application to robot audition," in *ICIR*, 2007.

[6] C. Wooters and M. Huijbregts, *The ICSI RT07s Speaker Diarization System*, Springer-Verlag, 2008.

[7] L. R. Rabiner, "A tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257–285, 1989.

[8] A. Stolcke and S. Omohundro, "Best-first Model Merging for Hidden Markov Model Induction," Technical Report TR-94-003, Intl. Computer Science Institute, 1994.

[9] Cen Li and Gautam Biswas, "Temporal pattern generation using hidden markov model based unsupervised classification," 1999, vol. 1642, pp. 245–256.

[10] M. Ostendorf and H. Singer, "HMM topology design using maximum likelihood successive state splitting," *Computer Speech and Language*, vol. 11, pp. 17–41, 1997.

[11] Sajid Siddiqi, Geoffrey J. Gordon, and Andrew Moore, "Fast state discovery for HMM model selection and learning," in *Proc. AISTATS*, 2007.

[12] G. Schwarz, "Estimating the dimension of a model," *Annals of Statistics*, 1978.