MULTIVARIATE AUTOREGRESSIVE MIXTURE MODELS FOR MUSIC AUTO-TAGGING

Emanuele Coviello University of California, San Diego ecoviell@ucsd.edu

Yonatan Vaizman University of California, San Diego yvaizman@eng.ucsd.edu

Antoni B. Chan City University of Hong Kong abchan@cityu.edu.hk

Gert R.G. Lanckriet University of California, San Diego gert@ece.ucsd.edu

ABSTRACT

We propose the multivariate autoregressive model for content based music auto-tagging. At the song level our approach leverages the multivariate *autoregressive mixture* (*ARM*) model, a generative time-series model for audio, which assumes each feature vector in an audio fragment is a linear function of previous feature vectors. To tackle tag-model estimation, we propose an efficient hierarchical EM algorithm for *ARM*s (HEM-ARM), which summarizes the acoustic information common to the *ARM*s modeling the individual songs associated with a tag. We compare the *ARM* model with the recently proposed dynamic texture mixture (DTM) model. We hence investigate the relative merits of different modeling choices for music time-series: i) the flexibility of selecting higher memory order in *ARM*, ii) the capability of DTM to learn specific frequency basis for each particular tag and iii) the effect of the hidden layer of the DT versus the time efficiency of learning and inference with fully observable *AR* components. Finally, we experiment with a support vector machine (SVM) approach that classifies songs based on a kernel calculated on the frequency responses of the corresponding song *ARM*s. We show that the proposed approach outperforms SVMs trained on a different kernel function, based on a competing generative model.

1. INTRODUCTION

Browsing and discovery of new music can largely benefit from semantic search engines for music, which represent songs within a vocabulary of semantic tags, i.e., words or short phrases describing songs' attributes. By just typing the desired tags as in a standard text search engines (e.g., Bing or Google), users can find the music they desire.

E.C. and Y.V. contributed equally to this work. Y.V. was at ICNC, Hebrew University during this work. E.C., A.B.C. and G.R.G.L. acknowledge support from Google, Inc. E.C. and G.R.G.L. acknowledge support from Qualcomm, Inc, Yahoo!, Inc., the Hellman Fellowship Program, the Sloan Foundation, and NSF Grants CCF-0830535 and IIS-1054960. A.B.C. was supported by the Research Grants Council of the Hong Kong Special Administrative Region, China [9041552 (CityU 110610)]. This research was supported in part by the UCSD FWGrid Project, NSF Research Infrastructure Grant Number EIA-0303622.

Historically, attempts to map songs onto a semantic vocabulary have initially relied on available metadata (e.g. artist names, genre annotation, critical reviews), or manual labeling from expert human annotators and social networks. More recently, distributed human computation games, such as TagATune [15] and HerdIt [2], have attempted to scale up manual labeling to larger collections by recruiting non-expert users through engaging or rewarding games. However, these efforts have so far covered only a small portion of the songs available in modern music collections. [1] This motivates the development of content-based auto-tagging systems, i.e., intelligent algorithms that, by analyzing and understanding the acoustic content of songs, can automatically index them with semantic tags.

1.1 Related work

A large number of content-based auto-taggers are trained on a database of songs annotated with respect to a semantic vocabulary following a common scheme. First, a time series of low-level spectral features (e.g., Mel Frequency Cepstral Coefficients (MFCCs)) is extracted from each song in the database. Then, for each tag, a representative statistical model is fine tuned to capture the most predictive patterns common to the songs annotated with that tag. Once a new song is available, the auto-tagger uses the learned tag-models to process its low-level features and produces a vector of tag-affinities. The tag-affinities are then mapped onto a semantic multinomial (SMN), which represents the song within the semantic vocabulary.

A variety of auto-taggers, based on either generative models [13, 19, 23, 24] or discriminative models [4, 9, 11, 16, 21, 26], rely on a Bag-of-Features (BoF) representation of the spectral content of songs, which ignores temporal dynamics by treating all feature vectors as independent. While augmenting the spectral features with their first and second instantaneous derivatives has represented a common choice to enrich the BoF representation with temporal information (e.g. [3, 23]), more principled solutions have been implemented in recently proposed auto-taggers. The dynamic texture mixture (DTM) treats short fragments of audio as the output of a linear dynamical system [7]. The multi-scale learning algorithm in [12] leverages several pooling functions for summarization of the features over time. The Bag-of-Systems (BoS) approach represents

[1] Pandora annotated catalog and TagATune labeled clips represent less than 5% and 0.15% of the iTunes' collection, respectively.

songs with a codebook of time-series models [10]. The multivariate autoregressive (AR) model was used in [17] in a semi-parametric approach for genre classification of short musical snippets. In [20] various methods for temporal integration, including the AR model, were examined for musical instrument classification.

1.2 Original contribution

In this paper we introduce the autoregressive mixture (ARM) model [1] for automatic music annotation and retrieval. We first model each song as an ARM, estimated from a collection of audio-fragments extracted from the song. Note that this is different from estimating single AR models from *individual* audio clips as done in [17], since each mixture component of a song-level ARM models the music content of *several* (perceptually similar) audio fragments.

In order to model tag-level distributions as ARMs, we propose a novel efficient hierarchical expectation maximization algorithm for ARMs (HEM-ARM). Starting from all the song-ARMs that are relevant for a specific tag, the proposed algorithm summarizes the common music content by clustering similar AR components together, and learning a tag-ARM model with fewer components. We compare our HEM-ARM with previous auto-taggers that used GMMs [23] and DTMs [7], to model tag-level distribution, in tandem with an efficient HEM algorithm for learning. In particular, we obtain that HEM-DTM generally performs better than HEM-ARM (e.g., the annotation F-scores are 0.264, 0.254, respectively). However, relative to HEM-DTM, our HEM-ARM has significantly lower time requirements, both for training (two orders of magnitude) and for annotation (one order of magnitude). These results are explained by the differences in the *graphical* structures of the models. The DT model has an observed layer (which models the spectral content) and a hidden layer (that encodes the temporal dynamics). As a consequence, using DTMs can learn different frequency bases that better adapt to specific tags, but requires marginalization over the hidden variables — and hence delays — at each training iteration and for inference at annotation. On the opposite, the AR is a fully observable model. Hence, training and annotation can be implemented efficiently by computing sufficient statistics for each song a single time.

In addition, once songs are modeled with ARMs, we investigate a kernel-SVM method upon these song-ARMs for semantic retrieval, similar to the work done in [3] over GMMs and in [17] over single ARs. We test several kernel functions, some of which represent each song by the quantized frequency responses (QFR) of its AR components.

The remainder of this paper is organized as follows. In section 2 we present the autoregressive (mixture) model, and in section 3 we derive the hierarchical EM algorithm for ARMs. In Section 4 we present our kernel-SVM approach. In Section 5 we report our experiments.

2. THE AUTOREGRESSIVE MIXTURE MODEL

In this section we present the autoregressive (AR) model and the autoregressive mixture (ARM) model for music time series.

2.1 The AR model

A multivariate autoregressive (AR) model is a generative time-series model for audio fragments. Given a time series of T $d-$dimensional feature vectors $x_{1:T} \in \mathbb{R}^{d\times T}$, the AR model assumes each audio feature x_t at time t is a linear combination of the previous p audio features. Specifically, the AR model is described by the equation

$$x_t = \sum_{j=1}^{p} A_j x_{t-j} + \nu_t \tag{1}$$

where $\{A_j\}_{j=1}^{p}$ are p transition matrices of dimension $d \times d$. ν_t is a driving noise process and is i.i.d. zero-mean Gaussian distributed, i.e., $\nu_t \sim \mathcal{N}(0, Q)$, where $Q \in \mathbb{R}^{d\times d}$ is a covariance matrix. The initial condition is specified by $x_1 \sim \mathcal{N}(\mu, S)$, where $S \in \mathbb{R}^{d\times d}$ is a covariance matrix. We can express (1) in a vectorial form:

$$x_t = \widetilde{A} x_{t-p}^{t-1} + \nu_t \tag{2}$$

where $\widetilde{A} = [A_1 \dots A_p] \in \mathbb{R}^{d\times dp}$ and $x_a^b = [x_b' ... x_a']' \in \mathbb{R}^{dp\times 1}$. Note that, for convenience, we assume $x_t = 0$ for $t \in \{-p+1, \dots, 0\}$, and hence assume that x_1 triggered the generation of the whole time series. An AR model is hence parametrized by $\Theta = \{\mu, S, \widetilde{A}, Q\}$. The likelihood of a sequence $x_{1:T}$ is

$$p(x_{1:T}|\Theta) = \mathcal{N}(x_1|\mu, S) \prod_{t=2}^{T} \mathcal{N}(x_t| \sum_{j=1}^{p} A_j x_{t-j}, Q) \tag{3}$$

where $\mathcal{N}(\cdot|\mu, \Sigma)$ is the pdf of a Gaussian distribution with mean μ and covariance matrix Σ.

The parameters of an AR model can be estimated from a time-series $x_{1:T}$ with various optimization criteria [17,18].

2.2 The ARM model

An ARM model treats a group of audio fragments as samples from K AR models. Specifically, for a given sequence, an assignment variable $z \sim \text{categorical}(\pi_1, \cdots \pi_K)$ selects one of the K AR models, where the i^{th} AR model is selected with probability π_i. Each mixture component is specified by the parameters $\Theta_i = \{\mu^i, S^i, \widetilde{A}^i, Q^i\}$, and the ARM model is specified by $\Theta = \{\pi_i, \Theta_i\}_{i=1}^{K}$. Whereas a single AR model suffices to describe an individual audio fragment, the ARM model is a more appropriate modeling choice for an entire song. This is motivated by the observation that a song usually shows significant structural variations within its duration, and hence multiple AR components are necessary to model the heterogeneous sections.

The likelihood of an audio fragment $x_{1:T}$ under an ARM model is

$$p(x_{1:T}|\Theta) = \sum_{i=1}^{K} \pi_i p(x_{1:T}|z = i, \Theta_i), \tag{4}$$

where the likelihood of $x_{1:T}$ under the i^{th} AR component $p(x_{1:T}|z = i, \Theta_i)$ is given by (3).

The parameters of an ARM model can be estimated from a collection of audio-fragments using the expectation maximization (EM) algorithm [8], which is an iterative procedure that alternates between estimating the assignment variables given the current estimate of the parameters, and re-estimating the parameters based on the estimated assignment variables.

3. THE HEM ALGORITHM FOR ARM MODELS

In this paper we proposed to model tag distributions as ARM models. One way to estimate a tag-level ARM model is to run the EM algorithm directly on all the audio-fragments extracted from the relevant songs. However, this approach would require excessive memory and computation time, to store all the input audio-sequences in RAM and to compute their likelihood at each iteration. In order to avoid this computational bottleneck, we propose a novel hierarchical EM algorithm for ARM models (HEM-ARM), which allows to learn ARM models using an efficient hierarchical estimation procedure. In a first stage, intermediate ARM models are estimated in parallel for each song, using the EM algorithm for ARMs on the song's audio fragments. Then, the HEM-ARM algorithm estimates the final model by summarizing the common information represented in the relevant song-ARMs. This is achieved by aggregating together all the relevant song-ARMs into a single big ARM model, and clustering similar AR models together to form the final tag-level ARM model.

At a high level, the HEM algorithm consists in maximum-likelihood estimation of the ARM tag model from virtual samples distributed according to the song ARM models. However, since using the virtual samples can be approximated with a marginalization over the song ARM distribution (for the law of large numbers, see (8)), the estimation is carried out in an efficient manner that requires only knowledge of the parameters of the song models without the need of generating actual samples. The HEM algorithm was originally proposed by Vasconcelos and Lipmann [25] to reduce a GMM with a large number of mixture components to a compact GMM with fewer components, and extended to DTMs by Chan et al. [5]. The HEM algorithm has been successfully applied to the estimation of GMM tag-distribution [23] and DTM tag-distribution [7]. We now derive the HEM algorithm for ARMs.

3.1 Derivation of the HEM for ARMs

Formally, let $\Theta^{(s)} = \{\pi_i^{(s)}, \Theta_i^{(s)}\}_{i=1}^{K^{(s)}}$ be an ARM model with $K^{(s)}$ components, which pools together the ARM models of all the songs relevant for a tag. The goal of the HEM-ARM algorithm is to learn a tag-level ARM model $\Theta^{(t)} = \{\pi_j^{(t)}, \Theta_j^{(t)}\}_{j=1}^{K^{(t)}}$ with fewer components (i.e., $K^{(t)} < K^{(s)}$), that represents $\Theta^{(s)}$ well. The likelihood of the tag ARM $\Theta^{(t)}$ is given by (4).

The HEM algorithm uses a set of N *virtual* samples generated from the base model $\Theta^{(s)}$, where the $N_i = N\pi_i^{(s)}$ samples $X_i = \{x_{1:\tau}^{(i,m)}\}_{m=1}^{N_i}$ are from the i^{th} component, i.e., $x_{1:\tau}^{(i,m)} \sim \Theta_i^{(s)}$. We assume that samples within each X_i are assigned to the same component of the tag model $\Theta^{(t)}$, and we denote the entire set of virtual samples with $X = \{X_i\}_{i=1}^{K^{(s)}}$.

The log likelihood of the incomplete data under $\Theta^{(t)}$ is

$$\begin{aligned}\log p(X|\Theta^{(t)}) &= \log \prod_{i=1}^{K^{(s)}} p(X_i|\Theta^{(t)}) \\ &= \log \prod_{i=1}^{K^{(s)}} \sum_{j=1}^{K^{(t)}} \pi_j^{(t)} p(X_i|\Theta_j^{(t)}).\end{aligned} \tag{5}$$

The HEM algorithm consists of the maximum likelihood estimation of the parameters of $\Theta^{(t)}$ from (5). Since (5) involves marginalizing over the hidden assignment variables $z_i^{(s)} \in \{1, \ldots, K^{(t)}\}$, its maximization can be solved with the EM algorithm. Hence, we introduce an indicator variable $\mathbf{z}_{i,j}$ for when the virtual audio sample set X_i is assigned to the j^{th} component of $\Theta^{(t)}$, i.e., when $z_i^{(s)} = j$. The complete data log-likelihood is then:

$$\begin{aligned}&\log p(X, Z|\Theta^{(t)}) = \\ &= \sum_{i=1}^{K^{(s)}} \sum_{j=1}^{K^{(t)}} \mathbf{z}_{i,j} \log \pi_j^{(t)} + \mathbf{z}_{i,j} \log p(X_i|\Theta_j^{(t)})\end{aligned} \tag{6}$$

The $\mathcal{Q}$-function is obtained by taking the conditional expectation of (6) with respect to Z, and the dependency on the virtual samples is removed by using the law of large numbers, i.e.,

$$\log p(X_i|\Theta_j^{(t)}) = N_i \frac{1}{N_i} \sum_{m=1}^{N_i} \log p(x_{1:\tau}^{(i,m)}|\Theta_j^{(t)}) \tag{7}$$

$$\approx N_i \, \mathbb{E}_{x_{1:\tau}|\Theta_i^{(s)}} \left[\log p(x_{1:\tau}|\Theta_j^{(t)})\right]. \tag{8}$$

Note that (8) can be computed using the chain rule of the expected log-likelihood and (1) to break the expectation

$$\mathbb{E}_{x_{1:\tau}|\Theta_i^{(s)}} \left[\log p(x_{1:\tau}|\Theta_j^{(t)})\right] = \tag{9}$$

$$= \sum_{t=1}^{\tau} \mathbb{E}_{x_{1:t}|\Theta_i^{(s)}} \left[\log p(x_t|x_{1:t-1}, \Theta_j^{(t)})\right] \tag{10}$$

$$= \sum_{t=1}^{\tau} \mathbb{E}_{x_{1:t}|\Theta_i^{(s)}} \left[\log p(x_t|x_{t-p:t-1}, \Theta_j^{(t)})\right] \tag{11}$$

$$= \sum_{t=1}^{\tau} \mathbb{E}_{x_{1:t-1}|\Theta_i^{(s)}} \left[\mathbb{E}_{x_t|x_{t-p:t-1},\Theta_i^{(s)}} \left[\log p(x_t|x_{t-p:t-1}, \Theta_j^{(t)})\right]\right] \tag{12}$$

The inner expectation in (12) is the expected log-likelihood of a Gaussian, and its closed form solution depends on the first and second order statistics of $x_{t-p,t-1} \sim \Theta_i^{(s)}$. The outer expectation involves the computation of the expected first and second order statistics of $\Theta_i^{(s)}$, which can be carried out with the recursion presented in Algorithm 1. Note that, since the AR model $\Theta_j^{(t)}$ has no hidden variables, the computation of the expected sufficient statistics in Algorithm 1 is independent of $\Theta_j^{(t)}$, and hence needs to be executed only once for each input component $\Theta_i^{(s)}$.

Algorithm 1 Expected sufficient statistics

Input: song-level AR model $\Theta_i^{(s)} = \{\mu, S, \widetilde{A}, Q\}$, length of virtual samples τ.

Compute expected sufficient statistics for $t = 1, \ldots, \tau - 1$:

$$\tilde{E}_1^{(i)} = \mathbb{E}_{x_1 \Theta_i^{(s)}}[x_1 x_1'] = \mu\mu' + S$$

$$\hat{E}_1^{(i)} = \mathbb{E}_{x_{-p+1:1}\Theta_i^{(s)}}\left[x_{-p+1}^1 {x_{-p+1}^1}'\right] = \begin{bmatrix} \tilde{E}_1^{(i)} & 0_{d\times(d-1)p} \\ 0_{(d-1)p\times d} & 0_{(d-1)p\times(d-1)p} \end{bmatrix}$$

For $t = 1, \ldots, \tau - 1$

$$\hat{E}_t^{(i)} = \mathbb{E}_{x_{1:t}|\Theta_i^{(s)}}\left[x_{t-p+1}^t {x_{t-p+1}^t}'\right] = \begin{bmatrix} \widetilde{A}\hat{E}_{t-1}^{(i)}\widetilde{A}' + Q & A\hat{E}_{t-1}^{(i)} \\ \hat{E}_{t-1}^{(i)}\widetilde{A}' & \hat{E}_{t-1}^{(i)} \end{bmatrix}_{(1:dp,1:dp)}$$

Endfor

Compute expected sufficient statistics:

$$\hat{E}^{(i)} = \textstyle\sum_{t=1}^{\tau-1} \hat{E}_t^{(i)} \tag{13}$$

Output: expected sufficient statistics: $\hat{E}^{(i)}$.

If hidden variables are present (which is the case for the DT components of the DTM model, but not for the AR model), computing the expected sufficient statistics of a song component $\Theta_i^{(s)}$ involves marginalizing over the hidden variables of $\Theta_j^{(t)}$, and hence needs to be repeated at every iteration for each $j = 1, \ldots, K^{(t)}$.

The E-step of the HEM consists of computing of the expected sufficient statistics in Algorithm 1, the assignments variables in (14) and (15), and the cumulative expected sufficient statistics in (16). The M-step maximizes the $\mathcal{Q}$-function with respect to $\Theta^{(t)}$, giving the updates in (17). The full HEM-ARM scheme is presented in Algorithm 2.

4. KERNEL-SVM APPROACH

We then used a semi-parametric approach that leverages the ARM model at the song level, and kernel support vector machine (SVM) for retrieval. In particular, we first model each song as an ARM using the EM algorithm. Then, for each tag, we learn a binary SVM classifier over the train set, based on a notion of similarity between ARM models defined in terms on their proximity in parameter space. Finally, following [3], we use the SVMs' decision values as the relevance of a song for a tag, and use it for retrieval of test songs based on one-tag queries.

Since the AR parameters lie on a non-linear manifold, naïvely treating them as Euclidean vectors would not necessary produce a correct similarity score. Hence, in the remainder of this section, we present several kernel functions based on more appropriate similarity scores between autoregressive (mixture) models. In previous work, Meng and Shawe-Taylor [17] specialize the Probability Product Kernel [14] to the AR case, which depends non-linearly on the AR parameters, and is define as:

$$\mathcal{K}_{\text{AR}}(\Theta_a, \Theta_b) = \textstyle\int_{x_{1:p}} (p(x_{1:p}|\Theta_a)p(x_{1:p}|\Theta_b))^\rho, \tag{18}$$

where $\rho = 0.5$ corresponds to the Battaccharyya affinity. Since a song-ARM is associated with several AR components, for retrieval we collect a decision value for each AR component, and then rank the songs according to the average of the corresponding decision values (PPK-AR). Note that we compute PPK between individual AR components of the song-ARMs. This is different from [17], which uses *single* ARs on individual audio snippets.

In addition, we experiment SVM classification in tandem with a probability product kernel between auotoregressive *mixture* models (PPK-ARM). Following an approximation by Jebara et al. [14], the PPK-ARM can be computed from the PPK between individual components as

$$\mathcal{K}_{\text{ARM}}(\Theta^{(1)}, \Theta^{(2)}) = \sum_{a=1}^{K_s}\sum_{b=1}^{K_s} (\pi_a^{(1)}\pi_b^{(2)})^\rho \mathcal{K}_{\text{AR}}(\Theta_a^{(1)}, \Theta_b^{(2)}). \tag{19}$$

We finally propose a novel descriptor of AR models based on their frequency responses, and compute a kernel between these descriptors. Since an AR is a linear time invariant (LTI) system, its dynamics can be characterized

Algorithm 2 HEM algorithm for ARM

Input: combined song-level ARM $\{\pi_i^{(s)}, \Theta_i^{(s)}\}_{i=1}^{K^{(s)}}$, number of virtual samples N.

Compute cumulative expected sufficient statistics $\hat{E}^{(i)}$ for each $\Theta_i^{(s)}$ using Algorithm 1

Initialize tag-level ARM, $\{\pi_j^{(t)}, \Theta_j^{(t)}\}_{j=1}^{K^{(t)}}$.

repeat

{E-step}

Compute expected log-likelihood for each $\Theta_i^{(s)}$ and $\Theta_j^{(t)}$:

$$\begin{aligned}\ell_{i|j} &= \mathbb{E}_{x_{1:\tau}|\Theta_i^{(s)}}[\log p(x_{1:\tau}|\Theta_j^{(t)})] \\ &= -\tfrac{d\tau}{2}\log 2\pi - \tfrac{1}{2}\log|S_j^{(t)}| \\ &\quad -\tfrac{1}{2}\text{trace}{S_j^{(t)}}^{-1}[S_i^{(s)} + (\mu_j^{(t)} - \mu_i^{(s)})'(\mu_j^{(t)} - \mu_i^{(s)})] \\ &\quad -\tfrac{\tau-1}{2}\text{trace}{Q_j^{(t)}}^{-1}Q_i^{(s)} - \tfrac{\tau-1}{2}\log|Q_j^{(t)}| \\ &\quad -\tfrac{1}{2}\text{trace}[{Q_j^{(t)}}^{-1}(\widetilde{A}_j^{(t)} - \widetilde{A}_i^{(s)})\hat{E}^{(i)}(\widetilde{A}_j^{(t)} - \widetilde{A}_i^{(s)})']\end{aligned}$$

{M-step}

Compute assignment probability and weighting:

$$\hat{\mathbf{z}}_{i,j} = \frac{\pi_j^{(t)}\exp\left(N_i\ell_{i|j}\right)}{\sum_{j'=1}^{K^{(t)}}\pi_{j'}^{(t)}\exp\left(N_i\ell_{i|j'}\right)} \tag{14}$$

$$\hat{\mathbf{w}}_{i,j} = \hat{\mathbf{z}}_{i,j}N_i = \hat{\mathbf{z}}_{i,j}\pi_i^{(s)}N \tag{15}$$

Computed aggregated expectations for each $\hat{\Theta}_j^{(t)}$:

$$\begin{aligned}&\hat{N}_j = \textstyle\sum_i \hat{\mathbf{z}}_{i,j}, && \hat{M}_j = \textstyle\sum_i \hat{\mathbf{w}}_{i,j}, \\ &\hat{S}_j = \textstyle\sum_i \hat{\mathbf{w}}_{i,j}[S_i^{(s)} + \mu_i^{(s)}(\mu_i^{(s)})'] && \hat{m}_j = \textstyle\sum_i \hat{\mathbf{w}}_{i,j}\mu_i^{(s)} \\ &\hat{V}_j = \textstyle\sum_i \hat{\mathbf{w}}_{i,j}A_i^{(s)}\hat{E}^{(i)}(\widetilde{A}_i^{(s)})' && \hat{P}_j = \textstyle\sum_i \hat{\mathbf{w}}_{i,j}\hat{E}^{(i)} \\ &\hat{R}_j = \textstyle\sum_i \hat{\mathbf{w}}_{i,j}\hat{E}^{(i)}(\widetilde{A}_i^{(s)})' && \hat{Q}_j = \textstyle\sum_i \hat{\mathbf{w}}_{i,j}Q_i^{(s)}\end{aligned} \tag{16}$$

{M-step}

Recompute parameters for each component $\hat{\Theta}_j^{(t)}$:

$$\begin{aligned}&\widetilde{A}_j^* = \hat{R}_j'\hat{P}_j^{-1} && Q_j^* = \tfrac{1}{(\tau-1)\hat{M}_j}(\hat{V}_i - A_j^*\hat{R}_j + \hat{Q}_j), \\ &\mu_j^* = \tfrac{1}{\hat{M}_j}\hat{m}_j, && S_j^* = \tfrac{1}{\hat{M}_j}\hat{S}_j - \mu_j^*(\mu_j^*)', \\ &\pi_j^* = \tfrac{\sum_{i=1}^{K^{(s)}}\hat{\mathbf{z}}_{i,j}}{K^{(s)}}.\end{aligned} \tag{17}$$

until convergence

Output: tag-level ARM $\{\pi_j^{(t)}, \Theta_j^{(t)}\}_{j=1}^{K^{(t)}}$.

by a transfer function defined as:

$$H(s) = (I_d - \sum_{j=1}^{p} A_j s^{-j})^{-1} \in \mathbb{C}^{d \times d} \qquad (20)$$

where $s \in \mathbb{C}$ is a complex number and I_d is the $d-$dimensional identity matrix. The transfer function describes the cross influences of each pair of components of the audio feature vectors. In particular, we sample the transfer function at 200 equally spaced points on the unit circle, and then sum the the absolute values of these matrices over 30 linearly spaced frequency bins, to get a representation of the system's frequency response. By concatenating the AR's μ parameter and the log values of these 30 frequency response matrices, we get a descriptor $\Delta \in \mathbb{R}^{(d+30d^2)\times 1}$, which we call quantized frequency response (QFR). Finally, we use a SVM over QFRs based on cosine-similarity (CS) kernel and radial basis function (RBF) kernel. [2]

5. EXPERIMENTS

5.1 Data

We performed automatic music annotation on the CAL500 dataset (details in [23] and references therein), which is a collection of 502 popular Western songs by as many different artists, and provides binary annotations with respect to a vocabulary of semantic tags. In our experiments we consider the 97 tags associated to at least 30 songs in CAL500 (11 genre, 14 instrumentation, 25 acoustic quality, 6 vocal characteristics, 35 mood and 6 usage tags).

The acoustic content of a song (resampled at $22,050Hz$) is represented by computing a time-series of 34-bin Mel-frequency spectral (MFS) features, extracted over half overlapping windows of 92 msec of audio signal, i.e., every $\sim$ 46 msec. Following the insight in recent work of Hamel et al. [12], MFS features where further projected on the first $d = 20$ principal components, which we estimated over the MFSs collected from the $10,870$ songs in the CAL10K dataset [22].

Song level ARMs were learned with $K = 4$ components and memory of $p = 5$ steps, from a dense sampling of audio fragments of length $T = 125$ (i.e., approximately $6s$), extracted with 80% overlap.

5.2 Results with HEM-ARM

For each tag, all the relevant song ARMs were pooled together to form a big ARM, and a tag-level ARM with $K^{(t)} = 8$ components was learned with the HEM-ARM algorithm (with $N = 1000$ virtual samples of length $\tau = 10$). To reduce the effects of low likelihood in high dimension, for annotation we smooth the likelihood (3) by $T \cdot d \cdot p$. We compare our HEM-ARM with hierarchically trained Gaussian mixture models (HEM-GMM) [23] and dynamic texture mixture models (HEM-DTM) [7].

On the test set, a novel test song is annotated with the 10 most likely tags, corresponding to the peaks in its semantic multinomial. Retrieval given a one tag query involves rank ordering all songs with respect to the corresponding entry in their semantic multinomials. Annotation is measured with average per-tag precision (P), recall (R), and f-score (F). Retrieval is measured by per-tag area under the ROC (AROC), mean average precision (MAP), and precision at the first 10 retrieved objects (P@10). Refer to [23] for a detailed definition of the metrics. All reported metrics are the result of 5 fold-cross validation.

	annotation			retrieval			time	
	P	R	F	AROC	MAP	P@10	train	test
HEM-ARM	0.468	0.203	0.254	0.696	0.421	0.412	$198m$	$41m$
HEM-DTM	0.446	0.217	0.264	0.708	0.446	0.460	$424h$	$482m$
HEM-GMM	0.474	0.205	0.213	0.686	0.417	0.425	$41m$	$38m$

Table 1. Annotation and retrieval on CAL500, for HEM-ARM, HEM-DTM and HEM-GMM.

In Table 1 we report annotation and retrieval results for HEM-ARM, HEM-DTM and HEM-GMM. In addition, we register the total time for the training stage, which consist in the estimation of the 97 tag models over the 5 folds (and also includes the estimation of the 502 song-level models), as well as for the test stage, i.e., the automatic-annotation of the 502 songs with the 97 tags.

From Table 1 we note that the advantages of the proposed HEM-ARM relative to HEM-DTM are in terms of *computation efficiency*. While HEM-DTM performs better than HEM-ARM on each metric (except on annotation precision where HEM-ARM is better), HEM-ARM has significantly lower time requirements. Specifically, the training time for our HEM-ARM is two orders of magnitude lower than that for HEM-DTM. Similarly, our auto-tagger requires approximately 42 minutes for the test-stage, while the auto-tagger based on DTMs requires 482 minutes to accomplish the same task. These results are explained by comparing the *graphical structures* of the AR and DT models. While the AR is a fully observable model, the DT consists of an observed layer, which model the spectral content, and a hidden layer that encodes the temporal dynamics. Hence, DTMs have the advantage of learning different frequency basis to best represent specific tags [7]. However, the computation of the (expected) sufficient statistics with respect to each DT component requires marginalization of the hidden variables (see [5]). Hence, during training, it needs to be executed at each iteration of the learning algorithms for each input datum (i.e., audio-fragments for EM, and DTs for HEM) and for each individual component of the learned model; during annotation, it needs to be repeated for each audio-fragment and each DT component of the tag models. On the opposite, the corresponding statistics for ARMs involve no marginalization of hidden variables. Hence, during training, they need to be computed only a single time for each input datum (i.e., audio-fragments for EM, and ARs for HEM). In addition, during annotation, the sufficient statistics can be collected a single time for each song.

Finally, HEM-ARM performs favorably relative to HEM-GMM (which does not model temporal dynamics), while still requiring limited computation times. Since learning and inference are performed efficiently, HEM-ARM can leverage higher order memories to model temporal dynam-

[2] The CS kernel is defined as $\mathcal{K}(a,b) = a'b/\sqrt{||a||_2||b||_2}$. The RBF kernel is defines as $\mathcal{K}(a,b) = \exp\{-||a-b||_2^2/\sigma\}$. We set the bandwidth σ of the RBF kernel to the descriptor dimension $dim(\Delta)$.

p	1	2	3	4	5	6	7	8	9
F-score	0.234	0.247	0.252	0.255	0.254	0.245	0.244	0.242	0.237
AROC	0.650	0.672	0.686	0.693	0.696	0.695	0.694	0.695	0.694

Table 2. Annotation (F-score) and retrieval (AROC) performance of HEM-ARM, for different memories $p \in [1{:}9]$.

kernel		AROC	MAP	P@10	train	test
ARM based	PPK-ARM	0.717	0.448	0.459	$233m$	$194m$
	PPK-AR	0.727	0.463	0.484	$287m$	$194m$
	QFR-CS	0.717	0.461	0.479	$125m$	$65m$
	QFR-RBF	0.723	0.469	0.488	$137m$	$74m$
GMM-PPK [3]		0.696	0.436	0.454		
MFCC-PPK-AR [17]		0.706	0.447	0.463		

Table 3. Retrieval for the kernel-SVM approach. Including train and test times

ics, without incurring in large delays. In particular, in Table 2 we plot annotation (F-score) and retrieval (AROC) performance as a a function of the memory p of the AR models. Performance are fairly stable for $p = 4, 5, 6$. Shorter memories (e.g., $p = 1, 2, 3$) do not suffices to capture the interesting dynamics, while too large values deteriorate annotation performance.

5.3 Results with kernel-SVM.

We implemented the kernel-SVM approach as described in Section 4. In particular, we learned song-ARMs with $K = 4$ components and memory $p = 5$, estimated from the $d = 20$ dimensional PCA-MFS features. We then computed the QFR-CS and QFR-RBF kernels based on the QFR descriptors, the PPK kernel between ARM (PPK-ARM), and the PPK kernel between individual AR components (PPK-AR). For comparison, we also considered PPK similarity between song-GMMs estimated on MFCC features [3] (GMM-PPK) and PPK similarity between single AR models estimated on the MFCC features of entire songs as proposed in [17] (MFCC-PPK-AR). We used the LibSVM software package [6] for the SVM, with all parameters selected using validation on the training set.

Retrieval scores are reported in Table 3, and are result of 5-fold cross validation. We note that these results are generally superior to those in Table 1, since the SVM is a discriminative algorithm and hence tends to be more robust on strongly labeled datasets such as CAL500. In particular, the best performance was registered with the QFR-RBF and PPK-AR systems (score differences between them are not statistically significant). In addition, PPK similarity on ARMs (PPK-ARM) proves less performant, suggesting that the approximation in (19) may be not enough accurate for our task. Finally, PPK similarity on GMM performs the worst, since it does not leverage termporal dynamics, and MFCC-AR-PPK, which doesn't leverage mixtures, is also significantly behind.

6. DISCUSSION

In this paper we have proposed the ARM model for music auto-tagging. We have derived a hierarchical EM algorithm for efficiently learning tag ARMs. We have showed that our HEM-ARM can estimate tag models significantly more efficiently than HEM-DTM, at the price of a small reduction in performance. We have also successfully tested a kernel-SVM approach based on several similarity functions based on the ARM model.

7. REFERENCES

[1] A. Agarwal and B. Triggs. Tracking articulated motion using a mixture of autoregressive models. In *ECCV - Lecture notes in computer science*, pages 54–65, 2004.

[2] L. Barrington, D. O'Malley, D. Turnbull, and G. Lanckriet. User-centered design of a social game to tag music. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, pages 7–10. ACM, 2009.

[3] L. Barrington, M. Yazdani, D. Turnbull, and G. Lanckriet. Combining feature kernels for semantic music retrieval. *Proc. ISMIR 2008*, pages 723–728, 2008.

[4] M. Casey, C. Rhodes, and M. Slaney. Analysis of minimum distances in high-dimensional musical spaces. *IEEE Transactions on Audio, Speech and Language Processing*, 16(5):1015–1028, 2008.

[5] A.B. Chan, E. Coviello, and G. Lanckriet. Clustering dynamic textures with the hierarchical EM algorithm. In *Proc. IEEE CVPR*, 2010.

[6] C.C. Chang and C.J. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[7] E. Coviello, A. Chan, and G. Lanckriet. Time Series Models for Semantic Music Annotation. *Audio, Speech, and Language Processing, IEEE Transactions on*, 19(5):1343–1359, July 2011.

[8] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38, 1977.

[9] D. Eck, P. Lamere, T. Bertin-Mahieux, and S. Green. Automatic generation of social tags for music recommendation. In *Advances in Neural Information Processing Systems*, 2007.

[10] K. Ellis, E. Coviello, and G. Lanckriet. Semantic annotation and retrieval of music using a bag of systems representation. In *ISMIR*, 2011.

[11] A. Flexer, F. Gouyon, S. Dixon, and G. Widmer. Probabilistic combination of features for music classification. In *Proc. ISMIR*, pages 111–114, 2006.

[12] P. Hamel, S. Lemieux, Y. Bengio, and D. Eck. Temporal pooling and multiscale learning for automatic annotation and ranking of music audio. ISMIR, 2011.

[13] M. Hoffman, D. Blei, and P. Cook. Easy as CBA: A simple probabilistic model for tagging music. In *Proc. ISMIR*, pages 369–374, 2009.

[14] T. Jebara, R. Kondor, and A. Howard. Probability product kernels. *The Journal of Machine Learning Research*, 5:819–844, 2004.

[15] E. Law and L. Von Ahn. Input-agreement: a new mechanism for collecting data using human computation games. In *Proceedings of the 27th international conference on Human factors in computing systems*, pages 1197–1206, 2009.

[16] M.I. Mandel and D.P.W. Ellis. Multiple-instance learning for music information retrieval. In *Proc. ISMIR*, pages 577–582, 2008.

[17] A. Meng and J. Shawe-Taylor. An investigation of feature models for music genre classification using the support vector classifier. In *Proc. ISMIR*, pages 604–609, 2005.

[18] A. Neumaier and T. Schneider. Estimation of parameters and eigenmodes of multivariate autoregressive models. *ACM Transactions on Mathematical Software (TOMS)*, 27(1):27–57, 2001.

[19] J. Reed and C.H. Lee. A study on music genre classification based on universal acoustic models. In *Proc. ISMIR*, pages 89–94, 2006.

[20] C. Joder S. Essid and G. Richard. Temporal integration for audio classification with application to musical instrument classification. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(1):174–186, 2009.

[21] M. Slaney, K. Weinberger, and W. White. Learning a metric for music similarity. In *Proc. ISMIR*, pages 313–318, 2008.

[22] Derek Tingle, Youngmoo E. Kim, and Douglas Turnbull. Exploring automatic music annotation with "acoustically-objectiv" tags. In *Proc. MIR*, New York, NY, USA, 2010.

[23] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE Transactions on Audio, Speech and Language Processing*, 16(2):467–476, February 2008.

[24] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on speech and audio processing*, 10(5):293–302, 2002.

[25] N. Vasconcelos and A. Lippman. Learning mixture hierarchies. In *Advances in Neural Information Processing Systems*, pages 606–612, 1998.

[26] B. Whitman and D. Ellis. Automatic record reviews. In *Proc. ISMIR*, 2004.