

CODEBOOK-BASED SCALABLE MUSIC TAGGING WITH POISSON MATRIX FACTORIZATION

Dawen Liang, John Paisley, Daniel P. W. Ellis

Department of Electrical Engineering
Columbia University

{dliang, dpwe}@ee.columbia.edu, jpaisley@columbia.edu

ABSTRACT

Automatic music tagging is an important but challenging problem within MIR. In this paper, we treat music tagging as a matrix completion problem. We apply the Poisson matrix factorization model jointly on the vector-quantized audio features and a “bag-of-tags” representation. This approach exploits the shared latent structure between semantic tags and acoustic codewords. Leveraging the recently-developed technique of stochastic variational inference, the model can tractably analyze massive music collections. We present experimental results on the CAL500 dataset and the Million Song Dataset for both annotation and retrieval tasks, illustrating the steady improvement in performance as more data is used.

1. INTRODUCTION

Automatic music tagging is the task of analyzing the audio content (waveform) of a music recording and assigning to it human-relevant semantic tags [16] – which may relate to style, genre, instrumentation, or more subtle aspects of the music, such as those contributed by users on social media sites. Such “autotagging” [5] relies on labelled training examples for each tag, and performance typically improves with the number of training examples consumed, although training schemes also take longer to complete. In the era of “Big Data”, it is necessary to develop models which can rapidly handle massive amount of data; a starting point for music data is the Million Song Dataset [2], which includes user tags from Last.fm [1].

In this paper, we treat the automatic music tagging as a matrix completion problem, and use the techniques of stochastic variational inference to be able to learn from large amounts of data presented in an online fashion [9]. The “matrix completion” problem treats each track as a row in a matrix, where the elements describe both the acoustic properties (represented, for instance, as a histogram of occurrences of vector-quantized acoustic features) and the relevance of a large vocabulary of tags: We can regard the

tag information as incomplete or missing for some of the rows, and seek to “complete” these rows based on information inferred from the complete, present rows.

1.1 Related work

There have been a large number of papers on automatic tagging of music audio in recent years. In addition to the papers mentioned above, work particularly relevant to this paper includes the Codeword Bernoulli Average (CBA) approach of Hoffman *et al.* [7], which uses a similar VQ histogram representation of the audio to build a simple but effective probabilistic model for each tag in a discriminative fashion. Xie *et al.* [17] directly fits a regularized logistic regression model to the normalized acoustic codeword histograms to predict each tag and achieves state-of-the-art results, and Ellis *et al.* [6] further improves tagging accuracy by employing multiple generative models that capture different characteristics of a music piece, which are combined in an optimized “bag-of-systems”.

Much of the previous work has been performed on the CAL500 dataset [16] of 502 Western popular music tracks that were carefully labelled by at least three human annotators with their relevance to 149 distinct labels spanning instrumentation, genre, emotions, vocal characteristics, and use cases. This small dataset tends to reward approaches that can maximize the information extracted from the sparse data regardless of the computational cost. A relatively larger dataset in this domain is CAL10k [15] with over 10,000 tracks described by over 500 tags, mined from Pandora’s website¹. However, neither of these datasets can be considered industrial scale, which implies handling millions of tracks and tens of thousands of tags.

Matrix factorization techniques, in particular, nonnegative matrix factorization (NMF), have been widely used to analyze music signals [8, 11] in the context of source separation. Paisley *et al.* [12] derived scalable Bayesian NMF for topic modeling, which we develop here. To our knowledge, this is the first application of matrix factorization to VQ acoustic features for automatic music tagging.

2. DATA REPRESENTATION

For our automatic tagging system, the data comes from two sources: vector-quantized audio features and a “bag-



© Dawen Liang, John Paisley, Daniel P. W. Ellis.

Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Dawen Liang, John Paisley, Daniel P. W. Ellis. “Codebook-based scalable music tagging with Poisson matrix factorization”, 15th International Society for Music Information Retrieval Conference, 2014.

¹<http://www.pandora.com/>

of-tags” representation.

- **Vector-quantized audio features** Instead of directly working with audio features, we vector quantize all the features following the standard procedure: We run the K -means algorithm on a subset of randomly selected training data to learn J cluster centroids (codewords). Then for each song, we assign each frame to the cluster with the smallest Euclidean distance to the centroid. We form the VQ feature $y_{VQ} \in \mathbb{N}^J$ by counting the number of assignments to each cluster across the entire song.
- **Bag-of-tags** Similar to the bag-of-words representation, which is commonly used to represent documents, we represent the tagging information (whether or not the tag applies to a song) with a binary bag-of-tags vector $y_{BoT} \in \{0, 1\}^{|V|}$, where V is the set of all tags.

For each song, we will simply concatenate the VQ feature y_{VQ} and the bag-of-tags vector y_{BoT} , thus the dimension of the data is $D = J + |V|$. When we apply the matrix factorization model to the data, the latent factors we learn will exploit the shared latent structure between semantic tags and acoustic codewords. Therefore, we can utilize the shared latent structure to predict tags when only given the audio features.

3. POISSON MATRIX FACTORIZATION

We adopt the notational convention that bold letters (e.g. $\mathbf{y}, \boldsymbol{\theta}, \boldsymbol{\beta}$) denote matrices. $i \in \{1, \dots, I\}$ is used to index songs. $d \in \{1, \dots, D\}$ is used to index feature dimensions. $k \in \{1, \dots, K\}$ is used to index latent factors from the matrix factorization model. Given the data $\mathbf{y} \in \mathbb{N}^{I \times D}$ as described in Section 2, the Poisson matrix factorization model is formulated as follows:

$$\begin{aligned} \theta_{ik} &\sim \text{Gamma}(a, ac), \\ \beta_{kd} &\sim \text{Gamma}(b, b), \\ y_{id} &\sim \text{Poisson}(\sum_{k=1}^K \theta_{ik} \beta_{kd}), \end{aligned} \quad (1)$$

where $\beta_k \in \mathbb{R}_+^D$ denote the k th latent factors and $\theta_i \in \mathbb{R}_+^K$ denote the weights for song i . a and b are model hyperparameters. c is a scalar on the weights that we tune to maximize the likelihood.

There are a couple of reasons to choose a Poisson model over a more traditional Gaussian model [14]. First, the Poisson distribution is a more natural choice to model count data. Secondly, real-world tagging data is extremely noisy and sparse. If a tag is not associated with a song in the data, it could be either because that tag does not apply to the song, or simply because no one has labelled the song with the tag yet. The Poisson matrix factorization model has the desirable property that it does not penalize values of 0 as strongly as the Gaussian distribution [12]. Therefore, even weakly labelled data can be used to learn the Poisson model.

4. VARIATIONAL INFERENCE

To learn the latent factors $\boldsymbol{\beta}$ and the corresponding decomposition weights $\boldsymbol{\theta}$ from the training data \mathbf{y} , we need to compute the posterior distribution $p(\boldsymbol{\theta}, \boldsymbol{\beta} | \mathbf{y})$. However, no closed-form expression exists for this hierarchical model. We therefore employ mean-field variational inference to approximate this posterior [10].

The basic idea behind mean-field variational inference is to choose a factorized family of variational distributions,

$$q(\boldsymbol{\theta}, \boldsymbol{\beta}) = \prod_{k=1}^K \left(\prod_{i=1}^I q(\theta_{ik}) \right) \left(\prod_{d=1}^D q(\beta_{kd}) \right), \quad (2)$$

to approximate the posterior $p(\boldsymbol{\theta}, \boldsymbol{\beta} | \mathbf{y})$, so that the Kullback-Leibler (KL) divergence between the variational distribution and the true posterior is minimized. Following a further approximation discussed in the next section, the factorized distribution allows for a closed-form expression of this variational objective, and thus tractable inference. Here we choose variational distributions from the same family as the prior:

$$\begin{aligned} q(\theta_{ik}) &= \text{Gamma}(\theta_{ik}; \gamma_{ik}, \chi_{ik}), \\ q(\beta_{kd}) &= \text{Gamma}(\beta_{kd}; \nu_{kd}, \lambda_{kd}). \end{aligned} \quad (3)$$

Minimizing the KL divergence is equivalent to maximizing the following variational objective:

$$\mathcal{L} = \mathbb{E}_q[\ln p(\mathbf{y}, \boldsymbol{\theta}, \boldsymbol{\beta})] + H(q), \quad (4)$$

where $H(q)$ is the entropy of the variational distribution q . We can optimize the variational objective using coordinate ascent via two approaches: batch inference, which requires processing of the entire dataset for every iteration; or stochastic inference, which only needs a small batch of data for each iteration and can be potentially scale to much larger datasets where batch inference is no longer computationally feasible.

4.1 Batch inference

Although the model in Equation (1) is not conditionally conjugate by itself, as demonstrated in [4] we can introduce latent random variables $z_{idk} \sim \text{Poisson}(\theta_{ik} \beta_{kd})$ with the variational distribution being $q(z_{idk}) = \text{Multi}(z_{id}; \phi_{id})$, where $z_{id} \in \mathbb{N}^K$, $\phi_{idk} \geq 0$ and $\sum_k \phi_{idk} = 1$. This makes the model conditionally conjugate, which means that closed-form coordinate ascent updates are available.

Following the standard results of variational inference for conditionally conjugate model (e.g. [9]), we can obtain the updates for θ_{ik} :

$$\begin{aligned} \gamma_{ik} &= a + \sum_{d=1}^D y_{id} \phi_{idk}, \\ \chi_{ik} &= ac + \sum_{d=1}^D \mathbb{E}_q[\beta_{kd}]. \end{aligned} \quad (5)$$

The scale c is updated as $c^{-1} = \frac{1}{IK} \sum_{i,k} \mathbb{E}_q[\theta_{ik}]$.

Similarly, we can obtain the updates for β_{kd} :

$$\begin{aligned}\nu_{kd} &= b + \sum_{i=1}^I y_{id} \phi_{idk}, \\ \lambda_{kd} &= b + \sum_{i=1}^I \mathbb{E}_q[\theta_{ik}].\end{aligned}\quad (6)$$

Finally, for the latent variables z_{idk} , the following update is applied:

$$\phi_{idk} \propto \exp\{\mathbb{E}_q[\ln \theta_{ik} \beta_{kd}]\}.\quad (7)$$

The necessary expectations for θ_{ik} are:

$$\begin{aligned}\mathbb{E}_q[\theta_{ik}] &= \gamma_{ik} / \chi_{ik}, \\ \mathbb{E}_q[\ln \theta_{ik}] &= \psi(\gamma_{ik}) - \ln \chi_{ik},\end{aligned}\quad (8)$$

where $\psi(\cdot)$ is the digamma function. The expectations for β_{kd} have the same form, but use ν_{kd} and λ_{kd} .

4.2 Stochastic inference

Batch inference will alternate between updating θ and β using the entire data at each iteration until convergence to a local optimum, which could be computationally intensive for large datasets. We can instead adopt stochastic optimization by selecting a subset (mini-batch) of the data at iteration t , indexed by $B_t \subset \{1, \dots, I\}$, and optimizing over a noisy version of the variational objective \mathcal{L} :

$$\mathcal{L}_t = \frac{I}{|B_t|} \sum_{i \in B_t} \mathbb{E}_q[\ln p(y_i, \theta_i | \beta)] + \mathbb{E}_q[\ln p(\beta)] + H(q).\quad (9)$$

By optimizing \mathcal{L}_t , we are optimizing \mathcal{L} in expectation.

The updates for weights θ_{ik} and latent variables z_{idk} are essentially the same as batch inference, except that now we are only inferring weights for the mini-batch of data for $i \in B_t$. The optimal scale c is updated accordingly:

$$c^{-1} = \frac{1}{|B_t|K} \sum_{i \in B_t, k} \mathbb{E}_q[\theta_{ik}].\quad (10)$$

After alternating between updating weights θ_{ik} and latent variables z_{idk} until convergence, we can take a gradient step, preconditioned by the inverse Fisher information matrix of variational distribution $q(\beta_{kd})$, to optimize β_{kd} (see [9] for more technical details),

$$\begin{aligned}\nu_{kd}^{(t)} &= (1 - \rho_t) \nu_{kd}^{(t-1)} + \rho_t \left(b + \frac{I}{|B_t|} \sum_{i \in B_t} y_{id} \phi_{idk} \right), \\ \lambda_{kd}^{(t)} &= (1 - \rho_t) \lambda_{kd}^{(t-1)} + \rho_t \left(b + \frac{I}{|B_t|} \sum_{i \in B_t} \mathbb{E}_q[\theta_{ik}] \right),\end{aligned}\quad (11)$$

where $\rho_t > 0$ is a step size at iteration t . To ensure convergence [3], the following conditions must be satisfied:

$$\sum_{t=1}^{\infty} \rho_t = \infty, \quad \sum_{t=1}^{\infty} \rho_t^2 < \infty.\quad (12)$$

One possible choice of ρ_t is $\rho_t = (t_0 + t)^{-\kappa}$ for $t_0 > 0$ and $\kappa \in (0.5, 1]$. It has been shown [9] that this update corresponds to stochastic optimization with a natural gradient step, which better fits the geometry of the parameter space for probability distributions.

4.3 Generalizing to new songs

Once the latent factor $\beta \in \mathbb{R}_+^{K \times D}$ is inferred, we can naturally divide it into two blocks: the VQ part $\beta_{VQ} \in \mathbb{R}_+^{K \times J}$, and the bag-of-tags part $\beta_{BoT} \in \mathbb{R}_+^{K \times |V|}$.

Given a new song, we can first obtain the VQ feature y_{VQ} and fit it with β_{VQ} to compute posterior of the weights $p(\theta | y_{VQ}, \beta_{VQ})$. We can approximate this posterior with the variational inference algorithm in Section 4.1 with β fixed. Then to predict tags, we can compute the expectation of the dot product between the weights θ and β_{BoT} under the variational distribution:

$$\hat{y}_{BoT} = \mathbb{E}_q[\theta^T \beta_{BoT}].\quad (13)$$

Since for different songs the weights θ may be scaled differently, before computing the dot product we normalize $\mathbb{E}_q[\theta]$ so that it lives on the probability simplex. To do automatic tagging, we could annotate the song with top M tags according to \hat{y}_{BoT} . To compensate for a lack of diversity in the annotations, we adopt the same heuristic used in [7] by introducing a ‘‘diversity factor’’ d : For each predicted score, we subtract d times the mean score for that tag. In our system, we set $d = 3$.

5. EVALUATION

We evaluate the model’s performance on an annotation task and a retrieval task using CAL500 [16] and Million Song Dataset (MSD) [2]. Unlike the CAL500 dataset where tracks are carefully-annotated, the Last.fm dataset [1] associated with MSD comes from real-world user tagging, and thus contains only weakly labelled data with a tagging vocabulary that is much larger and more diverse. We compare our results on these tasks with two other sets of codebook-based methods: Codeword Bernoulli Average (CBA) [7] and ℓ_2 regularized logistic regression [17]. Like the Poisson matrix factorization model, both methods are easy to train and can scale to relatively large dataset on a single machine. However, since both methods perform optimization in a batch fashion, we will later refer to them as ‘‘batch algorithms’’, along with the Poisson model with batch inference described in Section 4.1.

For the hyperparameters of the Poisson matrix factorization model, we set $a = b = 0.1$, and the number of latent factors $K = 100$. To learn the latent factors β , we followed the procedure in Section 4.1 for batch inference until the relative increase on the variational objective is less than 0.05%. For stochastic inference, we used a mini-batch size $|B_t| = 1000$ unless otherwise specified and took a full pass of the randomly permuted data. As for the learning rate, we set $t_0 = 1$ and $\kappa = 0.6$. All the source code in Python is available online².

5.1 Annotation task

The purpose of annotation task is to automatically tag unlabelled songs. To evaluate the model’s ability for annotation, we computed the average per-tag precision, recall,

² http://github.com/dawenl/stochastic_PMF

and F-score on a test set. Per-tag precision is defined as the average fraction of songs that the model annotates with tag v that are actually labelled v . Per-tag recall is defined as the average fraction of songs that are actually labelled v that the model also annotates with tag v . F-score is the harmonic mean of precision and recall, and is one overall metric for annotation performance.

5.2 Retrieval task

The purpose of the retrieval task is, when given a query tag v , to provide a list of songs which are related to tag v . To evaluate the models’ retrieval performance, for each tag in the vocabulary we ranked each song in the test set by the predicted score from Equation (13). We evaluated the area under the receiver-operator curve (AROC) and mean average precision (MAP) for each ranking. AROC is defined as the area under the curve, which plots the true positive rate against the false positive rate, and MAP is defined as the mean of the average precision (AP) for each tag, which is the average of the precisions at each possible level of recall.

5.3 Results on CAL500

Following the procedure similar to that described in [7, 17], we performed a 5-fold cross-validation to evaluate the annotation and retrieval performance on CAL500. We selected the top 78 tags, which are annotated more than 50 times in the dataset, and learned a codebook of size $J = 2000$. For the annotation task, we labelled each song with the top 10 tags based on the predicted score. Since CAL500 is a relatively small dataset, we only performed batch inference for Poisson matrix factorization model.

The results are reported in Table 1, which shows that the Poisson model has comparable performance on the annotation task, and does slightly worse on the retrieval task. As mentioned in Section 3, the Poisson matrix factorization model is particularly suitable for noisy and sparse data where 0’s are not necessarily interpreted as explicit observations. However, this may not be the case for CAL500, as the vocabulary was well-chosen and the data was collected from a survey where the tagging quality is understandably higher than the actual tagging data in the real world, like the one from Last.fm. Therefore, this task cannot fully exploit the advantage brought by the Poisson model. Meanwhile, the amount of data in CAL500 is fairly small – the data \mathbf{y} fit to the model is simply a 502-by-2078 matrix. This prevents us from adopting stochastic inference, which will be shown being much more effective than batch inference even on a 10,000-song dataset in Section 5.4.

5.4 Results on MSD

To demonstrate the scalability of the Poisson matrix factorization model, we conducted experiments using MSD and the associated Last.fm dataset. To our knowledge, there has not been any previous work where music tagging results are reported on the MSD.

Model	Prec	Recall	F-score	AROC	MAP
CBA	0.41	0.24	0.29	0.69	0.47
ℓ_2 LogRegr	0.48	0.26	0.34	0.72	0.50
PMF-Batch	0.42	0.23	0.30	0.67	0.45

Table 1. Results for the top 78 popular tags on CAL500, for Codeword Bernoulli Average (CBA), ℓ_2 regularized logistic regression (ℓ_2 LogRegr), and Poisson matrix factorization with batch inference (PMF-Batch). The results for CBA and ℓ_2 LogRegr are directly copied from [17].

Since the Last.fm dataset contains 522,366 unique tags, it is not realistic to build the model with all of them. We first selected the tags with more than 1,000 appearances and removed those which do not carry discriminative information (e.g. “my favorite”, “awesome”, “seen live”, etc.). Then we ran the stemming algorithm implemented in *NLTK*³ to further reduce the potential duplications and correct for alternate spellings (e.g. “pop-rock” v.s. “pop rock”, “love song” v.s. “love songs”), which gave us a vocabulary of 561 tags. Using the default train/test artist split from MSD, we filtered out the songs which have been labelled with tags from the selected vocabulary. This gave us 371,209 songs for training. For test set, we further selected those which have at least 20 tags (otherwise, it is likely that this song is very weakly labelled). This gave us a test set of 2,757 songs. The feature we used is the Echo Nest’s timbre feature, which is very similar to MFCC.

We randomly selected 10,000 songs as the data which can fit into the memory nicely for all the batch algorithms, and trained the following models with different codebook sizes $J \in \{256, 512, 1024, 2048\}$: Codeword Bernoulli Average (CBA), ℓ_2 regularized logistic regression (ℓ_2 LogRegr), Poisson matrix factorization with batch inference (PMF-Batch) and stochastic inference by a single pass of the data (PMF-Stoc-10K). Here we used batch size $|B_t| = 500$ for PMF-Stoc-10K, as otherwise there will only be 10 mini-batches from the subset. However, given enough data, in general larger batch size will lead to relatively superior performance, since the variance of the noisy variational objective in Equation (9) is smaller. To demonstrate the effectiveness of the Poisson model on massive amount of data (exploiting the stochastic algorithm’s ability to run without loading the entire dataset into memory), we also trained the model with the full training set with stochastic inference (PMF-Stoc-full). For the annotation task, we labelled each song with the top 20 tags based on the predicted score.

The results are reported in Table 2. In general, the performance of Poisson matrix factorization is comparably better for smaller codebook size J . Specifically, for stochastic inference, even if the amount of training data is relatively small, it is not only significantly faster than batch inference, but can also help improve the performance by quite a large margin. Finally, not surprisingly, PMF-Stoc-full dominates all the metrics, regardless of the size of the codebook, because it is able to learn from more data.

³<http://www.nltk.org/>

Codebook size	Model	Precision	Recall	F-score	AROC	MAP
$J = 256$	CBA	0.112 (0.007)	0.121 (0.008)	0.116	0.695 (0.005)	0.112 (0.006)
	ℓ_2 LogRegr	0.091 (0.008)	0.093 (0.006)	0.092	0.692 (0.005)	0.110 (0.006)
	PMF-Batch	0.113 (0.007)	0.105 (0.006)	0.109	0.647 (0.005)	0.094 (0.005)
	PMF-Stoc-10K	0.116 (0.007)	0.127 (0.007)	0.121	0.682 (0.005)	0.105 (0.006)
	PMF-Stoc-full	0.127 (0.008)	0.143 (0.008)	0.134	0.704 (0.005)	0.115 (0.006)
$J = 512$	CBA	0.120 (0.007)	0.127 (0.008)	0.124	0.689 (0.005)	0.117 (0.006)
	ℓ_2 LogRegr	0.096 (0.008)	0.108 (0.007)	0.101	0.693 (0.005)	0.113 (0.006)
	PMF-Batch	0.111 (0.007)	0.108 (0.006)	0.109	0.645 (0.005)	0.098 (0.005)
	PMF-Stoc-10K	0.112 (0.007)	0.128 (0.007)	0.120	0.687 (0.005)	0.110 (0.006)
	PMF-Stoc-full	0.130 (0.008)	0.154 (0.008)	0.141	0.715 (0.005)	0.122 (0.006)
$J = 1024$	CBA	0.118 (0.007)	0.126 (0.007)	0.122	0.692 (0.005)	0.117 (0.006)
	ℓ_2 LogRegr	0.113 (0.008)	0.129 (0.008)	0.120	0.698 (0.005)	0.115 (0.006)
	PMF-Batch	0.112 (0.007)	0.109 (0.006)	0.111	0.635 (0.005)	0.098 (0.006)
	PMF-Stoc-10K	0.111 (0.007)	0.127 (0.007)	0.118	0.687 (0.005)	0.111 (0.006)
	PMF-Stoc-full	0.127 (0.008)	0.146 (0.008)	0.136	0.712 (0.005)	0.120 (0.006)
$J = 2048$	CBA	0.124 (0.007)	0.129 (0.007)	0.127	0.689 (0.005)	0.117 (0.006)
	ℓ_2 LogRegr	0.115 (0.008)	0.137 (0.008)	0.125	0.698 (0.005)	0.118 (0.006)
	PMF-Batch	0.109 (0.007)	0.110 (0.006)	0.110	0.637 (0.005)	0.098 (0.006)
	PMF-Stoc-10K	0.107 (0.007)	0.124 (0.007)	0.115	0.682 (0.005)	0.106 (0.006)
	PMF-Stoc-full	0.120 (0.007)	0.147 (0.008)	0.132	0.712 (0.005)	0.118 (0.006)

Table 2. Annotation (evaluated using precision, recall, and F-score) and retrieval (evaluated using area under the receiver-operator curve (AROC) and mean average precision (MAP)) performance on the Million Song Dataset with various codebook sizes, from Codeword Bernoulli Average (CBA), ℓ_2 regularized logistic regression (ℓ_2 LogRegr), Poisson matrix factorization with batch inference (PMF-Batch) and stochastic inference by a single pass of the subset (PMF-Stoc-10K) and full data (PMF-Stoc-full). One standard error is reported in the parenthesis.

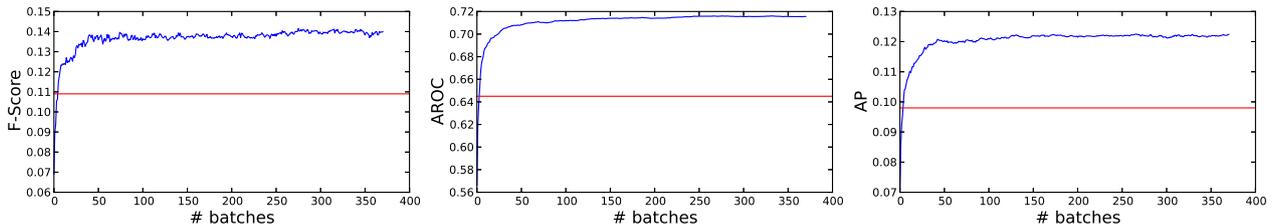


Figure 1. Improvement in performance with the number of mini-batches consumed for the PMF-Stoc-full system with $J = 512$. Red lines indicate the performance of PMF-Batch which is trained on 10k examples; that system’s performance is exceeded after fewer than 5 mini-batches.

Figure 1 illustrates how the metrics improve as more data becomes available for the Poisson matrix factorization model, showing how the F-score, AROC, and MAP improve with the number of (1000-element) mini-batches consumed up to the entire 371k training set. We see that initial growth is rapid, thanks to the natural gradient, with much of the benefit obtained after just 50 batches. However, we see continued improvement beyond this; it is reasonable to believe that if more data becomes available, the performance can be further improved.

Table 3 contains information on the qualitative performance of our model. The tagging model works by capturing correlations between semantic tags and acoustic codewords in each latent factor β_k . As discussed, when a new song arrives with missing tag information, only the portion of β_k corresponding to acoustic codewords is used, and the semantic tag portion of β_k is used to make predictions of the missing tags. Similar to related topic models [9], we

can therefore look at the highly probable tags for each β_k to understand what portion of the acoustic codeword space is being captured by that factor, and whether it is musically coherent. We show an example of this in Table 3, where we list the top 7 tags from 9 latent factors β_k learned by our model with $J = 512$. We sort the tags according to expected relevance under the variational distribution $\mathbb{E}_q[\beta_{kd}]$. This shows which tags are considered to have high probability for a song that has the given factor expressed. As is evident, each factor corresponds to a particular aspect of a music genre. We note that other factors contained similarly coherent tag information.

6. DISCUSSION AND FUTURE WORK

We present a codebook-based scalable music tagging model with Poisson matrix factorization. The system learns the joint behavior of acoustic features and semantic tags, which

“Pop”	“Indie”	“Jazz”	“Classical”	“Metal”	“Reggae”	“Electronic”	“Experimental”	“Country”
pop	indie	chillout	piano	metal	reggae	house	instrumental	country
female vocal	rock	loungue	instrumental	death metal	funk	electro	ambient	classic country
dance	alternative	chill	ambient	thrash metal	funky	electronic	experimental	male vocal
electronic	indie rock	downtempo	classic	brutal death metal	dance	dance	electronic	blues
sexy	post punk	smooth jazz	beautiful	grindcore	hip-hop	electric house	psychedelic	folk
love	psychedelic	relax	chillout	heavy metal	party	techno	progressive	love songs
synth pop	new wave	ambient	relax	black metal	sexy	minimal	rock	americana

Table 3. Top 7 tags from 9 latent factors for PMF-Stoc-full with $J = 512$. For each factor, we assign the closest music genre on top. As is evident, each factor corresponds to a particular aspect of a music genre.

can be used to infer the most appropriate tags given the audio alone. The Poisson model is naturally less sensitive to zero values than some alternatives, making it a good match to “noisy” training examples derived from real users’ taggings, where the fact that no user has applied a tag does not necessarily imply that the term is irrelevant. By learning this model using stochastic variational inference, we are able to efficiently exploit much larger training sets than are tractable using batch approaches, making it feasible to learn from an entire set of over 370k tagged examples. Although much of the improvement comes in the earlier iterations, we see continued improvement implying this approach can benefit from much larger, effectively unlimited sources of tagged examples, as might be available on a commercial music service with millions of users.

There are a few areas where our model can be easily developed. For example, stochastic variational inference requires we set the learning rate parameters t_0 and κ , which is application-dependent. By using adaptive learning rates for stochastic variational inference [13], model inference can converge faster and to a better local optimal solution. From a modeling perspective, currently the hyperparameters for weights θ are fixed, indicating that the sparsity level of the weight for each song is assumed to be the same *a priori*. Alternatively we could put *song-dependent* hyper-priors on the hyperparameters of θ to encode the intuition that some of the songs might have denser weights because more tagging information is available. This would offer more flexibility to the current model.

7. ACKNOWLEDGEMENTS

The authors would like to thank Matthew Hoffman for helpful discussion. This work was supported in part by NSF grant IIS-1117015.

8. REFERENCES

- [1] Last.fm dataset, the official song tags and song similarity collection for the Million Song Dataset. <http://labrosa.ee.columbia.edu/millionsong/lastfm>.
- [2] T. Bertin-Mahieux, D. Ellis, B. Whitman, and P. Lamere. The Million Song Dataset. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 591–596, 2011.
- [3] L. Bottou. Online learning and stochastic approximations. *On-line learning in neural networks*, 17(9), 1998.
- [4] A. T. Cemgil. Bayesian inference for nonnegative matrix factorisation models. *Computational Intelligence and Neuroscience*, 2009.
- [5] 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*, pages 385–392, 2007.
- [6] K. Ellis, E. Coviello, A. Chan, and G. Lanckriet. A bag of systems representation for music auto-tagging. *Audio, Speech, and Language Processing, IEEE Transactions on*, 21(12):2554–2569, 2013.
- [7] M. Hoffman, D. Blei, and P. Cook. Easy as CBA: A simple probabilistic model for tagging music. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 369–374, 2009.
- [8] M. Hoffman, D. Blei, and P. Cook. Bayesian nonparametric matrix factorization for recorded music. In *Proceedings of the 27th Annual International Conference on Machine Learning*, pages 439–446, 2010.
- [9] M. Hoffman, D. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.
- [10] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.
- [11] D. Liang, M. Hoffman, and D. Ellis. Beta process sparse non-negative matrix factorization for music. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 375–380, 2013.
- [12] J. Paisley, D. Blei, and M.I. Jordan. Bayesian nonnegative matrix factorization with stochastic variational inference. In E.M. Airoldi, D. Blei, E.A. Erosheva, and S.E. Fienberg, editors, *Handbook of Mixed Membership Models and Their Applications*. Chapman and Hall/CRC, 2015.
- [13] R. Ranganath, C. Wang, D. Blei, and E. Xing. An adaptive learning rate for stochastic variational inference. In *Proceedings of The 30th International Conference on Machine Learning*, pages 298–306, 2013.
- [14] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*, pages 1257–1264, 2008.
- [15] D. Tingle, Y.E. Kim, and D. Turnbull. Exploring automatic music annotation with acoustically-objective tags. In *Proceedings of the international conference on Multimedia information retrieval*, pages 55–62. ACM, 2010.
- [16] 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, 2008.
- [17] B. Xie, W. Bian, D. Tao, and P. Chordia. Music tagging with regularized logistic regression. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 711–716, 2011.