ONLINE LEARNING OF TIME-FREQUENCY PATTERNS

Jose F. Ruiz-Muñoz¹, Raviv Raich², Mauricio Orozco-Alzate¹ and Xiaoli Z. Fern²

¹Universidad Nacional de Colombia - Sede Manizales - Signal Processing and Recognition Group, Manizales, 17004-Colombia
²School of EECS, Oregon State University, Corvallis, OR 97331-5505, USA

ABSTRACT

We present an online method to learn recurring time-frequency patterns from spectrograms. Our method relies on a convolutive decomposition that estimates sequences of spectra into time-frequency patterns and their corresponding activation signals. This method processes one spectrogram at a time such that in comparison with a batch method, the computational cost is reduced proportionally to the number of considered spectrograms. We use a first-order stochastic gradient descent and show that a monotonically decreasing learning-rate works appropriately. Furthermore, we suggest a framework to classify spectrograms based on the estimated set of time-frequency patterns. Results, on a set of synthetically generated spectrograms and a real-world dataset, show that our method finds meaningful time-frequency patterns and that it is suitable to handle a large amount of data.

Index Terms— dictionary learning, non-negative matrix factorization, online learning, classification.

1. Introduction

Learning time-frequency patterns is helpful for both supervised and unsupervised analyses of acoustic signals. For this purpose, the mathematical model known as dictionary learning (DL) has been used. Estimation of such a model is usually formulated as a constrained optimization problem that includes a data fit term between the signal and a combination of a set of patterns —called dictionary—and their corresponding coefficients for weighting those patterns—called activations.

Depending on the problem, a physical meaning can be attributed to patterns and coefficients [1]. For example, for bioacoustic signals, dictionary patterns can be associated with different sound sources, e.g., bird species vocalization, and coefficients can be related to the time when the vocalizations are emitted. For later analysis, a DL algorithm should appropriately recover the original signal and satisfy the constraints, e.g., norm-constraints or non-negativity. Nevertheless, those algorithms are usually computationally expensive; therefore, to scale up and allow handling a large amount of data, it is important to consider complexity and memory requirements [2].

One approach for DL, which has been widely applied in machine learning and digital signal processing, is based on nonnegative matrix factorization (NMF) [3]. Particularly, NMF allows extracting meaningful information from audio recordings that contain mixtures of sounds [4, 5]. In order to apply NMF, the audio signal is usually represented by its spectrogram [6–8]. NMF has been successfully applied to various audio applications including automatic transcription, music analyses and blind source separation [9, 10]. NMF is formulated as an optimization problem (sparsity constraints are often added) that minimizes the least-squares error or the generalized Kullback-Leibler divergence [11] between the measured signal and its decomposition.

Using NMF a spectrum is decomposed into a product of two matrices: one corresponding to a collection of 1-D spectra (which forms the dictionary) and another corresponding to their activations in time. An alternative model is the convolutive non-negative matrix factorization (cNMF) in which each pattern of the dictionary is a matrix that corresponds to a sequence of 1-D spectra (time-frequency pattern) [12, 13]. The resulting time-frequency patterns provide useful information related to relevant temporal structures contained in the recordings [14]. Nevertheless, when dealing with large data (e.g., in bioacoustics), traditional cNMF algorithms become computationally expensive and demand large memory resources. To reduce the computational complexity and memory consumption, low-rank approximations are applied [15]. However, this approach generally results in information loss. An alternative approach to alleviate the processing requirements is using online algorithms. For instance, in [16], an algorithm for learning 1-D patterns using stochastic gradient descent is proposed and in [17], an online version of the cNMF algorithm proposed in [18] is introduced.

In this paper, we propose an unsupervised online version of the algorithm originally presented in [19]. For this purpose, we use a first-order stochastic gradient descent approach. Our algorithm progressively updates the dictionary with each incoming spectrogram. Additionally, we propose a scheme for classifying audio signals based on features extracted from the convolutive decomposition of the spectrograms. We evaluate and compare the proposed approach on synthetic and real-world datasets.

2. Learning time-frequency patterns

2.1. Convolutive decomposition

We approximate a spectrogram \( Y \in \mathbb{R}^{F \times T} \) by the linear combination of \( K \) shifted time-frequency patterns \( \mathbf{D}_k = [d_{k1} \ldots d_{kF}]^T \in \mathbb{R}^{F \times W} \) where \( d_{kf} \in \mathbb{R}^{W \times 1} \) is the \( k \)-th time
pattern at frequency $f$, and $W$ is the length of each time-frequency pattern. This approximation is expressed by the discrete convolution operation\(^1\) as follows:

$$\mathbf{Y}(f, t) \approx \sum_{k=1}^{K} (\mathbf{a}_k \ast \mathbf{d}_{k,f})(t)$$

where $\mathbf{Y}(f, t)$ is an entry of $\mathbf{Y} \in \mathbb{R}^{F \times T}$ at frequency $f \in [1, F]$ and time $t \in [1, T]$, and $\mathbf{a}_k = [a_k(1) \ldots a_k(L)]^\top \in \mathbb{R}^{F \times L}$ ($L = T + W - 1$) is the activation signal corresponding to $\mathbf{D}_k$. The convolution is performed without zero-padded edges; therefore, the convolution between $\mathbf{a}_k \in \mathbb{R}^{L \times 1}$ and $\mathbf{d}_{k,f} \in \mathbb{R}^{W \times 1}$ produces a vector of length $T$, i.e., $[\mathbf{a}_k \ast \mathbf{d}_{k,f}] \in \mathbb{R}^{F \times 1}$. The full dictionary $\mathbf{D}$ is built by stacking all $\mathbf{D}_k$, such that $\mathbf{D} \in \mathbb{R}^{K \times F \times W}$. Similarly, the set of activation signals $\mathbf{a}_k$ forms the matrix $\mathbf{A} = [\mathbf{a}_1 \ldots \mathbf{a}_K] \in \mathbb{R}^{L \times K}$.

Dictionary and activations are estimated by solving an optimization problem that aims to minimize the least-squares error and the $L_1$-norm of the activations to induce sparsity:

$$\min_{\mathbf{D}, \mathbf{A}} \ell(\mathbf{Y}, \mathbf{D}, \mathbf{A}) = \frac{1}{2} \sum_{f=1}^{F} \sum_{t=1}^{T} \left( \mathbf{Y}(f, t) - \sum_{k=1}^{K} (\mathbf{a}_k \ast \mathbf{d}_{k,f})(t) \right)^2 + \lambda \sum_{k=1}^{K} \sum_{t=1}^{T} |\mathbf{a}_k(t)|$$

subject to $\sum_{f=1}^{F} \sum_{t=1}^{T} (\mathbf{d}_{k,f}(t))^2 \leq 1$, $\forall 1 \leq k \leq K$.

In [19], an iterative rule for updating the $k$-th time-frequency pattern is proposed (based on a convexization procedure by a surrogate loss function [20]) as follows:

$$\mathbf{D}_{k(p)} = \Pi(\mathbf{D}_{k(p-1)} + \eta_d \nabla_{\mathbf{D}_k} \ell(\mathbf{Y}, \mathbf{D}_{k(p-1)}, \mathbf{A})),$$  \hspace{1cm} (3)

where $(p)$ denotes the current iteration, the projection $\Pi$ is defined as

$$\Pi(\mathbf{D}) = \begin{cases} \mathbf{D} & \text{if } \|\mathbf{D}\| \leq 1 \\ \mathbf{D} + \frac{1}{\|\mathbf{D}\|} \mathbf{D} & \text{otherwise} \end{cases}, \forall \mathbf{D}$$

$\eta_d$ is the step-size, and the gradient of the loss function wrt $\mathbf{D}_k$ is

$$\nabla_{\mathbf{D}_k} \ell(\mathbf{Y}, \mathbf{D}, \mathbf{A}) = [\mathbf{v}_{d_{k,1}} \ldots \mathbf{v}_{d_{k,F}}]^\top \in \mathbb{R}^{F \times W}$$

where $\mathbf{v}_{d_{k,f}} = \mathbf{T}_{ak}^\top \mathbf{y}_f - \sum_{k=1}^{K} \mathbf{T}_{ak} \mathbf{d}_{k,f}^{(p-1)} \in \mathbb{R}^{W \times 1}$, $\mathbf{y}_f = \mathbf{Y}(f, 1) \ldots \mathbf{Y}(f, T) \in \mathbb{R}^{F \times T}$ and $\mathbf{T}_{ak} = \text{TOEPLITZ}(\mathbf{a}_k, W, L, W) \in \mathbb{R}^{T \times W}$ (see Appendix A). A safe step-size $\eta_t = 1/\gamma_f \max\gamma_f$ where $\gamma_f = \lambda_{\text{max}}([\mathbf{u}_1, \mathbf{u}_2]^\top \mathbf{T}_{A}^\top \mathbf{T}_{A} [\mathbf{u}_1, \mathbf{u}_2])$, $\lambda_{\text{max}}(\cdot)$ denotes the maximum eigenvalue, $\mathbf{u}_1 = \mathbf{v}_{d_{k,f}}/\|\mathbf{v}_{d_{k,f}}\| \in \mathbb{R}^{W \times 1}$, $\mathbf{v}_{d_{k,f}} = [\mathbf{v}_{d_{k,1}} \ldots \mathbf{v}_{d_{k,F}}]^\top \in \mathbb{R}^{F \times W}$, $\mathbf{u}_2 = \mathbf{d}_{k,f}/\|\mathbf{d}_{k,f}\| \in \mathbb{R}^{W \times 1}$, $\mathbf{d}_{k,f} = [\mathbf{d}_{k,1} \ldots \mathbf{d}_{k,F}]^\top \in \mathbb{R}^{F \times W}$, and $\mathbf{T}_{A} = [\mathbf{T}_{a_1} \ldots \mathbf{T}_{a_K}] \in \mathbb{R}^{T \times kW}$. An update rule for activations is also given in [19]:

$$\mathbf{A}_{(p)} = \arg\min_{\mathbf{A}_{(p-1)}} \ell(\mathbf{Y}, \mathbf{D}, \mathbf{A}_{(p-1)}).$$  \hspace{1cm} (5)

\(^1\)Discrete convolution operation: $$(u \ast v)(n) = \sum_{m} a(n - m + 1)v(m)$$

2.2. Online dictionary learning

In order to learn a dictionary from a set of $N$ stacked spectrograms \{$\mathbf{Y}^{(1)} \ldots \mathbf{Y}^{(N)}$\}, in [19], the update rule of (3) is applied as follows:

$$\mathbf{D}_{k(p)} = \Pi(\mathbf{D}_{k(p-1)} + \frac{1}{N} \sum_{i=1}^{N} \eta_d^{(i)} \nabla_{\mathbf{D}_k} \ell(\mathbf{Y}^{(i)}, \mathbf{D}_{(p-1)}, \mathbf{A}^{(i)}))$$

Alternatively, we propose an online algorithm that updates the time-frequency patterns according to the current spectrogram and the ones observed in the past. Therefore, we define the following loss function:

$$g_N(\mathbf{D}) := \frac{1}{N} \sum_{i=1}^{N} \ell(\mathbf{Y}^{(i)}, \mathbf{D}, \mathbf{A}^{(i)})$$

where $\mathbf{A}^{(i)}$ is the estimated activation matrix that corresponds to the $i$-th spectrogram $\mathbf{Y}^{(i)}$. Hence, the dictionary learning task consists of minimizing the expected cost

$$g(\mathbf{D}) := \mathbb{E}_{\mathbf{Y}^{(i)}}[\ell(\mathbf{Y}, \mathbf{D}, \mathbf{A})] := \lim_{N \to \infty} g_N(\mathbf{D}).$$

For this purpose, we update $\mathbf{D}_k$ by using the first-order stochastic gradient descent algorithm \[16, 21\] as follows:

$$\mathbf{D}_{k(p)} = \Pi(\mathbf{D}_{k(p-1)} + \mu \eta_d \nabla_{\mathbf{D}_k} \ell(\mathbf{Y}^{(i)}, \mathbf{D}_{(p-1)}, \mathbf{A}^{(i)}))$$  \hspace{1cm} (7)

where $i = \begin{cases} N \text{ if } \text{mod}(p, N) = 0 \\ \text{mod}(p, N) \text{ otherwise} \end{cases}$, and $\mu$ is the factor for scaling the gradient, also known as learning-rate. Notice that one iteration of (6) requires computing $N$ times the gradient $\nabla_{\mathbf{D}_k} \ell(\mathbf{Y}, \mathbf{D}, \mathbf{A})$ but (7) requires computing this gradient only once. According to [22], two learning-rate schedules commonly used in matrix factorization are:

- **Fixed Schedule (FS):** the learning rate $\mu_p = \alpha \forall p$ is fixed throughout the online learning process.
- **Monotonically Decreasing Schedule (MDS):** the learning rate monotonically decreases each time that a new spectrogram is observed. Two options are: i) MDS1: $\mu_p = \frac{\alpha}{p}$, and ii) MDS2: $\mu_p = \frac{\alpha}{p^2}$.

Our online DL process, which aims to compute $\mathbf{D}$ and $\mathbf{A}$, alternatively updates both of them. Therefore, in the $p$-th iteration, activations are updated, as indicated in (5), by

$$\mathbf{A}_{(p)} = \arg\min_{\mathbf{A}_{(p-1)}} \ell(\mathbf{Y}^{(i)}, \mathbf{D}_{(p-1)}, \mathbf{A}^{(i)}_{(p-1)}) \mathbb{P}_i.$$  \hspace{1cm} (A)

Subsequently, the dictionary $\mathbf{D}_{(p-1)}$ is updated by (7). Note that due to the non-convex nature of the problem convergence to a global optimum is not guaranteed.

3. Classifying spectrograms

Our classification task consists of mapping the vector representation of a spectrogram $\mathbf{x} \in \mathbb{R}^N$ to a categorical (class) label $\mathcal{y} \in \{-1, 1\}$. The label in the binary classification setting indicates the presence ($\mathcal{y} = 1$) or absence ($\mathcal{y} = -1$) of the target class in a given spectrogram.

We divide the experiments into two stages: training and test. In the training stage, the dictionary is estimated by
using the proposed online DL method, which receives a sequence of spectrograms. The estimated dictionary is used to extract the feature vector \( x_t = [x_{t1}, \ldots, x_{tK}] \in \mathbb{R}^K \) for the \( i \)-th spectrogram \( Y_i \in \mathbb{R}^{F \times T} \) in a training set, whose entries are computed as follows: \( x_{tk} = \max_i (\sum_j \langle a_{kj} \circ y_i(j) \rangle, 0) \) where \( h_{kj}^{(i)} = \mathbf{d}_{kj} \ast y_{i}^{(j)} \in \mathbb{R}^{(T+W-1) \times 1} \), and \( \mathbf{d} \) denotes the vector in reversed order.

A support vector machine (SVM) classifier is trained by using this representation, and the dictionary estimated in the training stage is used to compute the vector representation of the test set. Labels are assigned by the trained SVM.

4. Experiments

4.1. Experiments on an artificial dataset

Initially, we perform experiments in a collection of 1000 synthetically generated spectrograms containing some of different time-frequency patterns. The three dimensional binary label vector of each spectrogram \( Y_i \in \mathbb{R}^{16 \times 30} \) indicates the presence or absence of each class in the spectrogram. For each class, two types of time-frequency patterns of length 10 are considered. Therefore, the original dictionary is formed by six basic time-frequency patterns (see Fig. 3a). The free parameters in the proposed online DL method are: length of window \( W \), number of dictionary words \( K \), learning-rate \( \mu \), and \( \ell_1 \)-norm regularization parameter \( \lambda \). We fix \( W = 10 \), since this parameter is known beforehand, and \( K = 8 \) (we over-estimate the size of the dictionary in order to avoid missing a time-frequency pattern). Estimation of the remaining parameters is described below.

We compare the schedules of \( \mu \) described in Sec. 2.2 and tune the parameter \( \alpha \). Figure 1 contains the reconstruction error of a test set of 30 spectrograms and the actual sparsity of their activations (rate of non-zero entries) as a function of the number of observed spectrograms for a set of different values of \( \alpha \) (for \( \lambda = 0.1 \)). According to this experiment, the FS schedule works well for moderate values of \( \alpha \) tuning off initial instability at a large value of \( \alpha \) with slow convergence for a small value of \( \alpha \).

Figure 2 shows the reconstruction error and the rate of non-zero entries in function of \( \lambda \) after observing 1000 spectrograms (the learning-rate is MDS1 for a set of different values of \( \alpha \)). Results confirm the trade-off in the objective function between the reconstruction error and the \( \ell_1 \)-norm constraint. Figures 3a and 3b show the original set of time-frequency patterns and the estimated ones (with MDS1, \( \alpha = 100 \) and \( \lambda = 0.1 \), respectively).

4.2. Experiments on real-world datasets

To validate the proposed method, we perform experiments on the MLSP 2013 Bird Classification Challenge dataset,\(^2\) which was collected in the H. J. Andrews (HJA) Long-Term Experimental Research Forest in Oregon (USA). Table 1 shows the number of recordings and classes of this dataset.

The classification experiments consider the following: i) for each class a binary (presence/absence) classification problem is considered; ii) the dataset is randomly divided into

\(^2\)https://www.kaggle.com/c/mlsp-2013-birds
produces the best performance when classifying GSFL, CBCH, and WETA.

4.3. Computational cost: Batch Learning vs Online Learning

In order to show the computational benefits of our method (online learning), we compare it against a batch learning approach. We call batch learning to the DL method that updates the time-frequency patterns by (6), which requires the whole set of spectrograms to estimate the gradient. These experiments were carried out on a CPU with Processor 2.20GHz × 8 and Memory 3.8 GB.

Figure 4 compares the time needed to reconstruct 20 (randomly selected) spectrograms from the MLSP 2013 dataset by batch learning and online learning. Note that since there are more than 20 iterations, the spectrograms are observed several times in the online case. We observe that the error is not monotonically decreasing at the beginning for online learning. However, the reconstruction error for both the online and batch methods converges to a similar value after several iterations. Furthermore, as expected, the online learning is faster than the batch learning by a factor of the number of spectrograms reconstructed at each iteration.

![Figure 3: Original and estimated sets of time-frequency patterns.](image)

![Figure 4: Comparison of reconstruction error (top) and computational cost (bottom) between batch learning and online learning.](image)

5. Conclusion

We described an online DL method based on stochastic gradient descent, which learns time-frequency patterns from large datasets of spectrograms. Our algorithm is based on a convolutive DL method with the additive update rule. The proposed method handles better the computational resources than its batch counterpart. Therefore, it could be preferred for analyzing large datasets. Experiments on an artificial dataset and a real-world dataset show that the method recovers appropriately the original spectrograms and finds meaningful time-frequency patterns for classification outperforming a state-of-the-art DL method and the classification based on the raw frequency information.

A Toepplitz matrix $\text{TOEPLITZ}(x, \tau_c, \tau_1, \tau_2)$ is constructed as follows:

$$
\begin{bmatrix}
\phi(x, \tau_c) & \phi(x, \tau_c - 1) & \cdots & \phi(x, \tau_c - \tau_2 + 1) \\
\phi(x, \tau_c + 1) & \phi(x, \tau_c) & \cdots & \phi(x, \tau_c - \tau_2 + 2) \\
\vdots & \vdots & \ddots & \vdots \\
\phi(x, \tau_1) & \phi(x, \tau_1 - 1) & \cdots & \phi(x, \tau_1 - \tau_2 + 1)
\end{bmatrix}
$$

where $x \in \mathbb{R}^m$, $\tau_c, \tau_1, \tau_2 \in \mathbb{N}$ and

$$
\phi(x, \tau) = \left\{ \begin{array}{ll}
\phi(x, \tau), & 1 \leq \tau \leq \dim(x) \\
0, & \text{otherwise}
\end{array} \right.
$$

6. References


