Online PLCA for Real-time Semi-supervised Source Separation

Zhiyao Duan[1]*, Gautham J. Mysore[2] and Paris Smaragdis[2,3]

[1] EECS Department, Northwestern University,
[2] Advanced Technology Labs, Adobe Systems Inc.,
[3] University of Illinois at Urbana-Champaign

Abstract. Non-negative spectrogram factorization algorithms such as probabilistic latent component analysis (PLCA) have been shown to be quite powerful for source separation. When training data for all of the sources are available, it is trivial to learn their dictionaries beforehand and perform supervised source separation in an online fashion. However, in many real-world scenarios (e.g. speech denoising), training data for one of the sources can be hard to obtain beforehand (e.g. speech). In these cases, we need to perform semi-supervised source separation and learn a dictionary for that source during the separation process. Existing semi-supervised separation approaches are generally offline, i.e. they need to access the entire mixture when updating the dictionary. In this paper, we propose an online approach to adaptively learn this dictionary and separate the mixture over time. This enables us to perform online semi-supervised separation for real-time applications. We demonstrate this approach on real-time speech denoising.

1 Introduction

In recent years, non-negative matrix factorization (NMF) and its probabilistic counterparts such as probabilistic latent component analysis (PLCA) have been widely used for source separation [1]. The basic idea is to represent the magnitude spectrum of each time frame of the mixture signal as a linear combination of dictionary elements from source dictionaries. In the language of PLCA, for a sound mixture of two sources, this can be written as:

$$P_t(f) \approx \sum_{z \in \mathcal{S}_1 \bigcup \mathcal{S}_2} P(f|z) P_t(z) \text{ for } t = 1, \cdots, T \tag{1}$$

where T is the total number of frames; $P_t(f)$ is the normalized magnitude spectrum of the t-th frame of the mixture; $P(f|z)$ for $z \in \mathcal{S}_1$ and $z \in \mathcal{S}_2$ represent the elements (analogous to basis vectors) of the dictionaries of source 1 and source 2 respectively. $P_t(z)$ represents the activation weights of the different dictionary elements at time t. All distributions are discrete.

* This work was performed while interning at Adobe Systems Inc.

Given a mixture spectrogram, we can estimate the dictionary elements and activation weights using the expectation–maximization (EM) algorithm. The source spectra in the t-th frame can then be reconstructed as $\sum_{z\in\mathcal{S}_1} P(f|z)P_t(z)$ and $\sum_{z\in\mathcal{S}_2} P(f|z)P_t(z)$, respectively. This is unfortunately a highly underconstrained problem and rarely leads to useful parameter estimates. One way to address this issue is to perform *supervised* source separation [1], in which we first learn the dictionaries for both sources from their isolated training data. Then in the separation stage, we fix these dictionaries and only the estimate the activation weights, from which we can reconstruct the spectra of each source.

However, in a lot of real-world problems, training data for one source might be hard to obtain beforehand. For example, in the application of speech denoising, we want to separate speech from noise. It is relatively easy to obtain training data for noise, but hard for speech. In these cases, we need to perform *semi-supervised* source separation [1], where we first learn the dictionary for one source (e.g. noise) from its training data beforehand, and then learn the dictionary for the other source (e.g. speech) in addition to the activation weights of both sources from the mixture. Finally, separation can be performed.

For supervised separation, the algorithm in [1] is intrinsically online, since the activation weights in different frames are estimated independently. For semi-supervised separation, however, the algorithm in [1] needs to access the entire mixture to learn the dictionary for the un-pretrained source, hence is offline.

In recent years, researchers have proposed several online NMF algorithms for dictionary learning in different applications (e.g. dictionary learning for image databases [2], document clustering [3], audio reconstruction [4]). The idea is to learn a dictionary to well explain the entire input data, after processing all the inputs, in an online fashion. However, we argue that these algorithms are not suitable for real-time semi-supervised source separation. The reason is that these algorithms only care about the final learned dictionary, after processing all of the input frames. They do not care about the intermediate estimates of the learned dictionary during processing the input frames. Therefore, the dictionary learned after receiving the current frame is not necessarily good enough to explain that frame and to separate it. In fact, processing all of the input frames once is often not enough and it has been shown that cycling over the input data set several times and randomly permuting samples at each cycle [2–4] improves the results.

In this paper, we propose an online PLCA algorithm tailored for real-time semi-supervised source separation. We learn the dictionary for the source that does not have training data, from the mixture, and apply it to separate the mixture, in an online fashion. When a new mixture frame comes in, the dictionary is adaptively updated to explain the current frame instead of explaining the entire mixture frames. In this way, we can use a much smaller-sized dictionary compared to the offline PLCA. We show that the performance of the proposed algorithm is almost as good as that of the offline PLCA algorithm (numerically equivalent to offline NMF using KL divergence), but significantly better than an existing online NMF algorithm for this application.

2 Proposed Algorithm

For the real-time source separation problem of two sources, assuming that some isolated training excerpts of source 1 ($\mathcal{S}_1$) are available beforehand and are long enough to capture $\mathcal{S}_1$'s characteristics, we can follow the semi-supervised source separation paradigm presented in Section 1 [4]. We first learn a dictionary of $\mathcal{S}_1$ from the spectrogram of its training excerpts beforehand. Then during separation, as each incoming mixture frame arrives, we learn and update the dictionary of $\mathcal{S}_2$ using the proposed online PLCA algorithm, with $\mathcal{S}_1$'s dictionary fixed.

2.1 Online Separation and Dictionary Learning

In order to separate the t-th frame of the mixture signal, we need to decompose its magnitude spectrum using Eq. (1). Here, $P(f|z)$ for $z \in \mathcal{S}_1$ is the pre-learned dictionary of $\mathcal{S}_1$ from training excerpts, and is kept fixed in this decomposition. We need to estimate the dictionary $P(f|z)$ for $z \in \mathcal{S}_2$ and activation weights $P_t(z)$ for all z, such that the decomposition is as accurate as possible, i.e.

$$\underset{P(f|z) \text{ for } z \in \mathcal{S}_2,\ P_t(z) \text{ for all } z}{\arg\min} \quad d_{KL}(P_t(f)||Q_t(f)) \tag{2}$$

where d_{KL} is the KL divergence between two distributions. $P_t(f)$ is the normalized mixture spectrum at time t and $Q_t(f)$ is the reconstructed mixture spectrum i.e. the LHS and RHS of Eq. (1).

However, this is a highly unconstrained problem, since the number of parameters to estimate is much more than the number of equations (i.e. the number of frequency bins in Eq. (1)), even if there is only one element in $\mathcal{S}_2$'s dictionary. A trivial solution that makes the KL divergence in Eq. (2) equal to zero is to use only one dictionary element in $\mathcal{S}_2$, such that the dictionary element is the same as the mixture and the corresponding activation weight equals to one (with all other weights being zero). In practice, this trivial solution is almost always achieved, essentially making the separated source 2, the same as the mixture.

We therefore need to constrain the dictionary of source 2 to avoid this overfitting. We do this by requiring $\mathcal{S}_2$'s dictionary to not only explain $\mathcal{S}_2$'s spectrum in the current frame, but also those in a number of previous frames. We denote this set of frames as $\mathcal{B}$, representing a running buffer. We update $\mathcal{S}_2$'s dictionary in every frame using $\mathcal{B}$. We also set the size of $\mathcal{S}_2$'s dictionary to be much smaller than the size of $\mathcal{B}$. This avoids the overfitting because a compact dictionary will now be used to explain a much larger number of frames.

Clearly these buffer frames need to contain $\mathcal{S}_2$'s spectra, otherwise the dictionary will be incorrectly learned. We will describe how to determine if a mixture frame contains $\mathcal{S}_2$'s spectrum or not in Section 2.2. Suppose we can identify the previous mixture frames that contain $\mathcal{S}_2$'s spectra, we need to decide which ones

[4] It is straightforward to extend this to N sources if isolated training excerpts for N-1 sources are available.

to include in $\mathcal{B}$. On one hand, $\mathcal{S}_2$'s spectra in the buffer frames need to be different from those in the current frame, so that the learned $\mathcal{S}_2$'s dictionary does not overfit the mixture spectra in the current frame. On the other hand, we do not want $\mathcal{S}_2$'s spectra in the buffer frames to be too different from those in the current frame so that we have a more "localized" and compact dictionary. In the real-time source separation problem, it is intuitive to use the L most recent identified mixture frames to balance the tradeoff, as they are not the same as the current frame but tend to be similar. Based on this, the objective becomes:

$$\underset{P(f|z) \text{ for } z\in\mathcal{S}_2,\ P_t(z) \text{ for all } z}{\arg\min} d_{KL}(P_t(f)||Q_t(f)) + \frac{\alpha}{L}\sum_{s\in\mathcal{B}} d_{KL}(P_s(f)||Q_s(f)) \quad (3)$$

where α is the tradeoff between the original objective (good reconstruction of the current frame) and the added constraint (good reconstruction of the L frames in the running buffer $\mathcal{B}$).

With this new objective, we learn $\mathcal{S}_2$'s dictionary and the current frame's activation weights. However, we fix the activation weights of the buffer frames as the values learned when separating them. There are two advantages of fixing them than updating them: First, it makes the algorithm faster. Second, it imposes a heavier constraint on $\mathcal{S}_2$'s dictionary that the newly learned dictionary must not deviate from those learned in the buffer frames too much. We use the EM algorithm to optimize Eq. (3), which is described in Algorithm 1.

Algorithm 1 Single Frame Dictionary Learning

Require: $\mathcal{B}$ (buffer frames set), V_{fs} for $s \in \mathcal{B}\bigcup\{t\}$ (normalized magnitude spectra of buffer frames and current frame), $P(f|z)$ for $z \in \mathcal{S}_1$ ($\mathcal{S}_1$'s dictionary), $P(f|z)$ for $z \in \mathcal{S}_2$ (initialization of $\mathcal{S}_2$'s dictionary), $P_s(z)$ for $s \in \mathcal{B}\bigcup\{t\}$ and $z \in \mathcal{S}_1\bigcup\mathcal{S}_2$ (input activation weights of buffer frames and current frame), α (tradeoff between reconstruction of buffer frames and current frame), M (number of EM iterations).

1: **for** $i = 1$ to M **do**

2: E Step:

$$P_s(z|f) \leftarrow \frac{P_s(z)P(f|z)}{\sum_{z\in\mathcal{S}_1\bigcup\mathcal{S}_2} P_s(z)P(f|z)},\ \text{for } s \in \mathcal{B}\bigcup\{t\}. \quad (4)$$

3: M Step:

$$\phi(f|z) \leftarrow V_{ft}P_t(z|f) + \frac{\alpha}{|\mathcal{B}|}\sum_{s\in\mathcal{B}} V_{fs}P_s(z|f),\ \text{for } z \in \mathcal{S}_2, \quad (5)$$

$$\phi_t(z) \leftarrow \sum_f V_{ft}P_t(z|f),\ \text{for } z \in \mathcal{S}_1\bigcup\mathcal{S}_2. \quad (6)$$

Normalize $\phi(f|z)$ and $\phi_t(z)$ to get $P(f|z)$ and $P_t(z)$ respectively.

4: **end for**

5: **return** learned dictionary $P(f|z)$ for $z \in \mathcal{S}_2$ and activation weights $P_t(z)$ for $z \in \mathcal{S}_1\bigcup\mathcal{S}_2$ of the current frame t.

2.2 Mixture Frame Classification

The problem that has not been addressed in Section 2.1 is how to determine if a mixture frame contains $\mathcal{S}_2$'s spectrum or not. For a mixture of two sound

sources, we can address this by decomposing the magnitude spectrum of the mixture frame using only the learned $\mathcal{S}_1$'s dictionary as follows:

$$P_t(f) \approx \sum_{z \in \mathcal{S}_1} P(f|z)P_t(z) \tag{7}$$

Since the dictionary is fixed, we learn only the activation weights. If the KL divergence between $P_t(f)$ and the RHS is smaller than a threshold θ_{KL}, it means the mixture spectrum can be well explained by only using $\mathcal{S}_1$'s dictionary, hence $\mathcal{S}_2$'s spectrum is not likely to be present. Otherwise, $\mathcal{S}_1$'s dictionary is not enough to explain the spectrum, hence $\mathcal{S}_2$'s spectrum is likely to be present.

We learn the threshold θ_{KL} by decomposing $\mathcal{S}_1$'s training excerpts again, with its pre-learned dictionary. We calculate the mean and standard deviation of the KL divergences of all the frames, and set the threshold as $\theta_{KL} = mean + std$.

If the current frame is classified as not containing $\mathcal{S}_2$, then we do not include it in the running buffer $\mathcal{B}$. However, just in case there is some amount of $\mathcal{S}_2$ in the frame, we still perform supervised separation on the frame using the pre-learned dictionary of $\mathcal{S}_1$ and the previously updated dictionary of $\mathcal{S}_2$. If the current frame is classified as containing $\mathcal{S}_2$, we run Algorithm 1 on it to update $\mathcal{S}_2$'s dictionary and separate the frame. After separation, we include this frame into the running buffer $\mathcal{B}$ for future use.

2.3 Algorithm Summary

The whole online semi-supervised source separation algorithm is summarized in Algorithm 2. Note that in Line 6 we make a "warm" initialization of $\mathcal{S}_2$'s dictionary using the one learned in the previous frame. This makes Algorithm 1 converge fast, as spectra in successive frames do not often change much.

3 Experiments

We test the proposed online semi-supervised source separation algorithm for real-time speech denoising. The two sources are therefore noise and speech. We learn the dictionary of noise from its training excerpts beforehand [5], and learn and update the dictionary of speech during real-time separation.

We use clean speech files and clean noise files to construct a noisy speech dataset for our experiments. For clean speech files, we use the full speech corpus in the NOIZEUS dataset [6]. This corpus has thirty short English sentences (each about three seconds long) spoken by three female and three male speakers. We concatenate sentences from the same speaker into one long sentence, and therefore obtain six long sentences, each of which is about fifteen seconds long.

[5] Training excerpts for noise is relatively easy to obtain in applications such as teleconferencing, since a few seconds at the beginning in which no one is talking are likely to be long enough to capture the noise characteristics throughout the teleconference.

[6] `http://www.utdallas.edu/~loizou/speech/noizeus/`

Algorithm 2 Online Semi-supervised Source Separation

Require: V_{ft} for $t = 1, \cdots, T$ (magnitude spectra of the mixture signal), $P(f|z)$ for $z \in \mathcal{S}_1$ ($\mathcal{S}_1$'s dictionary), $P^{(0)}(f|z)$ for $z \in \mathcal{S}_2$ (random initialization of $\mathcal{S}_2$'s dictionary), θ_{KL} (threshold to classify a mixture frame), $\mathcal{B}$ (buffer frames set).

for $t = 1$ to T **do**
  Decompose normalized magnitude spectrum $P_t(f) = \frac{V_{ft}}{\sum_f V_{ft}}$ by Eq. (7).
  if $d_{KL}(P_t(f) || \sum_{z \in \mathcal{S}_1} P(f|z)P_t(z)) < \theta_{KL}$ **then**
    Supervised separation using $P(f|z)$ for $z \in \mathcal{S}_1$ and $P^{(t-1)}(f|z)$ for $z \in \mathcal{S}_2$ and $P^{(t)}(f|z) \leftarrow P^{(t-1)}(f|z)$.
  else
    Learn $\mathcal{S}_2$'s dictionary $P^{(t)}(f|z)$ for $z \in \mathcal{S}_2$ and activation weights $P_t(z)$ using Algorithm 1, with $P^{(t)}(f|z)$ for $z \in \mathcal{S}_2$ initialized as $P^{(t-1)}(f|z)$.
    Set $\mathcal{S}_2$'s magnitude spectrum as:

$$V_{ft} \frac{\sum_{z \in \mathcal{S}_2} P^{(t)}(f|z)P_t(z)}{\sum_{z \in \mathcal{S}_1} P(f|z)P_t(z) + \sum_{z \in \mathcal{S}_2} P^{(t)}(f|z)P_t(z)}. \quad (8)$$

    Replace the oldest frame in $\mathcal{B}$ with the t-th frame.
  end if
end for
return separated magnitude spectra of the current frame.

For clean noise files, we collected ten different types of noise, including *birds*, *casino*, *cicadas*, *computer keyboard*, *eating chips*, *frogs*, *jungle*, *machine guns*, *motorcycles* and *ocean*. Each noise file is at least one minute long. The first twenty seconds are used to learn the noise dictionary. The rest are used to construct the noisy speech files.

We generate a noisy speech file by adding a clean speech file and a random portion of a clean noise file with one of the following signal-to-noise ratios (SNR): -10dB, -5dB, 0dB, 5dB and 10dB. By exploring all combinations of speech, noise and SNRs, we generate a total of 300 noisy speech files, each of which is about fifteen seconds long. The sampling rate of all the files is 16kHz.

For comparison, we run offline semi-supervised PLCA [1] (denoted as "PLCA") on this dataset. We segment the mixture into frames of 64ms long and 48ms overlap. We set the speech dictionary size as 20, since we find it is enough to get a perceptually good reconstruction of the clean speech files. We use different sizes of the noise dictionary for different noise types, due to their different characteristics and inherent complexities. We set this value by choosing from $\{1, 2, 5, 10, 20, 50, 100, 200\}$ the size that achieves the best denoising results in the condition of SNR of 0dB. The number of EM iterations is set to 100 as it always converged in that many iterations in our experiments.

We also implement an existing online NMF algorithm [4] (denoted as "O-IS-NMF"), which is designed for audio reconstruction. We apply it to this dataset in the semi-supervised paradigm. We use the same frame sizes and dictionary

sizes as PLCA. We set the mini-batch parameter β to 1 to avoid inherent delay, and set the scaling factor ρ to 1 as suggested to match β.

For the proposed algorithm, we use the same frame sizes and noise dictionary sizes as PLCA. We set the buffer size L as 60, which is about one second long. Since the speech dictionary is only supposed to explain the speech spectra in the current frame and buffer frames, we can use a much smaller size of speech dictionary. We set this value to 7 (opposed to 20 in PLCA), since we find that the average KL divergence in decomposing one second of speech spectra with seven dictionary elements is about the same as that of the average KL divergence in decomposing fifteen seconds of speech spectra with twenty dictionary elements. We choose the tradeoff factor α for each different noise, from the set $\{1, 2, \cdots, 20\}$ as the one that achieves the best denoising results in the condition of SNR of 0dB. We run only 20 EM iterations in processing each frame, which we find almost assures convergence due to the "warm" initialization as described in Section 2.3.

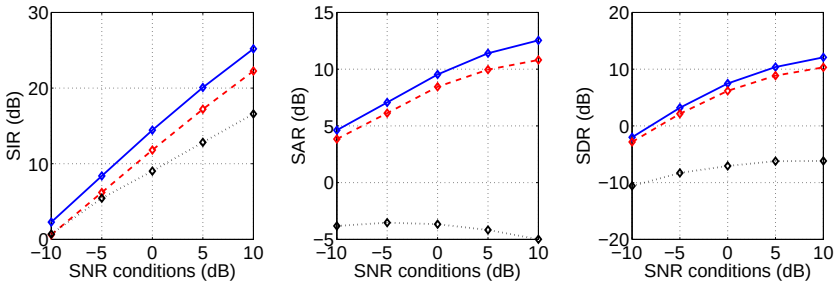

Fig. 1. Average performances on all types of noise of PLCA (blue solid line), O-IS-NMF (black dotted line) and the proposed algorithm (red dash line).

We use the BSS-EVAL metrics [5] to evaluate the separated speech files. Figure 1 shows the average results over all noise types and speakers, for each algorithm and SNR condition. Source-to-interference ratio (SIR) reflects noise suppression, source-to-artifacts ratio (SAR) reflects the artifacts introduced by the separation process, and source-to-distortion ratio (SDR) reflects the overall separation performance. It can be seen that for all the three metrics, the proposed algorithms achieves almost as good of a performance as PLCA. This is a promising result, since the proposed algorithm is an online algorithm and it uses a much smaller speech dictionary than PLCA. The performance of O-IS-NMF is significantly worse than PLCA and the proposed algorithm. As argued in Section 1, we think this algorithm is not suitable for real-time source separation.

Table 1 presents the performances of PLCA and the proposed algorithm for different noise types in the SNR condition of 0dB. The noise-specific parameters for the two algorithms are also presented. It can be seen that for different noise types, the results vary significantly. This is due to the inherent complexity of the noise and whether the training data can cover the noise characteristics or not. For some noise, like *birds*, *cicadas* and *frogs*, the performance of PLCA is

Table 1. Performances and noise-specific parameters for different noise types in the SNR condition of 0dB. K_n is the noise dictionary size and α is the tradeoff factor.

	SIR		SAR		SDR			
Noise type	PLCA	Proposed	PLCA	Proposed	PLCA	Proposed	K_n	α
birds	20.0	18.4	10.7	8.9	10.1	8.3	20	14
casino	5.3	7.5	8.6	7.2	3.2	3.9	10	13
cicadas	29.9	18.1	14.8	10.5	14.7	9.7	200	12
computer keyboard	18.5	12.2	8.9	10.2	8.3	7.9	20	3
eating chips	14.0	13.3	8.9	7.0	7.3	5.7	20	13
frogs	11.9	10.9	9.3	7.2	7.1	5.0	10	13
jungle	8.5	5.3	5.6	7.0	3.2	2.5	20	8
machine guns	19.3	16.0	11.8	11.5	10.9	10.0	10	2
motorcycles	10.2	8.0	7.9	7.0	5.6	4.5	10	10
ocean	6.8	7.4	8.8	8.0	4.3	4.3	10	10

significantly better than the proposed algorithm. For other noise like *casino*, *computer keyboard*, *machine guns* and *ocean*, the proposed algorithm achieves similar results to PLCA. The K_n parameter does not change much, except for the *cicada* noise. The α parameter is usually around 12, with the exception of *computer keyboard* and *machine gun* noise. Since these two noises are pulse-like noise with relatively simple spectra, the optimal α values are much smaller to have a weaker constraint.

The Matlab implementation of the proposed algorithms takes about 25 seconds to denoise each noisy speech file (which is about 15 seconds long), in a modern laptop computer with a 4-core 2.13GHz CPU. It would be easy to make it work in real-time in a C++ implementation or in a more advanced computer.

4 Conclusions

In this paper, we presented an online PLCA algorithm for real-time semi-supervised source separation. For the real-time speech denoising application, we showed that it achieves almost as good results as offline PLCA and significantly better results than an existing online NMF algorithm.

References

1. Smaragdis, P., Raj, B., Shashanka, M.V.: Supervised and Semi-Supervised Separation of Sounds from Single-Channel Mixtures. ICA (2007)
2. Mairal, J., Bach, F., Ponce, J., Sapiro, G.: Online Learning for Matrix Factorization and Sparse Coding. J. Machine Learning Research 11, 19–60 (2010)
3. Wang, F., Tan, C., König, A.C., Li, P.: Efficient Document Clustering via Online Nonnegative Matrix Factorizations. SDM (2011)
4. Lefèvre, A., Bach, F., Févotte, C.: Online Algorithms for Nonnegative Matrix Factorization with the Itakura-Saito Divergence. WASPAA (2011)
5. Vincent, E., Fevotte, C., Gribonval, R.: Performance Measurement in Blind Audio Source Separation, IEEE Trans. on Audio Speech Lang. Process. 14, 4, 1462–1469 (2006)