

TP12 – Séparation de sources

Exercice 1 : séparation harmonique/percussive

La *séparation harmonique/percussive* est un exemple très simple de séparation de sources. Comme son nom l'indique, elle consiste à séparer les composantes *harmoniques* (voix, instruments mélodiques) des composantes *percussives* (batterie) d'un morceau de musique. Cette méthode se fonde sur les observations suivantes :

- Le contenu spectral d'un son harmonique varie peu au cours du temps : son sonagramme fait apparaître des lignes horizontales.
- Au contraire, un son percussif est bref mais très « coloré » (riche en fréquences) : son sonagramme fait apparaître des lignes verticales.

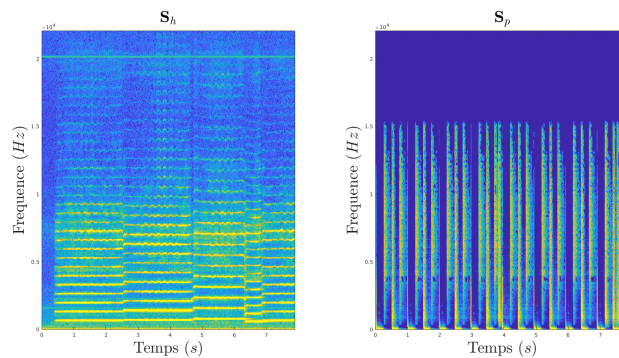


FIGURE 1 – Sonagramme \mathbf{S}_h d'un morceau de violon et sonagramme \mathbf{S}_p d'un morceau de batterie.

Étant donné le signal $y = y_h + y_p$, obtenu par superposition de deux signaux y_h et y_p , nous cherchons à séparer sa TFCT $\mathbf{Y} = \mathbf{Y}_h + \mathbf{Y}_p$ en une partie harmonique \mathbf{Y}_h et une partie percussive \mathbf{Y}_p . Attention à ne pas confondre la TFCT \mathbf{Y} , qui est complexe, avec le sonagramme \mathbf{S} , qui est réel et positif. En particulier, comme $\mathbf{S} = |\mathbf{Y}| = |\mathbf{Y}_h + \mathbf{Y}_p|$, il s'avère que $\mathbf{S} \neq |\mathbf{Y}_h| + |\mathbf{Y}_p| = \mathbf{S}_h + \mathbf{S}_p$.

Pour ce faire, commençons par renforcer les composantes horizontales \equiv harmoniques (resp. verticales \equiv percussives) du sonagramme \mathbf{S} , en lui appliquant un filtrage médian horizontal (resp. vertical) :

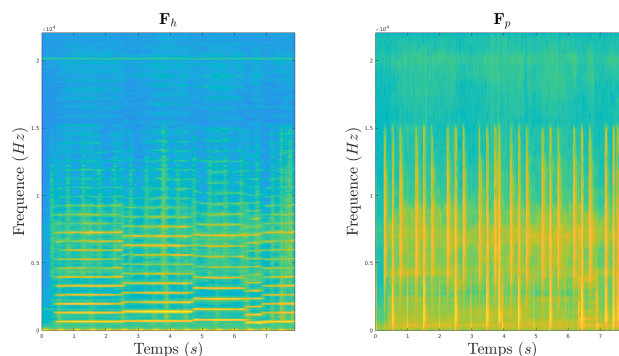


FIGURE 2 – Par filtrage du sonagramme \mathbf{S} avec un filtre médian horizontal (à gauche) ou vertical (à droite), on obtient \mathbf{F}_h et \mathbf{F}_p , respectivement.

Il est ensuite possible de créer des masques correspondant aux parties harmonique et percussive. Pour cela, on peut créer deux masques binaires, un élément du sonagramme étant soit *harmonique* soit *percussif* :

$$\mathbf{M}_h = (\mathbf{F}_h \geq \mathbf{F}_p) \quad ; \quad \mathbf{M}_p = (\mathbf{F}_h < \mathbf{F}_p) \quad (1)$$

mais on peut également créer des masques « doux » (\odot désigne la division élément par élément) :

$$\mathbf{M}_h = \mathbf{F}_h \odot (\mathbf{F}_h + \mathbf{F}_p) \quad ; \quad \mathbf{M}_p = \mathbf{F}_p \odot (\mathbf{F}_h + \mathbf{F}_p) \quad (2)$$

En appliquant cette paire de masques à la TFCT \mathbf{Y} du signal d'origine (\odot désigne le produit élément par élément), on obtient bien une décomposition $\mathbf{Y} = \hat{\mathbf{Y}}_h + \hat{\mathbf{Y}}_p$, puisque \mathbf{M}_h et \mathbf{M}_p sont complémentaires :

$$\hat{\mathbf{Y}}_h = \mathbf{M}_h \odot \mathbf{Y} \quad ; \quad \hat{\mathbf{Y}}_p = \mathbf{M}_p \odot \mathbf{Y} \quad (3)$$

Le résultat de cette décomposition peut être écouté en inversant la TFCT.

Écrivez la fonction `HPSS`, appelée par le script `exercice_1`, permettant de réaliser la décomposition harmonique/percussive d'un morceau de musique. Il vous est également demandé de compléter la partie du script `exercice_1` permettant de créer les masques \mathbf{M}_h et \mathbf{M}_p . Les valeurs des paramètres données dans le script `exercice_1` devraient vous permettre d'obtenir des résultats satisfaisants.

Pour réaliser les filtrages médians, vous pourrez utiliser la fonction `medfilt2` de Matlab. La taille des fenêtres dépend des paramètres utilisés pour le calcul de la TFCT (qui définissent la résolution temporelle et la résolution fréquentielle). Vous pourrez utiliser une fenêtre horizontale de taille $1 \times n_1$ et une fenêtre verticale de taille $n_2 \times 1$, avec $n_1 = n_2 = 17$ pour commencer. Testez différentes tailles de fenêtres, ainsi que les deux types de masques (binaires ou doux) et commentez les résultats.

Résolution du problème NMF

Le but de la *factorisation en matrices non négatives* (NMF, pour *Non-negative Matrix Factorization*) est d'approcher une matrice \mathbf{S} à coefficients non négatifs, c'est-à-dire positifs ou nuls, par le produit de deux matrices \mathbf{D} et \mathbf{A} à coefficients non négatifs, de rang R généralement très inférieur à celui de \mathbf{S} .

Intuitivement, l'idée est de décrire \mathbf{S} avec des *vecteurs de base* contenus dans \mathbf{D} , pouvant être activés à différents instants par la matrice \mathbf{A} .

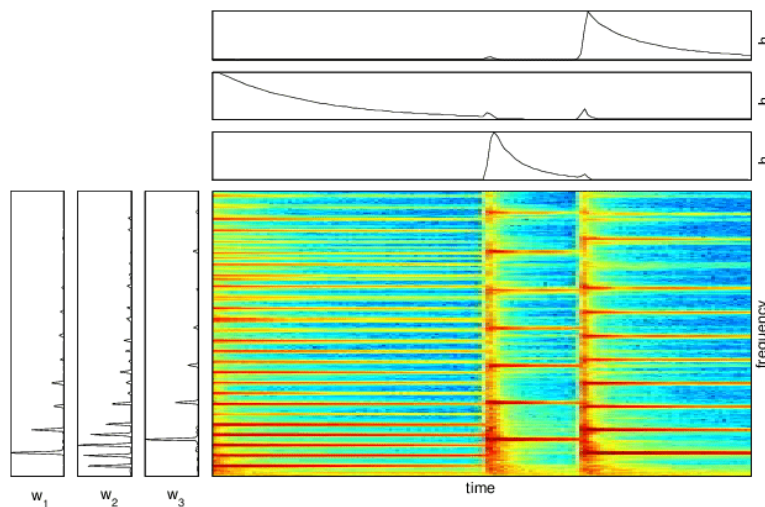


FIGURE 3 – Illustration de la décomposition d'un sonagramme par la méthode NMF, par Juan José Burred : $\mathbf{D} = [w_1, w_2, w_3]$ et $\mathbf{A} = [h_1; h_2; h_3]$. On a donc ici $R = 3$.

Mathématiquement, il s'agit de résoudre le problème d'optimisation sous contraintes suivant :

$$\{\hat{\mathbf{D}}, \hat{\mathbf{A}}\} = \underset{\mathbf{D}, \mathbf{A}}{\operatorname{argmin}} \mathcal{E}(\mathbf{S} - \mathbf{DA}) \quad \text{s.c.} \quad \mathbf{D} \in \mathcal{M}_{K,R}(\mathbb{R}^+) \quad \text{et} \quad \mathbf{A} \in \mathcal{M}_{R,N}(\mathbb{R}^+) \quad (4)$$

où \mathcal{E} désigne une fonction d'erreur. Le choix le plus courant (et le premier, historiquement) pour \mathcal{E} est le carré de la norme de Frobenius, qui constitue une généralisation aux matrices de la norme euclidienne :

$$\mathcal{E}(\mathbf{S} - \mathbf{DA}) := \frac{1}{2} \|\mathbf{S} - \mathbf{DA}\|_F^2 := \sum_i \sum_j [\mathbf{S} - \mathbf{DA}]_{i,j}^2 \quad (5)$$

Ce problème est cependant non convexe. Une manière de contourner cette difficulté consiste à effectuer une *descente de gradient*, en alternant les mises à jour de \mathbf{A} et de \mathbf{D} . À \mathbf{D} fixée, il est facile de déduire de (4) et (5) l'expression suivante pour la mise à jour de \mathbf{A} dans la direction opposée au gradient, où $p^{(k)}$ désigne le pas de descente à l'itération k :

$$\mathbf{A}^{(k+1)} = \mathbf{A}^{(k)} - p^{(k)} (\mathbf{D}^\top \mathbf{D} \mathbf{A}^{(k)} - \mathbf{D}^\top \mathbf{S}) \quad (6)$$

L'astuce consiste à ne pas utiliser le même pas $p^{(k)}$ pour tous les éléments de $\mathbf{A}^{(k)}$, mais autant de pas de descente qu'il y a d'éléments dans cette matrice. Plus précisément, il s'avère judicieux d'utiliser la matrice de pas de descente suivante :

$$\mathbf{P}^{(k)} = \mathbf{A}^{(k)} \oslash (\mathbf{D}^\top \mathbf{D} \mathbf{A}^{(k)}) \quad (7)$$

En effet, cela permet de simplifier l'itération (6) :

$$\mathbf{A}^{(k+1)} = \mathbf{A}^{(k)} \odot (\mathbf{D}^\top \mathbf{S}) \oslash (\mathbf{D}^\top \mathbf{D} \mathbf{A}^{(k)}) \quad (8)$$

L'avantage de cette expression est que, si $\mathbf{A}^{(k)}$ est à valeurs positives, alors $\mathbf{A}^{(k+1)}$ l'est aussi. Par un raisonnement en tous points similaire, la mise à jour de \mathbf{D} s'écrit de la façon suivante :

$$\mathbf{D}^{(k+1)} = \mathbf{D}^{(k)} \odot (\mathbf{S} \mathbf{A}^\top) \oslash (\mathbf{D}^{(k)} \mathbf{A} \mathbf{A}^\top) \quad (9)$$

Cette expression garantit que, si $\mathbf{D}^{(k)}$ est à valeurs positives, cela est également le cas de $\mathbf{D}^{(k+1)}$.

Afin d'éviter les divisions par 0 dans les mises à jour (8) et (9), il est quand même nécessaire d'ajouter une valeur $\epsilon > 0$ aux dénominateurs des divisions de matrices élément par élément.

Enfin, on peut remarquer que si une décomposition du type $\mathbf{S} = \mathbf{DA}$ existe, alors il en existe une infinité. Tout d'abord, en multipliant une colonne de \mathbf{D} par un réel $\alpha > 0$, il suffit de diviser la ligne correspondante dans \mathbf{A} par α pour compenser cet effet. Afin de garantir une certaine stabilité, il est demandé de normaliser les colonnes de \mathbf{D} en utilisant la norme infinie (max). Ensuite, si l'on échange deux colonnes de \mathbf{D} , il suffit d'échanger les deux lignes correspondantes dans \mathbf{A} pour compenser cet effet. Ce problème sera traité dans l'exercice 3.

Exercice 2 : décomposition d'un sonogramme par NMF

Écrivez la fonction `nmf`, appelée par le script `exercice_2`, qui permet de calculer et d'afficher, à partir du sonogramme \mathbf{S} de l'enregistrement musical `Au clair de la lune.wav`, une paire de matrices (\mathbf{D}, \mathbf{A}) vérifiant l'égalité $\mathbf{S} \approx \mathbf{DA}$, grâce à la méthode de factorisation NMF décrite ci-dessus. Dans ce script, le rang de \mathbf{D} et \mathbf{A} est fixé à $R = 3$, qui est le nombre de notes différentes entendues dans l'extrait analysé. Les matrices \mathbf{D} et \mathbf{A} sont initialisées avec des valeurs positives aléatoires (fonction `rand` de Matlab).

Enfin, une fois \mathbf{D} et \mathbf{A} obtenues, chaque composante est séparée en utilisant un système de masquage similaire à celui de l'exercice 1, le masque \mathbf{M}_r correspondant à la r -ème composante étant obtenu ainsi :

$$\mathbf{M}_r = (\mathbf{D}(:, r) \mathbf{A}(r, :)) \oslash (\mathbf{DA}) \quad (10)$$

Les signaux résultant du masquage de chaque composante sont enregistrés dans le répertoire `Resultats`.

Lancez plusieurs exécutions de ce script : observez et écoutez les décompositions ainsi obtenues. Pouvez-vous leur donner un sens ? Testez avec différentes valeurs de R . Que remarquez-vous ?

Méthodes par apprentissage profond

Les deux méthodes présentées précédemment ne sont pas entièrement satisfaisantes : la première ne résout qu'un problème très précis (séparation des percussions d'un morceau), tandis que la deuxième nécessite ou bien un post-traitement ou bien une initialisation plus fine des dictionnaires afin de pouvoir utiliser le résultat à des fins de séparation.

Depuis quelques années, des méthodes par apprentissage profond constituent l'état de l'art en matière de séparation de sources (avec souvent à l'origine des méthodes issues du traitement d'image).

Exercice 3 : U-Net au service de la séparation de sources

Faites une copie du notebook suivant dans votre espace personnel et complétez-le, en accord avec ce qui est indiqué.