Institut Mines-Télécom

Practical work on Non-negative matrix factorization

Roland Badeau
from the subjects written by U. Simsekli, B. David and P. Magron

TELECOM Paris
IP PARIS

The files necessary for this practical work are available on the ATIAM Moodle. You can load a sound file in Matlab by typing the command `[x,Fs] = wavread('file.wav')`. To listen to it, you can type `soundsc(x,Fs)`. The template of a Python notebook `NMF_TP.ipynb` is also provided.

1 Non-Negative Matrix Factorization with β-divergence

In this practical work, we will deal with non-negative matrix factorization (NMF) with the β-divergence. The problem that we aim to solve is defined as follows :

$$(\mathbf{W}^\star, \mathbf{H}^\star) = \text{argmin}_{\mathbf{W}\geq 0, \mathbf{H}\geq 0} \sum_{f=1}^{F}\sum_{t=1}^{T} d_\beta(x_{ft}||\hat{x}_{ft}), \tag{1}$$

where x_{ft} is an element of $\mathbf{X} \in \mathbb{R}_+^{F\times T}$, that is the *non-negative* data matrix, and $\mathbf{W} \in \mathbb{R}_+^{F\times R}$ and $\mathbf{H} \in \mathbb{R}_+^{R\times T}$ are the unknown *non-negative* factor matrices. We also define $\hat{x}_{ft} = \sum_r w_{fr}h_{rt}$. The cost function that we are minimizing is called the β-divergence, which is defined as follows :

$$d_\beta(x||\hat{x}) = \frac{x^\beta}{\beta(\beta-1)} - \frac{x\hat{x}^{\beta-1}}{\beta-1} + \frac{\hat{x}^\beta}{\beta}. \tag{2}$$

When $\beta = 2$, we obtain the squared Euclidean distance (EUC), when $\beta = 1$, we obtain the Kullback-Leibler (KL) divergence, when $\beta = 0$ we obtain the Itakura-Saito (IS) divergence.

One of the most popular algorithms for NMF is called the multiplicative update rules (MUR). The MUR algorithm has the following update rules :

$$\mathbf{W} \leftarrow \mathbf{W} \circ \frac{(\mathbf{X}\circ\hat{\mathbf{X}}^{\beta-2})\mathbf{H}^\top}{\hat{\mathbf{X}}^{\beta-1}\mathbf{H}^\top} \tag{3}$$

$$\mathbf{H} \leftarrow \mathbf{H} \circ \frac{\mathbf{W}^\top(\mathbf{X}\circ\hat{\mathbf{X}}^{\beta-2})}{\mathbf{W}^\top\hat{\mathbf{X}}^{\beta-1}}, \tag{4}$$

where $\circ$ denotes element-wise multiplication and / and $\frac{\cdot}{\cdot}$ denote element-wise division.

By following the technique that we used in the lecture, derive the MUR algorithm by yourselves.

2 Variations on a simple example

The aim here is to program the NMF as described above. For that, we will rely on the script `test_nmf.m` which manages the display part.

2.1 Construction of the simple example

We propose to synthesize the NMF of a simple musical example : a C, followed by an E, followed by a C/E chord, which we will note C-E-C/E.

This example is obtained by constructing the matrix $\mathbf{W}$ of the spectral basis and the matrix $\mathbf{H}$ of the activations in a synthetic way. No sound is calculated here. For this we can for example :

— use the sampling frequency $F_e = 8000$ Hz, the DFT order : $N_{\text{fft}} = 512$.

— calculate the fundamental frequencies F_0, of C and E located under A 440.

Roland Badeau
from the subjects written by U. Simsekli, B. David and P. Magron

— for each note r, create a (non-negative) harmonic spectrum of the form $\mathbf{W}_r(f) = \sum_{k=1}^{K_r} a_k w(f - kF_{0,r})$ where w is a usual window (e.g. Hann of width 80 Hz) and a_k describes the amplitude of the spectral lines (e.g. decreasing, of the form $a_k = exp(-k/K_r)$). K_r is fixed using the Shannon criterion. Deduce the spectral matrix $\mathbf{W}_s$.
— for the activations, we will use a function $h_r(t) = \sum_p b_{r,p} h(t - p\Delta T)$ where $h(t) = e^{-t/\tau}$ for $t \in [0, \Delta T]$ with ΔT of the order of 0.5 seconds and $h(t) = 0$ elsewhere. b_p is 0 or 1. Deduce the matrix $\mathbf{H}_s$. Attention, to put ourselves in realistic conditions, we will suppose that the representation is obtained with frames of length $N = N_{\text{fft}}$ and an overlap of 75%, i.e. a sampling rate of $F_e/(N_{\text{fft}}/4)$ for the temporal activations. τ will be selected nearby $\Delta T/3$.
— then construct the time-frequency matrix $\mathbf{X}_s = \mathbf{W}_s\mathbf{H}_s$.

2.2 Standard NMF

— Use the multiplicative update rules to factorize matrix $\mathbf{X}_s$ into a product of non-negative matrices. In order to avoid that the NMF indeterminacies lead to numerical problems, at the end of every iteration we renormalize the columns of $\mathbf{W}$ and we multiply the rows of $\mathbf{H}$ by the appropriate factors in order to keep the product $\mathbf{WH}$ unchanged. The matrices $\mathbf{W}$ and $\mathbf{H}$ will be initialized with random values.
— Draw the graph of the cost function as a function of the iteration. What do you notice ?

2.3 Variations

Test your algorithm by varying the data and parameters :
— by repeating with several draws of initialization (what differences do you notice on the results ?)
— with addition of noise (be careful to respect the non-negativity, use `rand`)
— with different values of β (0, 1 and 2 typically)
— with more notes and indeterminations like C/E, C/G, C/E/G
— by making synthetic notes (waveforms) as sum of sinusoids and by calculating the STFT of the obtained temporal signal.

3 Automatic transcription using semi-supervised NMF

The isolated notes of a piano are provided, as well as a piece of music played with the same instrument. The semi-supervised NMF consists in defining the spectral matrix $\mathbf{W}$ using the isolated notes (provided in the form of `xxx.wav` where `xxx` is the MIDI number, 21-108 for the tessitura of the piano), by simply calculating an estimate of the associated power spectral density. The only remaining update is therefore that of $\mathbf{H}$. We will work on signals sub-sampled at 22 kHz.

The provided Matlab script or Python function `build_signal` allows, with the help of a simple syntax, to build a piece from notes defined by their MIDI number, their attack time, their duration and the associated velocity.

1. Build the basis $\mathbf{W}$ corresponding to the 88 notes of the piano.
2. Synthesize a test signal corresponding to the C3-C4 scale.
3. Program the semi-supervised NMF using the results of the previous part and apply it to the test signal. Observe the temporal activations obtained.

Roland Badeau
from the subjects written by U. Simsekli, B. David and P. Magron

Contexte académique } sans modifications
Voir page 3

4. Program an attack detection function to be applied to the activations in order to estimate the times when the notes are played.