Nonnegative Matrix Factorisation

Dan Jacobellis, Tyler Masthay

Motivations for Nonnegative Matrix Factorisation : Audio Source Separation

Suppose we have a matrix $\mathbf{V}$ containing nonnegative data; for example, the time-frequency image of an audio recording.

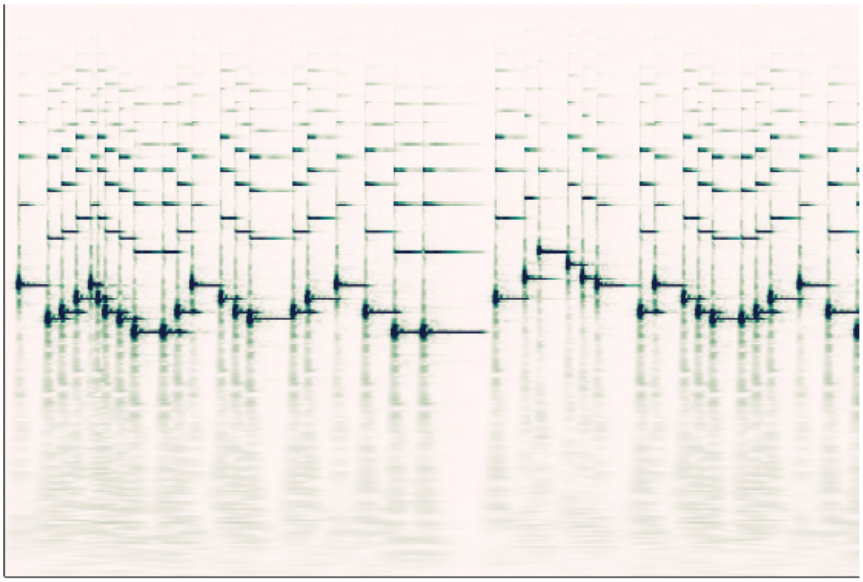

Figure 1: *Time-frequency representation of 'Korobeiniki' played on piano. The frequency is on a logarithmic scale.*

The problem of nonnegative matrix factorisation (NMF) amounts to factorising $\mathbf{V}$ into two factors $\mathbf{W}$ and $\mathbf{H}$ which are also nonnegative. That is,

$$\mathbf{V} \approx \hat{\mathbf{V}} = \mathbf{WH}$$

If the number of rows in $\mathbf{W}$ is restricted to be less than the number of rows in $\mathbf{V}$, then it may not be possible for $\hat{\mathbf{V}}$ to exactly match $\mathbf{V}$. By searching for a factorization that approximately matches $\mathbf{V}$, we perform unsupervised learning, where $\mathbf{W}$ contains a learned dictionary and $\mathbf{H}$ contains a representation of $\mathbf{V}$ in terms of this dictionary.

This unsupervised learning method has proven to be effective for audio analysis, especially for source separation and automatic music transcription [1]. To understand why, consider the structure of the recording in ***figure 1***. Although the recording consists of a single instrument (piano) playing a monophonic melody, the harmonics of each note create replicas of the actual melody at integer multiples of the fundamental frequency. The complexity that arises from mixing multiple intruments or adding any amount of polyphony

makes direct analysis of the time-frequency image intractable for most tasks. Nonnegative matrix factorisation allows us to learn the harmonic structure of a mixture of sources, dramatically simplifying analysis.

Algorithms

The most widely used algorithm for NMF is the so-called multiplicative update rule based on the pioneering work of Lee and Sung [2]. The algorithm consists of the following steps:

- Initialize $\mathbf{W}$ and $\mathbf{H}$ with non-negative values
- Iteratively update $\mathbf{W}$ and $\mathbf{H}$ using the following rules: (n is the iteration)

$$\mathbf{H}^{n+1}_{[i,j]} \leftarrow \mathbf{H}^{n}_{[i,j]} \odot \frac{\left((\mathbf{W}^n)^\top \mathbf{V}\right)_{[i,j]}}{\left((\mathbf{W}^n)^\top \mathbf{W}^n \mathbf{H}^n\right)_{[i,j]}}$$

$$\mathbf{W}^{n+1}_{[i,j]} \leftarrow \mathbf{W}^{n}_{[i,j]} \odot \frac{\left(\mathbf{V}(\mathbf{H}^{n+1})^\top\right)_{[i,j]}}{\left(\mathbf{W}^n \mathbf{H}^{n+1}(\mathbf{H}^{n+1})^\top\right)_{[i,j]}}$$

where $\odot$ is elementwise multiplication.

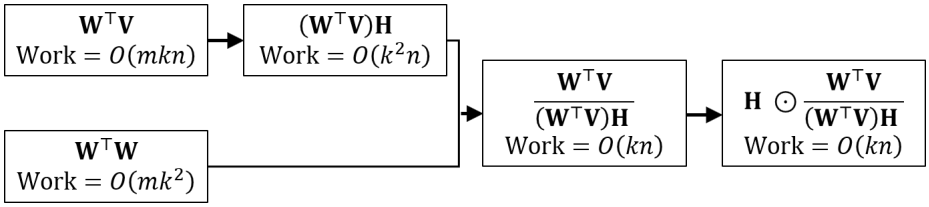

Figure 2: *DAG for the update of the H matrix showing the order of operations.*

We see that each iteration consists of matrix-matrix multiplication, elementwise multiplication, and elementwise division. As a result, the algorithm is embarrassingly parallel. Furthermore, since each iteration uses the same matricies, the algorithm demands much more computation than memory operations.

Careful attention should be paid to the order in which the matricies are multiplied. Suppose that $\mathbf{V}$ has dimension $m \times n$, and that we are interested in approximating using a dictionary of k components, where $k < m$. Then, $\mathbf{W}$ has size $m \times k$ and $\mathbf{H}$ has size $k \times n$. Typically for audio, $m << n$, since the maximum number of frequency bins is limited by the Gabor limit for a desired time resolution.

References

[1] S. Makino, Ed., Audio source separation. New York, NY: Springer Berlin Heidelberg, 2018.

[2] Lee, D.D., Seung, H.S., 2001. Algorithms for Non-negative Matrix Factorization, in: Advances in Neural Information Processing Systems 13. MIT Press, pp. 556–562.

[3] E. Vincent, T. Virtanen, and S. Gannot, Audio source separation and speech enhancement. 2018.