THE UNIVERSITY OF BRITISH COLUMBIA

# Unsupervised Signal Demixing with Tensor Factorization

*Nicholas Richardson[1] and Michael Friedlander[12]*
Department of [1]Mathematics and [2]Computer Science

INSTITUTE OF APPLIED MATHEMATICS
University of British Columbia

**Keywords** — source separation, optimization

## I Abstract

We propose a signal demixing framework and implementation in Julia using constrained tensor factorization. We use this tool to separate real signal mixtures in applications like geology, biology, and music without prior supervised training.

## II Setting: Multiple Unlabeled Mixtures

Unmix $\{\mathbf{y}_i\}$ into a small number of unknown sources $\{\mathbf{b}_r\}$ with unknown weights $\{a_{ir}\}$:

$$\begin{aligned}\mathbf{y}_1 &= a_{11}\mathbf{b}_1 + a_{12}\mathbf{b}_2 + \ldots + a_{1R}\mathbf{b}_R\\ &\vdots\\ \mathbf{y}_I &= a_{I1}\mathbf{b}_1 + a_{I2}\mathbf{b}_2 + \ldots + a_{IR}\mathbf{b}_R.\end{aligned} \tag{1}$$

Mixtures can be multivariable functions $\mathbf{y}_i : \mathbb{R}^N \to \mathbb{R}$ or directly measured vectors, matrices, or tensors $\mathbf{y}_i[j_1,\ldots,j_N]$. Either way, we package the data into a tensor $Y$ by sampling the mixtures $\{\mathbf{y}_i\}$ or stacking the observations:

$$Y[i,j_1,\ldots,j_N] = \mathbf{y}_i(x_1[j_1],\ldots,x_N[j_N]) \quad \text{or} \quad \mathbf{y}_i[j_1,\ldots,j_N]. \tag{2}$$

## III Model: Tucker-1 Tensor Factorization

Factorize $Y$ into a mixing matrix $A$ times a source tensor $B$ using the Tucker-1 model [1]:

$$\begin{aligned}Y &\to AB\\ Y[i,j_1,\ldots,j_N] &= \sum_{r=1}^{R} A[i,r]\cdot B[r,j_1,\ldots,j_N].\end{aligned} \tag{3}$$

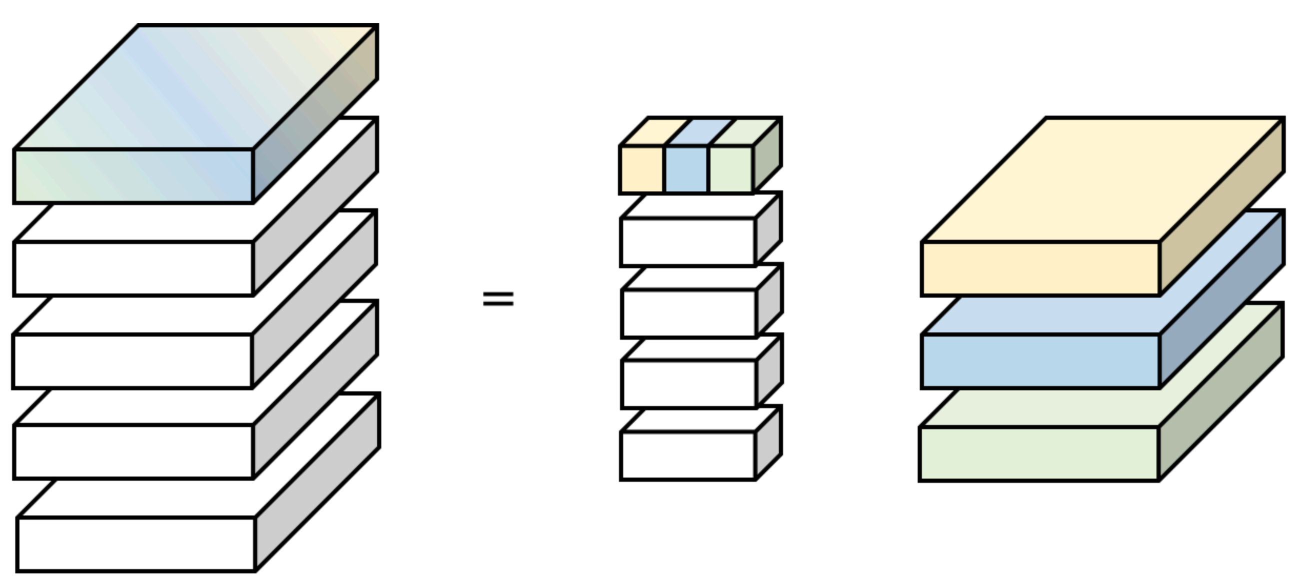


Figure 1: Example Tucker-1 decomposition for a 3rd-order tensor.

## IV Method: Least-Squares Optimization

Rather than directly computing a factorization $Y = AB$, we minimize the error between the model $AB$ and the data $Y$:

$$\min_{A,B} \ell(A,B) := \frac{1}{2}\|AB - Y\|_F^2 \qquad \text{s.t} \qquad A \in \mathcal{C}_A,\, B \in \mathcal{C}_B. \tag{4}$$

Constraint sets $\mathcal{C}_A$ and $\mathcal{C}_B$ are application dependent. For example, if want to ensure mixtures $Y[i,\ldots]$ are convex combinations of sources $B[r,\ldots]$, rows of $A$ should be nonnegative and sum to one,

$$A \in \mathcal{C}_A = \left\{ A \in \mathbb{R}_+^{I\times R} \,\middle|\, \textstyle\sum_{r=1}^{R} A[i,r] = 1 \text{ for } i = 1,2,\ldots,I \right\}. \tag{5}$$

## V Algorithm: Block Projected Gradient Descent

Alternately update $A$ and $B$ with projected gradient descent [2]:

$$\begin{aligned}A^{t+1} &= P_{\mathcal{C}_A}\left(A^t - \frac{1}{L_A}\nabla_A \ell(A^t,B^t)\right)\\ B^{t+1} &= P_{\mathcal{C}_B}\left(B^t - \frac{1}{L_B}\nabla_B \ell(A^{t+1},B^t)\right).\end{aligned} \tag{6}$$

This converges to a *block-wise* minimum and stationary point,

$$\ell(A^*,B^*) \le \min_{A\in\mathcal{C}_A, B\in\mathcal{C}_B} \{\ell(A^*,B), \ell(A,B^*)\}. \tag{7}$$

## VI Application: Geological Sediment Analysis

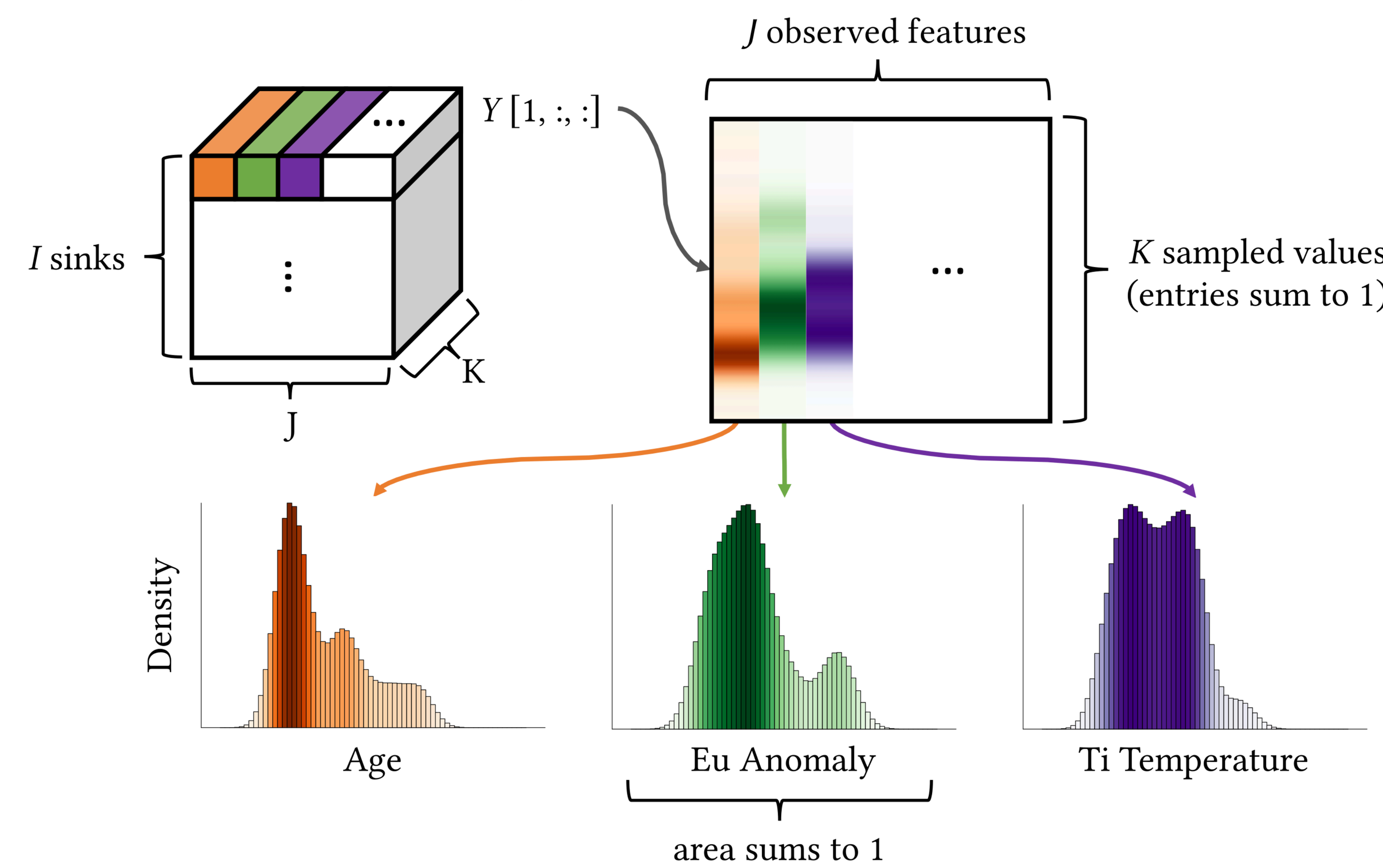


Figure 2: Input data tensor $Y$. Each 3-fibre $Y[i,j,:]$ is a discretized probability density for a different geological feature. The decomposed source distributions can be used to classify grains. See our paper in *Mathematical Geosciences* [3], and implementation on *GitHub* [4].

## VII Application: Spatial Transcriptomics

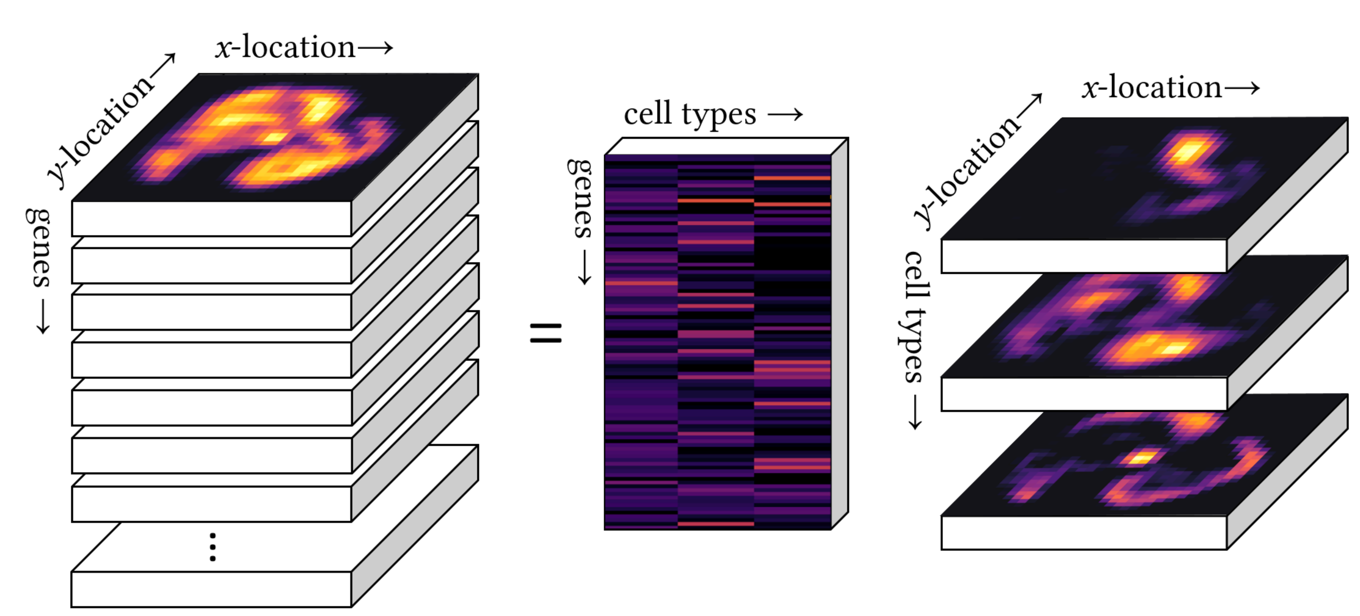


Figure 3: Spatial transcriptomics factorization model. Spatial distribution of many genes can be decomposed into few cell types. We uncover the gene expression and spatial distribution of these cell types, and can label distinct regions accordingly.

## VIII Application: Musical Instrument Separation

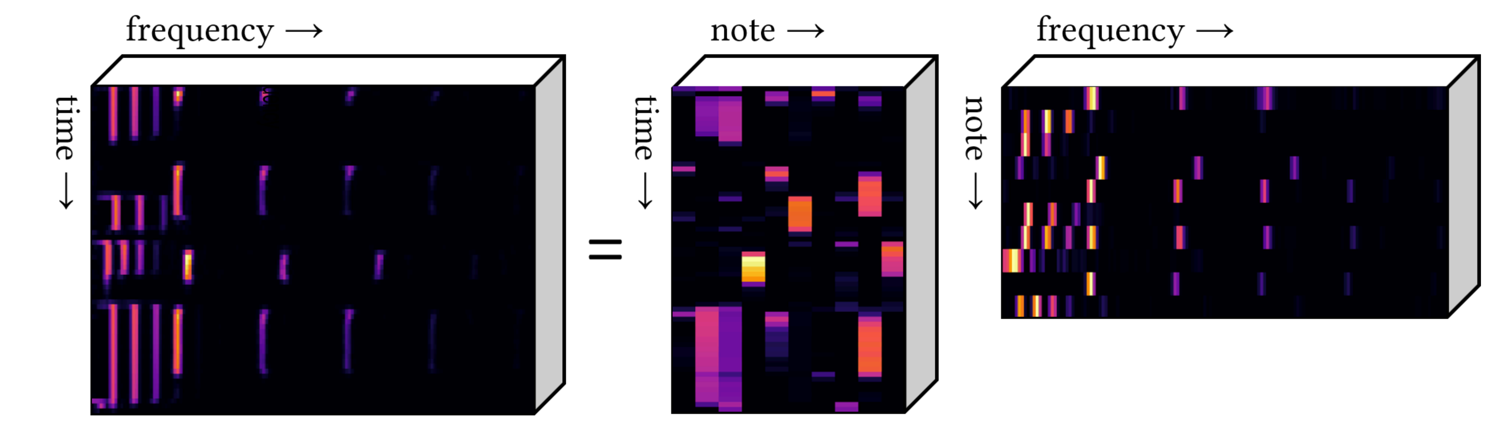


Figure 4: Audio source separation model. The short-time Fourier transform of a mixture can be separated into harmonically distinct notes. These can be grouped by their spectral similarity to recover instrument sources.

## IX Implementation: BlockTensorDecomposition.jl

The following is an example call to factorize a tensor `Y` with our Julia package `BlockTensorDecomposition.jl` [4] where the 1st-order slices must be nonnegative and sum to one.

```
simplex_1slices! = ProjectedNormalization(isnonnegative_sumtoone,
        projsplx!; whats_normalized=(x -> eachslice(x; dims=1)))
factorize(Y;rank=R,model=Tucker1,constraints=simplex_1slices!)
```

## X Current Development: Multi-Scaled Decomposition

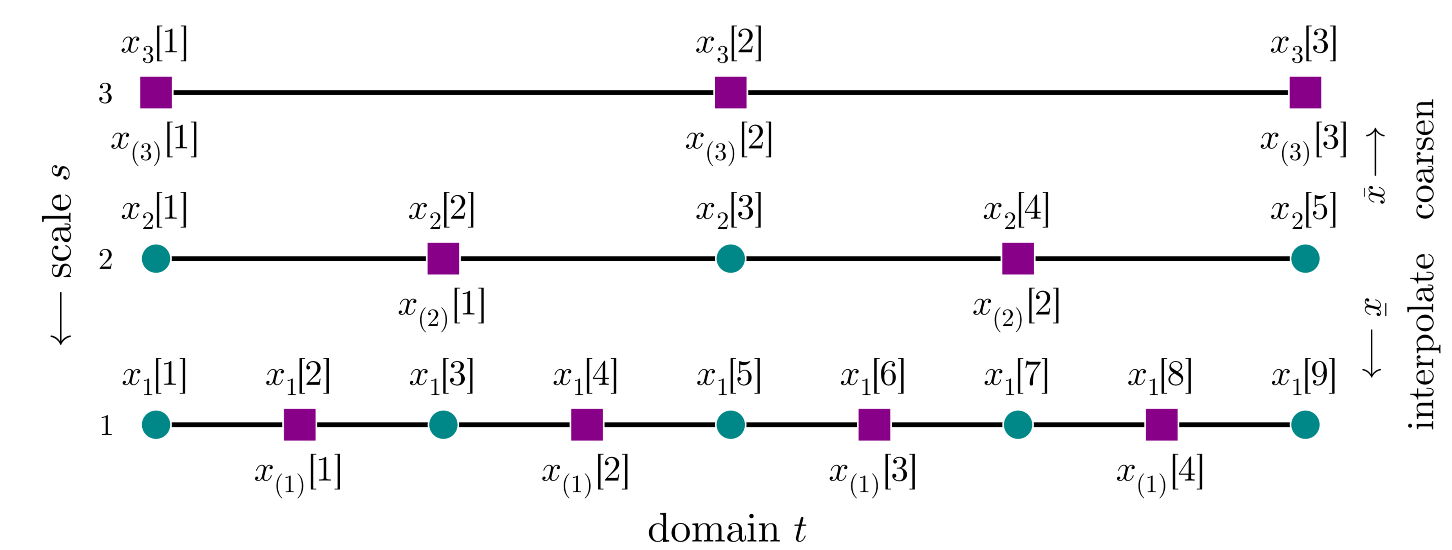


Figure 6: Rather than discretizing the mixtures $\{\mathbf{y}_i\}$ on a fine grid from the start, optimize over cheaper, coarse discretizations with fewer points. Then gradually refine.

## Bibliography

[1] T. G. Kolda and B. W. Bader, "Tensor Decompositions and Applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, Aug. 2009, doi: 10.1137/07070111X.

[2] Y. Xu and W. Yin, "A Block Coordinate Descent Method for Regularized Multiconvex Optimization with Applications to Nonnegative Tensor Factorization and Completion," *SIAM Journal on Imaging Sciences*, Jan. 2013, doi: 10.1137/120887795.

[3] N. Graham, N. Richardson, M. P. Friedlander, and J. Saylor, "Tracing Sedimentary Origins in Multivariate Geochronology via Constrained Tensor Factorization," *Mathematical Geosciences*, Feb. 2025, doi: 10.1007/s11004-024-10175-0.

[4] N. Richardson, N. Marusenko, and M. P. Friedlander, "BlockTensorDecomposition.jl," 2025. https://github.com/MPF-Optimization-Laboratory/MatrixTensorFactor.jl/tree/general-block-decomposition