BUILDING AN NMF SOURCE SEPARATION TOOLBOX FOR MUSICAL AUDIO

MIDTERM REPORT

MATTY BOI CCCCCC

# Abstract

Non-Negative Matrix Factorization (NMF) has proven to be an effective tool in source separation problems for musical audio. This report presents a MATLAB framework for source separation using NMF. Several related algorithms have been implemented and benchmarked, and the software is highly modular and extensible. We also present a discussion and timeline of future work, including score-aware implementations and public release as a MATLAB toolbox.

# Contents

# Introduction

*Source Separation* is the name given to the problem of extracting a set of individual sound *sources* from one or several *mixtures*, where a mixture is a weighted sum of the sources whose weighting may change with time. Mixtures may be *Instantaneous* – affected only by the present values of the sources – or *convolutive* – affected by present and past values. In some cases, the problem can be exactly solved, in theory allowing perfect reconstruction of the source signals. There are inherent ambiguities, however, in *underdetermined* mixtures with more sources than mixture channels. Algorithms for this class of problems must make prior assumptions about the source signals, such as statistical independence, harmonicity, or sparseness under some frequency transform.

Applications of source separation algorithms are numerous and include noise reduction, speech enhancement and analysis of hyperspectral images [1]. In music, source separation can be used for *upmixing* mono to stereo or stereo to surround sound, and for *remastering* existing recordings – perhaps by extracting the sound of a single instrument, editing it, and replacing it in the mix. More exotic uses include automatic generation of karaoke backing tracks [2]. These applications often involve underdetermined mixtures while requiring high quality reconstruction, placing heavy demands on the source separation algorithm. Musical Source Separation is an area of active research.

This project aims to create a robust MATLAB framework for Source Separation of musical audio using Non-Negative Matrix Factorization (NMF), a technique which approximates the Short Time Fourier Transform (STFT) matrix of a mixture as a product of two non-negative matrices of much smaller rank. There are a range of possible NMF-based algorithms depending on the choice of approximation cost function and the application of various constraints. *Score-aware* approaches which incorporate information from a musical score will be a particular focus.

The goals of the project are as follows:

1. Produce a flexible software framework for source separation using NMF which can accommodate a wide range of algorithms
2. Produce a framework which is modular and trivially easy to extend, and therefore useful for other researchers
3. Implement a range of blind and score-aware source separation algorithms, and compare their performance using established benchmarks.
4. Distribute the project codebase online for free as a MATLAB toolbox

For these goals to be met, the code must be production quality throughout the codebase, with effective error handling, extensive documentation and commenting, and minimal coupling between modules. High performance, though desirable in the end product, is a secondary concern at present. Reproducibility of results is ensured by versioning and publicly releasing code, using publicly available datasets, and benchmarking with standard benchmarks. A detailed project plan and logbook have been maintained throughout and will be taken forward into the second half of the project.

This Report is structured as follows:

<blah blah>

# Literature Review

## Approaches to Source separation

### History

Source separation (or *signal separation*, since the problem is not limited to audio) was first studied by Colin Cherry in 1953, who proved that humans distinguish between interfering speech sources based on their physical - not semantic - properties [3]. The problem was approached statistically in the 1980s [4] and by 1991 approaches using Independent Components Analysis were appearing [5]. NMF based approaches took off in 1999 with a well-known paper by Lee and Seung [6], providing an iterative method with easy to implement update rules. NMF became very popular due to the ease with which the basic algorithm could be implemented, as well as the ability to add constraints by zeroing out elements of the initialization matrices. Source separation and other rank-reduction algorithms using NMF are still areas of active research.

### Problem Formulation

Throughout this section, the definition of a source separation problem given by G. Evangelista in [7] will be used. Under this definition all sources are single point sources and source *m* can be represented using a single channel . If there are mixture channels, and the mixtures depend only on the present source values, then the *i-*th mixture is given by:

where is a scalar coefficient and *n* an integer valued time index. This is the *instantaneous mixing model.* If instead the mixture depends on the present input and a number of past inputs we refer to an FIR *mixing filter* :

We consider and to be constant with time.

While this representation is accurate, a frequency based representation is often more useful. Denote by

Blah blah give freq representation. Mention unmixing matrix

### Beamforming and Spatial Approaches

Beamforming works by creating unmixing filter matrix. Inherently spatial, requires I <= M. Simple case Sets up phases such that all in alignment for one source, assumes interfering sources will in general be out of phase. Clever implementations actively steer zeros over interfering angles. Adaptive solutions exist to get round needing theta cf “LCMV” (find citation from evangelista). Unsuitable for underdetermined mixtures

### Statistical approaches – Independent Components Analysis

Statistical approach – make the assumption that the sources are INDEPENDENT, IDENTICALLY DISTRIBUTED. Try to build unmixing matrix s.t. independence of STFT coeffs is maximised. This approach is FD ICA. Various choices can be made about assumed distribution of signal etc etc

some popular distributions introduce a scaling indeterminacy – eg circular generalised gaussian requires us to pick \beta. Normally scale so avg is 1. Can only talk about “contribution to each mixture of each source”

All ICA approaches introduce a permutation indeterminacy – nothing to distinguish! This may or my not be a problem depending on application, can use properties of expected spectra toresolve if known. In a spatial setting can use eg location information of micropones and expected DOA

Can use ICA or other statistical approaches in underdetermined mixtures but need to fold in assumptions about the sources. And difficult for v v underdetermined. Google around for Bayesian approaches eg IS measure paper? Or put in extensions to NMF

### NMF

What if I >> M? e.g. many instruments on a stereo recording?? Humans can pull information even out of mono recordings. Based on semantic but also physical information.

Lets go down to one mixture channel. Approach similar with stereo. Y is a matrix of magnitudes – must be positive. Make the sparseness assumption that our signal is made up of some small number of fixed spectra, with different excitations at different times.

Diag. row and column of matrix with interpretations highlighted.

Spectra all positive, excitations all positive. So we have ((V ~= WH)) st positive semidefiniteness. Define K and show can be much smaller than v. when we observe the signal we only see V but update rules given by BIGMANS lee and seung allow us to approx W,H together.

## Nmf In Depth

### Overview

Find W,H st WH ~= V s.t. some cost function C & W,H,V all elems positive.

Restate (or only state lel) as an optimisation problem. Minimise C s.t. blah blah.

Choice of cost functions corresponding to different fields of study, importance of scaling etc etc. see below. It turns out there are multiplicative update rules for many measures. Larger steps than grad descent, simple to implement. Nonincreasing so convergence can be detected.

Here are two update rules from Lee and Seung and one from that IS measure paper. <some LaTEX>

Note that “sources” under NMF are really “notes”. Need to unify the templates to one source. Not looking at this at present. Approaches include slidey NMF. Permutation indeterminacy a la statistical. Inherent to problem when spatial info not present.

Recovery – outer product of a spectrum by its excitation to get its contribution to the stft V. by def, sum of these contributions will be ~= V. but they don’t have phase!! Got to make it up or find it from the spectrum somehow. Spectrum painting solid but imperfect. See "reconstruction and stft processing considerations"

### Overcoming The Drawbacks of NMF using Score Alignment

There are some problems with using NMF in practice – notes can shadow eg if pno G/ gtr A always played together they will seem like one note. Also broadband signals at start of notes are more similar to each other than to the rest of the note – so can throw algo off.

What if there was a score involved? Assume prealignment for now. Theres a key property of the NMF update steps which is MULTIPICATIVENESS. So can zero out forbidden regions of W, H and constrain eg a col of W to a specific note, and a row of H to timing of that note. Then the NMF only learns the specific properties of the spectrum, and its specific volume over time. Jobs a goodun.

A chroma feature is a subdivision of the spectrum into eg “C#”. need to look up exact definition re periodicity in freq. by building expected array of chroma features from (aligned) score we can do the zeroing we need from above sect.

### Alignment using DTW

“assume prealignment” is a pretty huge assumption. From a DSP point of view a score is an extremely vague way of transmitting information. But if can get score to a set of (right-ish time correct order) time:note events we can use various alignment techniques to “warp” it to fit audio. Assume for now we have time:note. Can build very simply or go directly from midi, etc etc.

Dynamic time warping. Warping = either copying or deleting frames from one to make it “match” the other. Build a cost function on elements of your sequence (in this case will be a whole STFT frame, but can think of as single numbers. Whats important is cost function is single valued). Trying to align x[n], y[n]. so build a cost matrix where each elem is the cost between x[i] and y[j]. trying to find a low cost path from 0,0 to n,n st step sizes being 01 10 or 11. can populate a new matrix D with the lowest-possible-cost up to that particular I,j by building from bottom left. Then follow lowest path from top right to “realise” the number you get. Several improvements including more flexible step sizes, and constraining certain parts of the path by detecting note onsets, etc etc.

There also exist ML and HMM approaches eg [8]

### Reconstruction and STFT processing considerations

STFT windows the signal both on STFT and ISTFT. On STFT this is to extract the frame in question. In ISTFT this is to turn the periodic signal back into a time limited one. It is important for every sample to contribute equally otherwise significant errors can be introduced:

DIag – image from bench of a bad and a good reconstruction graph.

Let Wsynth be the synth window and Wanal be the analysis window. Let P = Wsynth \* Wanal. Essentially the signal is being windowed by P in each frame, before being summed back up so if hop size is h and N is length of window, then “Sigma (i = -inf, inf) p(n + ih) = 1 for all n” implies PR as we have multiplied each sample by 1 when we count over all the hops. If for some n sigma (I = -inf….) was not 1, that sample would have a lesser weighting in freq calculations. BAD TIMES INNIT

Is PR enough? In general STFT processing looks like this:

Diag – research book 4.2.4.2.

The “arbitrary transform block” can be represented as a set of gains, ie a linear, time varying, generally phase nonlinear filter, with an impulse response. If gains are sharp implied impulse response may be long. In order to avoid circular convolution errors we need fft\_length > N + K -1 where N = window length, K = implied IR length.

Options – ignore. Introduce errors, which depend on filter sharpness. Sufficiently pad. may introduce overhead, and anyway hard to bound K. find out implied gains on the fly, multiply by the FFT of a window func (SHORT circ conv) before applying. Fixes problem, but may be a sledgehammer to crack a nut. Planning to benchmark this – see future work section.

NMF does not provide phases. Can paint spectrum which is pretty good. But if two instruments are playing with different phase in the same time-frequency block then the resultant phase is between the two. Account for this by looking at phase spectrum where they don’t overlap? ML? ignore?

### Extensions to NMF

CITE ME BABY

## Existing Resources

### Performance assessment for source separation

When sources >> mixtures the implied mixing matrix in <freq equation> is degenerate and non invertible. So we cannot calculate the implied unmixing matrix and compare to mixing matrix. Instead benchmark directly on the extracted source signals. Signal to noise ratio can be found by comparing signal with ground truth but is rather uninformative and does not correlate well with perceptive judgments. This is because there are multiple classes of error. In PROPOSALS FOR, they define SIR as signal to interference from other sources, SAR as “musical noise” from the algorithm, and SNR to be the signal to additive noise ratio – ie the noise that remains AFTER accounting for SIR and SAR.

BSS\_EVAL (psyte) implements SAR and SIR measures along with an SDR measure for total distortion

PEASS (psyte) is a perceptually motivated decomposition which uses a different decomposition based on ŝj-sj=eTarget+eInterf+eArtif

### Datasets

Mercy. Think about whether this is in 5.1 already

### Other NMF toolboxes

Will probably omit

# Work To Date

## Overview

Work on the project to date has consisted of background research, design and architecture tasks, implementation, and testing. A generic framework for source separation has been fully architected and implemented in a GitLab repository along with several blind NMF-based algorithms and a series of test scripts. Score alignment and score-aware source separation algorithms have yet to be implemented, though their place in the architecture has been carefully mapped out.

Test data comes from the TRIOS [9] and PHENICX [10], [11], [12] HOW CITE source separation datasets, which also include scores and score alignment information for the next phase of the project. Testing included full pipeline benchmarking in several configurations, as well as a more targeted look at ISTFT reconstruction quality. Fuzz testing was used to assess the robustness of the NMF algorithms. When assessing the whole pipeline, two preexisting benchmarks from the literature were used - BSS\_EVAL [13] and PEASS [14]. When assessing STFT reconstruction.

The full gitlab repository is at XXXXXXX. Access to or a copy of the codebase are available upon request.

## Framework design

### Choice of Language

The first task in architecting the system was to choose a language. MATLAB, Python with Numpy, and lower-level approaches including C and C++ were considered. C was ruled out due to its error prone nature and lack of portability. C++ fares a little better on these two counts but lacks native matrix operations. MATLAB and Python are both feasible contenders from a technical point of view – MATLAB was chosen for its widespread adoption and integrated debugging tools despite Python’s more expressive syntax.

### The Generic Source separation algorithm

Reiterate interpretations of nmf values. Therefore source sep happens in four parts – spect, set init matrices, converge, reconstruct. describe steps required to reconstruct phases. Mention inherent coupling between stft and istft

Diag – source sep algo.

Needed a highly generic and extensible format but with structure. So pass functions around! “source sep” algo just calls them in turn and combines the results. V v generic but ensures common bare-minimum interface (even though interface of func itself can change!). If args need passing can use function partials and @ notation. Explain what a function partial is but leave til impl to show how it works. Managing args since no named args. Making pipeline reconfigurable. Describe interfaces. Programmers responsibility to ensure sensible functions.

Diag – sep\_sources

### Proposed score aware source separation architecture

Most NMF algorithms work by constraining W\_init, H\_init – can get v far on that alone. Eg score align by passing audio and score to nmf\_init\_aligned, which calls out to a score alignment function.

Diag – proposed score aligned source separation architecture

Score alignment using DTW will be architected as follows

Diag – how to score align using a dtw algorithm

## Implementation - Blind Source Separation

### Repository Structure

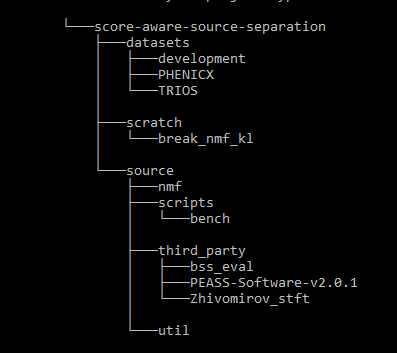


fig blahblah - the top level repository structure.



Fig blahblah – files in /source

The repository contains three directories at the top level:

**/datasets** contains the PHENICX and TRIOS datasets in full. Several simpler testcases have been collected in the “development” subfolder for prototyping, including random signals, chirps, and short snippets from the TRIOS dataset.

**/scratch** is a place for unclassified or experimental work which would cause problems if it appeared on the MATLAB path – most work in here is expected to either become superseded or move to /source eventually. break\_nmf\_kl contains a set of inputs which is found to violate the expected monotonicity of the nmf\_kl update implementation – see results – fuzz testing pp. XXX. The folder is otherwise empty.

**/source** contains source separation framework code, along with algorithm implementations, scripts, benchmark code and third party libraries. All code in the framework expects /source and all its subdirectories to be on the MATLAB path.

There are a further four subdirectories within /source:

**/source/nmf** contains nmf\_separate\_sources, which implements the generic source separation framework discussed in SECT 5.2.2 The numerical code to meet the nmf\_init, nmf, and nmf\_spect/nmf\_reconstruct interfaces fills the other subdirectories.

**/source/scripts** contains all MATLAB scripts directly runnable from the MATLAB terminal. Test-specific scripts are in **/source/scripts/bench.**

**/source/third\_party** is for third party code.

**/source/utils** is a place for low level functions which are useful in multiple places around the codebase.

Diag – arch ?

### nmf\_separate\_sources

mention source sep POC and Visualiser

nmf separate sources implements the generic algorithm design thingy. Takes as input the audio, the four functions in question along with a plot\_level flag – this allows the same function to be used in performant code and development, reducing the number of files that need maintaining.

A note on partial function applications:

MATLAB anonymous function syntax uses the @ sign – ie func = @(x,y) x+y; func(1,2) returns 3. By calling a named function in the anonymous function body we can partially apply a function (ie “fix” some of its arguments), as follows –

P\_func = @(free1, free2) myFunc (bound1, free1, free2, bound2)

P\_func (a,b)

P\_func(c,d)

The values of free1 and free2 can change between function calls. Every call to myFunc through p\_func qill have the same values for bound1/bound2.

A typical call to nmf\_separate\_sources is in eNorm\_source\_sep\_POC:. Can see the setup of partial functions before calling at end. DIAG – source sep POC code

### NMF functions - /source/nmf/nmf

Considerations: range of different measures each with its own update step. Need to implement a distance measure and NMF algorithm for each. All NMF algorithms fairly similar – just different enough that a generic template would be excessive (not to mention slow!). nmf\_euclidian included as an example in appendix 1

First converged using absolute threshold. Actually turned out to be a blunt tool – convergence detection by checking for a givern % improvement in last 1000 iter much better. Euclidian norm written to deal with scaling problems in edone thresh– now superfluous but included in bench etc for history. Show that done thresh not much use

Existing NMF functions – Euclidian. Uses square Euclidian distance as distance measure and the following update rule

Euclidian\_norm. uses a normed Euclidian distance. Update rule is the same and in terms of stationary points, both will reach the same. Difference is in done thresh

Nmf\_kl – uses the KL divergence as shown in lee and seung. Possible error – see fuzz testing

Nmf\_is – uses the IS divergence measure. See (psyte). Some evidence that this measure is better for audio applications.

### Init functions - /source/nmf/init

Folder for initialization functions. Nmf\_init\_rand takes an average and a K-value as input. Set average to average val of matrices, or 1. K-value is how many distinct (instr, note) points you think you have. Algorithm is quite sensitive to this when randomly initialized – see benchmarks.

Future work – nmf\_init\_chroma, nmf\_init\_aligned

### Spectrogram and reconstruction functions - /source/nmf/reconstruct

STFT/ISTFT is using Zhivomirov STFT at base (cyte (properly)). This is because spectrogram() is really for display so somewhat flabby, and doesn’t come with a matched istft pair. When taking spectrogram it is sufficient to build a partial application of stft suitable for the application (PR, etc. dependency of istft/stft pair unavoidable. See lit review spect considerations). However, the reconstruction step has two extra parts – build contributions and pick up phases. Currently two functions – reconstruct\_keepPhases and recons\_noPhases. Recons\_pitchTrack is a novel idea to track vib by taking strongest part near each spect line – still in dev. Flexible structure allows free development of novek algorithms w/no time cost.

Future work – pitchtrack, implementation of circ\_conv issues. See lit review.

### Scripting, Utilities, and repository management.

Script folder contains all current entry points to the project code – hence why bench is a subfolder. Framework should stand alone without scripts folder. Setpaths.m is important – sets up MATLAB path etc. run by other scripts on startup as a convenience. Enorm\_source\_sep\_POC is a proof of concept application of nmf\_separate\_sources. Amply tested by bench scripts but useful syntax reference and eventually will be an example for new users. Plot\_spectra allows hacking audio together/apart and looking at it.

Various utils developed while researching PR. Also matrix fuzzer, two UI quality of life functions. Two resons why useful – one, use across codebase. Two testing.

Repo managed with a .gitignore file and careful folder structuring and not much else.no version tagging system – if relaeasing to the public this will be necessary.

### Third party tools

MATLAB IDE, sublime, beyond compare, git and gitlab, git bash, agent ransack, Mendeley.

## Testing Methodology and Architecture

### STFT benchmarking

After reading up on PR found that is was necessary to build an stft bench. Also a stepping stone for later stft -> arbitrary filter -> istft bench, which will be more complex and important.

File paths list, series of named windows -> params list with name. call each param list and accumulate MSE and central MSE in results array. Can log and plot. By adding to file paths, windows or testcases can bench a wide range of things.

Cute lil diag

Improvements – could have a list of “benchmarks” similar to bench source\_sep. already tripped over this adding central\_MSE. Third times a problem. Printing code mixed in with algorithmic making it unwieldy.

It did find a key bug which was (literally) ~1000x improvement. Quote actual benchmark here.

### Nmf Fuzz Testing

Found that nmf\_is -> nmf\_kl on resulting matrices gives muuuch better convergence by kl measure than nmf\_kl alone. Could be running into a local stationary point but it smells funny.

Decided to fuzz test – ie randomly change matrix values at stationary point to see if can “manually” violate monotonicity. Added monotonicity assert, wrote the fuzz tester and left running for an hour or two. Found a situation in which monotonicity is violated and stored in scratch/break\_nmf\_kl/. Investigation pending – time limited in the test phase. Other algorithms passed fuzz test with flying colours, so can be confident problem is limited to kl.

Fuzz test design – wrote matfuzz util. table of V’s and init functions, including init using another nmf convergence. table of partially applied fuzz functions at different levels. Table of nmf funcs with their corresponding distance measure. Apply each fuzz many times, remembering rand. catch a drop in dist and collate in results array. Veeeeeeery slow.

Diag of fuzz test structure.

List what stored in break\_nmf\_kl

### Full Source Separation Benchmark

All full system benching happens through the same script -> DRY. Flexible testcase definition, potential to add new benches etc etc etc.

Tables:

Audio filepaths -> audio vectors. Multi col with fs and ground truths.

Testdefs – all partial functions needed for bench

Benchmarks.

Perform Source sep in loop. can suppress errors and record, or rethrow. Accumulate in cellArr.

Print results to log. Can call plotting functions at various points.

Diag.

Advantages – highly highly extensible. Unified testdef and printing arch saves lots of boilerplate code.

Improvements – testdef array v long making script unwieldt. Could pass in?? anything else from [Logbook:04/01/09:Stocktake] that I’m missing…

Bench code in appendix 2

## Test Results

Nmf measures

Effect of k

(show that done\_thresh strictly worse)

Nfft

PR/no PR

Reconstruction methods

Prove Fs agnostic

# Future Work

## Plannned Deliverables

Include: docs. SASS, SA. Matlab toolboxing.

## Project plan and (gantt chart)

LMAO

## Costings

😊 No cost has been incurred on the project so far – free tools, no use of heavy computing power etc. main asset is my MATLAB license which I got from the university license for free. No cost expected over the remaining course of the project.

# summary

# Conclusions

# references

# Appendices

## nmf\_euclidian

## nmf benchmark code