Insert here your thesis' task.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master's thesis

Exploring use of non-negative matrix factorization for lossy audio compression

Bc. Tomáš Drbota

Department of Theoretical Computer Science
Supervisor: doc. Ing. Ivan Šimeček, Ph.D.

March 22, 2019

Acknowledgements

TODO

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46(6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity.

In Prague on March 22, 2019 .....

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Drbota, Tomáš. *Exploring use of non-negative matrix factorization for lossy audio compression.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2019.

Abstrakt

TODO

Klíčová slova TODO

Abstract

Non-negative matrix factorization has been successfully applied in various scenarios, mostly for analyzing large chunks of data and finding patterns in them for later use. Due to the nature of NMF, it has also seen some use in the field of image compression.

The purpose of this thesis is to research possible uses of non-negative matrix factorization in the problem of audio compression. A reference audio encoder and decoder using NMF will be implemented and various experiments using this encoder will be conducted. The results will be measured and compared to existing audio compressing solutions.

Keywords lossy, audio, compression, processing, nmf, encoding

Contents

List of Figures

Introduction

In today's age of smartphones and other portable electronic devices capable of connecting to the internet, nearly everyone has access to this giant (and still growing) library of various media, including music and other audio. However, to transmit or store all of this data in its raw uncompressed form, a large amount of bandwidth and storage would be required.

.. need for compression ..
.. common methods of audio compression ..
.. mp3 opus ..
.. this work tries nmf ..
.. state of art ..
.. then design and implement ..
.. measure results ..

Part I

Background

Chapter 1

Digital audio

Sound as we know it can be defined as a physical wave travelling through air or another means. [1] It can be measured as change in air pressure surrounding an object. Once we have this electrical representation of the wave, we can convert it back and consequently play using speakers.

In the real world, these sound waves are generally composed of many different kinds of waves, with differing frequencies and amplitudes. The human ear can tell the difference between high (whistling) and low frequencies (drums), and knowledge of this will be useful later when we are discussing audio encoding.

- TODO image of audio signal -

1.1 Important terms

.. TODO .. sampling nyquist frequency/limit quantization transient aliasing spectral leakage windowing

1.2 Digital audio representation

Most commonly, the amount of air pressure is sampled many times a second and after being processed this information is stored as a discrete-time signal using numerical representations - this is what's known as a *digital audio signal*. This entire process is called *digital audio encoding*.

By sampling the audio signal, we will potentially be losing out on some information, but given a high enough sampling rate, the result will be imperceptible to the human ear. For general purpose audio and music, the standard sampling rate is 48 kHz, alternatively 44.1 kHz from the compact disk era.

Once we have our digital signal, there are two distinct kinds of ways we can represent, or, encode it. Both of them have many different data models

for encoding [1], but in this work I am only going to focus on the most relevant ones.

- TODO what compromises are taken when encoding -

1.2.1 Time domain representation

In the time domain, the signal is simply represented as a function of time, where t is the time and $x(t)$ is the raw amplitude, or air pressure, at that point. [2]

This is the most straightforward representation since it directly correlates to how the signal is being captured in the first place. However, as we will see later, this format is not ideal for storing audio data with any sort of compression.

1.2.1.1 PCM

In the time domain, the most basic encoding we can use is PCM (Pulse Code Modulation). After sampling a signal at uniform intervals, the discrete values are quantized; that is, each range of values is assigned a symbol in (usually) binary code.

For example using 16-bit signed PCM, each sample will be represented as a 16-bit signed integer, or in the case of multiple channels, N 16-bit signed integers, where N is the amount of channels.

PCM serves as a good base for what we are going to talk about next - Frequency domain representation and encoding.

1.2.2 Frequency domain representation

While it's simple to understand and work with for the computer with samples in the form of a sequence of amplitudes, it's difficult to run any sort of meaningful analysis on such data. To better grasp the structure of the audio we're working with, it would be helpful to be able to decompose it into its basic building blocks, so to speak. And that's where frequency based representation comes in.

The goal here is to represent the signal as not a function of time, but rather a function of frequency $X(f)$. That is, instead of having a simple sequence of amplitudes, we will have information about the magnitude for each component from a set of frequency ranges. This description alone is generally more compact than the PCM representation [2] on top of providing us with useful information about the signal, so it will serve as a good entry point to our compression schemes.

1.2.2.1 Fourier transform

Fourier transform is the first and arguably the most used tool for converting a signal from a function of time $x(t)$ into a function of frequency $X(f)$.

It is based on the *Fourier series*, which is essentially a representation of a periodic function as the linear combination of sines and cosines. [3] However, the main difference is that our function need not be periodic.

The Fourier transform of a continuous signal s is defined as: [4]

$$S(\xi) = \int_{-\infty}^{\infty} s(t) e^{-2\pi i t \xi} dt \tag{1.1}$$

If we inspect the formula, we can notice that Fourier transform essentially projects our signal into infinity - this wouldn't be a problem if it was a periodic signal, but sampled audio is generally constrained by time. To prevent spectral leakage, we must window the signal before processing it.

The output is a complex number, which provides us with the means to find the magnitude and phase offset for the sinusoid of each frequency ξ.

The Fourier transform can also be inverted, providing us with an easy way to obtain the original signal back from its frequency components. The inverse transform is defined as:

$$s(t) = \int_{-\infty}^{\infty} S(\xi) e^{2\pi i t \xi} d\xi \tag{1.2}$$

However, seeing as our samples are discretely sampled, we will need to modify our transform accordingly.

The discrete Fourier transform of a discrete signal $s_0, s_1, ..., s_{N-1}$ is: [4]

$$S_k = \sum_{n=0}^{N-1} s_n e^{-2\pi i k n / N} \tag{1.3}$$

And our inverse is:

$$s_n = \frac{1}{N} \sum_{k=0}^{N-1} S_k e^{2\pi i k n / N} \tag{1.4}$$

The issue is, due to the nature of this process, if we run the Fourier transform on our whole signal, we will only be able to analyse it as a whole, e.g. we won't be able to tell which parts of for example a song are quiet or if there are any parts with very high frequencies - we lose our temporal data.

To alleviate this problem, we can run Fourier transform on smaller chunks of the signal, analyse them separately and later join them back into the original signal. That is the essence of the Short-time Fourier transform.

1.2.2.2 Short-time Fourier transform

When using Short-time Fourier transform, or STFT for short, we first split the signal into smaller segments of equal size and then run Fourier transform on those separately. As such, our output can be projected into two dimensions - namely a frequency spectrum as a function of time.

Doing it this way will let us see how the frequency components change over time instead of taking the spectrum of the entire signal.

As with regular Fourier transform, we'll need to window each segment of the signal, but there is a caveat. Since we have windowed segments, we may be losing some information at the edge of each segment leading to artifacts, and furthermore we may be losing information about transients. To solve this, we'll need to introduce overlapping windows - however, having an overlap will increase the amount of coefficients required.

The continuous version is defined as: [4]

$$S(\tau, \xi) = \int_{-\infty}^{\infty} s(t)w(t-\tau)e^{-2\pi it\xi}dt \tag{1.5}$$

where w is the window function.

But again, as we have discrete samples, we will need to use a discrete short-time Fourier transform, specifically:

$$S_{k,\xi} = \sum_{n=-\infty}^{\infty} s_n w_{n-k} e^{-2\pi i\xi n} \tag{1.6}$$

And similarly to the regular Fourier Transform, short-time Fourier Transform is also invertible. [5]

STFT is most commonly used for audio analysis (TODO source) but in this case it will be used as a means for our NMF compression.

1.2.2.3 Modified discrete cosine transform

Modified discrete cosine transform, or MDCT for short, has become the dominant means of high-quality audio coding. [6]

It is what's known as a *lapped transform*. This means that when transforming a block into its MDCT coefficients, the basis function overlaps the block's boundaries. [7] In practice, what this means is that while we have blocks with overlapping windows as in the short-time Fourier transform, the number of coefficients remains the same as without while retaining the relevant properties.

As the name suggests, MDCT is based on the Discrete cosine transform, namely *DCT-IV*, where the main difference is the addition of lapping mentioned above.

What makes MDCT simpler to work with compared to Fourier transform is that not only do we not need more coefficients despite overlapping, they are also real numbers as opposed to complex numbers, lowering the amount of bytes necessary to store them.

It is a linear function $f : \mathbf{R}^{2N} \to \mathbf{R}^{N}$, defined as: [8]

$$X(k) = \sum_{n=0}^{N-1} x(n) \cos\left\{ \frac{(2n + 1 + \frac{N}{2})(2k + 1)\pi}{2N} \right\} \tag{1.7}$$

for $k = 0, 1, \ldots, \frac{N}{2} - 1$.

It is assumed that $x(n)$ is already windowed by an appropriate windowing function w.

MDCT is also invertible, and its inversion is defined as:

$$\bar{x}(n) = \sum_{k=0}^{\frac{N}{2}-1} X(k) \cos\left\{ \frac{(2n + 1 + \frac{N}{2})(2k + 1)\pi}{2N} \right\} \tag{1.8}$$

for $n = 0, 1, \ldots, N - 1$.

It's important to note that the inverted transformed sequence does not correspond to the original signal [9]. To achieve perfect invertibility, we must add subsequent overlapping blocks of the inverted MDCT (IMDCT). This method is called *time domain aliasing cancellation* [10], or TDAC for short. As the name suggests, it mainly helps remove artifacts on the boundaries between transform blocks.

1.3 Psychoacoustics

Apart from time-frequency representations being generally more compact, they also give us the ability to analyse, isolate or modify the frequency composition of a given signal. This comprises a large chunk of the audio compressing process.

The field of psychoacoustics studies sound perception - that is, how our ears work and how we perceive different kinds of sounds. There are many different characteristics to sound that need to be taken into account for a proper psychoacoustic analysis [11], split into several categories, namely:

tonal includes pitch, timbre, melody harmony

dynamic based on loudness

temporal involves time, duration, tempo and rhythm

qualitative represents harmonic constitution of the tone

For music, it's important to balance these four qualities appropriately. For compression, the most important qualities for us in scope of this work are going to be tonal (pitch) and dynamic (loudness).

1.3.1 Pitch

Pitch is a characteristic that comes from a frequency. The difference between the two is that pitch is our subjective perception of the tone whereas a frequency is an objective measure. Despite this fact, pitch is often quantified as a frequency using Hertz as its unit.

The lower bound of human hearing is around 20 Hz whereas the upper bound is most commonly cited as 20 000 Hz, or 20 kHz. [12] In a laboratory environment, people have been found to hear as low as 12 Hz. As people age, our hearing gets progressively worse and a healthy adult younger than 40 years can generally perceive frequencies only up to 15 kHz. [11]

The human ear is capable of distinguishing different frequencies fairly accurately, though accuracy gets lower with increasing frequency. It's easier for our ears to tell a difference between 500 Hz and 520 Hz compared to the difference between 5000 Hz and 5020 Hz. [13]

Furthermore, if we hear two different tones simultaneously, but their frequencies are close enough to one another, we may perceive them as a combination of tones rather than separate tones. Frequency ranges, or bands, where this phenomenon happens, are called *critical bands*. [14] It's also possible for one tone to mask the other entirely, and then we get what's called *auditory masking*. [15]

Based on the knowledge of the existence of these critical bands, it's possible to devise a system that specifies the range of each band in human hearing. One such scale that is commonly used is called the *Bark scale*.

1.3.1.1 Bark scale

The Bark scale ranges from 1 to 24 Barks, where each Bark corresponds to a single critical band of human hearing. [16] The perceived difference in pitch between each band should be the same, despite the scale not growing linearly in terms of frequency ranges. Specifically, until around 500 Hz, the scale is roughly linear, but above that it has a more logarithmic growth. [17]

The Bark scale is commonly used as reference for audio encoding codecs, as we will see later. Knowledge of these critical bands allows for more educated byte allocation during the quantization process when compressing a frequency domain representation.

.. TODO image/table of scale ..

1.3.2 Loudness

What people often decide as loudness is really called *sound pressure level* and it's measured in decibels (dB), however it has some shortcomings when it comes to psychoacoustic analysis.

It is defined as following: [18]

$$L_p = 20 \log_{10} \left(\frac{p}{p_0} \right) \text{dB} \tag{1.9}$$

where p is a sound's sound pressure and p_0 is a reference sound pressure, also called the threshold of human hearing.

While this metric is very popular, it doesn't account for the fact that different frequencies have a different perceived loudness for a person's ears. [11] There is a lot of research in recent years into how different frequencies impact our perception and hearing [19], but that is out of scope of this work. For more information about the exact definitions of loudness, refer to [11].

1.3.3 Auditory masking

As mentioned above, when it comes to audio masking, and therefore audio compression, we must not only take into account the critical bands as per e.g. the Bark scale, but also their intensity.

For example a lower frequency sound may mask one of a higher frequency, but the other way around does not apply. [15] Modern audio encoders take this into account and using this knowledge are able to eliminate sounds that exist in the original signal, but are not perceivable by humans.

There are two important different kinds of masking effects - *simultaneous* masking and *temporal* masking. [20]

Simultaneous masking is what I have hinted at above - when there are two sounds within the same critical band, the dominant one may mask other frequencies within the same band. .. TODO image ..

Temporal masking does not occur in the frequency domain, but the time domain. The essence is that a stronger tonal component may mask a weaker one if they appear within a small window of time in succession. .. TODO image ..

Chapter 2

Audio compression

Compression can be split into two kinds - lossy and lossless. Using ”lossless” in the context of audio is a bit misleading, since sampling itself is a lossy process, but using a high enough sampling rate, we will not notice any difference, so sampled audio without any lossy compression will be our baseline.

For audio, lossless compression generally means taking some form of digital audio representation and losslessly compressing this data. This will preserve the signal in its entirety with a reduced bit-rate. However, due to size of such audio (an audio CD could only fit about 80 minutes of such music sampled at 44.1 kHz), it's become more common to use a lossy format.

Lossy compression implies that there will be loss of data, and while this is true, thanks to the application of various psychoacoustic principles size of audio can be greatly reduced without altering human perception, leading to vastly smaller bit-rates for no real cost.

This work focuses on lossy audio compression, therefore only lossy codecs will be considered for comparison.

2.1 State of the art

Due to its qualities of efficiently compacting energy and mitigating artifacts at block boundaries, MDCT is the most commonly used transformation in modern lossy audio coding, and is employed in the most popular audio formats including MP3, Opus, Vorbis or AAC.

In this section, I will elaborate on some of the more popular ones to get an idea of what considerations go into writing a modern audio codec.

2.1.1 MP3

.. TODO diagram ..

MP3, or MPEG-1 Layer III has been standardized in 1991 and has since become widespread throughout a multitude of electronic devices as the de-facto standard for music storage.

It's a very powerful compression/decompression scheme capable of reducing the bit-rate of an audio stream by up to a factor of 12 without any noticeable (to humans) quality degradation. In other words, to transmit CD quality audio, it needs a bitrate of 128 kbps. [20]

The core of MP3 compression is the Modified discrete cosine transform. The signal represented in its PCM form is first split into 32 subbands using an analysis polyphase filterbank, and each of those is further split into 18 MDCT bins, so overall we end up with 576 MDCT frequency bins per frame.

These bins are then sorted into 22 scalefactor bands, which roughly correlate to the 24 bands of human hearing. The point of these bands is that you may individually scale each of them up or down depending on how much precision you need for that specific frequency range. This usually done by dividing and rounding the values in the band, losing a certain amount of information; this process will be reversed during decoding.

The signal is also analyzed using the Fourier transformation, which gives us frequency information for the signal in the same frame, and we can use this information to determine how much to scale each scalefactor band - e.g. if there's some weak sound that will be masked by another, we can assign the band it's in lower precision, saving data. [21]

Once we have the scaled and quantized data, we use the Huffman encoding to losslessly compress these values, and format this output into the final bitstream, encoding our audio.

2.1.2 Opus (CELT)

.. TODO diagram ..

The Opus codec has been standardized fairly recently [22] compared to other widespread audio codecs. Opus was created from two core technologies - Skype's SILK codec based on linear prediction, and Xiph.Org's CELT codec, based on the MDCT. [23]

As a result, Opus is capable of performing in three different modes:

SILK used for speech signals

CELT used for high-bitrate speech and music

Hybrid both SILK and CELT used simultaneously

This thesis will mainly focus on Opus CELT due to it being more general purpose than SILK, which will make it easier to use for comparison with a NMF-based codec. SILK uses a technique known as Linear predictive coding,

whose complexity and difficulty of implementation exceeds the scope of this work.

Opus is designed with real-time constraints in mind, that is, for example network music performances which require very low delays. Despite that, however, its compression is comparable to codecs with higher delays intended for streaming or storage. This makes it a good candidate to test against in this work, alongside MP3.

CELT (Constrained Energy Lapped Transform) is based on MDCT similar to MP3, but the main difference is that Opus uses specifically sized bands with ranges similar to the Bark scale in order to preserve the spectral envelope of the signal.

Similar to MP3, Opus CELT works on the basis of quantizing MDCT coefficients, but utilizes various kinds of prediction and focuses more on handling transients. Through various optimizations Opus achieves similar results but at a reduced bitrate. .. TODO source listening test ..

2.1.3 Opus (SILK)

.. TODO if I have time ..

Chapter 3

Non-negative matrix factorization

.. what is nmf ..
- .. how is nmf defined ..
- .. what is nmf used for ..
- .. why is nmf non-negative ..
- .. different kinds of nmf ..
- .. use in audio ..

Part II

Audio compression using NMF

Chapter 4

Design

.. time domain compression ..
compressing raw
.. frequency domain compression ..
- compressing STFT
STFT
NMF
mu-law quantization, non-uniform, formula in p128 wrong (missing sgn)
- compressing MDCT
unusable, needs to be compressed consistently / losslessly
.. file structure ..
diagram for both time domain and frequency domain compression

Chapter 5

Implementation

5.1 Encoder

.. process of encoding ..
.. application of NMF ..
.. variables ..

5.2 Decoder

Chapter 6

Evaluation

.. how is audio evaluated ..
- .. gstpeaq ..
- .. how does gstpeaq work ..
- .. how and what did I test ..
- .. comparison to other formats ..

Conclusion

.. what went right ..
.. what went wrong ..
.. what could be improved .. add psychoacoustics try compressing LPC/SILK

Bibliography

[1] You, Y. *Audio Coding Theory and Applications*. Springer US, 2010.

[2] Bosi, M.; Goldberg, R. E. *Introduction to digital audio coding and standards*. Kluwer Academic Publishers, 2003.

[3] Shatkay, H. The Fourier Transform - A Primer. Technical report, Providence, RI, USA, 1995.

[4] Recoskie, D. Constrained Nonnegative Matrix Factorization with Applications to Music Transcription. 2014.

[5] Selesnick, I. W. Short-Time Fourier Transform and Its Inverse. Apr 2009. Available from: `http://eeweb.poly.edu/iselesni/EL713/STFT/stft_inverse.pdf`

[6] Wang, Y.; Vilermo, M. Modified discrete cosine transform - Its implications for audio coding and error concealment. *Advances in Engineering Software - AES*, volume 51, 01 2012.

[7] Malvar, H. S. *Signal Processing with Lapped Transforms*. Norwood, MA, USA: Artech House, Inc., 1992, ISBN 0890064679.

[8] Babu, S. P. K.; Subramanian, K. Fast and Efficient Computation of MDCT / IMDCT Algorithms for MP 3 Applications. 2013.

[9] Princen, J.; BRADLEY, A. Analysis/Synthesis filter bank design based on time domain aliasing cancellation. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, volume 34, 11 1986: pp. 1153 – 1161, doi:10.1109/TASSP.1986.1164954.

[10] Princen, J.; Johnson, A.; et al. Subband/Transform coding using filter bank designs based on time domain aliasing cancellation. 05 1987, pp. 2161 – 2164, doi:10.1109/ICASSP.1987.1169405.

[11] Olson, H. *Music, Physics and Engineering*. Dover Books, Dover Publications, 1967, ISBN 9780486217697. Available from: https://books.google.cz/books?id=RUDTFBbb7jAC

[12] Rosen, S.; Howell, P. Signals and Systems for Speech and Hearing. *Acoustical Society of America Journal*, volume 94, 12 1993: p. 163, doi: 10.1121/1.407176.

[13] "smacdon". Critical Bands in Human Hearing. Oct 2018. Available from: https://community.plm.automation.siemens.com/t5/Testing-Knowledge-Base/Critical-Bands-in-Human-Hearing/ta-p/416798

[14] Fletcher, H. Auditory Patterns. *Rev. Mod. Phys.*, volume 12, Jan 1940: pp. 47–65, doi:10.1103/RevModPhys.12.47. Available from: https://link.aps.org/doi/10.1103/RevModPhys.12.47

[15] Gelfand, S. *Hearing: An Introduction to Psychological and Physiological Acoustics*. 01 1990, 187 pp., doi:10.1201/b14858.

[16] Fastl, H.; Zwicker, E. *Psychoacoustics: Facts and Models*. Berlin, Heidelberg: Springer-Verlag, 2006, ISBN 3540231595.

[17] Hermes, D. J. The auditory filter. Available from: http://home.ieis.tue.nl/dhermes/lectures/soundperception/04AuditoryFilter.html

[18] Behar, A. Intensity and sound pressure level. *Journal of The Acoustical Society of America - J ACOUST SOC AMER*, volume 76, 08 1984, doi: 10.1121/1.391117.

[19] Kuwano, S.; Namba, S.; et al. Advantages and Disadvantages of A-weighted Sound Pressure Level in Relation to Subjective Impression of Environmental Noise. *Noise Control Engineering Journal - NOISE CONTR ENG J*, volume 33, 11 1989, doi:10.3397/1.2827748.

[20] Raissi, R. The Theory Behind Mp3. 2002.

[21] Wilburn, T. The AudioFile: Understanding MP3 compression. Oct 2007. Available from: https://arstechnica.com/features/2007/10/the-audiofile-understanding-mp3-compression/2/

[22] Valin, e. a. Definition of the Opus Audio Codec. RFC 6716, RFC Editor, September 2012. Available from: http://www.rfc-editor.org/rfc/rfc6716.txt

[23] Valin, J.-M.; Maxwell, G.; et al. High-Quality, Low-Delay Music Coding in the Opus Codec. *135th Audio Engineering Society Convention 2013*, 01 2013: pp. 73–82.

Appendix A

Acronyms

todo TODO

Appendix B

Contents of enclosed CD

- `readme.txt` ..... the file with CD contents description
- `exe` ..... the directory with executables
- `src` ..... the directory of source codes
  - `wbdcm` ..... implementation sources
  - `thesis` ..... the directory of LaTeX source codes of the thesis
- `text` ..... the thesis text directory
  - `thesis.pdf` ..... the thesis text in PDF format
  - `thesis.ps` ..... the thesis text in PS format