Automatic Music Summarization

Jules Scholler
ENS Paris-Saclay
MSc Mathematics, Vision, Learning
jules.scholler@gmail.com

March 29, 2017

Abstract

This project, based on [1] presents different methods for automatically producing representative excerpts for audio signal. To this purpose we compute the maximum average segment similarity to the full audio signal. We start by computing the signal spectrogram and then a quantitative similarity measure is calculated between every pair of windows. By summing the similarity matrix along a segment support we obtained a measure of how similar this segment is from the rest. We then look for the maximum to find the segment that is the most representative to the whole music. All our code is freely available on GitHub [1].

Introduction

Nowadays there are several hours of videos, including audio signals, that are posted each minute. It is impossible to process such amount of data by human workforce. We treat in this project the problem of automatic summarization for audio signals. Such an application can be useful for music selling platform and any music content provider for broadcasting previews before listening or buying a song. The main difficulty is that the method should work with a broad types of audio signal (live hard rock concert, classical opera, etc.) in order to be universal. The proposed features should therefore be independent of music style or recording conditions. To this aim, a self-similarity approach is applied on a specific signal parametrization to find the most representative excerpt for a given duration in the whole signal. In the first section we present the proposed method for automatic music summarization with two parametrizations. In the second section we investigate the method stability with respect to noise and pathological cases.

[1] https://github.com/JulesScholler/AutoMusicSumm

1 Proposed framework

Drawing inspiration from [1] we exhibit a self-similarity measure for audio signal. This measure is designed to capture the likelihood between different parts of a signal and must be robust to noise. Indeed, the majority of audio signal that can be heard nowadays are compressed (e.g. MP3 compression) which can be represented as adding noise. Before introducing the similarity framework we discuss the signal representation. There exists more suitable ways of representing the data for our application than the simple amplitude variation in time.

1.1 Signal parametrization

Motivated by quantum mechanics, in 1946 the physicist Gabor proposed to decompose signals over time-frequency atoms [3]. By showing that such decompositions are closely related to our perception of sounds Gabor demonstrated the importance of localized time-frequency signal processing.

1.1.1 Fourier spectrum

Following this idea we started by computing the signal spectrogram by following these steps:

1. Divide the signal into N overlapping segments (the segment length and the overlapping degree must be carefully chosen in order to obtain the correct frequency range).
2. Multiply each segment by a window ϕ to avoid aliasing.
3. Compute the Fourier transform of each windowed segment giving a feature vector (each feature vector is called a frame).
4. Concatenate each feature vector forming a time frequency representation.

The choice of the window form ϕ is crucial for obtaining satisfactory results. In fact, from the convolution theorem we have:

$$\mathcal{F}(s \times \phi) = \tilde{s} * \tilde{\phi} = \sum_k \tilde{s}[k]\tilde{\phi}[n-k] \tag{1}$$

where $\mathcal{F}$ is the Fourier transform operator and $\tilde{s}$ is the discrete time Fourier transform of s. The Fourier transform of the segment is convoluted by the window Fourier transform. It is necessary to use a regular window to avoid gibbs effect in the Fourier domain. We decided to use the causal Hamming window [2] defined by:

$$\forall\, n \in [0, N],\ w(n) = 0.54 - 0.46cos(2\pi\frac{n}{N}) \qquad (2)$$

Hamming window was empirically designed to obtain a partial cancellation of sidelobes:

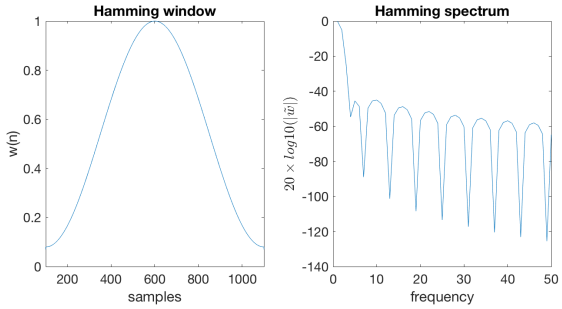

Figure 1: Hamming window effects

In practice the method gives similar results with Hann and Blackman windows. We applied the previous parametrization framework on a famous music signal and we obtained the results shown on Fig.2.

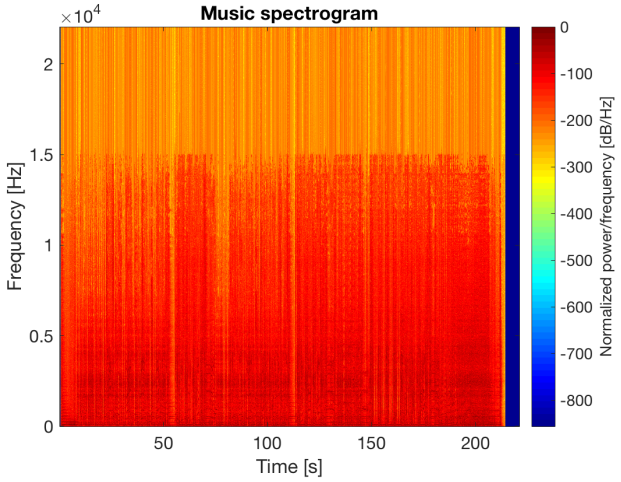

Figure 2: Spectrogram computed on ACDC - TNT with Hamming window on 46.4 ms frames with 23.2 ms overlap between samples.

1.1.2 Mel-Frequency Cepstral Coefficient

It is possible to draw another parametrization using Mel-Frequency Cepstral Coefficient (MFCC) instead of frequencies. The Mel scale is a perceptual scale of pitches judged by listeners to be equal in distance from one another and therefore approximates the human auditory system's response more closely than the linearly-spaced frequency bands used in the classical representation. It is possible to convert a frequency f in hertz into m in mels:

$$m = 2595 \times log_{10}(1 + \frac{f}{700}) \qquad (3)$$

In practice, in our setup, $f \in [0, 22050]\ Hz$ so $m \in [0, 3923]\ mel$ and applying this new scale (3) we obtained the power spectrums presented on Fig.3.

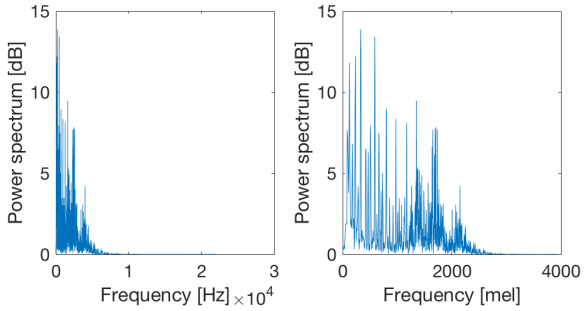

Figure 3: Linear frequency and mel frequency on the 300th frames of ACDC - TNT

This parametrization boils down to a non-linear rescaling of the original frequency axis, increasing the importance of lower frequencies and decreasing the importance of high frequencies. Indeed, because of the logarithm, the low frequencies domain is outstretched whereas the high frequency domain is shrinked. The shifting between the two regimes is for:

$$\frac{dm}{df} = 1 \qquad (4)$$

$$\iff f = 1895\ Hz \qquad (5)$$

$$\iff m = 1477\ mel \qquad (6)$$

This is schematically represented on Fig.4.

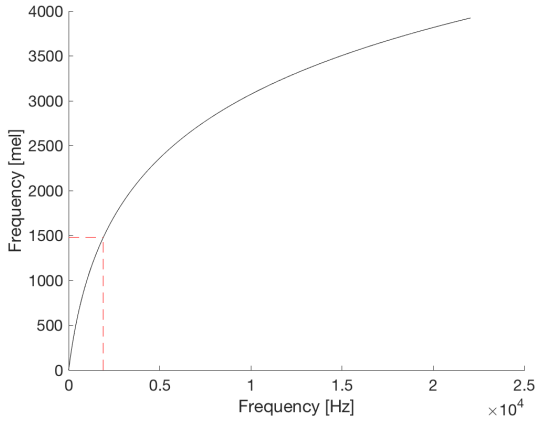

Figure 4: Comparison between linear frequency and MFCC parametrization

To obtain MFCC coefficients one needs to inverse the Fourier transform giving:

$$\begin{aligned} S(t) &\underset{Windowing}{\longrightarrow} s(t) \underset{FFT}{\longrightarrow} \hat{s}(w) \\ &\underset{(3)}{\longrightarrow} \sim log(\hat{s}(w)) \underset{iFFT}{\longrightarrow} MFCC \end{aligned} \tag{7}$$

In practice, we obtained similar results by using whether the linear frequency representation or the MFCC coefficients. In the literature, MFCC coefficients are widely used as features for voice recognition because it as the property to separate the source and the filter and allow better signal interpretation by machine learning algorithm. In the presented method it does not lead to an improvement because we do not seek to interpret signals in them-selves but to compare them and a linear frequency representation is therefore sufficient.

1.2 Similarity measure

Now that we have N feature vectors v we compute the similarity between each pair of vector to assess how similar is a given frame from all the others. To do that we construct the similarity matrix defined as:

$$D(i,j)^2 = \begin{cases} \frac{<v_i, v_j>}{||v_i||.||v_j||} & \text{if } v_i, v_j \neq 0 \\ \frac{<v_i+\epsilon, v_j>}{||v_i+\epsilon||.||v_j||} & \text{if } v_i = 0 \\ \frac{<v_i+\epsilon, v_j+\epsilon>}{||v_i+\epsilon||.||v_j+\epsilon||} & \text{if } v_i, v_j = 0 \end{cases} \tag{8}$$

where $<,>$ denotes the dot product and $||.||$ denotes the L^2 norm. Note that D is symmetric because we used the dot product as distance measurement and that $D(i,i) = 1$ i.e. maximum values are concentrated along the principal diagonal and are all equal to 1 as we have normalized D with the product of the vectors L^2 norm. The result obtained with the spectrogram presented on Fig.2 is shown on Fig.5.

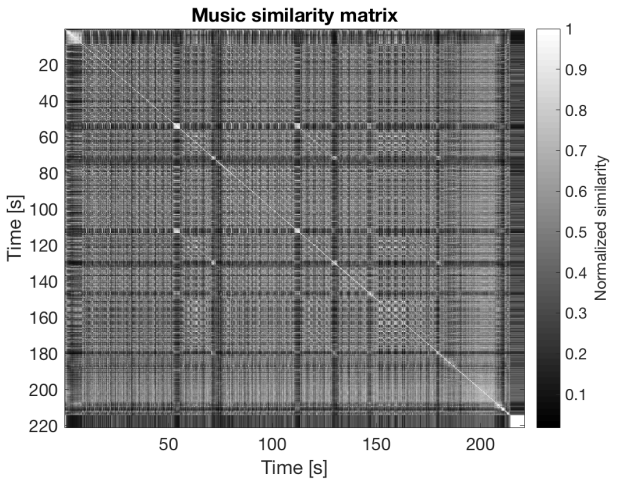

Figure 5: Similarity matrix computed on the spectrogram shown in Fig.2

[2] ϵ represents the spacing of floating point numbers, it is used to avoid numerical issues when evaluating $D(i,j)$

One can see that the diagonal is maximal and that the end of the signal is very cross-correlated which makes sens because one can see on Fig.2 that there is a silence at the end of the song.

1.3 Finding the best representative excerpt

Given a certain time duration L we want to find the excerpt of length L that is the most representative to the whole music. To do that, we simply need to sum the similarity evaluation on the excerpt support. Doing that, we obtain similarity score q of a given excerpt with all its pairs. Repeating this operation for all the segments of length L we construct the excerpt score array Q which in our setup is defined as follows:

$$Q(i) = \frac{1}{NL} \sum_{j=1}^{N} \sum_{k=i}^{i+L} S(j,k) \tag{9}$$

We then find the maximum of Q and retrieve the most representative except location α:

$$\alpha = argmax_i \frac{1}{NL} \sum_{j=1}^{N} \sum_{k=i}^{i+L} S(j,k) \tag{10}$$

The result obtained on the similarity matrix presented on Fig.5 is shown on Fig.6.

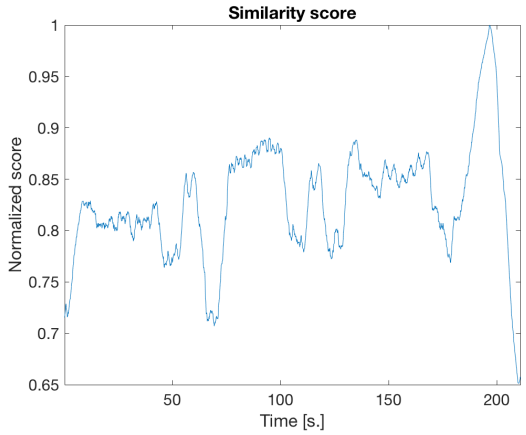

Figure 6: Excerpt score function computed with the similarity matrix shown in Fig.5

2 Method analysis

In this we try to understand the strengths and the weaknesses of the proposed approach and carry out both theoretical analysis and numerical experiments.

2.1 Robustness to noise

In order to be applied on huge data-set (e.g. for picking the right preview of a song before a customer can buy it) the presented method should be robust to noise. By noise we mean all the perturbations that can be found in an audio signal, such as:

- Compression artifacts
- Acquisition noise (non-ideal microphone, quantification errors, etc.)
- Live recording (background noise)

In general the signal noise is independently and identically distributed (i.i.d). This type of noise is called white noise because it can be shown that if the noise is i.i.d then its spectrum contains equally all the frequencies.

Wiener Kinchin theorem [2]: This theorem applies on finite energy signals and proves that the Fourier transform of the auto-correlation gives the frequency distribution of the signals energy, representing the energy spectrum.

White noise contains all the frequencies and thus has a constant Fourier spectrum. By Wiener Kinchin theorem we deduce that the white noise auto-correlation is a Dirac impulsion for infinite length signal and a cardinal sinus for finite length signals. The longer the signal length, the thinner the cardinal sinus:

$$\mathcal{F}(C) = \frac{1}{\sqrt{2\pi}} \int_{-L/2}^{+L/2} C.e^{-iwt} dt \tag{11}$$

$$= C' \left[\frac{e^{-iwt}}{-iw} \right]_{-L/2}^{+L/2} \tag{12}$$

$$= C' \left(\frac{e^{-\frac{iwL}{2}} - e^{\frac{iwL}{2}}}{-iw} \right) \tag{13}$$

$$= C'' sinc(\frac{wL}{2}) \underset{L \to \infty}{\longrightarrow} \delta(t) \tag{14}$$

This result is shown on Fig.7.

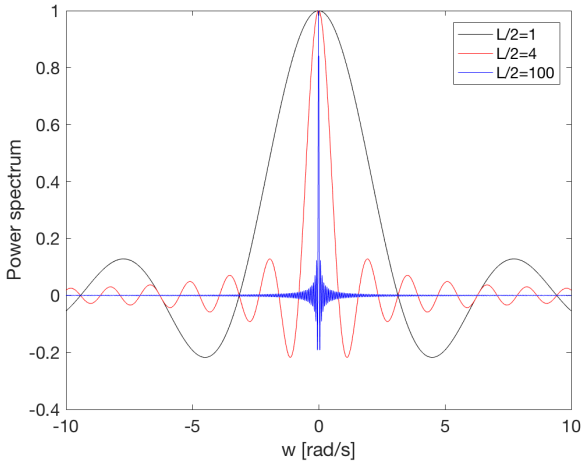

Figure 7: Cardinal sinus converging to a Dirac impulse

To illustrate this result we constructed a white Gaussian vector (with 0 mean and variance 1) with 10^6 dimensions. We run the method on this vector and we show (Fig.9) that the cross-correlation of white Gaussian noise only plays a role on the diagonal. Because the diagonal is always 1, white noises are not supposed to affect the algorithm. We can also see that by looking at the noise similarity score on Fig.10. Indeed, it is almost constant for all excerpts. This can be shown by considering the dot product between 2 noisy frames v_1 and v_2:

$$\begin{cases} v_1 = \bar{v}_1 + w_1 \\ v_2 = \bar{v}_2 + w_2 \end{cases}$$

with w_1 and w_2 i.i.d following $\mathcal{N}(0, \sigma^2)$ with $\sigma << \bar{v}_i$. One can write:

$$\begin{aligned} < v_1, v_2 > &= \sum_{f=0}^{f_{max}} v_1[f] v_2[f] \\ &= \sum_{f=0}^{f_{max}} (\bar{v}_1[f] + w_1[f])(\bar{v}_2[f] + w_2[f]) \\ &= < \bar{v}_1, \bar{v}_2 > + ... \\ &+ \sum_{f=0}^{f_{max}} \bar{v}_1[f] w_2[f] + \bar{v}_2[f] w_1[f] + w_2[f] w_1[f] \end{aligned}$$

We have $\mathbb{E}(w_2[f] w_1[f]) = 0$ because the noises are supposed to be independent, $\mathbb{E}(\sum_{f=0}^{f_{max}} \bar{v}_i[f] w_j[f]) = \sum_{f=0}^{f_{max}} \bar{v}_i[f] \mathbb{E}(w_j[f]) = 0$ because we have supposed that the noises have 0 mean. So finally, one can write:

$$\mathbb{E}(< v_1, v_2 >) \underset{\text{Probability}}{\longrightarrow} \mathbb{E}(< \bar{v}_1, \bar{v}_2 >) \tag{15}$$

We can make an analogous calculation on the variances (see appendix) and show that the relative error is:

$$\epsilon = \frac{var(< v_1, v_2 >)}{\mathbb{E}(< v_1, v_2 >)} \sim \frac{\sigma^2(\sigma^2 + \mathbb{E}(\bar{v_1})^2 + \mathbb{E}(\bar{v_2})^2)}{\mathbb{E}(\bar{v_1})\mathbb{E}(\bar{v_2})} \tag{16}$$

For small σ we have $\epsilon \sim \sigma^2$ (we can observe it on Fig.11) and as we have $\sigma << \bar{v}$ the error will remain small.

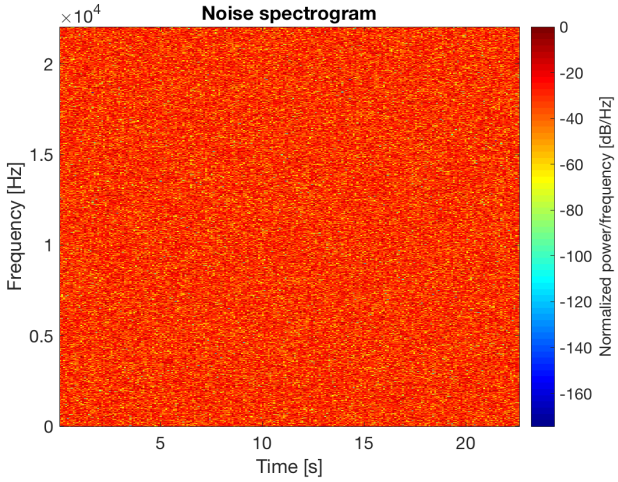

Figure 8: White Gaussian noise spectrogram

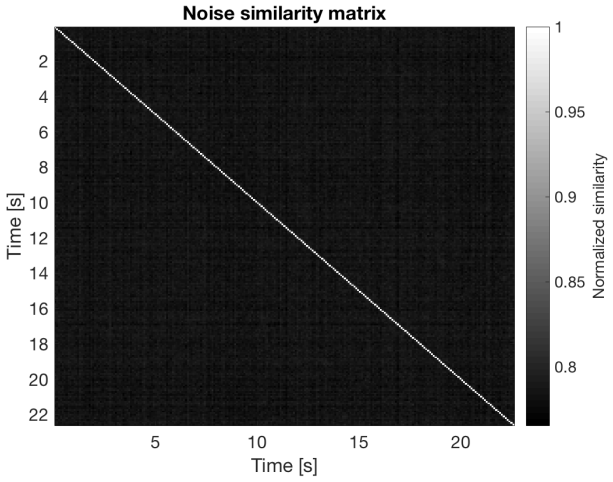

Figure 9: White Gaussian noise similarity matrix

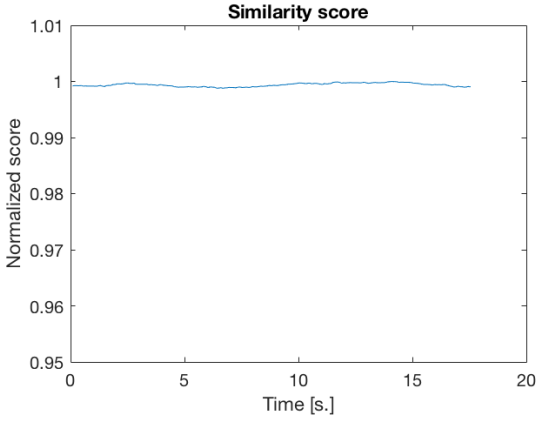

Figure 10: White Gaussian noise score

In order to validate our reasoning, we added white Gaussian noise with increasing intensity to the ACDC - TNT audio signal and retrieved the score function. We then calculated the average L^2 distance between the original score function and the score function obtained with noise.

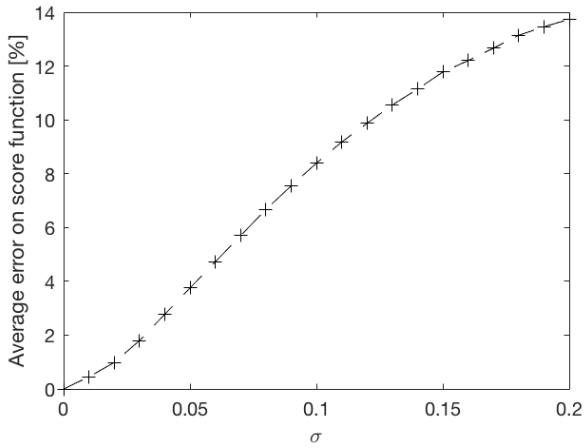

Figure 11: Average error when adding white Gaussian noise. Audio signal was normalized so that σ is the ratio between the standard deviation of the noise and the signal maximum amplitude

We can see that even with high noise levels (σ=0.2 is by far superior than any meetable noise for traditional application) the score function is not much affected. In fact the noise spread on the score function but as one can see in Fig.6 the score function is often peaked around its maximum and the selected excerpt with $\sigma = 0.2$ is the same as without noise.

2.2 Score function weighting

For some application it could be convenient to weight the score function with some characteristics computed on the input signal. For instance, if the input audio signal contains too much silences, then the algorithm will find that the best representative excerpt is a silent one. This may not be satisfactory in a preview application. To ensure that the algorithm will select an excerpt that contains energy there are several approaches:

- Weight the score function by a function that depends on energy such as an Heaviside step (put score function to zero under a certain energy and does not bias other scores). This corresponds to threshold the score function with the energy.
- Delete silent frame in the spectrogram.

According to the use of the algorithm, one can fine-tune its parameters (frequency scale, energy influence, etc.) in order to promote certain characteristics as more representatives. This is done by changing the parametrization or by weighting the score function but the similarity assessment is universal in this case which make this method very convenient.

Conclusion

We have presented a quantitative approach to automatic music summarization without any assumptions on the source audio. The reasoning is based on the fact that the best representative excerpt should be the one that is the more similar with the others. We tested the method on a broad (jazz, rock, techno, classical, etc.) music set and we obtained satisfactory results as we were able to recognize the song only with a 10 s. excerpt. The retrieved excerpt was always a "popular" part of the song. We tested the method on artificially noisy signals (by adding salt-and-pepper and gaussian noises) and the retrieved excerpt was almost the same (less than 0.1 s. shifted).

In order to improve this approach one can implement an automatic excerpt length determination. Indeed, in the presented method, the excerpt duration is an input and the excerpt is often cut in the middle of a sentence/instrumental which could be unpleasant in a music preview. A possibility could be to look at the excerpt extremities and retrieve the time where the score function derivative is minimal, which boils down to stop when the excerpt first becomes less representative. Another possibility could be to perform an audio segmentation around the retrieved excerpt which is not trivial, our attempts to compute such algorithm were not convincing.

Appendix

Relative error calculation:

$$var(< v_1, v_2 >) = var(< \bar{v_1} + w_1, \bar{v_2} + w_2 >)$$

We can separate each term into the variance because the covariances are nulls.

$$\begin{aligned} var(< v_1, v_2 >) = var(< \bar{v_1}, \bar{v_2} >) + var(< \bar{v_1}, w_2 >) + ... \\ + var(< w_1, \bar{v_2} >) + var(< w_1, w_2 >) \end{aligned}$$

We calculate each term separately:

$$\begin{aligned} var(< \bar{v_1}, \bar{v_2} >) &= 0 \\ var(< \bar{v_1}, w_2 >) &= var(\sum_f \bar{v_1}[f] w_2[f]) \\ &\underset{i.i.d}{=} \sum_f \bar{v_1}[f]^2 var(w_2[f]) \\ &= N\mathbb{E}(\bar{v_1})^2 \sigma^2 \\ var(< \bar{v_2}, w_2 >) &= N\mathbb{E}(\bar{v_2})^2 \sigma^2 \\ var(< w_1, w_2 >) &= var(\sum_f w_1[f] w_2[f]) \\ &\underset{i.i.d}{=} \sum_f var(w_1[f] w_2[f]) \\ &\underset{mean=0}{=} N var(w_1) var(w_2) \\ &= N\sigma^4 \end{aligned}$$

Finally:

$$var(< v_1, v_2 >) = N\sigma^2(\sigma^2 + \mathbb{E}(\bar{v_1})^2 + \mathbb{E}(\bar{v_2})^2)$$

References

[1] Matthew Cooper and Jonathan Foote. Automatic music summarization via similarity analysis. *Proc. of the 3rd ISMIR Conf.*, pages 94–100, 2002.

[2] Digital Signal Processing and Spectral Analysis for Scientists. *Silvia Maria Alessio*. Springer, 2016.

[3] Mallat Stephane. Chapter 2 - the fourier kingdom. In *A Wavelet Tour of Signal Processing (Third Edition)*. Academic Press, Boston, third edition edition, 2009.