

Bootcamp Data Science

Zajęcia 4

Przemysław Spurek

$\mathbb{C} = \mathbb{R} \times \mathbb{R}$ - zbiór liczb zespolonych z działaniami:

- $\forall (a, b), (c, d) \in \mathbb{C}: (a, b) + (c, d) = (a + c, b + d),$
- $\forall (a, b), (c, d) \in \mathbb{C}: (ac - bd, ad + bc).$

Liczby zespolone możemy zatem zapisywać także w postaci:

- $i = \sqrt{-1}$ - jednostka urojona
- $\forall (a, b) \in \mathbb{C}: (a, b) = a + bi$

Kilka specjalnych funkcji oraz sposobów reprezentacji:

- $\operatorname{Re}(a + bi) = a$ - **część rzeczywista**,
- $\operatorname{Im}(a + bi) = b$ - **część urojona**,
- $\overline{a + bi} = a - bi$ - **sprzężenie** liczby zespolonej,
- $|a + bi| = \sqrt{a^2 + b^2}$ - **moduł** liczby zespolonej (odległość od zera $(0, 0)$) lub inaczej $|z| = \sqrt{(\operatorname{Re} z)^2 + (\operatorname{Im} z)^2}$,
- $\phi = \begin{cases} \arccos \frac{a}{|z|}, & \text{jeżeli } b \geq 0 \text{ i } z \neq (0, 0) \\ -\arccos \frac{a}{|z|}, & \text{jeżeli } b < 0 \text{ i } z \neq (0, 0) \end{cases}$, to **argument główny** liczby zespolonej ($\phi \in [0, 2\pi]$).

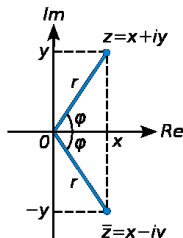
Liczby zespolone

- Liczba zespolona $z \neq (0, 0)$ może być przedstawiona jako:

$$z = |z|(\cos \phi + i \sin \phi), \quad (1)$$

gdzie:

- $|z|$, to **moduł** liczby zespolonej,
- ϕ , to argument główny liczby zespolonej,



- Liczba zespolona $z \in \mathbb{C}$ może być przedstawiona jako:

$$z = |z| \cdot e^{i\phi}$$

gdzie:

- $|z|$, to **moduł** liczby zespolonej,
- ϕ , to argument główny liczby zespolonej,

Stąd na podstawie układu równań

$$\begin{cases} e^{i\phi} = \cos \phi + i \sin \phi \\ e^{-i\phi} = \cos \phi - i \sin \phi \end{cases}$$

otrzymujemy $\sin \phi = \frac{e^{i\phi} - e^{-i\phi}}{2i}$ oraz $\cos \phi = \frac{e^{i\phi} + e^{-i\phi}}{2}$.

Własności fal



Własności fal



- amplituda – maksymalne odchylenie A od położenia równowagi

Własności fal



- amplituda – maksymalne odchylenie A od położenia równowagi
- długość fali – odległość λ pomiędzy kolejnymi powtórzeniami kształtu fali (np. grzbiety, doliny)

Własności fal



- amplituda – maksymalne odchylenie A od położenia równowagi
- długość fali – odległość λ pomiędzy kolejnymi powtórzeniami kształtu fali (np. grzbiety, doliny)
- okres – odstęp czasu T między momentami, gdy grzbiety (doliny) dwóch sąsiadujących fal przechodzą przez ten sam punkt.

https://github.com/przem85/bootcamp/blob/master/bootcamp_extra/D02_Z01.ipynb

Szereg Fouriera

Niech dana będzie funkcja okresowa $f : \mathbb{R} \rightarrow \mathbb{R}$ o okresie $T \in \mathbb{R}^+$, bezwzględnie całkowalna w przedziale $[-\frac{T}{2}, \frac{T}{2}]$.

Trygonometrycznym szeregiem Fouriera funkcji f nazywamy szereg funkcyjny następującej postaci:

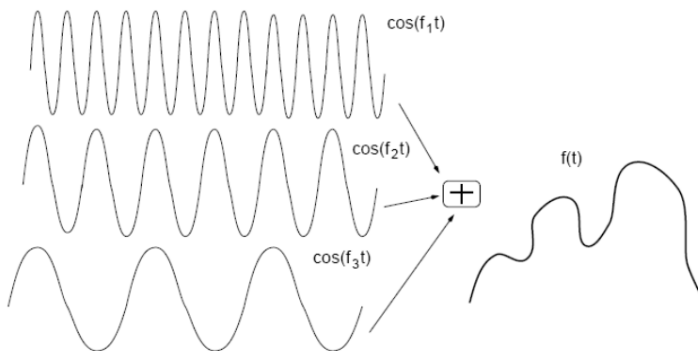
$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \left(a_n \cos\left(\frac{2n\pi}{T}x\right) + b_n \sin\left(\frac{2n\pi}{T}x\right) \right) \quad (1.1)$$

O współczynnikach określonych następującymi wzorami:

$$a_n = \frac{2}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(x) \cos\left(\frac{2n\pi}{T}x\right) dx, \quad n = 0, 1, 2, \dots$$

$$b_n = \frac{2}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(x) \sin\left(\frac{2n\pi}{T}x\right) dx, \quad n = 1, 2, 3, \dots$$

Transformata Fouriera



https://github.com/przem85/bootcamp/blob/master/bootcamp_extra/D02_Z02.ipynb

Transformata Fouriera

Transformacja Fouriera rozkłada funkcję okresową na szereg funkcji okresowych tak, że uzyskana transformata podaje w jaki sposób poszczególne częstotliwości składają się na pierwotną funkcję.

Definition

Dla funkcji $f(x)$ **transformatę Fouriera** definiujemy następująco:

$$F(f(x)) = F(s) = \int f(x) \cdot e^{-2\pi ixs} dx.$$

Definition

Dla funkcji $f(x)$ **transformatę Fouriera** definiujemy następująco:

$$F(f(x)) = F(s) = \int f(x) \cdot e^{-2\pi ixs} dx.$$

Zauważmy, że przy tak postawionej definicji możemy $f(x)$ zapisać w następujący sposób:

$$F^{-1}(F(s)) = f(x) = \int F(s) \cdot e^{2\pi ixs} ds.$$

gdzie

$$e^{\pm i\alpha} = \cos(\alpha) \pm i \sin(\alpha)$$

W praktyce często zmienna x oznacza czas (w sekundach), a argument transformaty s oznacza częstotliwość (w $\text{Hz} = \frac{1}{s}$).

Zauważmy, że:

$$f(x) = E(x) + O(x),$$

gdzie $E(x)$ jest pewną funkcją parzystą zmiennej x , natomiast $O(x)$ jest pewną funkcją nieparzystą.

Wtedy transformata Fouriera funkcji f redukuje się do postaci:

$$2 \int_0^{\infty} E(x) \cdot \cos(2\pi xs) dx - 2i \int_0^{\infty} O(x) \sin(2\pi xs) dx$$

Stąd łatwo widać, że jeżeli funkcja jest parzysta, to jej transformata jest parzysta, a jeżeli funkcja jest nieparzysta, to jej transformata jest również nieparzysta.

$$2 \int_0^{\infty} E(x) \cdot \cos(2\pi xs) dx - 2i \int_0^{\infty} O(x) \sin(2\pi xs) dx$$

Ponieważ w praktyce w wyniku pomiarów otrzymujemy dane o charakterze dyskretnym, a nie ciągłym, konieczne jest zdefiniowanie dyskretnego odpowiednika ciągłej transformaty Fouriera (zastępuje się całkę poprzez sumę):

Definition

Dla N -elementowego ciągu x_n **dyskretną transformatę Fouriera** definiujemy następująco:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} nk}, \quad k = 0, 1, \dots, N-1$$

Liczenie DFT z definicji:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} nk}, \quad k = 0, 1, \dots, N-1$$

gdzie x_0, \dots, x_{N-1} to próbki sygnału, wymagało (w latach 60-tych) tak ogromnych mocy obliczeniowych, że maszyny z tego okresu ograniczały użycie tego algorytmu.

https://github.com/przem85/bootcamp/blob/master/statistics/D16_Z02.ipynb

Rok 1965 przyniósł rewolucję. J. Cooley i J. Tuckey opublikowali pracę pod tytułem “An Algorithm for the machine computation of complex Fourier series”, w której opracowali szybszy algorytm liczenia dyskretnej transformaty Fouriera powszechnie znany jako szybka transformata Fouriera (ang. FFT – Fast Fourier Transform).

FFT jest to DFT ze zmniejszoną liczbą niezbędnych operacji arytmetycznych. Celem FFT jest zmniejszenie długiego algorytmu obliczeniowego przez jego podział na krótsze i prostsze obliczenia DFT i skrócenie czasu obliczeń. Istnieją różne algorytmy FFT.

Transformata Fouriera

Sama idea algorytmu opiera się na tzw. lemacie Danielsona-Lanczosa. Odkryli oni, że pojedyncza DFT o długości N , jest równoważna sumie dwóch transformat o długości $N/2$, jedna z nich jest złożona z nieparzystych punktów spośród oryginalnych N , a druga z parzystych.

$$\begin{aligned} X_k = & \underbrace{\sum_{m=0}^{N/2-1} x_{2m} e^{-\frac{2\pi i}{N/2} mk}}_{\text{DFT of even-indexed part of } x_m} + e^{-\frac{2\pi i}{N} k} \underbrace{\sum_{m=0}^{N/2-1} x_{2m+1} e^{-\frac{2\pi i}{N/2} mk}}_{\text{DFT of odd-indexed part of } x_m} = \\ & = E_k + e^{-\frac{2\pi i}{N} k} O_k. \end{aligned}$$

http:

[//en.wikipedia.org/wiki/Coolley%E2%80%93Tukey_FFT_algorithm](http://en.wikipedia.org/wiki/Coolley%E2%80%93Tukey_FFT_algorithm)

Zadania - filtrowanie sztucznego sygnału

https://github.com/przem85/bootcamp/blob/master/bootcamp_extra/D02_Z03.ipynb

https://github.com/przem85/bootcamp/blob/master/bootcamp_extra/D02_Z04.ipynb

https://github.com/przem85/bootcamp/blob/master/bootcamp_extra/D02_Z05.ipynb

https://github.com/przem85/bootcamp/blob/master/bootcamp_extra/D02_Z06.ipynb

Independent Component Analysis

(ICA)

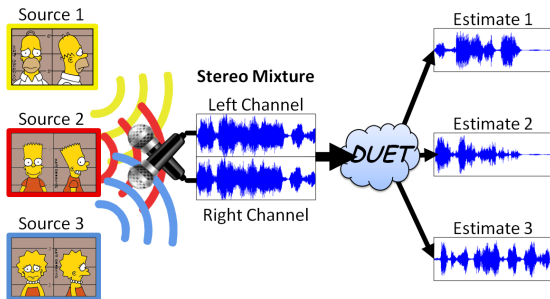
Cocktail-party problem

Imagine that you are in a room where two people are speaking simultaneously. We could express this as a linear equation:

$$\begin{cases} x_1(t) = a_{11}s_1 + a_{12}s_2 \\ x_2(t) = a_{21}s_1 + a_{22}s_2 \end{cases}$$

It would be very useful if you could now estimate the two original speech signals $s_1(t)$ and $s_2(t)$, using only the recorded signals $x_1(t)$ and $x_2(t)$.

Cocktail-party problem



- https://github.com/przem85/bootcamp/blob/master/bootcamp_extra/D03_Z01_ICA_signals.ipynb

Vector-matrix notation

- Let us denote by \mathbf{x} the random vector whose elements are the mixtures x_1, \dots, x_n ,
- Let us denote by \mathbf{s} the random vector with elements s_1, \dots, s_n .
- Let us denote by \mathbf{A} the matrix with elements a_{ij} .

The above mixing model is written as

$$\mathbf{x} = \mathbf{A}\mathbf{s}.$$

- Without loss of generality, we can assume that both the mixture variables and the independent components have zero mean.

After estimating the matrix \mathbf{A} , we can compute its inverse, say \mathbf{W} , and obtain the independent component simply by:

$$\mathbf{s} = \mathbf{W}\mathbf{x}.$$

Ambiguities of ICA

- We cannot determine the variances (energies) of the independent components. The reason is that, both \mathbf{s} and \mathbf{A} being unknown, any scalar multiplier in one of the sources s_i could always be cancelled by dividing the corresponding column \mathbf{a}_i of \mathbf{A} by the same scalar. The most natural way to do this is to assume that each has unit variance: $E\{s_i^2\} = 1$.
- Note that this still leaves the ambiguity of the sign: we could multiply the an independent component by -1 without affecting the model.
- We cannot determine the order of the independent components.

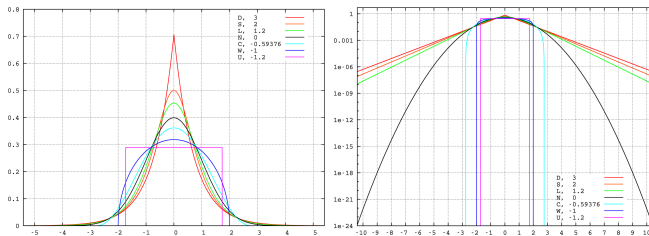
https://github.com/przem85/bootcamp/blob/master/bootcamp_extra/D03_Z02_ICA_signals.ipynb

Why Gaussian variables are forbidden

The fundamental restriction in ICA is that the independent components must be nongaussian for ICA to be possible.

<http://fourier.eng.hmc.edu/e161/lectures/ica/node3.html>

- https://github.com/przem85/bootcamp/blob/master/bootcamp_extra/D03_Z03_ICA_nongaussian.ipynb
- https://github.com/przem85/bootcamp/blob/master/bootcamp_extra/D03_Z04_ICA_nongaussian.ipynb



Example

A typical example is the Laplace distribution, whose pdf (normalized to unit variance) is given by

$$p(y) = \frac{1}{\sqrt{2}} \exp(\sqrt{2}|y|)$$

- https://github.com/przem85/bootcamp/blob/master/bootcamp_extra/D03_Z05_ICA_sound.ipynb
- https://github.com/przem85/bootcamp/blob/master/bootcamp_extra/D03_Z06_ICA_sound.ipynb
- https://github.com/przem85/bootcamp/blob/master/bootcamp_extra/D03_Z07_ICA_img.ipynb
- https://github.com/przem85/bootcamp/blob/master/bootcamp_extra/D03_Z08_EEG.ipynb

Outlier detection

- https://github.com/przem85/bootcamp/blob/master/bootcamp_extra/D04_Z01_outlier_detection.ipynb
- https://github.com/przem85/bootcamp/blob/master/bootcamp_extra/D04_Z02_outlier_detection.ipynb