

1 Kontinuierlicher Fall

Gegeben eine integrierbare (in L^2 -Norm konvergente) Funktion $f : \mathbb{C} \rightarrow \mathbb{C}$ mit Periode T (meist $T = 2\pi$), also mit $f(t) = f(t + T)$ für alle $t \in \mathbb{C}$ oder $f : [0, T] \rightarrow \mathbb{C}$. Dann ist ihre Fourier-Reihe gegeben durch

$$f_N(t) = \sum_{n=-N}^N c_n e^{in\omega t} \text{ mit } \omega = \frac{T}{2\pi}, c_n = \frac{1}{T} \int_0^T f(t) e^{in\omega t} dt$$

Dies gilt jedoch nicht für auf dem ganzen Raum definierte Funktionen. Deshalb geht man von der Summenbildung über in eine Integraltransformation:

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt$$
$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{i\omega t} d\omega$$

1.1 Beispiel

Die Stufenfunktion

$$f(t) = \begin{cases} 1, & \text{für } -1 \leq t \leq 1 \\ 0, & \text{sonst} \end{cases}$$

hat die Fouriertransformierte

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt = \int_{-1}^1 e^{-i\omega t} dt = \int_0^1 e^{-i\omega t} + e^{i\omega t} dt = \int_0^1 \cos(\omega t) dt = \left[\frac{\sin(\omega t)}{\omega} \right]_{t=0}^1 = \frac{\sin(\omega)}{\omega}$$

1.2 Beispiel

Allgemeiner hat für $c > 0$ die Stufenfunktion

$$f(t) = \begin{cases} 1, & \text{für } -c \leq t \leq c \\ 0, & \text{sonst} \end{cases}$$

die Fouriertransformierte

$$F(\omega) = \int_{-c}^c e^{-i\omega t} dt = \left[\frac{\sin(\omega t)}{\omega} \right]_{t=0}^c = \frac{\sin(\omega c)}{\omega}$$

Je grösser also c ist, desto gestauchter ist die Fouriertransformierte.

2 Diskreter Fall

Für die Praxis am Computer sind keine kontinuierlichen Funktionen, sondern diskrete Werte zu gebrauchen. Deswegen approximiert man die Integraltransformation durch eine diskrete Transformation. Gegeben ist also eine endliche Menge von Werten $\{a_0, \dots, a_{N-1}\}$, also ein N -dimensionaler Vektor a , dem ein N -dimensionaler Vektor \hat{a} zugeordnet wird:

$$\hat{a}_k = \sum_{j=0}^{N-1} w^{jk} a_j$$
$$a_k = \frac{1}{N} \sum_{j=0}^{N-1} w^{-jk} \hat{a}_j$$

Dabei ist $w = e^{2\pi i/N}$ die N -te Einheitswurzel.

2.1 Beispiel

$$a = \begin{pmatrix} 0 \\ 1 \\ 0 \\ -1 \end{pmatrix}$$

Hier ist $N = 4$ und damit $w^0 = 1, w^1 = i, w^2 = -1, w^3 = -i, w^4 = 1, \dots$

$$\hat{a}_0 = \sum_{j=0}^{N-1} w^0 a_j = w^0 a_0 + w^0 a_1 + w^0 a_2 + w^0 a_3 = 0 + 1 + 0 - 1 = 0$$

$$\hat{a}_1 = \sum_{j=0}^{N-1} w^j a_j = w^0 a_0 + w^1 a_1 + w^2 a_2 + w^3 a_3 = 0 + i + 0 + i = 2i$$

$$\hat{a}_2 = \sum_{j=0}^{N-1} w^{2j} a_j = w^0 a_0 + w^2 a_1 + w^4 a_2 + w^6 a_3 = 0 - 1 + 0 + 1 = 0$$

$$\hat{a}_3 = \sum_{j=0}^{N-1} w^{3j} a_j = w^0 a_0 + w^3 a_1 + w^6 a_2 + w^9 a_3 = 0 - i + 0 - i = -2i$$

$$\hat{a} = \begin{pmatrix} 0 \\ 2i \\ 0 \\ -2i \end{pmatrix}$$

2.2 Alternative Darstellung

Man kann diese Berechnung auch als Matrix-Vektor-Multiplikation auffassen. F ist die Fouriermatrix.

$$\hat{a} = F \cdot a$$

$$F = \begin{pmatrix} w^0 & w^0 & w^0 & \dots & w^0 \\ w^0 & w^1 & w^2 & \dots & w^{N-1} \\ w^0 & w^2 & w^4 & \dots & w^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w^0 & w^{N-1} & w^{2(N-1)} & \dots & w^{(N-1)(N-1)} \end{pmatrix}$$

Man sieht leicht, dass dieser Berechnungsansatz einen quadratischen Aufwand hat. Deshalb wird als nächstes ein Algorithmus vorgestellt, der die diskrete Fouriertransformation schneller durchführen kann.

3 Algorithmus

Dieser Algorithmus funktioniert nur für einen Vektor der Länge einer Zweier-Potenz. Wir nehmen daher nun an, dass a ein 2^m -dimensionaler Vektor ist, wobei $m \in \mathbb{N}_0$.

3.1 Herleitung

Der Korrektheitsbeweis des Algorithmus geht induktiv über $m \in \mathbb{N}_0$. Ist $m = 0$, so ist $\hat{a} = a$, denn $\hat{a}_0 = w^0 a_0 = a_0$. Ist $m > 0$, so ist a ein $2N$ -dimensionaler Vektor mit $N = 2^{m-1}$. Nun kann nach Induktions-Vorraussetzung angenommen werden, dass die diskrete Fouriertransformation für einen N -dimensionalen Vektor bereits durchgeführt werden kann. Ziel ist also nun das Problem auf diesen Fall zu reduzieren. Dazu spaltet man die Berechnungsvorschrift in gerade und ungerade Terme auf.

$$\hat{a}_k = \sum_{j=0}^{2N-1} w^{jk} a_j = \sum_{j=0}^{N-1} w^{2jk} a_{2j} + \sum_{j=0}^{N-1} w^{(2j+1)k} a_{2j+1} = \sum_{j=0}^{N-1} w^{2jk} a_{2j} + w^k \cdot \sum_{j=0}^{N-1} w^{2jk} a_{2j+1}$$

Es bezeichne nun $v = w^2$ die N -te Einheitswurzel. Zudem seien nun $g_0 = a_0, g_1 = a_2, \dots, g_{N-1} = a_{2(N-1)}$ die geraden und $u_0 = a_1, \dots, u_{N-1} = a_{2N-1}$ die ungeraden Koeffizienten.

$$\hat{a}_k = \underbrace{\sum_{j=0}^{N-1} v^{jk} g_j}_{\hat{g}_k} + v^{k/2} \cdot \underbrace{\sum_{j=0}^{N-1} v^{jk} u_j}_{\hat{u}_k}$$

Für $k < N$ liefert dieser Ausdruck bereits das richtige Ergebnis. Für $k \geq N$ beachte man, dass $v^{N/2}$ eine halbe Drehung darstellt, also ist dann das

richtige Ergebnis gegeben durch

$$\hat{a}_k = \hat{g}_{k-N} - v^{k/2} \hat{u}_{k-N}$$

3.2 Komplexität

$$T(1) = c$$

$$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) + c \cdot 2 \frac{n}{2} = 2 \cdot T\left(\frac{n}{2}\right) + c \cdot n$$

Mit dem Mastertheorem $a = 2, b = 2, \alpha = 1$ folgt $T(n) \in O(n \log n)$.

4 Anwendung

Der Algorithmus kommt in sehr vielen Bereichen zur Anwendung. Zwei von ihnen werden hier kurz vorgestellt.

4.1 Polynom-Multiplikation

Sind $p(x) = p_0 + p_1x + p_2x^2 + \dots + p_nx^n$ und $q(x) = q_0 + q_1x + q_2x^2 + \dots + q_nx^n$ zwei Polynome. Dann hat der naive Berechnungsansatz quadratischen Aufwand. Fasst man die Koeffizienten als $1 + n + m$ -dimensionale Vektoren p und q auf, wobei $p_j = q_k = 0$ für $j > n$ und $k > n$, so kann man zeigen, dass $f^{-1}(\hat{p} * \hat{q})$ der Vektor der Koeffizienten des Produktpolynoms ist. Dabei soll f^{-1} die inverse schnelle Fouriertransformation darstellen und $*$ komponentenweise Multiplikation. Da der schnelle Fouriertransformations-Algorithmus einen Aufwand von $n \log n$ hat und komponentenweise Multiplikation sogar nur linearen Aufwand, hat dieser Algorithmus zur Polynom-Multiplikation einen Aufwand von $n \log n$.

4.2 Kompressions-Algorithmen

Gegeben ist ein grosser Datensatz (zum Beispiel .BMP- oder .WAV- Datei) und man wendet den schnellen Fouriertransformations-Algorithmus an. Die so entstandenen neuen Werte können geeignet abgeändert werden, sodass eine Menge an Speicherplatz gespart wird, aber bei der Rücktransformation die alten Daten im Wesentlichen rekonstruiert werden können.