KTH Computer Science and Communication

Övning 3

Snabb Fouriertransform

Den snabba Fouriertransformen, FFT, var känd av Gauss redan på tidigt 1800-tal. Det var emellertid när den återupptäcktes på sextiotalet av Cooley och Tukey och började användas på datorer som dess styrka och användbarhet blev uppenbar. Idag är det en av de i praktiken mest använda numeriska algoritmerna. FFT tillämpas i huvudsak i ämnen som signalteori och bildanalys men används t.ex. även i så kallade spektralmetoder för att lösa partiella differentialekvationer.

Bakgrund

När f tillhör $L^2([0, 2\pi])$ kan funktionen utvecklas i en Fourierserie,

$$f(x) = \sum_{k=-\infty}^{\infty} \hat{f}_k e^{ikx}, \qquad \hat{f}_k = \frac{1}{2\pi} \int_0^{2\pi} f(x) e^{-ikx} dx.$$

Här kallas $\{\hat{f}_k\}$ funktionens Fourierkoefficienter. Koefficienten $\hat{f}_k$ kan tolkas som funktionens frekvensinnehåll vid frekvens k. Vi vill kunna beräkna dessa koefficienter snabbt.

När f är bandbegränsad till $n/2$ frekvenser (n jämt tal), dvs när

$$f(x) = \sum_{k=-n/2}^{n/2-1} \hat{f}_k e^{ikx},$$

kan $\hat{f}_k$ beräknas exakt från n ekvidistanta samples av $f(x)$. Detta är innehållet i Shannons samplingssats. Man kan uttrycka det som att man behöver två samplingspunkter per våglängd (hos den högsta frekvensen) för att kunna återskapa funktionen exakt.

Vi inför lite mer notation. Låt

$$f_j = f(x_j), \qquad x_j = jh, \qquad h = \frac{2\pi}{n},$$

och definiera den diskreta innerprodukten,

$$\langle f, g \rangle_n := \sum_{j=0}^{n-1} f(x_j) \overline{g(x_j)}.$$

Då har vi speciellt att

$$\begin{aligned} \langle e^{ikx}, e^{i\ell x} \rangle_n &= \sum_{j=0}^{n-1} e^{i(k-\ell)x_j} = \sum_{j=0}^{n-1} e^{i(k-\ell)hj} = \sum_{j=0}^{n-1} \left(e^{i(k-\ell)h} \right)^j \\ &= \begin{cases} \frac{1-e^{i(k-\ell)hn}}{1-e^{i(k-\ell)h}}, & k \neq \ell, \\ n, & k = \ell, \end{cases} = \begin{cases} 0, & k \neq \ell, \\ n, & k = \ell, \end{cases} \end{aligned}$$

eftersom $hn = 2\pi$. Om vi tar innerprodukten mellan f och $\exp(i\ell x)$ och utnyttjar detta result får vi

$$\sum_{j=0}^{n-1} f_j e^{-i\ell x_j} = \langle f, e^{i\ell x}\rangle_n = \left\langle \sum_{k=-n/2}^{n/2-1} \hat{f}_k e^{ikx}, e^{i\ell x} \right\rangle_n = \sum_{k=-n/2}^{n/2-1} \hat{f}_k \langle e^{ikx}, e^{i\ell x}\rangle_n = n\hat{f}_\ell.$$

Alltså ges $\hat{f}_\ell$ av

$$\hat{f}_\ell = \frac{1}{n}\sum_{j=0}^{n-1} f_j e^{-i\ell x_j}.$$

Detta är den *diskreta Fouriertransformen* (DFT), som beräknar Fourierkoefficienterna $\{\hat{f}_\ell\}$ från de samplade värdena $\{f_j\}$.

Omformulering till matrisform

Vi kan nu skriva detta i matrisform. Låt

$$\boldsymbol{f} = \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_{n-1} \end{pmatrix}, \qquad \hat{\boldsymbol{f}} = n\begin{pmatrix} \hat{f}_{-n/2} \\ \hat{f}_{-n/2+1} \\ \vdots \\ \hat{f}_{n/2-1} \end{pmatrix}, \qquad \boldsymbol{f}, \hat{\boldsymbol{f}} \in \mathbb{C}^n.$$

och definiera

$$\omega_n = \exp(-i2\pi/n).$$

Transformen kan skrivas som,

$$\hat{\boldsymbol{f}} = \boldsymbol{F}_n \boldsymbol{f},$$

där $\boldsymbol{F}_n \in \mathbb{C}^{n\times n}$ är en matris, för vilken elementet i rad ℓ och kolumn j är givet av

$$e^{-i(-n/2+\ell)x_j} = e^{-i(-n/2+\ell)jh} = e^{-2\pi i(-n/2+\ell)j/n} = e^{\pi ij - 2\pi i\ell j/n} = (-1)^j \omega_n^{\ell j},$$

med $\ell, j = 0, \ldots, n-1$, det vill säga

$$\boldsymbol{F}_n = \begin{pmatrix} 1 & -1 & 1 & \cdots & 1 & -1 \\ 1 & -\omega_n & \omega_n^2 & \cdots & \omega_n^{n-2} & -\omega_n^{n-1} \\ 1 & -\omega_n^2 & \omega_n^4 & \cdots & \omega_n^{2(n-2)} & -\omega_n^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & -\omega_n^{n-1} & \omega_n^{2(n-1)} & \cdots & \omega_n^{(n-2)(n-1)} & -\omega_n^{(n-1)^2} \end{pmatrix}. \tag{1}$$

(För tydlighetens skull: $\omega_n^{kj} = (\omega_n)^{kj} = \exp(-i2\pi kj/n)$.) Den diskreta Fouriertransformen är alltså i grunden en matris-vektormultiplikation. En naiv implementation av transformen har därför den vanliga kostnaden för en matris-vektormultiplikation, nämligen $\mathcal{O}(n^2)$.

Snabba transformen

Kostnaden $\mathcal{O}(n^2)$ minskas radikalt om man använder den snabba Fouriertransformen. Normalt beskrivs denna för fallet då längden av $\boldsymbol{f}$ är en jämn potens av 2, alltså $n = 2^m$ med m ett positivt heltal. Den är då baserad på observationen att $\boldsymbol{F}_n$ kan delas upp i enkla operationer i termer av $\boldsymbol{F}_{n/2}$.

Låt $\boldsymbol{f}^{\text{even}}$ och $\boldsymbol{f}^{\text{odd}}$ vara vektorerna som innehåller *vartannat* element av $\boldsymbol{f}$,

$$\boldsymbol{f}^{\text{even}} = \begin{pmatrix} f_0 \\ f_2 \\ \vdots \\ f_{n-2} \end{pmatrix}, \qquad \boldsymbol{f}^{\text{odd}} = \begin{pmatrix} f_1 \\ f_3 \\ \vdots \\ f_{n-1} \end{pmatrix}, \qquad \boldsymbol{f}^{\text{even}}, \boldsymbol{f}^{\text{odd}} \in \mathbb{C}^{n/2}.$$

Efter att ha kastat om motsvarande kolumner i $\boldsymbol{F}_n$ har vi då

$$\boldsymbol{F}_n \boldsymbol{f} = \begin{pmatrix} 1 & 1 & 1 & \cdots & -1 & -1 & \cdots \\ 1 & \omega_n^2 & \omega_n^4 & \cdots & -\omega_n & -\omega_n^3 & \cdots \\ 1 & \omega_n^4 & \omega_n^8 & \cdots & -\omega_n^2 & -\omega_n^6 & \cdots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \cdots & \\ 1 & \omega_n^{2(n-1)} & \omega_n^{4(n-1)} & \cdots & -\omega_n^{n-1} & -\omega_n^{3(n-1)} & \cdots \end{pmatrix} \begin{pmatrix} \boldsymbol{f}^{\text{even}} \\ \boldsymbol{f}^{\text{odd}} \end{pmatrix} =: A\boldsymbol{f}^{\text{even}} - \boldsymbol{\Lambda} A \boldsymbol{f}^{\text{odd}},$$

där $\boldsymbol{\Lambda}$ är en diagonal matris, $\boldsymbol{\Lambda} = \operatorname{diag}(1, \omega_n, \omega_n^2, \ldots, \omega_n^{n-1})$ och $A \in \mathbb{C}^{n \times \frac{n}{2}}$ är en rektangulär (obs, ej kvadratisk) matris given av

$$A = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & \omega_n^2 & \cdots & \omega_n^{n-2} \\ 1 & \omega_n^4 & \cdots & \omega_n^{2(n-2)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{2(n-1)} & \cdots & \omega_n^{(n-1)(n-2)} \end{pmatrix} =: \begin{pmatrix} a_{00} & a_{01} & \cdots & a_{0,n/2-1} \\ a_{10} & a_{11} & \cdots & a_{1,n/2-1} \\ a_{20} & a_{21} & \cdots & a_{2,n/2-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n-1,0} & a_{n-1,1} & \cdots & a_{n-1,n/2-1} \end{pmatrix}.$$

Matriselementen a_{jk} kan alltså uttryckas $a_{jk} = \omega_n^{2jk}$.

Vi visar nu att både A och $\boldsymbol{\Lambda}$ går att skriva om på en form som förenklar beräkningarna. Först noterar vi att

$$a_{jk} = \omega_n^{2jk} = \exp(-i4\pi jk/n) = \omega_{n/2}^{jk}.$$

Det implicerar att övre kvadratiska delmatrisen av A, med indexen $j, k = 0, \ldots, n/2 - 1$, är identisk med $\boldsymbol{F}_{n/2}$. Vidare gäller att

$$a_{j+n/2,k} = \omega_n^{2(j+n/2)k} = \exp(-i4\pi(j + n/2)k/n) = \omega_{n/2}^{jk} \exp(-i2\pi k) = \omega_{n/2}^{jk},$$

vilket visar att även den undre delmatrisen av A är lika med $\boldsymbol{F}_{n/2}$. Slutligen har vi att

$$\omega_n^{n/2+k} = \omega_n^{n/2} \omega_n^k = \exp(-i\pi)\omega_n^k = -\omega_n^k.$$

De övre diagonalelementen hos $\boldsymbol{\Lambda}$ är därför samma som de undre med omvänt tecken. Med andra ord,

$$A = \begin{pmatrix} \boldsymbol{F}_{n/2} \\ \boldsymbol{F}_{n/2} \end{pmatrix}, \qquad \boldsymbol{\Lambda} = \begin{pmatrix} \boldsymbol{D}_{n/2} & 0 \\ 0 & -\boldsymbol{D}_{n/2} \end{pmatrix}, \qquad \boldsymbol{D}_{n/2} = \operatorname{diag}(1, \omega_n, \omega_n^2, \ldots, \omega_n^{n/2-1}).$$

Till slut får vi det enkla sambandet

$$\boldsymbol{F}_n \boldsymbol{f} = \begin{pmatrix} \boldsymbol{F}_{n/2} & -\boldsymbol{D}_{n/2}\boldsymbol{F}_{n/2} \\ \boldsymbol{F}_{n/2} & \boldsymbol{D}_{n/2}\boldsymbol{F}_{n/2} \end{pmatrix} \begin{pmatrix} \boldsymbol{f}^{\text{even}} \\ \boldsymbol{f}^{\text{odd}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{F}_{n/2}\boldsymbol{f}^{\text{even}} \\ \boldsymbol{F}_{n/2}\boldsymbol{f}^{\text{even}} \end{pmatrix} + \begin{pmatrix} -\boldsymbol{D}_{n/2}\boldsymbol{F}_{n/2}\boldsymbol{f}^{\text{odd}} \\ \boldsymbol{D}_{n/2}\boldsymbol{F}_{n/2}\boldsymbol{f}^{\text{odd}} \end{pmatrix}. \tag{2}$$

En transform av n variabler kan alltså beräknas genom att göra två transformer av $n/2$ variabler: $\boldsymbol{F}_{n/2}\boldsymbol{f}^{\text{even}}$ och $\boldsymbol{F}_{n/2}\boldsymbol{f}^{\text{odd}}$. I den snabba Fouriertransformen utnyttjar man detta rekursivt.

Vardera $\boldsymbol{F}_{n/2}$-transformen delas på samma sätt upp i två $\boldsymbol{F}_{n/4}$-transformer, vilka i sin tur delas upp i två $\boldsymbol{F}_{n/8}$-transformer, etc. Vi får

$$\boldsymbol{F}_n \to 2 \text{ st } \boldsymbol{F}_{n/2} \to 4 \text{ st } \boldsymbol{F}_{n/4} \to 8 \text{ st } \boldsymbol{F}_{n/8} \to \cdots \to n \text{ st } \boldsymbol{F}_1,$$

och $\boldsymbol{F}_1 = 1$ så transformen behöver aldrig utföras. För att beräkna den totala kostnaden av detta låter vi b_n beteckna kostnaden för en transform av n variabler. Om vi antar att alla element i $\boldsymbol{D}_{n/2}$ är givna (beräknade i förväg), får vi rekursionsformeln

$$b_n = 2b_{n/2} + \frac{3}{2}n, \qquad b_1 = 0, \tag{3}$$

där termen $3n/2$ kommer från multiplikationen med $\boldsymbol{D}_{n/2}$ och additionerna, och $b_1 = 0$ eftersom $\boldsymbol{F}_1 = 1$. Man kan nu via induktion visa att (3) ger

$$b_n = \frac{3}{2}n \log_2 n, \tag{4}$$

det vill säga att den snabba fouriertransformen har beräkningskostnaden $\mathcal{O}(n \log n)$. (Värdet på konstanten framför n är mindre intressant.)

(Bevis: För $n = 1$ ger (4) att $b_1 = 0$. Antag att (4) uppfylls för $n = 1, 2, 4, \ldots, 2^p$. Genom att använda (3) och (4) får vi för $n = 2^{p+1}$:

$$b_{2^{p+1}} = 2b_{2^{p+1}/2} + \frac{3}{2}2^{p+1} = 2b_{2^p} + 3\,2^p = 2\Big(\frac{3}{2}2^p \log_2(2^p)\Big) + 3\,2^p = 3p2^p + 3\,2^p = \frac{3}{2}2^{p+1}(p+1).$$

Via induktion följer att (4) gäller för alla $n = 2^m$, $m = 0, 1, \ldots$)

Matlab

I Matlab finns kommandot `fhat=fft(f)` för att beräkna Fourierkoefficienterna för vektorn `f`. Ordningen av koefficienterna i `fhat` är lite annorlunda än i $\hat{\boldsymbol{f}}$ ovan, nämligen

$$\texttt{fhat} = n[\hat{f}_0, \hat{f}_1, \ldots, \hat{f}_{n/2}, \hat{f}_{-n/2+1}, \ldots, \hat{f}_{-1}].$$

För att gå mellan de två ordningarna kan man använda kommandot `fftshift`. Kommandot `f=ifft(fhat)` ger den inversa Fouriertransformen. Se `help fft` för mer information.