Fourier Transformation

Hy Truong Son

August 2016

1 Discrete Fourier Transform 1D

Fourier transform:

$$\hat{f_k} = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} f_n . e^{-i2\pi k \frac{n}{N}}$$

In the case that the original signal (f_n) is real:

$$\hat{f_k} = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} f_n \left[\cos\left(2\pi k \frac{n}{N}\right) - i \sin\left(2\pi k \frac{n}{N}\right) \right]$$

$$Re(\hat{f_k}) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} f_n . \cos\left(2\pi k \frac{n}{N}\right)$$

$$Im(\hat{f_k}) = -\frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} f_n . \sin\left(2\pi k \frac{n}{N}\right)$$

Inverse Fourier transform:

$$f_n = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \hat{f_k} . e^{i2\pi k \frac{n}{N}}$$

In the case that the original signal (f_n) is real:

$$f_n = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \hat{f_k} \left[\cos\left(2\pi k \frac{n}{N}\right) + i \sin\left(2\pi k \frac{n}{N}\right) \right]$$

$$f_n = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \left[Re(\hat{f_k}) + iIm(\hat{f_k}) \right] \left[\cos\left(2\pi k \frac{n}{N}\right) + i \sin\left(2\pi k \frac{n}{N}\right) \right]$$

$$f_n = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \left[Re(\hat{f_k}) \cos\left(2\pi k \frac{n}{N}\right) - Im(\hat{f_k}) \sin\left(2\pi k \frac{n}{N}\right) \right]$$

Amplitude:

$$|\hat{f}_k| = \sqrt{Re(\hat{f}_k)^2 + Im(\hat{f}_k)^2}$$

Phase:

$$arg(\hat{f}_k) = atan2(Im(\hat{f}_k), Re(\hat{f}_k))$$

2 Discrete Fourier Transform 2D

Fourier transform:

$$\hat{f}_{x,y} = \frac{1}{\sqrt{MN}} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} f_{u,v}.e^{-i2\pi(\frac{xu}{M}+\frac{yv}{N})}$$

$$\hat{f}_{x,y} = \frac{1}{\sqrt{MN}} \sum_{u=0}^{M-1} e^{-i2\pi\frac{xu}{M}}.\left(\sum_{v=0}^{N-1} f_{u,v}.e^{-i2\pi\frac{yv}{N}} \right)$$

Let $P(u,y) = \sum_{v=0}^{N-1} f_{u,v}.e^{-i2\pi\frac{yv}{N}}$. Then:

$$\hat{f}_{x,y} = \frac{1}{\sqrt{MN}} \sum_{u=0}^{M-1} e^{-i2\pi\frac{xu}{M}}.P(u,y)$$

Inverse Fourier transform:

$$f_{u,v} = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \hat{f}_{x,y}.e^{i2\pi(\frac{xu}{M}+\frac{yv}{N})}$$

$$f_{u,v} = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} e^{i2\pi\frac{xu}{M}}.\left(\sum_{y=0}^{N-1} \hat{f}_{x,y}.e^{i2\pi\frac{yv}{N}} \right)$$

Let $H(x,v) = \sum_{y=0}^{N-1} \hat{f}_{x,y}.e^{i2\pi\frac{yv}{N}}$. Then:

$$f_{u,v} = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} e^{i2\pi\frac{xu}{M}}.H(x,v)$$

The Fourier image (to visualize amplitudes or phases) is shifted in such a way that the value (image mean) $\hat{f}(0,0)$ is displayed in the center of the image. The further way from the center an image point is, the higher is its corresponding frequency. The dynamic range of the Fourier coefficients (for example, the intensity values in the Fourier image) is too large to be displayed on the screen, therefore all other values appear as black. We need to apply a logarithmic transformation to the image as follows:

$$Q(i,j) = c\log(1 + |P(i,j)|)$$

where $P(i,j)$ is the original image, and $Q(i,j)$ is the transformed image. The scaling constant c is chosen so that the maximum output value is 255 (providing an 8-bit format). That means if R is the value with the maximum magnitude in the input image, c is given by:

$$c = \frac{255}{\log(1+|R|)}$$

3 Fast Fourier Transform 1D

Cooley-Tukey FFT algorithm was invented by Carl Friedrich Gauss and then rediscovered by Cooley and Tukey (the radix-2 Decimation In Time or DIT case). DFT is defined by the formula:

$$\hat{f}_k = \sum_{n=0}^{N-1} f_n . e^{-\frac{2\pi i}{N} nk} \quad as\ k = 0, 1, .., N-1$$

Radix-2 DIT first computes the DFTs of the even-indexed inputs ($f_{2m} = f_0, f_2, .., f_{N-2}$) and of the odd-indexed inputs ($f_{2m+1} = f_1, f_3, .., f_{N-1}$) and then combines those two results to produce the DFT of the whole sequence:

$$\hat{f}_k = \sum_{m=0}^{N/2-1} f_{2m} . e^{-\frac{2\pi i}{N}(2m)k} + \sum_{m=0}^{N/2-1} f_{2m+1} . e^{-\frac{2\pi i}{N}(2m+1)k}$$

$$\hat{f}_k = \sum_{m=0}^{N/2-1} f_{2m} . e^{-\frac{2\pi i}{N/2} mk} + e^{-\frac{2\pi i}{N}k} \sum_{m=0}^{N/2-1} f_{2m+1} . e^{-\frac{2\pi i}{N/2} mk}$$

$$\hat{f}_k = E_k + e^{-\frac{2\pi i}{N}k} . O_k$$

where E_k is the DFT of the even-indexed part of f_m, and O_k is the DFT of the odd-indexed part of f_m. Based on the periodicity of the DFT: $E_{k+\frac{N}{2}} = E_k$, $O_{k+\frac{N}{2}} = O_k$. We have the following:

$$\hat{f}_k = E_k + e^{-\frac{2\pi i}{N}k} . O_k \quad for\ 0 \leq k < \frac{N}{2}$$

$$\hat{f}_k = E_{k-\frac{N}{2}} + e^{-\frac{2\pi i}{N}k} . O_{k-\frac{N}{2}} \quad for\ \frac{N}{2} \leq k < N$$

The harmonic factor $e^{-2\pi ik/N}$ have the property that:

$$e^{-\frac{2\pi i}{N}(k+\frac{N}{2})} = e^{\frac{-2\pi ik}{N} - \pi i} = -e^{-\frac{2\pi ik}{N}}$$

We can cut the number of harmonic factor calculations in half. For $0 \leq k < \frac{N}{2}$:

$$\hat{f}_k = E_k + e^{-\frac{2\pi i}{N}k} . O_k$$

$$\hat{f}_{k+\frac{N}{2}} = E_k - e^{-\frac{2\pi i}{N}k}.O_k$$

Pseudocode:
01: $\hat{f}_{0,1,..,N-1} \leftarrow FFT(f, N, s):$ *returns the DFT of* $(f_0, f_s, f_{2s}, ..)$
02: *if* $N = 1$ *then*
03: $\hat{f}_0 \leftarrow f_0$
04: *else*
05: $\hat{f}_{0,..,N/2-1} \leftarrow FFT(f, N/2, 2s)$
06: $\hat{f}_{N/2,..,N-1} \leftarrow FFT(f + s, N/2, 2s)$
07: *for* $k = 0$ *to* $N/2 - 1$
08: $t \leftarrow \hat{f}_k$
09: $\hat{f}_k \leftarrow t + exp(-2\pi ik/N)\hat{f}_{k+N/2}$
10: $\hat{f}_{k+N/2} \leftarrow t - exp(-2\pi ik/N)\hat{f}_{k+N/2}$
11: *endfor*
12: *endif*
Note that: $f+s$ (in the context of $C++$ programming language) means moving the array pointer of f to the s-th element.

4 Fast Fourier Transform 2D

Discrete Fourier Transform 2D:

$$\hat{f}_{x,y} = \frac{1}{\sqrt{MN}} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} f_{u,v}.e^{-i2\pi\left(\frac{xu}{M}+\frac{yv}{N}\right)}$$

$$\hat{f}_{x,y} = \frac{1}{\sqrt{MN}} \sum_{u=0}^{M-1} e^{-i2\pi\frac{xu}{M}}.\left(\sum_{v=0}^{N-1} f_{u,v}.e^{-i2\pi\frac{yv}{N}}\right)$$

We can compute $P(u, y) = \sum_{v=0}^{N-1} f_{u,v}.e^{-i2\pi\frac{yv}{N}}$ by FFT-1D. Again, we use FFT-1D to compute $\hat{f}_{x,y} = \frac{1}{\sqrt{MN}} \sum_{u=0}^{M-1} e^{-i2\pi\frac{xu}{M}}.P(u, y)$.

Inverse Discrete Fourier Transform 2D:

$$f_{u,v} = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \hat{f}_{x,y}.e^{i2\pi\left(\frac{xu}{M}+\frac{yv}{N}\right)}$$

$$f_{u,v} = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} e^{i2\pi\frac{xu}{M}}.\left(\sum_{y=0}^{N-1} \hat{f}_{x,y}.e^{i2\pi\frac{yv}{N}}\right)$$

Compute $H(x, v) = \sum_{y=0}^{N-1} \hat{f}_{x,y}.e^{i2\pi\frac{yv}{N}}$ and then $f_{u,v} = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} e^{i2\pi\frac{xu}{M}}.H(x, v)$ by FFT-1D algorithm.

5 Fast Polynomial Multiplication by FFT

Given two polynomials:

$$A(x) = \sum_{i=0}^{n-1} a_i.x^i$$

$$B(x) = \sum_{i=0}^{m-1} b_i.x^i$$

Compute:

$$C(x) = A(x).B(x) = \sum_{i=0}^{k-1} c_i.x^i \quad (k = n + m - 1)$$

where $c_i = \sum_j a_j.b_{i-j}$.

Let N be the smallest power of 2 that is bigger than or equal to k. We can rewrite $A(x)$, $B(x)$, $C(x)$ as polynomials of degree $N-1$ as follows:

$$A(x) = \sum_{i=0}^{n-1} a_i.x^i + \sum_{i=n}^{N-1} 0.x^i$$

$$B(x) = \sum_{i=0}^{m-1} b_i.x^i + \sum_{i=m}^{N-1} 0.x^i$$

$$C(x) = \sum_{i=0}^{k-1} c_i.x^i + \sum_{i=k}^{N-1} 0.x^i$$

Let w be the N-th complex root of unity: $w = cos(\frac{2\pi}{N}) + i.sin(\frac{2\pi}{N})$. Some properties of w:

$$w^j = \left[cos\left(\frac{2\pi}{N}\right) + i.sin\left(\frac{2\pi}{N}\right)\right]^j = cos\left(\frac{2\pi}{N}j\right) + i.sin\left(\frac{2\pi}{N}j\right)$$

$$w^0 = 1,\ w^N = 1,\ w^{j+\frac{N}{2}} = -w^j,\ w^{j+N} = w^j$$

We will use the idea of the Fast Fourier Transform algorithm to compute the values of $A(x)$ and $B(x)$ at this special series $W = \{w^0, w^1, w^2, .., w^{N-1}\}$ in time complexity $O(NlogN)$. Then we can compute the values of $C(x)$ at W in time complexity $O(N)$. Finally, from these values, we can intepolate to compute the coefficients of $C(x)$ again by FFT.

Let $P(x)$ be an arbitrary polynomial of degree $N-1$: $P(x) = \sum_{i=0}^{N-1} p_i.x^i$. We have:

$$P(w^j) = \sum_{i=0}^{N-1} p_i.(w^j)^i$$

$$\Leftrightarrow P(w^j) = \sum_{k=0}^{N/2-1} p_{2k}.(w^j)^{2k} + \sum_{k=0}^{N/2-1} p_{2k+1}.(w^j)^{2k+1}$$

$$\Leftrightarrow P(w^j) = \sum_{k=0}^{N/2-1} p_{2k}.(w^{2j})^k + w^j.\sum_{k=0}^{N/2-1} p_{2k+1}.(w^{2j})^k$$

On the another hand:

$$P(w^{j+N/2}) = \sum_{k=0}^{N/2-1} p_{2k}.(w^{2j+N})^k + w^{j+N/2}.\sum_{k=0}^{N/2-1} p_{2k+1}.(w^{2j+N})^k$$

$$\Leftrightarrow P(w^{j+N/2}) = \sum_{k=0}^{N/2-1} p_{2k}.(w^{2j})^k - w^j.\sum_{k=0}^{N/2-1} p_{2k+1}.(w^{2j})^k$$

Pseudocode of computing the DFT $F(P)$ or the evaluation of $P(x)$ at $1, w, w^2, .., w^{N-1}$:
01: $F_{0,1,..,N-1} \leftarrow FFT(p, N, s, w) :$ $\quad returns\ the\ DFT\ of\ (p_0, p_s, p_{2s}, ..)$
02: $\quad if\ N = 1\ then$
03: $\quad\quad F_0 \leftarrow p_0$
04: $\quad else$
05: $\quad\quad F_{0,..,N/2-1} \leftarrow FFT(p, N/2, 2s, w^2)$
06: $\quad\quad F_{N/2,..,N-1} \leftarrow FFT(p+s, N/2, 2s, w^2)$
07: $\quad\quad x \leftarrow 1$
08: $\quad\quad for\ j = 0\ to\ N/2-1$
09: $\quad\quad\quad t \leftarrow F_j$
10: $\quad\quad\quad F_j \leftarrow t + x.F_{j+N/2}$
11: $\quad\quad\quad F_{j+N/2} \leftarrow t - x.F_{j+N/2}$
12: $\quad\quad\quad x \leftarrow x.w$
13: $\quad\quad endfor$
14: $\quad endif$

The final algorithm is:
Step 1. Fourier transform for evaluation - time $O(NlogN)$: $F_A \leftarrow FFT(A, N, 1, w)$
Step 2. Fourier transform for evaluation - time $O(NlogN)$: $F_B \leftarrow FFT(B, N, 1, w)$
Step 3. Time $O(N)$:
$for\ i = 1\ to\ N$
$F_C[i] \leftarrow F_A[i].F_B[i]$
$endfor$
Step 4. Fourier Transform for interpolation - time $O(NlogN)$: $C \leftarrow \frac{1}{N}.FFT(F_C, N, 1, w^{-1})$
Note that $w^{-1} = cos(\frac{2\pi}{N}) - i.sin(\frac{2\pi}{N})$

6 Fast Integer Multiplication by FFT

Given two decimal integer $A = (a_{n-1}, .., a_1, a_0)$ and $B = (b_{m-1}, .., b_1, b_0)$. A and B can be expressed in terms of polynomials:

$$A(10) = \sum_{i=0}^{n-1} a_i.10^i$$

$$B(10) = \sum_{i=0}^{m-1} b_i.10^i$$

Then the product of A and B is equal to $A(10).B(10)$. Apply the Fast Fourier Transform algorithm as described above, we can compute the multiplication polynomial $C(x) = A(x).B(x)$ faster, so the result will be $C(10)$.

7 Continuous Fourier Transform

Basic of Fourier analysis:

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \hat{f}(w).e^{ixw} dw$$

$$\hat{f}(w) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \hat{f}(x).e^{-ixw} dw$$

$\mathcal{F}(f) = \hat{f}(w)$ is the Fourier transform of $f(x)$.

Plancharel theorem:
- The Fourier transform preserves inner product, if $\hat{f}$ is the Fourier transform of f and $\hat{g}$ is the Fourier transform of g, then: $\langle \hat{f}(w), \hat{g}(w) \rangle = \langle f(x), g(x) \rangle$.
- $\|f(x)\|^2 = \|\hat{f}(w)\|^2$

Some other basic properties of Fourier Transform:
Assume that $\hat{f} = \mathcal{F}(f)$. Then:
(i) $\mathcal{F}(f(x-c))(w) = e^{-iwc}.\hat{f}(w)$
(ii) $\mathcal{F}(f(cx))(w) = \frac{1}{c}.\hat{f}(\frac{w}{c})$

8 Discrete Cosine Transform 2D

Discrete Cosine Transform:

$$\hat{f}_{u,v} = \frac{1}{4}\alpha(u)\alpha(b) \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f_{x,y} cos\left[\frac{(2x+1)u\pi}{2M}\right] cos\left[\frac{(2y+1)v\pi}{2N}\right]$$

where $\alpha(u) = \frac{1}{\sqrt{2}}$ if $u = 0$, and $\alpha(u) = 1$ otherwise.
Inverse Discrete Cosine Transform:

$$f_{x,y} = \frac{1}{4} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} \hat{f}_{u,v}.\alpha(u)\alpha(v)cos\left[\frac{(2x+1)u\pi}{2M}\right]cos\left[\frac{(2y+1)v\pi}{2N}\right]$$