Fast/Discrete Fourier Transform

Group 23

UM-SJTU Joint Institute

November 28, 2018

Fourier Transform

Definition

Fourier transform is a decomposition of a funtion of time (a signal) into frequencies that make it up.

- The main strategy is to see signals as acuumulations of sine waves of different frequencies, and this can be represented as the following

$$f(x) = \int_{-\infty}^{\infty} F(s) e^{2i\pi sx} ds$$

Discrete Fourier Transform

Limitation on FT

In fourier transform, both data in the time domain and frequency domain are continuous, which are not feasible to store and process in computers.

Discrete Fourier Transform (DFT) solves this question as it processes a finite sample sequence of time domain data and convert it into a same-length sample data for frequency domain.

Mathematical Formula

DFT will decompose a sequence $\{x_n\}$ of length N into the following style

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k \cdot e^{i2\pi kn/N}$$

And thus the sequence X_k which represents how much a certain frequency is in the sample can be written as

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{i2\pi kn/N}$$

Fast Fourier Transform

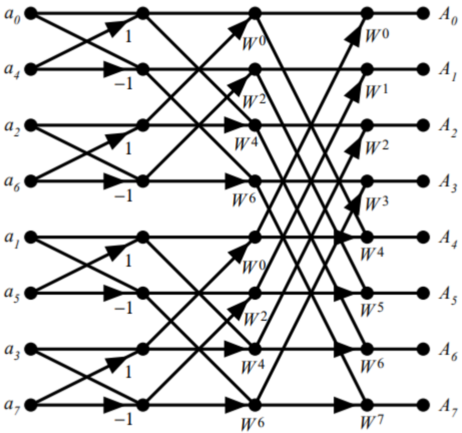
If the calculation of *DFT* is operated naively, then it is clear that the time complexity is $\mathcal{O}(n^2)$ for a length-n sequence.

And thus Fast Fourier Transform(FFT) is introduced, which serves to find the internal mathematical basis and simplify the calculation, usually to $\mathcal{O}(n \log n)$. And here the algorithm introduced will be Cooley-Tukey algorithm
The algorithm uses the idea of divide and conquer, so here the case when $n = 2^p$ is considered.

Cooley-Tukey Algorithm

a: input sequence, A: output sequence

W: $e^{-i2\pi/n}$, $weight_{j,k} = W^{\frac{8}{2^j}\times k}$



Discussion on Bitwise Reverse

The most naive method in bitwise reverse is iteratively divide the number by 2 to get all the bits. In this way reversing one number should be $\mathcal{O}(\log n)$ complexity.

I found that when the number of bits is a power of 2, this can do it in the way of flipping every adjacent 2,4,...,n/2 bits to reverse it.

So the algorithm I am using now is to find the smallest 2-power number larer than the number of bits now, flipping in that way then right shift the extra 0s in the end.

References

- https://www.cs.cmu.edu/afs/andrew/scs/cs/15-463/2001/pub/www/notes/fourier/fourier.pdf
- James W. Cooley, Peter A. W. Lewis, and Peter W. Welch, "Historical notes on the fast Fourier transform"
- https://en.wikipedia.org/wiki/Fast_Fourier_transform