Computational Complexity of DFT

Samantha R. Summerson

26 October, 2009

1 Review DTFT and DFT

Recall the formula for the DTFT and the inverse DTFT:

$$\begin{aligned} S\left(e^{j2\pi f}\right) &= \sum_{n=-\infty}^{\infty} s(n)e^{-j2\pi fn}, \\ s(n) &= \int_{-\frac{1}{2}}^{\frac{1}{2}} S\left(e^{j2\pi f}\right)e^{j2\pi fn}. \end{aligned}$$

The spectra of discrete-time signals are periodic with a period of 1. The DFT is the DTFT sampled at $f = \frac{k}{N}$. The formulas for the DFT and the inverse DFT are:

$$\begin{aligned} S(k) &= \sum_{n=0}^{N-1} s(n)e^{-j2\pi\frac{k}{N}n}, \\ s(n) &= \frac{1}{N}\sum_{k=0}^{N-1} S(k)e^{j2\pi\frac{k}{N}n}. \end{aligned}$$

2 Computational Complexity

The DFT requires $2N^2$ real multiplies and adds.

$$\sum_{n=0}^{N-1} s(n)\left(cos\left(2\pi\frac{k}{N}n\right) - jsin\left(2\pi\frac{k}{N}n\right)\right)$$

How can we make this faster? One thing to notice is that we can take advantage of the periodicity of *sine* and *cosine*. Unfortunately, this only reduces the number of operations to $O\left(N^2\right)$. We need to do something smarter. The *Fast Fourier Transform (FFT)* was invented by Gauss in 1805, and later re-discovered by Cooley and Tukey in 1965. This methods requires only $O\left(Nlog_2(N)\right)$ operations. The **trick**: assume that $N = 2^L$ (if this is not the case, we simply pad our signal with zeros to make its length a power of 2). We start by re-ordering the terms in the DFT.

$$\begin{aligned} S(k) &= s(0) + s(2)e^{-j2\pi\frac{2k}{N}} + ... + s(N-2)e^{-j2\pi\frac{(N-2)k}{N}} + s(1)e^{-j2\pi\frac{k}{N}} + s(3)e^{-j2\pi\frac{(2+1)k}{N}} + ... + s(N-1)e^{-j2\pi\frac{(N-1)k}{N}}, \\ &= s(0) + s(2)e^{-j2\pi\frac{2k}{N}} + ... + s(N-2)e^{-j2\pi\frac{(N-2)k}{N}} + e^{-j2\pi\frac{k}{N}}\left(s(1) + s(3)e^{-j2\pi\frac{2k}{N}} + ... + s(N-1)e^{-j2\pi\frac{(N-2)k}{N}}\right), \\ &= s(0) + s(2)e^{-j2\pi\frac{k}{N/2}} + ... + s(N-2)e^{-j2\pi\frac{(N/2-1)k}{N/2}} + e^{-j2\pi\frac{k}{N}}\left(s(1) + s(3)e^{-j2\pi\frac{k}{N/2}} + ... + s(N-1)e^{-j2\pi\frac{(N/2-1)k}{N/2}}\right) \end{aligned}$$

We notice that this looks like a sum of two length $\frac{N}{2}$ DFTs, with one of the DFTs scaled by $e^{-j2\pi\frac{k}{N}}$ (which we call the *twiddle factor*). We can continue dividing the DFTs in half, since the orginal DFT was of length that was a power of two, until we have a sum of DFTs of length 2. This method requires $O\left(N log_2(N)\right)$ operations.

L	2^L
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024
11	2048
12	4096