4 傅里叶变换

xht03

2025-04-13 18:51:20

DFT 与 FFT

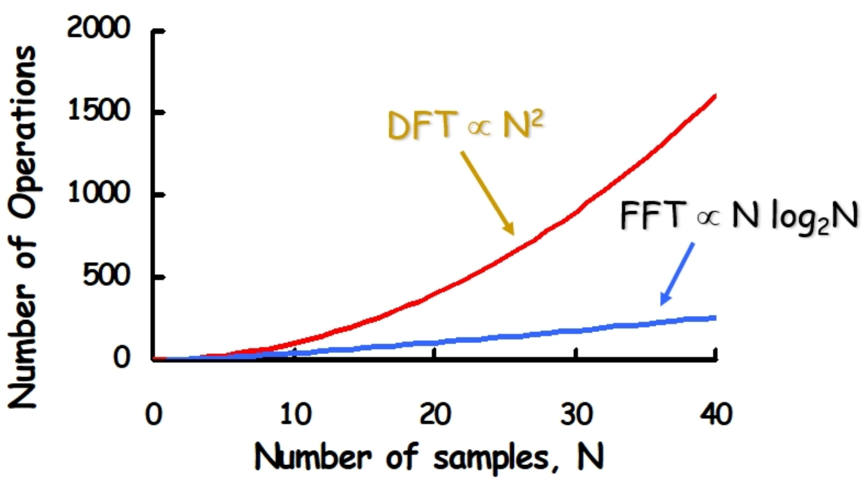
在实际的信号处理中，我们都采用**离散傅里叶变换**（DFT）来对信号进行处理。

$$X(k) = DFT[x(n)] = \sum_{n=0}^{N-1} x(n) W_N^{nk} \quad 0 \le k \le N-1$$

计算每个 $X(k)$ 需要 N 次乘法和 $N-1$ 次加法，而完成整个 DFT 需要 N^2 次乘法和 $N(N-1)$ 次加法。DFT 的时间复杂度为 $O(N^2)$ ，对于大规模的信号处理，这个计算量是不可接受的。

因此，我们需要一种更高效的算法来计算 DFT。**快速傅里叶变换**（FFT）就是为了解决这个问题而提出的。

FFT 本质是一种分治算法。FFT 利用旋转因子 W_N^{nk} （也就是 N 阶单位根）的性质，避免重复计算，将 DFT 的计算复杂度降低到 $O(N \log N)$ 。FFT 算法的基本思想是将一个长度为 N 的序列分解成两个长度为 $\frac{N}{2}$ 的子序列，然后递归地计算这两个子序列的 DFT，最后将结果合并起来。



FFT 原理

为了简单起见，我们只考虑长度为 $N = 2^m$ 的序列。

$$\begin{aligned}
X(k) &= \sum_{n=0}^{N-1} x(n) W_N^{nk} \quad 0 \le k \le N-1 \\
&= \sum_{n=0}^{N/2-1} x(2n) W_N^{2nk} + \sum_{n=0}^{N/2-1} x(2n+1) W_N^{(2n+1)k} \\
&= \sum_{n=0}^{N/2-1} g(n) \left(W_N^2\right)^{nk} + \sum_{n=0}^{N/2-1} h(n) \left(W_N^2\right)^{nk} W_N^k \\
&= \sum_{n=0}^{N/2-1} g(n) W_{N/2}^{nk} + W_N^k \sum_{n=0}^{N/2-1} h(n) W_{N/2}^{nk} \\
&= G(k) + W_N^k H(k) \quad (0 \le k \le N/2-1)
\end{aligned}$$

其中 $g(n) = x(2n)$ 和 $h(n) = x(2n+1)$ 分别是偶序列和奇序列。

但注意到上述推导中 k 的范围是 $0 \le k \le N-1$，而我们只计算了 $0 \le k \le \frac{N}{2}-1$ 的情况。

当 $\frac{N}{2} \le k \le N-1$，我们可以利用单位根的对称性：

$$\begin{aligned}
X(k+N/2) &= \sum_{n=0}^{N/2-1} g(n) W_N^{2n(k+N/2)} + \sum_{n=0}^{N/2-1} h(n) W_N^{(2n+1)(k+N/2)} \\
&= \sum_{n=0}^{N/2-1} g(n) W_N^{2nk} \cdot W_N^{Nn} + W_N^{\frac{(2n+1)N}{2}} \sum_{n=0}^{N/2-1} h(n) W_N^{(2n+1)k} \\
&= \sum_{n=0}^{N/2-1} g(n) W_N^{2nk} + (-1) \sum_{n=0}^{N/2-1} h(n) W_N^{(2n+1)k} \\
&= G(k) - W_N^k H(k), \quad (0 \le k \le N/2-1)
\end{aligned}$$

综上，我们可以得到：全部 N 个点的离散傅里叶变换。

$$\begin{aligned}
&X(k) = G(k) + W_N^k H(k) \\
&X(k+N/2) = G(k) - W_N^k H(k) \\
&\text{其中} \quad 0 \le k \le N/2-1
\end{aligned}$$

所以，我们可以将一个 N 点的 DFT 分解成两个 $\frac{N}{2}$ 点的 DFT，递归地计算这两个 DFT，最后将结果合并起来。

FFT 实现

详细代码实现在这里。