Convolutions and the Fast Fourier Transform - A Study

Surya Raghav B

June 2024

1 The Problem

Given two vectors $a = (a_0, a_1, \ldots, a_{n-1}$ and $b = (b_0, b_1, \ldots, b_{n-1}$, we can combine them by convolution $a * b$, which is a vector with $2n-1$ coordinates, where the k^{th} coordinate equals

$$\sum_{(i,j);i+j=k;i,j<n} a_i b_j.$$

It can be extended to vectors of different lengths also.

We have following circumstances where convolution can be used.

- Polynomial multiplication. The coefficients of the polynomials can be written as a vector $a = (a_0, a_1, \ldots, a_{n-1}$ and $b = (b_0, b_1, \ldots, b_{n-1}$, whose convolution gives a vector which contains the coefficients of multiplied polynomial.
- Signal Processing. Smoothing of signal can be done taking weighted average withing k steps to the left and right. For Gaussian smoothing we can define a mask $w = (w_{-k}, w_{-(k-1)}, \ldots, w_{-1}, w_0, w_1, \ldots, w_{k-1}, w_k)$, where $w_i = 1/Z(e^{-i^2})$. We can just convolve the signal and the mask to get the smoothed signal.
- We can also convolution for combining histograms, showing the number of pairs of (a,b) that have the combined frequency k.

2 Designing the algorithm

Computing the convolution normally takes $O(n^2)$ time as we have to find $a_i b_j$ for each i,j and perform $O(n^2)$ additions in addition to the multiplications.

Now we can treat the vectors as polynomial functions and multiply as following to get $C(x) = A(x)B(x)$:

1. Choose 2n values and $A(x)$ and $B(x)$ on them.

2. Now $C(x)$ for the 2n values is just the product of $A(x)$ and $B(x)$ at the points.

3. Since any d-degree polynomial can reconstructed from any set of d+1 or more points, we can the 2n valuations of $C(x)$ to recover its coefficients.

Steps (i), (ii) take only $O(n)$ time, so now we will look at performing step (iii) in $O(nlogn)$.

Complex Roots of Unity For the 2n sample points we can choose the $(2n)^{th}$ roots of unity. They are $w_{j,2n} = e^{2\pi ji/2n}$ (for $j = 0, 1, 2, \ldots, 2n-1$). The representation of d-degree polynomial P by its values on the $(d+1)^{st}$ roots of unity is the discrete Fourier transform of P.

Recursive Procedure for Polynomial Evaluation We design an algorithm to evaluate the polynomial A on the 2n roots of unity recursively. We have $A(x) = a_0 + a_1x + a_2x^2 + \ldots + a_{n-1}x^{n-1}$, which we can write as

$$A(x) = A_{even}(x^2) + xA_{odd}(x^2)$$

where

$$A_{even}(x) = a_0 + a_2x + \ldots + a_{n-2}x^{(n-2)/2}$$

and

$$A_{odd}(x) = a_1 + a_2x + \ldots + a_{n-1}x^{(n-2)/2}$$

Now we can evaluate A_{even} and A_{odd} on the nth roots of unity.

$$A(w_{j,2n}) = A_{even}(w_{j,2n}^2) + w_{j,2n}A_{odd}(w_{j,2n}^2)$$

Here $w_{j,2n}^2$ is actually the nth roots of unity. So now the problem is recursively broken into n/2 steps and combining takes $O(n)$ time. Thus the computation is bounded by $T(n) \leq 2T(n/2) + O(n)$, which gives us the bound of $O(nlogn)$.

Polynomial Interpolation Once we have found the value of C on the 2n values, we have reconstruct the coefficients of the polynomial C.

Let $C(x) = \sum_{s=0}^{2n-1} c_s x^s$ be the polynomial that we want to reconstruct from its values $C(w_{s,2n})$. We define new polynomial $D(x) = \sum_{s=0}^{2n-1} d_s x^s$, where

$d_s = C(w_{s,2n})$, now consider the values of $D(x)$ at the 2n roots of unity.

$$\begin{aligned}
D(w_{j,2n}) &= \sum_{s=0}^{2n-1} C(w_{s,2n}) w_{j,2n}^s \\
&= \sum_{s=0}^{2n-1} (\sum_{t=0}^{2n-1} c_t w_{s,2n}^t) w_{j,2n}^s \\
&= \sum_{t=0}^{2n-1} c_t (\sum_{s=0}^{2n-1} w_{s,2n}^t) w_{j,2n}^s \\
&= \sum_{s=0}^{2n-1} c_t (\sum_{t=0}^{2n-1} e^{(2\pi i)(st+js)/2n}) \\
&= \sum_{t=0}^{2n-1} c_t (\sum_{s=0}^{2n-1} w_{t+j,2n}^s)
\end{aligned}$$

To analyze the last line we that sum of n roots of unity is zero. Thus $\sum_{s=0}^{2n-1} w_{t+j,2n}^s$ will be zero for all values of $t+j$ except when $t+j$ is a multiple of 2n, that is, $t = 2n - j$. For this value, $\sum_{s=0}^{2n-1} w_{t+j,2n}^s = 2n$. So we get $D(w_{j,2n}) = 2nc_{2n-j}$. Thus evaluating $D(x)$ at 2n roots of units gives us the coefficients of $C(x)$ in the reverse order, where $c_s = \frac{1}{2n} D(w_{2n-s,2n})$.

The evaluations of the values of $D(x)$ can be done in $O(nlogn)$ time using divide and conquer. So in total our algorithm computes the convolution of two vectors in $O(nlogn)$ time.