Compressed Matrix Multiplication & Extensions

FFT and Convolution

$$[a \otimes b]_i = \sum_{j=1}^{n} a_j \; b_{(i-j) \bmod n}$$

(recall - continuous was $\int f(x)\, g(\tau - x)\, dx = [f \otimes g](\tau)$

by symmetry we have $a \otimes b = b \otimes a$

Since $u := i - j \Rightarrow j = i - u$

$$\Rightarrow \sum_{u} a_{i-u}\, b_u = [b \otimes a]_i$$

FFT (on discrete domain)

given sequence $a \in \mathbb{R}^n$, define

$$A_k = \sum_{j=1}^{n} a_j \; e^{-2\pi i \cdot \frac{jk}{n}}$$

$$\Rightarrow a_k = \frac{1}{n} \sum_{j=1}^{n} A_j \; e^{2\pi i \cdot \frac{jk}{n}}$$

$$A := F[a] \; ; \quad a := F^{-1}[A]$$

FFT - convolution theorem

$$F[a \otimes b]_\ell = \sum_{j=1}^{n} \left(\sum_{k=1}^{n} a_k \; b_{(j-k) \bmod n} \right) e^{-2\pi i \cdot \frac{(j-k+k)\,\ell}{n}}$$

$$= F[a]_\ell \; F[b]_\ell$$

$$\Rightarrow \boxed{a \otimes b = F^{-1}\left[F[a] \cdot F[b] \right]}$$

in $O(n \log n)$ time

Hash functions

Recall: pairwise indep. hash functions are

$$Pr_h \{ h(x) = i \; ; \; h(x') = j \} = \frac{1}{N^2}$$

for $x \neq x'$ and for all i, j

Likewise for Rademacher hash

$$Pr_b (b(x) = i \; ; \; b(x') = j \} = \frac{1}{4}$$

for $x \neq x'$ and for all $i, j \in \{\pm 1\}$

Lemma: given pairwise indep. hash funcs h_1, h_2 we have that $h(i,j) := h_1(i) + h_2(j) \bmod N$ is pairwise indep.

Likewise for $b(i,j) := b_1(i) \cdot b_2(j)$

Proof: $\{ h_1(x_1) + h_2(x_2) = i \}$

is $\{ h_1(x_1) = a \; ; \; h_2(x_2) = b$ and $a + b \bmod N = i \}$

this occurs with prob $\frac{1}{N}$

ditto for (x_1', x_2')

{due to independence all $(a, b), (a', b')$ occur with same probability $\frac{1}{N^4}$

$\Rightarrow$ Sum over all N^2 cases yields claim

$\Rightarrow$ analogous for b

Compressed Matrix Multiplication

Idea: $M_{il} = A \cdot B^+ = \sum_{j=1}^{k} A_{ij} B_{lj}$

$T_i = \sum_{j=1}^{k} \prod_{l=1}^{d} A_{i_l j}$

→ We want to store M, T efficiently

How we use

a) Count Sketch

for all (i,l)

$s(a, h_a(i,l)) \mathrel{+}= \sigma_a(i,l)\, M_{il}$

ditto for tensors T with multi-index i

b) count min sketch

$s(a, h_a(i,l)) \mathrel{+}= M_{il}$

retrieval a) Count Sketch

$\text{median}_a \{ \sigma_a(i,l)\, s(a, h_a(i,l)) \;\forall a \}$ $O(\frac{\|\cdot\|}{N})$ whp

b) Count-min Sketch

$\min_a \{ s(a, h_a(i,l)) \;\forall a \}$ $O(\frac{1}{N})$ whp

goal: compute S as fast as possible

→ naive: $O(n_1 n_2 k)$

or $O(\Pi n \cdot k)$

→ efficient $O(dN \log N \cdot k)$

this is much better and faster!

doing it (we start with matrix product. the rest is very similar)

$M = A \cdot B^+ = \sum_{j=1}^{k} a_j \cdot b_j^+$

by linearity we have that

$S(M) = \sum_{j=1}^{k} S(a_j \cdot b_j^+)$

how we focus on $S(a_j \cdot b^+)$

$S(a \cdot b^+, l) = \sum_{h_1(i)+h_2(j)=l} \sigma_1(i)\,\sigma_2(j)\, a_i\, b_j$

$= \sum_u \Big[\sum_{h_1(i)=u} \sigma_1(i)\, a_i \Big] \Big[\sum_{h_2(j)=l-u} \sigma_2(j)\, b_j \Big]$

$= S_1(a) \circledast S_2(b)$

overall algorithm:

compute $S_1(a_j)$, $S_2(b_j)$

$S(AB) \mathrel{+}= F^{-1}\big(F(S_1(a_j)) \odot F(S_2(b_j)) \big)$

this has cost $O\big(k \cdot \big(\underbrace{b \log b}_{\text{FFT}} + \underbrace{N_1 + N_2}_{\text{inverse FFT}} \big)\big)$

↑ dimension

the values

Can also compute as product of polynomials ...

Compressed Matrix Multiplication

useful properties :

Symmetric matrices / tensors :

$$S(AB^t) = \sum_j F^{-1}\left(\lambda_j F(\delta(a)^4)^2\right)$$

Linearity : for diagonal dominant matrices we can compute

$$S(AB^t) - S(\mathrm{diag}(AB^t))$$

Covariance matrix estimation :

$$\mathrm{Cov}(X) = XX^t - (X\cdot 1)\cdot(X\cdot 1)^t$$
$$= \sum_i S(x_i x_i^t) - S((X\cdot 1)_\bullet (X\cdot 1)^t) \checkmark$$

moreover we can sketch the diagonal counts via

$\mathrm{diag}\ \mathrm{Cov}(X)_i = \mathbb{E}[x_{(i)}^2] - \mathbb{E}[x_{(i)}]^2$ ✓

and then use original hash representation ✓

thus needs $O(n\cdot(d + n\log n))$ CPU

(data dim; hash output storage)

and $O(n+d)$ RAM ✓

tail gauss for power law sketch to count heavy hitter sketch

better reconstruction :

- use Counter Braid algorithm for compact approximate matrix storage / transmission

$$S_b(AB^t) \to S_b'(S(AB^t)) \ldots.$$

this could be used for covariance matrix estimation (most off-diagonal terms are small)

- for tight law we get the better rates from C / CM sketch

variance of sketch

$$\hat{X}_{ij} = \delta(i,j) \sum_{(i',j'):h(i,j)=h(i',j')} \delta(i',j')\, X_{i',j'}$$

$\mathbb{E}[\hat{X}_{ij}] = X_{ij}$

for variance exploit 2-ways indep in $\delta(ij)$ to obtain

$$\mathrm{Var}[\hat{X}_{ij}] = \sum_{i',j' \neq ij} X_{i',j'}^2 \cdot \frac{1}{n} - \cancel{\mathrm{Var}}$$
$$= \left[\|X\|_{Frob}^2 - X_{ij}^2\right]\cdot\frac{1}{n}$$

use Chebyshev for $\mathrm{Var}[\,]$ + median

Compressed Matrix Multiplicak

Finding heavy terms

key idea is to use ECC to exclude / include rows / colns in matrix such that heavy terms survive:

choose eg $E \in \{0,1\}^{n \times \ell}$ with $\ell = O(\log n)$ an error correcting code

now compute $S(\mathrm{diag}(E_{\cdot r}) \cdot A \cdot B^+) =: S_r$

and also $S(A \cdot B^+ \mathrm{diag}(E_{\cdot r})) =: S^r$

Decoding algorithm:

a) assume that there are no collisions (this is easy since only few nonzeros)

b) find all entries $> \Delta$ in sketch

i.e. for all large entries we get

$(0, 0, 1, 0, \dots)$ for a given position

$\Rightarrow$ decode rows
$\Rightarrow$ decode colns
$\Rightarrow$ done