Outline

• Factorization of Scalar Products in \( \mathbb{R}^d \) (compression)
  – Factorization in 2D.
  – Factorization in 3D.
  – Factorization in \( dD \).
  – Multinomial Coefficients.
  – Length of compressed vector.
  – Example.
  – Complexity of Fast Summation.
Compression

Compression operator:

\[ A^n = \text{Compress}(a^n) \]

Required Property:

\[ a^n \cdot b^n = \text{Compress}(a^n) \cdot \text{Compress}(b^n). \]

Consider \( \mathbf{R}^2 \):

\[ a^n \cdot b^n = (a \cdot b)^n = (a_1 b_1 + a_2 b_2)^n \]

\[ = a_1^n b_1^n + \binom{n}{1} a_1^{n-1} b_1^{n-1} a_2 b_2 + \binom{n}{2} a_1^{n-2} b_1^{n-2} a_2^2 b_2^2 + ... + a_2^n b_2^n \]

The length is only \((n + 1)\), not \(2^n\)

Let us define:

\[ A^n = \text{Compress}(a^n) = \left( a_1^n, \sqrt{\binom{n}{1} a_1^{n-1} a_2}, \sqrt{\binom{n}{2} a_1^{n-2} a_2^2}, ..., a_2^n \right) \]

\[ B^n = \text{Compress}(b^n) = \left( b_1^n, \sqrt{\binom{n}{1} b_1^{n-1} b_2}, \sqrt{\binom{n}{2} b_1^{n-2} b_2^2}, ..., b_2^n \right) \]
Compression Can be Performed for any Dimensionality (Example for 3D):

\[ a^n \cdot b^n = (a \cdot b)^n = (a_1 b_1 + a_2 b_2 + a_3 b_3)^n \]

\[ = [(a_1 b_1 + a_2 b_2) + a_3 b_3]^n = \sum_{m=0}^{n} \binom{n}{m} (a_1 b_1 + a_2 b_2)^{n-m} a_3^m b_3^m \]

\[ = \sum_{m=0}^{n} \sum_{l=0}^{n-m} \binom{n}{m} \binom{n-m}{l} a_1^{n-m-l} b_1^{n-m-l} a_2^l b_2^l a_3^m b_3^m \]

\[ = a_1^n b_1^n + \binom{n}{1} a_1^{n-1} b_1^{n-1} a_2 b_2 + \binom{n}{2} a_1^{n-2} b_1^{n-2} a_2^2 b_2^2 + \ldots + a_2^n b_2^n \]

\[ + \binom{n}{1} a_1^{n-1} b_1^{n-1} a_3 b_3 + \binom{n}{1} \binom{n-1}{1} a_1^{n-2} b_1^{n-2} a_2 b_2 a_3 b_3 + \ldots + a_3^n b_3^n \]

Compress(\(a^n\)) = \[\begin{pmatrix} a_1^n \\ \left(\begin{array}{c} n \\ 1 \end{array}\right) a_1^{n-1} a_2 \\ \left(\begin{array}{c} n \\ 2 \end{array}\right) a_1^{n-2} a_2^2, ..., a_2^n \\ \left(\begin{array}{c} n \\ 1 \end{array}\right) a_1^{n-1} a_3, ..., a_3^n \end{pmatrix}\]

The length of \(a^n\) is \((n+1)+n+\ldots+1= (n+1)(n+2)/2\)
Compression Can be Performed for any Dimensionality (General Case):

\[(a_1 + a_2 + \ldots + a_d)^n = \sum_{n_1 + \ldots + n_d = n} (n, n_1, n_2, \ldots, n_d) a_1^{n_1} a_2^{n_2} \ldots a_d^{n_d} .\]

\[(n, n_1, n_2, \ldots, n_d) = \frac{n!}{n_1! n_2! \ldots n_d!} .\]

Multinomial coefficients

\[\text{Compress}(a^n) = \left( a_1^n, \sqrt{(n, n-1, 1, 0, \ldots, 0)} a_1^{n-1} a_2, \ldots, \sqrt{(n, n_1, n_2, \ldots, n_d)} a_1^{n_1} a_2^{n_2} \ldots a_d^{n_d}, \ldots, a_d^n \right) .\]

So we have

\[a^n \cdot b^n = \text{Compress}(a^n) \cdot \text{Compress}(b^n)\]

\[= \sum_{n_1 + \ldots + n_d = n} (n, n_1, n_2, \ldots, n_d) a_1^{n_1} a_2^{n_2} \ldots a_d^{n_d} b_1^{n_1} b_2^{n_2} \ldots b_d^{n_d}\]

\[= (a_1 b_1 + a_2 b_2 + \ldots + a_d b_d)^n = (a \cdot b)^n.\]
What are multinomial coefficients?

(n ; n₁,n₂,…,n_d) is the number of ways of putting n different objects into d different boxes with n_k in the k-th box.

\[ n_1 + n_2 + \ldots + n_d = n \]
The length of the compressed vector

\[ d = 1 : \quad 1, \]
\[ d = 2 : \quad n + 1, \]
\[ d = 3 : \quad \frac{1}{2}(n + 1)(n + 2), \]

... 

**Theorem:** If \( \mathbf{a} \in \mathbb{R}^d \), then the length of compressed vector \( \text{Compress}(\mathbf{a}^n) \) is

\[
\binom{n + d - 1}{n} = \frac{(n + 1)...(n + d - 1)}{(d - 1)!}.
\]

**Proof:** We have a basis for induction (see above). Let this holds for \( d \) dimensions. Consider \( d + 1 \) dimensions:

\[
((a_1 + ... + a_d) + a_{d+1})^n = \sum_{m=0}^{n} \binom{n}{m} (a_1 + ... + a_d)^m a_{d+1}^{n-m}
\]

The number of terms is then

\[
\sum_{m=0}^{n} \binom{m + d - 1}{m} = \binom{d - 1}{0} + \binom{d}{1} + ... + \binom{n + d - 1}{n} = \binom{n + d}{n}
\]

This proves the theorem.
Example of Fast Computation

\[ v_j = \sum_{i=1}^{N} u_i \Phi(y_j, x_i) = \sum_{m=0}^{p-1} c_m \cdot (y_j - x \ast )^m + \text{Residual}, \quad c_m = \frac{1}{m!} \sum_{i=1}^{N} u_i e^{x \ast \ast x_i} x_i^m. \]

Equivalent to:

\[ v_j = \sum_{m=0}^{p-1} c_m \cdot \text{Compress}( (y_j - x \ast )^m ) + \text{Residual}, \quad c_m = \frac{1}{m!} \sum_{i=1}^{N} u_i e^{x \ast \ast x_i} \text{Compress}(x_i^m). \]

Number of multiplications (complexity) to obtain \( v_j \): \hspace{1cm} \text{(in 2D case!)}

\[ \text{Complexity} = 1 + 2 + \ldots + p = \frac{p(p+1)}{2}. \]

\[ C_0 = \sum_{i=1}^{N} u_i e^{x \ast \ast x_i}, \]

\[ C_1 = (C_{11}, C_{12}) = \sum_{i=1}^{N} u_i e^{x \ast \ast x_i} (x_{i1}, x_{i2}), \]

\[ C_2 = (C_{21}, C_{22}, C_{23}) = \sum_{i=1}^{N} u_i e^{x \ast \ast x_i} \left( x_{i1}^2, \sqrt{2} x_{i1} x_{i2}, x_{i2}^2 \right). \]
Complexity of Fast Summation

Let $\circ$ be a scalar product of vectors $A_i$ and $F_j$ of length $P(p)$ ($p$ is the truncation number). Complexity of summation over $i$ is then $O(PN)$. Complexity of scalar product operation is $P$. Complexity of $M$ scalar product operations is $O(PM)$ (for $j = 1, \ldots, M$). Total complexity is $O(PM + PN)$. Fast Method is more efficient than direct only if $O(PM + PN) < O(MN)$, so we should have

$$P(p) \ll \min(M, N)$$