Understanding Einsum

And it’s decomposition in terms of primitive operations

Nikita Vedeneev
nikitaved@github

Preliminaries

Index sets

- Index set $I(d) \coloneqq \{0, \ldots, d-1 | d > 0\}$
- Two index sets $I(a)$ and $J(b)$ are equal, i.e. $I(a) = J(b)$ if $a = b \vee a = 1 \vee b = 1$, or, alternatively, when the shapes $(a,)$ and $(b,)$ broadcast

Sets of index sets

Consider $I(1)$ and $J(3)$. Since $(1,)$ and $(3,)$ broadcast, $I(1) = J(3)$, so $\{I(1), J(3)\} = \{I(1)\} = \{J(3)\}$.

Indexing arrays

- Suppose a is a 2D array of shape (s_1, s_2), and $I(s_1), J(s_2)$ are index sets, then

  $a[I, J] = a[I(s_1), J(s_2)] \coloneqq$ a sequence $(a[i, j])_{ij}\ \forall (i, j) \in I(s_1) \times J(s_2)$,

  where $\times$ denotes the Cartesian product.

  Note that parameters s_1, s_2 are made implicit for lighter notation.

- If $I = J$, (i.e. $I(s_1) = J(s_2)$), then

$$a[I, J] \coloneqq \left(\begin{cases} a[i, j], i = j, \\ 0, i \neq j. \end{cases} \right)_{ij},\ \forall (i, j) \in i \times j$$

Note on indexing with equal index sets

If $I(s_1) = J(s_2)$, then $I(s_1) \times J(s_2) = J(s_2) \times I(s_1)$, and this implies $s_1 = s_2$. *This means we cannot use same index sets to index dimensions with different lengths!*

Indexing arrays

- Of course, we can generalize indexing with index sets to more than just 2 dimensions

$$\begin{aligned} a[I, I, J, J] &= (a[i_1, i_2, j_1, j_2])_{i_1 i_2 j_1 j_2} \\ &= \left(\begin{cases} a[i_1, i_2, j_1, j_2], i_1 = i_2, j_1 = j_2, \\ 0, \ \text{otherwise.} \end{cases} \right)_{i_1 i_2 j_1 j_2} \\ &\forall (i_1, i_2, j_1, j_2) \in I \times I \times J \times J. \end{aligned}$$

  with implicit constraints
  a.shape[0] = a.shape[1] and a.shape[2] = a.shape[3]

Labelling

Suppose x is a n-dim array

- A tuple $(L_1, \ldots, L_n) \in \mathbb{R}^n$ is a labelling of dimensions in x
- So we can replace index sets with labels when indexing, i.e. $x[L_1, \ldots, L_n] := x[I_1, \ldots, I_n]$, where $L_i = L_j \implies I_i = I_j$
- Extension to binary ops

$$x[L_1^x, \ldots, L_i^x] \cdot y[L_1^y, \ldots, L_j^y] := x[I_1^x, \ldots, I_i^x] \cdot y[I_1^y, \ldots, I_j^y]$$

$$\text{where } L_k^x = L_l^y \implies I(x.shape[k]) = I(y.shape[l])$$

$$\wedge[k = l \vee k = 1 \vee l = 1 : \forall(k, l) \in I(x.shape[k]) \times I(y.shape[l])]$$

Sequences and arrays

$x[I], y[J]$ are sequences of real numbers, so $x[I] \cdot y[J]$ is well-defined and is a sequence. Moreover, there is a bijection between sequences and arrays (they are isomorphic), so we can use these objects interchangeably.

Contractions

Suppose x is a n-dim array

- A contraction of x over a contraction set (of labels) C with a saving set (of labels) S is defined as

$$\sum_{C \setminus S} x[L_1, \ldots, L_2] \coloneqq \left(\sum_{i_k,\, k \in \{k | L_k \in C\} \setminus \{\max\{k | L_k \in S\}\}} x[i_1, \ldots, i_n] \right)_{(i_k,\, k \in \{k | L_k \notin C\} \cup \{\max\{k | L_k \in S\}\})}$$

$$\forall (i_1, \ldots, i_n) \in I_1 \times \cdots \times I_n$$

Einsum. Decomposition

Einsum

Given

- n arrays $x_1, \ldots, x_n$ of ranks $r_1, \ldots, r_n$
- and their corresponding labellings $(L_1^{x_1}, \ldots, L_{r_1}^{x_1}), \ldots, (L_1^{x_n}, \ldots, L_{r_n}^{x_n})$
- and the output's labelling $(L_1^o, \ldots, L_k^o)$ such that
  - $|\{L_1^o, \ldots, L_k^o\}| = k$
  - $\forall L_i^o \exists L_k^{x_j} : L_i^o = L_k^{x_j}$

einsum computes

$$o[\pi(L_1^o, \ldots, L_k^o)] = \sum_{\cup_i \{L_1^{x_i}, \ldots, L_{r_i}^{x_i}\} \setminus \{L_1^o, \ldots, L_k^o\}} \prod_{j=1}^{n} x_j[L_1^{x_j}, \ldots, L_{r_j}^{x_j}]$$

for some permutation of labels π.

Einsum is powerful

- $x \in \mathbb{R}^{n \times n}, C = \{I\}, S = \varnothing \implies \sum_{C \setminus S} x[I, I] \equiv \sum_{ii} x[i, i] = trace(x)$
- $x \in \mathbb{R}^{n \times n}, C = \{I\}, S = \{I\} \implies \sum_{C \setminus S} x[I, I] \equiv diag(x)$
- $x \in \mathbb{R}^{m \times n}, y \in \mathbb{R}^{n \times p}, C = \{N\}, S = \varnothing \implies \sum_{C \setminus S} x[M, N] \cdot y[N, P] \equiv matmul(x, y)$
- $x \in \mathbb{R}^{m}, y \in \mathbb{R}^{n}, C = \varnothing, S = \varnothing \implies \sum_{C \setminus S} x[M] \cdot y[N] = (x[i] \cdot y[j])_{ij} \equiv outer(x, y)$

Einsum. Decomposition

einsum generalization to multiple inputs is possible because

- $\sum$ and $\prod$ distribute
- $\prod / \cdot$ is a binary operation that is also associative

As such, einsum is a composition of

- "unary" contractions $\sum_{C \setminus S} x[L_1, \ldots, L_n]$
- "binary" contractions $\sum_{C \setminus S} x[L^x_1, \ldots, L^x_m] \cdot y[L^y_1, \ldots, L^y_n]$

Associativity - contraction order matters

Consider a sequence of matrix multiplications of shapes $(1, n), (n, n^2), (n^2, n^3)$. Multiplying from left to right will yield $O(n^5)$ and from right to left - $O(n^6)$. We use the opt_einsum package to generate better than naive orderings.

Einsum. Decomposition

"binary" contractions $\sum_{C\setminus S} x[L_1^x, \ldots, L_m^x] \cdot y[L_1^y, \ldots, L_n^y]$ are most interesting.

- Let $L_x = \{L_1^x, \ldots, L_m^x\}, L_y = \{L_1^y, \ldots, L_n^y\}$
- $U_x = L_x \cap C \setminus L_y$ and $U_y = L_y \cap C \setminus L_x$ - unique to x and y labels

U_x and U_y can be "unary" contracted on the spot. So let's assume $U_x = U_y = \varnothing$.

Einsum. Decomposition. Contraction with BMM

"binary" contractions $\sum_{C \setminus S} x[L_1^x, \ldots, L_m^x] \cdot y[L_1^y, \ldots, L_n^y]$ are most interesting. Output labels are either provided, or deduced, so $L_o = \{L_1^o, \ldots, L_p^o\}$ is defined.

- Let $L_b = L_o \cap L_x \cap L_y$ the sorted set of batch labels
- Let $L_C^x \subseteq L_x \cap C$ the sorted set of contraction labels in x where the order is determined by the dimension the labels appear in. Same for L_C^y
- Let $L_R^x \subseteq L_x \setminus (L_b \cup L_C^x)$ the sorted set of the remaining labels with the same order structure as L_C^x. Same for L_R^y

Einsum. Decomposition. Contraction with BMM

”binary” contractions $\sum_{C\setminus S} x[L^x_1, \ldots, L^x_m] \cdot y[L^y_1, \ldots, L^y_n]$ are most interesting

- partition/align labelled dimensions such that

$$x[L^x_1, \ldots, L^x_m] \mapsto \tilde{x}[L_b, L^x_R, L^x_C]$$
$$y[L^y_1, \ldots, L^y_n] \mapsto \tilde{x}[L_b, L^y_C, L^y_R]$$

- flatten dimensions $L^x_R, L^x_C, L^y_C, L^y_R$, do matmul, unflatten back to get an array with the labelling (L_b, L^x_R, L^y_R)
- (optionally) permute dimensions in (L_b, L^x_R, L^y_R) to match output

Einsum. Decomposition. Contraction with reductions

"binary" contractions $\sum_{C \setminus S} x[L^x_1, \dots, L^x_m] \cdot y[L^y_1, \dots, L^y_n]$ are most interesting

- partition/align labelled dimensions such that

$$x[L^x_1, \dots, L^x_m] \mapsto \tilde{x}[L_b, L^x_R, L^x_C]$$
$$y[L^y_1, \dots, L^y_n] \mapsto \tilde{x}[L_b, L^y_R, L^y_C]$$

- reshape $\tilde{x}, \tilde{y}$

$$(L_b, L^x_R, L^x_C) \mapsto (L_b, L^x_R, \underbrace{1, \dots, 1}_{|L^y_R| \text{ times}}, L^x_C)$$
$$(L_b, L^y_R, L^y_C) \mapsto (L_b, \underbrace{1, \dots, 1}_{|L^x_R| \text{ times}}, L^y_R, L^y_C)$$

- do elem-wise mul and sum-out last $|C|$ dims to get an array with the labelling (L_b, L^x_R, L^y_R)
- (optionally) permute dimensions in (L_b, L^x_R, L^y_R) to match output