Multi-dimensional arrays indexing

Vassillen Chizhov

July 2019

1 Indexing

Let us have a multidimensional array of dimensions $d_0 \times d_1 \times ...d_{D-1}$, and corresponding index sets $I_k = \{0, ..., d_k - 1\}$. Also let $F = \prod_{k=0}^{D-1} d_k$. We can construct a mapping between the set E - the Cartesian product of the index sets $I_0 \times ... \times I_{D-1}$, and I_F in the following way:

$$f(x_0, ..., x_{D-1}) = \sum_{k=0}^{D-1} x_k \prod_{j=0}^{k-1} d_j$$

Where $x_k \in I_k$ and thus $x = (x_0, ..., x_{D-1}) \in E$, and $f(x) \in I_F$. Note that f is from I_F since if we pick x so that $f(x)$ is as large as possible we get $x = (d_0 - 1, ..., d_{D-1} - 1)$, and thus:

$$f(x) = d_0 - 1 + (d_1 - 1)d_0 + ... + (d_{D-1} - 1)\frac{F}{d_{D-1}} =$$

$$-1 + d_0(1 + d_1 - 1) + (d_2 - 1)d_0 d_1 + ... + (d_{D-1} - 1)\frac{F}{d_{D-1}} =$$

$$-1 + d_0 d_1(1 + d_2 - 1) + ... + (d_{D-1} - 1)\frac{F}{d_{D-1}} =$$

$$... =$$

$$-1 + \frac{F}{d_{D-1}} + (d_{D-1} - 1)\frac{F}{d_{D-1}} = \frac{F}{d_{D-1}}(1 + d_{D-1} - 1) - 1 = F - 1$$

Which is clearly the largest index of I_F. Thus the mapping is surjective. We will prove that the mapping is also injective. Let $x \neq y \in E$, we will show that it follows that $f(x) \neq f(y)$ by induction. We know that y can be constructed from x by iteratively changing the coordinates of x that differ from the coordinates of y, and there are finitely many such (at most D) - let that count be K. This induces a sequence of vectors $x = x^{(0)}, x^{(1)}, ..., x^{(K)} = y$ each differing from y at one less coordinate than the previous vector in the sequence. Let $x^{(k)}$ and $x^{(k+1)}$ differ at index l: $x_l^{(k)} \neq x_l^{(k+1)}$. We'll show that $f(x^{(k)}) \neq f(x^{(k+1)})$. Since $x^{(k)}$ and $x^{(k+1)}$ differ only at index l by construction, if we assume $f(x^{(k)}) = f(x^{(k+1)})$, we get the equality:

$$x_l^{(k)} \prod_{j=0}^{l-1} d_j + \sum_{i=0,i\neq l}^{D-1} x_i^{(k)} \prod_{j=0}^{i-1} d_j = x_l^{(k+1)} \prod_{j=0}^{l-1} d_j + \sum_{i=0ik\neq l}^{D-1} x_i^{(k+1)} \prod_{j=0}^{i-1} d_j$$

$$x_l^{(k)} \prod_{j=0}^{l-1} d_j = x_l^{(k+1)} \prod_{j=0}^{l-1} d_j$$

$$x_l^{(k)} = x_l^{(k+1)}$$

Thus we get a contradiction, and the assumption that it is possible for $x \neq y$ and $f(x) = f(y)$ to be true is incorrect, and so $x \neq y \implies f(x) \neq f(y)$ which means that the mapping f is injective. Since f is both injective and surjective, it is bijective. Meaning that an inverse mapping exists f^{-1}. The inverse mapping derivation is straightforward.

We have already shown a statement for F which implies: $\prod_{j=0}^{D-2} d_j - 1 = \sum_{k=0}^{D-2}(d_k - 1) \prod_{j=0}^{D-3} d_j$.

So if we add 1 to the left-hand side we get a strict inequality meaning that:

$$x_{D-1} = \left\lfloor \frac{f(x)}{\prod_{j=0}^{D-2} d_j} \right\rfloor$$

Having x_{D-1}, we can compute:

$$f(x) - x_{D-1} \prod_{j=0}^{D-2} d_j - 1 = \sum_{k=0}^{D-2} x_k \prod_{j=0}^{D-3} d_j$$

Then we can apply the exact same procedure to find x_{D-2} and so on.

2 Implementation

We saw how to compute the forward and inverse mappings in the previous section. In this section we discuss some implementation details. For instance when computing the sum of products, one can optimize it for minimal number of multiplications, by using:

$$f(x_0, ..., x_{D-1}) = \sum_{k=0}^{D-1} x_k \prod_{j=0}^{k-1} d_j =$$

$$x_0 + d_0(x_1 + d_1(x_2 + ... + d_{D-3}(x_{D-2} + d_{D-2}x_{D-1})...)$$

This can easily be sequentially evaluated in code:

$$t_1 = x_{D-1}$$

$$t_i = x_{D-i} + d_{D-i}t_{i-1}$$

The last term being $t_D = f(x)$.

The inversion operation given $f(x)$ requires the subsequent division, truncation, multiplication and subtraction with the products in reverse order. This could be done by recomputing (or storing) the full product and then dividing sequentially to get lower products, this however may be inefficient. One may store the products however:

$$p_0 = d_0, p_1 = d_0 d_1, ..., p_{D-2} = \prod_{j=0}^{D-2} d_j$$

$$p_0 = d_0, p_1 = d_1 p_0, ..., p_{D-2} = d_{D-2} p_{D-3}$$

In which case the first step need not to be sequential and can be simply done through the dot product:

$$x \cdot p$$

$$p = (1, p_0, ..., p_{D-2}), x = (x_0, ..., x_{D-1})$$

This also makes the inversion easier. The inverse can be found through the following iteration:

$$t_1 = f(x)$$

$$x_{D-k} = \left\lfloor \frac{t_k}{p_{D-k}} \right\rfloor$$

$$t_{k+1} = t_k - x_{D-k} p_{D-k}$$

Which ends at $t_D = x_0$. For a non-sequential version, the following formula can be used:

$$x_k = \left\lfloor \frac{f(x)}{p_k} \right\rfloor \mod \frac{p_{k+1}}{p_k} = \left\lfloor \frac{f(x)}{p_k} \right\rfloor \mod d_k$$