Operators

Define the operator

$$\textbf{vec}_{d_1,\ldots,d_k} : \mathbb{R}^{d_1 \times \cdots \times d_k} \to \mathbb{R}^{\prod_{i=1}^{k} d_i}$$

which flattens a $d_1 \times \cdots \times d_k$ tensor into a $\left(\prod_{i=1}^{k} d_i\right)$-dimensional vector. Clearly, $\textbf{vec}_{d_1,\ldots,d_k}$ is linear and invertible. We will refer to the inverse of $\textbf{vec}_{d_1,\ldots,d_k}$ as $\textbf{ten}_{d_1,\ldots,d_k}$, which reshapes a $\left(\prod_{i=1}^{k} d_i\right)$-dimensional vector into a $d_1 \times \cdots \times d_k$ tensor.

Jacobian

Let $f : \mathbb{R}^{l \times m \times n} \to \mathbb{R}^{p \times q \times r}$ be a function of the form

$$f(\boldsymbol{x}) = \begin{bmatrix} f_{1,1,1}(\boldsymbol{x}) & \cdots & f_{1,1,r}(\boldsymbol{x}) \\ \vdots & \ddots & \vdots \\ f_{1,q,1}(\boldsymbol{x}) & \cdots & f_{1,q,r}(\boldsymbol{x}) \end{bmatrix} \quad \cdots \quad \begin{bmatrix} f_{p,1,1}(\boldsymbol{x}) & \cdots & f_{p,1,r}(\boldsymbol{x}) \\ & & \vdots \\ & & \cdots(\boldsymbol{x}) \end{bmatrix}$$

where the component functions $f_{i,j,k} : \mathbb{R}^{l \times m \times n} \to \mathbb{R}$ are scalar-valued. It follows that the difference $f(\boldsymbol{x}+\boldsymbol{\epsilon}) - f(\boldsymbol{x})$ is approximated by

$$\begin{bmatrix} \left(\frac{\partial}{\partial \boldsymbol{x}} f_{1,1,1}(\boldsymbol{x})\right)^T \textbf{vec}_{l,m,n}(\boldsymbol{\epsilon}) & \cdots & \left(\frac{\partial}{\partial \boldsymbol{x}} f_{1,1,r}(\boldsymbol{x})\right)^T \textbf{vec}_{l,m,n}(\boldsymbol{\epsilon}) \\ \vdots & \ddots & \vdots \\ \left(\frac{\partial}{\partial \boldsymbol{x}} f_{1,q,1}(\boldsymbol{x})\right)^T \textbf{vec}_{l,m,n}(\boldsymbol{\epsilon}) & \cdots & \left(\frac{\partial}{\partial \boldsymbol{x}} f_{1,q,r}(\boldsymbol{x})\right)^T \textbf{vec}_{l,m,n}(\boldsymbol{\epsilon}) \end{bmatrix} \quad \cdots \quad \begin{bmatrix} \left(\frac{\partial}{\partial \boldsymbol{x}} f_{p,1,1}(\boldsymbol{x})\right)^T \textbf{vec}_{l,m,n}(\boldsymbol{\epsilon}) & \cdots & \left(\frac{\partial}{\partial \boldsymbol{x}} f_{p,1,r}(\boldsymbol{x})\right)^T \textbf{vec}_{l,m,n}(\boldsymbol{\epsilon}) \\ & & (\boldsymbol{\epsilon}) \end{bmatrix} = \textbf{ten}_{p,q,r}\left(\left(\frac{\partial}{\partial \boldsymbol{x}} f(\boldsymbol{x})\right)^T \textbf{vec}_{l,m,n}(\boldsymbol{\epsilon})\right)$$

where $\frac{\partial}{\partial \boldsymbol{x}} f(\boldsymbol{x})$ is the Jacobian of f with respect to $\boldsymbol{x}$ — that is, the $lmn \times pqr$ matrix with columns given by the gradients $\frac{\partial}{\partial \boldsymbol{x}} f_{i,j,k}(\boldsymbol{x})$.

Gradient descent

If f is scalar-valued, then the above approximation becomes

$$\left(\frac{\partial}{\partial \boldsymbol{x}} f(\boldsymbol{x})\right)^T \textbf{vec}_{l,m,n}(\boldsymbol{\epsilon}) = \left\|\frac{\partial}{\partial \boldsymbol{x}} f(\boldsymbol{x})\right\|_2 \|\textbf{vec}_{l,m,n}(\boldsymbol{\epsilon})\|_2 \cos(\theta)$$

where θ is the angle between the gradient and $\textbf{vec}_{l,m,n}(\boldsymbol{\epsilon})$. Hence, the approximation is most negative when $\textbf{vec}_{l,m,n}(\boldsymbol{\epsilon}) = -\gamma \frac{\partial}{\partial \boldsymbol{x}} f(\boldsymbol{x})$ for some positive constant γ (i.e., when $\cos(\theta) = -1$). It follows that the direction from $\boldsymbol{x}$ in which f decreases most rapidly is the direction of $\boldsymbol{\epsilon} = -\gamma\ \textbf{ten}_{l,m,n}\left(\frac{\partial}{\partial \boldsymbol{x}} f(\boldsymbol{x})\right)$. Thus, the gradient descent update rule is given by

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \gamma\ \textbf{ten}_{l,m,n}\left(\frac{\partial}{\partial \boldsymbol{x}} f(\boldsymbol{x}_t)\right)$$

Chain rule

Consider the composition $h = f \circ g$ for functions $g : \mathbb{R}^{l \times m \times n} \to \mathbb{R}^{i \times j \times k}$ and $f : \mathbb{R}^{i \times j \times k} \to \mathbb{R}^{p \times q \times r}$. From the results above,

$$\begin{aligned}
h(\boldsymbol{x} + \boldsymbol{\epsilon}) - h(\boldsymbol{x}) &= f(g(\boldsymbol{x} + \boldsymbol{\epsilon})) - f(g(\boldsymbol{x})) \\
&= f(g(\boldsymbol{x}) + g(\boldsymbol{x} + \boldsymbol{\epsilon}) - g(\boldsymbol{x})) - f(g(\boldsymbol{x})) \\
&\approx f\left(g(\boldsymbol{x}) + \textbf{ten}_{i,j,k}\left(\left(\frac{\partial}{\partial \boldsymbol{x}} g(\boldsymbol{x})\right)^T \textbf{vec}_{l,m,n}(\boldsymbol{\epsilon})\right)\right) - f(g(\boldsymbol{x})) \\
&\approx \textbf{ten}_{p,q,r}\left(\left(\frac{\partial}{\partial g(\boldsymbol{x})} f(g(\boldsymbol{x}))\right)^T \textbf{vec}_{i,j,k}\left(\textbf{ten}_{i,j,k}\left(\left(\frac{\partial}{\partial \boldsymbol{x}} g(\boldsymbol{x})\right)^T \textbf{vec}_{l,m,n}(\boldsymbol{\epsilon})\right)\right)\right) \\
&= \textbf{ten}_{p,q,r}\left(\left(\frac{\partial}{\partial g(\boldsymbol{x})} f(g(\boldsymbol{x}))\right)^T \left(\frac{\partial}{\partial \boldsymbol{x}} g(\boldsymbol{x})\right)^T \textbf{vec}_{l,m,n}(\boldsymbol{\epsilon})\right) \\
&= \textbf{ten}_{p,q,r}\left(\left(\frac{\partial}{\partial \boldsymbol{x}} g(\boldsymbol{x}) \frac{\partial}{\partial g(\boldsymbol{x})} f(g(\boldsymbol{x}))\right)^T \textbf{vec}_{l,m,n}(\boldsymbol{\epsilon})\right)
\end{aligned}$$

Hence, the Jacobian of h is given by

$$\frac{\partial}{\partial \boldsymbol{x}} h(\boldsymbol{x}) = \frac{\partial}{\partial \boldsymbol{x}} g(\boldsymbol{x}) \frac{\partial}{\partial g(\boldsymbol{x})} f(g(\boldsymbol{x}))$$

Example

Consider the operation of cross-correlating a $C \times H \times W$ image $\boldsymbol{x}$ with a $C \times H' \times W'$ kernel $\boldsymbol{k}$. The dimension of the cross-correlation is given by

$$\left\lfloor \frac{H - H'}{s_H} + 1 \right\rfloor \times \left\lfloor \frac{W - W'}{s_W} + 1 \right\rfloor$$

where s_H and s_W denote the strides in the vertical and horizontal spatial dimensions, respectively. The $(i, j)^{\text{th}}$ element of the cross-correlation is given by

$$(\boldsymbol{x} * \boldsymbol{k})[i, j] = \sum_{c=0}^{C-1} \sum_{h=0}^{H'-1} \sum_{w=0}^{W'-1} \boldsymbol{x}[c, s_H i + h, s_W j + w] \boldsymbol{k}[c, h, w]$$

Defining the operation $\textbf{flat}_{d_1,\ldots,d_n} : \mathbb{N}^n \to \mathbb{N}$ by

$$\textbf{flat}_{d_1,\ldots,d_n}(i_1, \ldots, i_n) = \sum_{j=1}^{n} i_j \prod_{k=1}^{j-1} d_k$$

which takes an n-dimensional index of a $d_1 \times \cdots \times d_n$ tensor and returns the corresponding 1-dimensional index, it follows that rows $\textbf{flat}_{C,H,W}(c, s_H i + h, s_W j + w)$ of column $\textbf{flat}_{\left\lfloor \frac{H-H'}{s_H}+1 \right\rfloor, \left\lfloor \frac{W-W'}{s_W}+1 \right\rfloor}(i, j)$ in $\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{x} * \boldsymbol{k})$ are given by $\boldsymbol{k}[c,h,w]$ and that all other rows are zero. Hence, we have the following algorithm for computing $\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{x} * \boldsymbol{k})$.

$$
\begin{aligned}
&\text{Initialize } \frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{x} * \boldsymbol{k}) \text{ with all zeros}\\
&\textbf{for } i = 0, \ldots, \left\lfloor \frac{H-H'}{s_H} + 1 \right\rfloor - 1:\\
&\quad \textbf{for } j = 0, \ldots, \left\lfloor \frac{W-W'}{s_W} + 1 \right\rfloor - 1:\\
&\quad\quad \text{col} \leftarrow \textbf{flat}_{\left\lfloor \frac{H-H'}{s_H}+1 \right\rfloor, \left\lfloor \frac{W-W'}{s_W}+1 \right\rfloor}(i,j)\\
&\quad\quad \textbf{for } c = 0, \ldots, C-1:\\
&\quad\quad\quad \textbf{for } h = 0, \ldots, H'-1:\\
&\quad\quad\quad\quad \textbf{for } w = 0, \ldots, W'-1:\\
&\quad\quad\quad\quad\quad \text{row} \leftarrow \textbf{flat}_{C,H,W}(c, s_H i + h, s_W j + w)\\
&\quad\quad\quad\quad\quad \frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{x} * \boldsymbol{k})\,[\text{row}, \text{col}] \leftarrow \boldsymbol{k}[c,h,w]
\end{aligned}
$$

The above algorithm can be used to compute cross-correlation itself. Since cross-correlation is linear,

$$
\boldsymbol{x} * \boldsymbol{k} = ((\boldsymbol{x} + \boldsymbol{x}) * \boldsymbol{k}) - (\boldsymbol{x} * \boldsymbol{k}) = \textbf{ten}_{\left\lfloor \frac{H-H'}{s_H}+1 \right\rfloor, \left\lfloor \frac{W-W'}{s_W}+1 \right\rfloor} \left(\left(\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{x} * \boldsymbol{k}) \right)^T \textbf{vec}_{C,H,W}(\boldsymbol{x}) \right)
$$

Note that in the above computations, we consider elements of a tensor in channel, row, column order. For example, the operator **flat** returns a 1-dimensional index by counting elements of a tensor front to back (channel), top to bottom (row), left to right (column). The order doesn't matter as long as one is consistent.