CSE 5523: Lecture Notes 15
Convolutional Neural Networks

Contents

Large networks often have a lot of parameters that do similar things, so can be tied (re-used).

One way to do this is by re-using whole blocks of neural units across a larger grid.

15.1 Convolution

The idea of re-using blocks of units at different places in a system comes from signal processing.

Often responses to a **signal** f are defined by a **filter** function g that adds up when impulses repeat:

$$(f * g)(i) = \int_{-\infty}^{\infty} f(j)\, g(i - j)\, dj$$

(It's subtracted because the filter function tapers to the left so the response tapers to the right.)

This is called **convolution**.

The same principle can apply to discrete vectors $f(...) \in \mathbb{R}^J, g(...) \in \mathbb{R}^{J-I}$ as signals and filters:

$$(f(...) * g(...))_{[i]} = \sum_{j=i}^{i+J-I} f(...)_{[j]}\, g(...)_{[1+j-i]}$$

Note that $i-j$ is non-positive, so we invert the filter and use $1+j-i$.

For example if $I = 4$ and $J = 6$:

$$\begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} * \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \cdot 2 + 1 \cdot 3 + 1 \cdot 1 = 4 \\ 1 \cdot 2 + 1 \cdot 3 + 0 \cdot 1 = 5 \\ 1 \cdot 2 + 0 \cdot 3 + 0 \cdot 1 = 2 \\ 0 \cdot 2 + 0 \cdot 3 + 0 \cdot 1 = 0 \end{bmatrix}$$

15.2 Jacobians for signals

We can define Jacobians for backprop into signals (z is a weight downstream from $f(...)$):

$$\begin{aligned}
\frac{\partial (f(...) * g(...))_{[i]}}{\partial z} &= \frac{\partial}{\partial z} \sum_{j=i}^{i+J-I} g(...)_{[1+j-i]}\, f(...)_{[j]} && \text{definition of convolution} \\
&= \sum_{j=i}^{i+J-I} \frac{\partial}{\partial z} g(...)_{[1+j-i]}\, f(...)_{[j]} && \text{sum rule} \\
&= \sum_{j=i}^{i+J-I} g(...)_{[1+j-i]} \frac{\partial}{\partial z} f(...)_{[j]} && \text{product rule} \\
&= \left(\sum_{j=1}^{J} \delta_j^\top \begin{cases} g(...)_{[1+j-i]} & \text{if } 0 \le j-i \le J-I \\ 0 & \text{otherwise} \end{cases} \right) \frac{\partial}{\partial z} f(...) && \text{def. of inner product}
\end{aligned}$$

So $\left(\sum_{i=1}^{I} \sum_{j=1}^{J} \delta_i \delta_j^\top \begin{cases} g(...)_{[1+j-i]} & \text{if } 0 \le j-i \le J-I \\ 0 & \text{otherwise} \end{cases} \right) = \frac{\partial (f(...) * g(...))}{\partial f(...)}$ is a Jacobian.

For example if $I = 4$ and $J = 6$:

$$\frac{\partial (f(...) * g(...))}{\partial f(...)} = \begin{bmatrix} g(...)_{[1]} & g(...)_{[2]} & g(...)_{[3]} & 0 & 0 & 0 \\ 0 & g(...)_{[1]} & g(...)_{[2]} & g(...)_{[3]} & 0 & 0 \\ 0 & 0 & g(...)_{[1]} & g(...)_{[2]} & g(...)_{[3]} & 0 \\ 0 & 0 & 0 & g(...)_{[1]} & g(...)_{[2]} & g(...)_{[3]} \end{bmatrix}$$

15.3 Jacobians for filters

We can also define Jacobians for backprop into filters (z is a weight downstream from $g(...)$):

$$\begin{aligned}
\frac{\partial (f(...) * g(...))_{[i]}}{\partial z} &= \frac{\partial}{\partial z} \sum_{j=i}^{i+J-I} g(...)_{[1+j-i]}\, f(...)_{[j]} && \text{definition of convolution} \\
&= \frac{\partial}{\partial z} \sum_{k=1}^{1+J-I} g(...)_{[k]}\, f(...)_{[k+i-1]} && \text{change of variable } k = 1+j-i \\
&= \sum_{k=1}^{1+J-I} \frac{\partial}{\partial z} g(...)_{[k]}\, f(...)_{[k+i-1]} && \text{sum rule} \\
&= \sum_{k=1}^{1+J-I} f(...)_{[k+i-1]} \frac{\partial}{\partial z} g(...)_{[k]} && \text{product rule} \\
&= \left(\sum_{k=1}^{1+J-I} \delta_k^\top \begin{cases} f(...)_{[k+i-1]} & \text{if } 1 \le k+i-1 \le J \\ 0 & \text{otherwise} \end{cases} \right) \frac{\partial}{\partial z} g(...) && \text{def. of inner product}
\end{aligned}$$

So $\left(\sum_{i=1}^{I} \sum_{k=1}^{1+J-I} \delta_i \, \delta_k^\top \begin{cases} f(\ldots)_{[k+i-1]} & \text{if } 1 \le k+i-1 \le J \\ 0 & \text{otherwise} \end{cases} \right) = \frac{\partial (f(\ldots) * g(\ldots))}{\partial g(\ldots)}$ is a Jacobian.

For example if $I = 4$ and $J = 6$:

$$\frac{\partial (f(\ldots) * g(\ldots))}{\partial g(\ldots)} = \begin{bmatrix} f(\ldots)_{[1]} & f(\ldots)_{[2]} & f(\ldots)_{[3]} \\ f(\ldots)_{[2]} & f(\ldots)_{[3]} & f(\ldots)_{[4]} \\ f(\ldots)_{[3]} & f(\ldots)_{[4]} & f(\ldots)_{[5]} \\ f(\ldots)_{[4]} & f(\ldots)_{[5]} & f(\ldots)_{[6]} \end{bmatrix}$$

With these Jacobians we can backprop error to either operand of a convolution.

15.4 Multiple dimensions

Data for images and other multi-dimensional data can be flattened with modified convolution.

For example to convolve a 2×2 pattern around a 3×3 image (so, with $I = 4$ and $J = 9$:

$$\frac{\partial (f(\ldots) * \mathbf{W})}{\partial f(\ldots)} = \begin{bmatrix} \mathbf{W}_{[1,1]} & \mathbf{W}_{[1,2]} & 0 & \mathbf{W}_{[2,1]} & \mathbf{W}_{[2,2]} & 0 & 0 & 0 & 0 \\ 0 & \mathbf{W}_{[1,1]} & \mathbf{W}_{[1,2]} & 0 & \mathbf{W}_{[2,1]} & \mathbf{W}_{[2,2]} & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{W}_{[1,1]} & \mathbf{W}_{[1,2]} & 0 & \mathbf{W}_{[2,1]} & \mathbf{W}_{[2,2]} & 0 \\ 0 & 0 & 0 & 0 & \mathbf{W}_{[1,1]} & \mathbf{W}_{[1,2]} & 0 & \mathbf{W}_{[2,1]} & \mathbf{W}_{[2,2]} \end{bmatrix}$$