Convolutions

A concise overview over the convolutional forward and backward propagation process used in convolutional neural networks.

I. SPECIFICATIONS

We have an unbatched input X of shape S_X, where

$$X = \begin{bmatrix} x_{1,1} & x_{2,1} & x_{3,1} \\ x_{1,2} & x_{2,2} & x_{3,2} \\ x_{1,3} & x_{2,3} & x_{3,3} \end{bmatrix}$$

and $S_X = (3,3)$, describing the input X's width X_W and height X_H.

We additionally have a convolutional kernel K of shape S_K, where K is made up of a matrix of its parameters

$$K = \begin{bmatrix} k_{1,1} & k_{2,1} \\ k_{1,2} & k_{2,2} \end{bmatrix}$$

and $S_K = (2,2)$, describing the kernel K's width K_W and height K_H.

We will perform a "classic" convolutional forward operation using our input X and the convolutional kernel K. For simplicityWe assume no zero-padding and a stride of 1. The output dimension size O_S from a convolutional forward can be computed using

$$O_S = \frac{X_S - K_S + 2P}{S} + 1 \tag{1}$$

where X_S is the size of the input dimension, K_S is the size of the kernel dimension, P is the zero-padding size and S is the stride. For our input and kernel dimensions S_X and S_K, we compute the dimensions for our convolutional output Z:

$$Z_W = Z_H = \frac{3 - 2 + 2 * 0}{1} + 1 = 2 \tag{2}$$

Equation 2 describes both the output width and height for Z, since both our input X and our kernel K are square. Thus, we have determined the shape of Z: $S_Z = (2,2)$.

We will then apply a sigmoid activation function to Z to compute the final output O of our forward process. The sigmoid activation function is described by

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

σ is an element-wise operation applied to Z, and produces an equal shaped output O. Thus, $S_Z = S_O$.

Finally, we have a target output matrix Y of shape $S_Y = S_O$. O and Y are used to compute the loss L, where

$$L = \sum_{i=1}^{O_W} \sum_{j=1}^{O_H} (o_{i,j} - y_{i,j})^2 \tag{4}$$

The goal of this exercise is to show how we can minimize the loss L by tuning our kernel parameters in K. This is achieved by performing a forward propagation through the convolutional operation, applying the sigmoid activation function to the convolutional operation's outputs and finally computing the loss. Then, we perform a backward propagation, computing the flow of gradients from the loss all the way back to the convolutional kernel, finding the gradient for K with respect to the output loss L. This allows us to intelligently update our kernel parameters to decrease the loss L for our next forward propagation.

II. FORWARD PROPAGATION

The first step of our forward propagation is to compute $Z = conv(X, K)$. The forward convolutional operation with our input X of shape $S_X = (3,3)$, kernel K of shape $S_K = (2,2)$, output Z of shape $S_Z = (2,2)$, stride $S = 1$ and no zero padding can be described as

$$z_{x,y} = \frac{\sum_{i=1}^{K_W} \sum_{j=1}^{K_H} x_{i+x,j+y} k_{i,j}}{S_W S_H} \tag{5}$$

Using equation 5, we can see that

$$\begin{aligned}
z_{1,1} &= \frac{x_{1,1}k_{1,1} + x_{2,1}k_{2,1} + x_{1,2}k_{1,2} + x_{2,2}k_{2,2}}{4} \\
z_{2,1} &= \frac{x_{2,1}k_{1,1} + x_{3,1}k_{2,1} + x_{2,2}k_{1,2} + x_{3,2}k_{2,2}}{4} \\
z_{1,2} &= \frac{x_{1,2}k_{1,1} + x_{2,2}k_{2,1} + x_{1,3}k_{1,2} + x_{2,3}k_{2,2}}{4} \\
z_{2,2} &= \frac{x_{2,2}k_{1,1} + x_{3,2}k_{2,1} + x_{2,3}k_{1,2} + x_{3,3}k_{2,2}}{4}
\end{aligned}$$

We then compute $O = \sigma(Z)$ by applying the sigmoid activatio function, σ, element-wise to Z.

$$o_{i,j} = \sigma(z_{i,j}) = \frac{1}{1 + e^{-z_{i,j}}} \tag{6}$$

Using equation 6, we get

$$\begin{aligned}
o_{1,1} &= \frac{1}{1 + e^{-z_{1,1}}} \\
o_{2,1} &= \frac{1}{1 + e^{-z_{2,1}}} \\
o_{1,2} &= \frac{1}{1 + e^{-z_{1,2}}} \\
o_{2,2} &= \frac{1}{1 + e^{-z_{2,2}}}
\end{aligned}$$

Finally, we complete our forward propagation by computing the loss L. By using equation 4, we get

$$L = \sum_{i=1}^{O_W}\sum_{j=1}^{O_H}(o_{i,j} - y_{i,j})^2$$
$$= (o_{1,1} - y_{1,1})^2 + (o_{2,1} - y_{2,1})^2$$
$$+(o_{1,2} - y_{1,2})^2 + (o_{2,2} - y_{2,2})^2$$

III. BACKWARD PROPAGATION

In order to update and tune the parameters of K, we must find their partial gradients with respect to the computed loss L. In effect, this is achieved by computing the derivatives for every tunable parameter $k_{i,j}$ with respect to L.

To understand how we can backwards propagate from the loss L back to the kernel K, we must recall how the loss was computed using the kernel in the first place. Recall that, in slightly simplified terms

$$L = (O - Y)^2$$

where $O = \sigma(Z)$ and $Z = conv(X, K)$.

From this, we can then see that

$$\frac{\partial L}{\partial K} = \frac{\partial L}{\partial O}\frac{\partial O}{\partial Z}\frac{\partial Z}{\partial K}$$

Then, by looking at $k_{1,1}$, we can see that

$$\frac{\partial L}{\partial k_{1,1}} =$$
$$\frac{\partial L}{\partial o_{1,1}}\frac{\partial o_{1,1}}{\partial z_{1,1}}\frac{\partial z_{1,1}}{\partial k_{1,1}}$$
$$+\frac{\partial L}{\partial o_{1,2}}\frac{\partial o_{1,2}}{\partial z_{1,2}}\frac{\partial z_{1,2}}{\partial k_{1,1}}$$
$$+\frac{\partial L}{\partial o_{2,1}}\frac{\partial o_{2,1}}{\partial z_{2,1}}\frac{\partial z_{2,1}}{\partial k_{1,1}}$$
$$+\frac{\partial L}{\partial o_{2,2}}\frac{\partial o_{2,2}}{\partial z_{2,2}}\frac{\partial z_{2,2}}{\partial k_{1,1}}$$

where

$$\frac{\partial L}{\partial o_{1,1}} = 2(o_{1,1} - y_{1,1})$$

$$\frac{\partial o_{1,1}}{\partial z_{1,1}} = \frac{1}{1+e^{-z1,1}}(1 - \frac{1}{1+e^{-z1,1}})$$

$$\frac{\partial z_{1,1}}{\partial k_{1,1}} = x_{1,1}$$