Exercise 08: Conv Nets

1163150 - Ausgewählte Kapitel sozialer Webtechnologien

7. Dezember 2018

Definition Given an image I with shape $i \times j$, a kernel K with shape $k_1 \times k_2$ and a stride s with $s \in \mathbb{Z}$ we define the multidimensional discrete convolution $*$ as:

$$(I * K)_{ij} = \sum_{m=1}^{k_1} \sum_{n=1}^{k_2} K_{m,n} \cdot I_{s(i-1)+m, s(j-1)+n} \tag{1}$$

Definition The gradient with respect to a Kernel K is:

$$\frac{\partial (I * K)}{\partial K_{m',n'}} = \sum_{i=1}^{c_1} \sum_{j=1}^{c_2} I_{s(i-1)+m', s(j-1)+n'} \tag{2}$$

Note Remember that the shape of an output O of a convolutional layer can be calculated by:

- $c_1 = (i_1 + 2 * p - k_1)/s + 1$
- $c_2 = (i_2 + 2 * p - k_2)/s + 1$
- Output depth equals the number of kernels used in the convolution

1. **Convolution as a sum**

   For the exercises, assume the following parameter: stride $s = 2$, padding $p = 0$

   (a) You feed an image I with the dimensions 200x200x3 in the convolutional layer, which consists of 12323 kernels K_i each with the size 40x40x3. What dimesions of the output O do you expect?

   (b) Calculate $I * K$ with

$$I = \begin{bmatrix} 1 & 1 & -2 & 0 & 1 \\ 1 & 0 & 0 & 2 & 1 \\ 0 & 1 & 0 & 5 & -1 \\ -2 & 1 & 0 & -1 & 1 \\ 0 & 1 & 0 & 5 & -1 \end{bmatrix}, K = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

   (c) Calculate the gradient $\frac{\partial (I*K)}{\partial K}$

2. **Covolution as a matrix multiplication**

   If you do a naive implementation of the convolutional operation, you have nested loops. That is a bad idea, because it is always computationla expensive. A good solution to that problem is to vectorize the operation via matrix multiplication.

   (a) How can you optimize the calculation of the convolutional operation to be more effienct? *(Hint: Look up 'Convolution as matrix multiplication' with a search engine of your choice.)*

   (b) Calculate $I * K$ again like in (1.b) but with the method described.

   (c) Calculate the gradient $\frac{\partial(L)}{\partial K}$ using a matrix multiplication, where:

$$\frac{\partial(L)}{\partial K} = \frac{\partial(L)}{\partial(I * K)} * \frac{\partial(I * K)}{\partial(K)}$$

   Assume that the convolutional operation described is the last or only node of your computational graph. Note that the derivative of the matrix multiplication is:

$$\frac{\partial(I \cdot \vec{k})}{\partial \vec{k}} = I^T$$

   (d) Calculate the max-pooling operation on image I with kernel-size of $(3x3)$, stride $s = 2$ padding $p = 0$. *(Hint: Pooling is the same as a convolution, but with a different kernel-function.)*

3. **Some questions to gain a better understanding of the whole topic**

   (a) In the equations (1) and (2) we did not include a bias b. How would they change if you decide to a bias b to your convolutional layer?

   (b) We always looked at 2d-examples. How does convolution look like if you write down a 3d-example (as a sum and matrix multiplication)? Create a toy example of your choice, e.g., an image with 5x5x2 and a kernel with 3x3x2 and calculate the convolution.

   (c) What is the difference between cross-correlation $\star$ and convolution $*$.

Note Some additional informations at the end. For a modular implementation of a convolutional neural network, we need also the gradient with **respect to the Image I**. It can be expressed as:

Convolution as a sum

$$\frac{\partial(I * K)}{\partial I_{m',n'}} = \sum_{i=1} \sum_{j=1} K_{m'-s(i-1),n'-s(j-1)}$$

Convolution as matrix multiplication

$$\frac{\partial(I \cdot \vec{k})}{\partial I} = k^T$$