Assignment 9

Welcome to the ninth assignment of the lecture *2D Vision and Deep Learning*. **Please read all instructions carefully!** This assignment covers the concept of back-propagation in deep (convolutional) neural networks.

Submission is due on Tuesday, January 25th, 2020 at 2pm. Please note that late assignments will receive zero (0) marks, so you are strongly encouraged to start the assignment early. If you have any questions please contact Tim Heußler (`theussle@students.uni-mainz.de`).

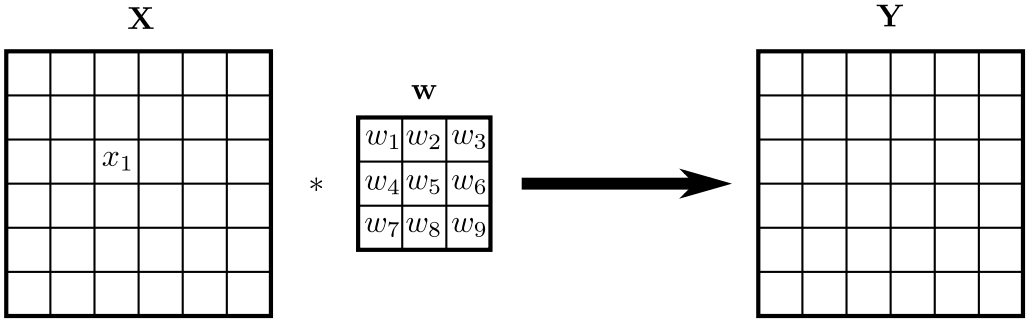
Exercise 1 (4 points)**.** Suppose you convert a function from a Cartesian coordinates to a polar coordinates, i.e. $f(x, y)$ is converted to $f(r, \theta)$. Prove that

$$\left(\frac{\partial f}{\partial r}\right)^2 + \frac{1}{r^2}\left(\frac{\partial f}{\partial \theta}\right)^2 = \left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2$$

Hint: Use the fact that $x = r \cdot \cos\theta, y = r \cdot \sin\theta$ and the chain rule.

Exercise 2 (6 points)**.** In a *pooling layer* there are no parameters, so for back-propagation only $\frac{\partial Loss}{\partial \mathbf{X}}$ has to be computed. In a *max pooling layer* only the maximum value in the local window survives in the forward pass. Thus, in the packward pass only these values should receive a gradient from the upper layer.
Describe a strategy to compute $\frac{\partial Loss}{\partial \mathbf{X}}$ for a *max pooling layer* with filter size 2×2 and stride 2.

Exercise 3 (6 points)**.** Given a single channel feature map $\mathbf{X}$, a 3×3 convolutional filter $\mathbf{w}$, and the output feature map $\mathbf{Y} = \mathbf{X} * \mathbf{w}$ (with padding). Now, let's see for a single neuron x_1 in $\mathbf{X}$ how back-propagation works for a convolution layer.



1. (1 point) There are nine neurons in $\mathbf{Y}$, namely $y_1, y_x, \ldots, y_9$ that are affected by x_1 during the convolution process. Label these neurons in the figure above.

2. (1 points) The activation value for each of these nine neurons can be written as $y_i = w_j \cdot x_1 + C_i$, where C_i are some terms that does not involve x_1. Write down such expression for each of these nine neurons.

3. (1 point) Suppose we use a vector $\mathbf{v} = [y_1, y_2, \ldots, y_9]^\top$ to represent the effected neurons. What is $\frac{\partial \mathbf{v}}{\partial x_1}$?

4. (1 point) Show that back-propagating the gradient through this convolution layer w.r.t $\mathbf{X}$ is equivalent to convolve $\frac{\partial Loss}{\partial \mathbf{Y}}$ with the matched filter $\mathbf{w}'$ of $\mathbf{w}$ which is the original filter being flipped both horizontal and vertically, i.e.

$$\mathbf{w}' = \begin{bmatrix} w_9 & w_8 & w_7 \\ w_6 & w_5 & w_4 \\ w_3 & w_2 & w_1 \end{bmatrix}.$$

5. (2 points) How about back-propagating the gradient through this convolutional layer w.r.t. $\mathbf{w}$?

Exercise 4 (6 points)**.** Compare different kind of optimizer provided by *PyTorch* to train the network you built in Assignment 8, Exercise 5. Use at least the optimizer

- Adadelta
- Adagrad
- Adam
- RSMprop
- SGD
- LBFGS

and describe their behavior and the results achieved.