Artificial Neural Networks (Gerstner). Exercises for week 9

Convolutional neural networks

Exercise 1. Backprop in ConvNets

Consider a very simple convolutional neural network with 3 dimensional input (e.g. RGB image), one convolution layer with 5x5x3 filters, stride 1, non-linearity σ and one linear layer, i.e.

$$a_{ijk} = b_k + \sum_{x=1}^{5}\sum_{y=1}^{5}\sum_{c=1}^{3} I_{i+x-1,j+y-1,c} w^{(1)}_{xyck} \tag{1}$$

$$x^{(1)}_{ijk} = \sigma(a_{ijk}) \tag{2}$$

$$\hat{y}_o = \sum_{ijk} w^{(2)}_{ijko} x^{(1)}_{ijk} \tag{3}$$

a. With loss $L = \frac{1}{2}\sum_o (t_o - \hat{y}_o)^2$, compute $\partial L / \partial w_{1111}$.

b. How does the result for the convolutional network change, if you introduce a max pooling layer after the convolutional layer?

c. Compare your result to the one you obtain with a dense (fully-connected) layer, i.e. $a_k = b_k + \sum_x \sum_y \sum_c I_{xyc} w^{(1)}_{xyck}$, $x^{(1)}_k = \sigma(a_k)$ and $\hat{y}_o = \sum_k w^{(2)}_{ok} x^{(1)}_k$.

Exercise 2. Computing volume sizes

Given a volume of width n, height n and depth c.

a. Show that the convolution with k filters of size $f \times f \times c$ with stride s and padding p leads to a new volume of size

$$\left(\frac{n+2p-f}{s}+1\right) \times \left(\frac{n+2p-f}{s}+1\right) \times k\,. \tag{4}$$

b. What padding p do you have to choose such that the input and output volumes have the same width and depth for stride $s = 1$. Check you result for the special case of $n = 4$ and $f = 3$.

Exercise 3. Test of translation invariance

We start with a 1-dimensional, simple case. Suppose we apply a 1-dimensional convolutional processing stage with one filter to our 1-dimensional image and obtain a filter bank x of size 21x1. The values of these filter outputs are denoted by $x(i)$ with $1 \le i \le 21$.

a. Suppose that the global maximum value of $x(i)$ is at $i = 11$. We apply max-pooling with stride 1 and a pooling window size of 11. How stable is the result of the pooling with respect to shifts in x that could stem from shifts in the original input image?

b. Suppose that the values $x(i)$ are monotonically increasing with a maximum at $i = 21$. We apply the same max-pooling (stride 1, window size 11). How stable is the result of the pooling in this case?

Now we move to a 2-dimensional case. On the right there is an image we would like to process with a 2-layer convolutional network. The network has one convolutional layer (two 3x3 filters, depicted below the image, stride 1, ReLU non-linearity) and one max-pooling layer (pool window 2x2, stride 2). The image is already padded (1x1 zero-padding) and no further padding is applied later.

0	0	0	0	0	**0**
0	1	2	0	3	0
0	1	2	0	1	0
0	4	1	1	2	0
0	0	0	2	1	0
0	0	0	0	0	**0**

image

0	1	0		0	1	1
1	0	1		0	1	0
0	1	0		1	1	0

filter 1 filter 2

c. Determine the width, height and depth of the volume after max-pooling.

d. Compute the output of the max-pooling layer. Use all biases = 0.

e. Move the image one pixel to the left (padding column with zeros). How does the output of the convolution layer change? How many output units of the max-pooling layer have changed?

f. Is it possible to find a translation of the original image under which the value for each output unit of the max-pooling layer is invariant?

Exercise 4. Reverse-mode automatic differentiation

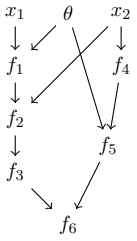
The backprop algorithm derived above is a special case of an algorithm called *Reverse-mode automatic differentation*. Modern deep learning software packages rely on this algorithm, as it allows flexible exploration of different architectures and cost functions without having to adapt the standard BackProp algorithm for each special case. To understand reverse-mode automatic differentation we look at the function

$$f(x_1, x_2, \theta) = \sin(\theta x_1 + x_2) + \theta x_2^2 \,. \tag{5}$$

Using the simple functions $f_1(x, y) = xy$, $f_2(x, y) = x + y$, $f_3(x) = \sin(x)$, $f_4(x) = x^2$, $f_5(x, y) = xy$ and $f_6(x, y) = x + y$, we can write the above function as

$$f(x_1, x_2, \theta) = f_6(f_3(f_2(f_1(x_1, \theta), x_2)), f_5(\theta, f_4(x_2))) \,, \tag{6}$$

that has the Abstract Syntax Tree (AST) or Computation Graph depicted on the right.



The reverse-mode Automatic Differentation algorithm proceeds now as follows:

AutoDiff

1. Determine all children of the variable(s) of interest. In the example, using θ, this includes f_1, f_2, f_3, f_5, f_6.
2. Find a reverse ancestral (backwards) schedule of nodes. All of the children of a node should be scheduled before the node itself. In the previous example with θ, the schedule could be $f_6, f_3, f_2, f_1, f_5, \theta$. For the full graph, this could be $f_6, f_3, f_2, f_1, x_1, f_5, \theta, f_4, x_2$.
3. Start with the first node n_1 in the reverse schedule and define $t_{n_1} = 1$, e.g. $t_{f_6} = 1$.
4. For the next node n in the reverse schedule, find the child nodes $\mathrm{ch}(n)$. Then define
$$t_n = \sum_{c \in \mathrm{ch}(n)} \frac{\partial f_c}{\partial f_n} t_c \,. \tag{7}$$
5. The total derivative of f with respect to node n is given by t_n.

a. Show that $t_\theta = \partial f / \partial \theta$ by following the steps of the algorithm, i.e. by computing $t_6, t_3, \ldots$, and comparing it to the result you get by differentiating Eq. 5 manually.

b. Draw the AST of a fully connected network with 2 hidden layers and a single output, and convince yourself that the backpropagation algorithm is a special case of reverse-mode automatic differentation.

c. Draw the AST of a network with one 1D convolutional layer (with filter length 3 and stride 1), followed by one max-pooling layer (with $k = 2$), one dense layer, and a single output. For simplicity, assume only 1 filter is used in the convolutional layer.