Ludwig-Maximilians-Universitaet Muenchen
Institute for Informatics
Prof. Dr. Volker Tresp
Rajat Koner
Andreas Lohrer
Sebastian Schmoll

14.05.2021

Machine Learning

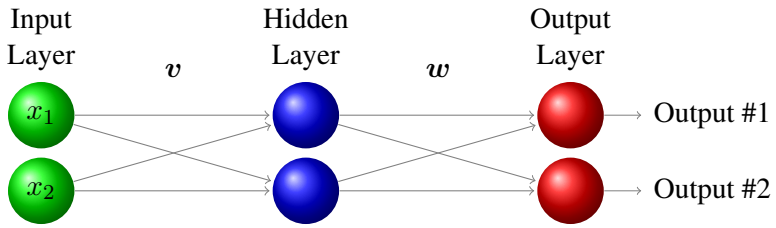
Summer 2021

Exercise Sheet 4

Exercise 4-1 Backpropagation

In order to classify a two-dimensional input $\mathbf{x} = (x_1, x_2)^T$ into two classes, we will use a 2-layer feed forward network, i.e, one hidden layer and one output layer with two units on each layer. As activation function, we use the sigmoid function in both layers, $\sigma(arg) = (1 + e^{-arg})^{-1}$. We will use the sum-of-squares error function, i.e.

$$\mathcal{L} = \frac{1}{2}\sum_{k=1}^{N}(\hat{y}_k - y_k)^2$$



The architecture is shown without biases. In the following use also biases for the input ($x_0 = 1$), resp., for the hidden layer:

(a) Derive the update rule for the weights $\boldsymbol{w}$ connecting the neurons of the hidden layer with the neurons of the output layer.

(b) Derive the update rule for the weights $\boldsymbol{v}$ connecting the neurons of the input layer with the neurons of the hidden layer.

Possible Solution

Let's σ_h, resp, σ_o denote the activation function of the hidden layer, resp., of the output layer. Let h_j^h be the weighted input signal to the j-th neuron in the hidden layer, i.e., $h_j^h = \sum_{i=0}^{2} v_{ij}x_i$ and h_k^o be the weighted input signal to the k-th neuron of the output layer, i.e., $h_k^o = \sum_{j=0}^{2} w_{jk}a_j^h$, where a_j^h denotes the output of the j-th neuron of the hidden layer ($a_j^h = \sigma_h(h_j^h)$)

(a) Using the chain rule,we get:

$$\frac{\partial \mathcal{L}}{\partial w_{jk}} = \frac{\partial \mathcal{L}}{\partial \hat{y}_k}\frac{\partial \hat{y}_k}{\partial h_k^o}\frac{\partial h_k^o}{\partial w_{jk}} \tag{1}$$

The last term resolves to:

$$\frac{\partial h_k^o}{\partial w_{jk}} = \frac{\partial}{\partial w_{jk}} \left[\sum_{j=0}^{2} w_{jk} a_j^h \right] = a_j^h \tag{2}$$

The first term resolves to:

$$\frac{\partial \mathcal{L}}{\partial \hat{y}_k} = \frac{\partial}{\partial \hat{y}_k} \left[\frac{1}{2} \sum_{k=1}^{N} (\hat{y}_k - y_k)^2 \right] = \hat{y}_k - y_k \tag{3}$$

The term in the middle is the derivative of the activation function of the output layer (here:sigmoid fnc; c.f. last exercise):

$$\frac{\partial \hat{y}_k}{\partial h_k^o} = \sigma_o(h_k^o)(1 - \sigma_o(h_k^o)) = \hat{y}_k(1 - \hat{y}_k) \tag{4}$$

Therefore, our update rule for the weights $\boldsymbol{w}$ is given by:

$$w_{jk} = w_{jk} - \eta \frac{\partial \mathcal{L}}{\partial w_{jk}} = w_{jk} - \eta(\hat{y}_k - y_k)\hat{y}_k(1 - \hat{y}_k)a_j^h = w_{jk} - \eta \delta_k^o a_j^h \tag{5}$$

where $(\hat{y}_k - y_k)\hat{y}_k(1 - \hat{y}_k)$ is substituted by δ_k^o.

(b) Propagating the error backwards, we notice that the value of $\frac{\partial \mathcal{L}}{\partial v_{ij}}$ is influenced by all N output nodes. Therefore, we need to consider all of the output nodes when updating the weight of v_{ij}.

Using the chain rule, we get:

$$\frac{\partial \mathcal{L}}{\partial v_{ij}} = \sum_{k=1}^{N} \frac{\partial \mathcal{L}}{\partial \hat{y}_k} \frac{\partial \hat{y}_k}{\partial h_k^o} \frac{\partial h_k^o}{\partial a_j^h} \frac{\partial a_j^h}{\partial h_j^h} \frac{\partial h_j^h}{\partial v_{ij}} \tag{6}$$

The last term evaluates to:

$$\frac{\partial h_j^h}{\partial v_{ij}} = \frac{\partial}{\partial v_{ij}} \left[\sum_{i=0}^{2} v_{ij} x_i \right] = x_i \tag{7}$$

The fourth term is the derivative of the activation function being used for the hidden layer (here: sigmoid fnc, c.f. last exercise):

$$\frac{\partial a_j^h}{\partial h_j^h} = \sigma_o(h_j^h)(1 - \sigma_o(h_j^h)) \tag{8}$$

The term in the middle evaluates to:

$$\frac{\partial h_k^o}{\partial a_j^h} = \frac{\partial}{\partial a_j^h} \sum_{j=0}^{2} w_{jk} a_j^h = w_{jk} \tag{9}$$

Putting all together, we have ($\delta_k^o = \frac{\partial \mathcal{L}}{\partial \hat{y}_k} \frac{\partial \hat{y}_k}{\partial h_k^o}$):

$$\sum_{k=1}^{N} \delta_k^o w_{jk} \sigma_o(h_j^h)(1 - \sigma_o(h_j^h)) x_i = \sigma_o(h_j^h)(1 - \sigma_o(h_j^h))(\sum_{k=1}^{N} \delta_k^o w_{jk}) x_i \tag{10}$$

Therefore, our update rule for the weights $\boldsymbol{v}$ is given by:

$$v_{ij} = v_{ij} - \eta \frac{\partial \mathcal{L}}{\partial v_{ij}} = v_{ij} - \eta \sigma_o(h_j^h)(1 - \sigma_o(h_j^h))(\sum_{k=1}^{N} \delta_k^o w_{jk}) x_i = v_{ij} - \eta \delta_j^h x_i \tag{11}$$

where $\sigma_o(h_j^h)(1 - \sigma_o(h_j^h))(\sum_{k=1}^{N} \delta_k^o w_{jk})$ is substituted by δ_j^h.

Exercise 4-2 Convolutions

Given the following 5x5 input image with one channel:

5	5	2	5	5
5	5	2	5	5
7	7	5	7	7
5	5	2	5	5
5	5	2	5	5

Let's assume we have the following 3x3 filters:

1	0	-1
2	0	-2
1	0	-1

1	2	1
0	0	0
-1	-2	-1

0	1	0
1	-4	1
0	1	0

After the filter size has been chosen (here: 3×3), we also have to define a *stride* and a *padding*. The former one controls the amount of shifting the filter at a time, i.e., how the filter convolves around the input volume. The latter one controls the spatial dimension of the output, e.g., a zero padding of size k indicates that the input volume is padded with k zeros around the border of the image such that the input volume can be preserved. The most prominent options for padding are called 'same' and 'valid': 'same' results in a padding of the input such that the output has the same volume as the original input, whereas the option 'valid' means that there is no padding at all.

(a) Apply the given filters (by cross-corellation) to the above dataset. The formula is given by:

$$S_{i,j} = (K \star X)_{i,j} = \sum_{m=0}^{2} \sum_{n=0}^{2} K_{m,n} X_{i+m,j+n}.$$

where S is the output, K denotes the kernel with dimensions $m \times n$ and X indicates the input of dimensions $i \times j$.

For this exercise use 'valid' padding and a stride of one.

Possible Solution

Solution: The output values are calculated as: $S_{i,j} = (K \star X)_{i,j} = \sum_{m=0}^{2} \sum_{n=0}^{2} K_{m,n} X_{i+m,j+n}$.

Channel 1:

11	0	-11
10	0	-10
11	0	-11

Example: $S_{0,0} = 1 \cdot 5 + 0 \cdot 5 + (-1) \cdot 2 + 2 \cdot 5 + 0 \cdot 5 + (-2) \cdot 2 + 1 \cdot 7 + 0 \cdot 7 + (-1) \cdot 5 = 11.$

Note that using convolution instead of cross-correlation would lead to inversed signs in the output (flip the kernel).

Channel 2:

-9	-10	-9
0	0	0
9	10	9

Channel 3:

-1	9	-1
-6	-2	-6
-1	9	-1

(b) Look at the structure of the filters. What do they do?

Possible Solution

Filter 1 and 2 are Sobel operators. They detect vertical (1) and horizontal (2) edges by computing an approximation of the gradient of the image intensity. Note that they can be decomposed as follows:

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} +1 & 0 & -1 \end{bmatrix}$$

where the first term is a smoothing term (the middle row of the convolved input section (3 x 3 part) counts twice as much as the outer ones) and the second term corresponds to the approximate derivative (between the left and the right column of the input).

The two Sobel operators can be combined to get the magnitude and angle of the gradients. Refer to https://de.wikipedia.org/wiki/Sobel_operator for more information.

Filter 3 is a Laplace operator. It detects diagonal edges (refer to https://de.wikipedia.org/wiki/Laplace-Filter).

Exercise 4-3 PyTorch - Convolutional Neural Network

On the course website you will find an ipython notebook leading you through the implementaion of a Convolutional Neural Network in PyTorch for classifying handwritten digits from the MNIST dataset. Fill in the missing parts and compare the results to the ones from the Feedforward Neural Network we used last week. What do you observe?