Exercise Sheet #12

Deadline: 05.02.2024, 12:00h

Problem 1 (*Gated Linear Units*) (5 points)

A symmetrized version of equation (1.9) from the last chapter of the lecture notes is given by

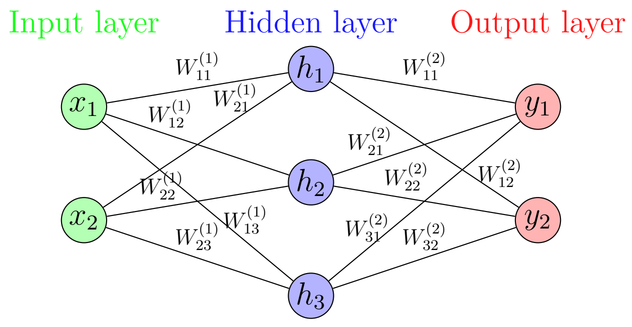
$$\sigma_{\mathrm{GLU}}(\vec{x}) = \tanh(\vec{w}\cdot\vec{x} - b_w)\tanh(\vec{v}\cdot\vec{x} - b_v),$$

where $\tanh(\,\cdot\,)$ has been used for the transfer function, and $\vec{w}, \vec{v}$ and b_w, b_v are the adaptable weight vectors and thresholds, respectively. Show that a single unit can encode both the AND and the XOR gate.

Hint: Find for each of the gates suitable choices for the weight vectors and thresholds, $\vec{w}, \vec{v}$ and b_w, b_v.

Problem 2 (*Pen and Paper Neuronal Network*) (15 points)

In this problem we will use and train a neuronal network with pen and paper. For that, consider the very simple neuronal network below with a single hidden layer and ReLU as the activation function, i.e. $\phi(x) = \max(x, 0)$. Assume that all external drivings / biases are zero.



$$W^{(1)} = \begin{pmatrix} 0.3 & -0.1 & 0.2 \\ 0.7 & -0.4 & -0.6 \end{pmatrix},$$

$$W^{(2)} = \begin{pmatrix} -0.3 & 0.2 \\ -0.8 & 0.2 \\ 0.4 & 0.6 \end{pmatrix}$$

(a) As a first step, we will see how input is propagated through the network. Assume you are given the input $x = (1, -1)^T$. Calculate the respective output vector y. Remember to apply the activation function. Include all intermediate steps to the output in your solution.

Now we want to optimize the network, i.e. train the weights by using training data. The basic road map is the following: We are given training data, that consists of input data with respective expected outputs (*targets*). We propagate the given input through the network and compare with the target by calculating the loss using a loss-function. To train the network, the goal is to minimize the loss. The minimization is achieved by following the gradient of the loss-function with respect to the system parameters (this is called the gradient decent algorithm). This algorithm requires the gradient of the function we want to minimize. To obtain that gradient we use backpropagation. Perform the following steps to train the network:

(b) We want to use MSE (mean square error) as our loss-function L, i.e.

$$L := (y - z)^2 ,$$

where z is the target for the output y of some training data x. Let the target vector for the input from part (a) be $z = (-0.2, 0.5)^T$. Calculate the loss for the given training data.

(c) To learn the weights that match the training data, we want to perform gradient descent, i.e. move the weights in the direction of the steepest descent. For that we first have to find the gradient of the loss-function with respect to the weights, i.e. find the partial derivatives

$$\frac{\partial L}{\partial W_{ij}} = \frac{\partial L}{\partial o_j}\frac{\partial o_j}{\partial W_{ij}} , \tag{1}$$

where on the right-hand side we used the chain-rule to rewrite the derivative. The neuron with activation o_j is the one we propagate backwards from along the weighted link W_{ij}. Using (1), calculate the derivative

$$\frac{\partial L}{\partial W_{21}^{(2)}} .$$

Hint: How can you express y_1 as a function of $W_{21}^{(2)}$?

(d) Next, calculate the gradient

$$\frac{\partial L}{\partial W_{12}^{(1)}} = \frac{\partial L}{\partial h_2}\frac{\partial h_2}{\partial W_{12}^{(1)}} .$$

Use the chain rule to find an expression for $\partial L/\partial h_2$.

(e) Using the gradient descent algorithm, we now want to update the weight matrices, such that the loss is minimized. To update a weight one step, use

$$W_{ij} \to W_{ij} - \alpha\frac{\partial L}{\partial W_{ij}} ,$$

where α is the learning rate.

Update the weights $W_{21}^{(2)}$ and $W_{12}^{(1)}$ one step with learning rate $\alpha = 0.1$.

(f) With the updated weights, calculate the loss for the given training data and compare with the result from part (b).