

# Knowledge Graph Analysis

## Solution For Exercise Sheet 7

---

Dr. Asja Fischer, Prof. Jens Lehmann

February 5, 2017

### 1 IN CLASS

#### 1. Transfer function

We know that

$$\left(\frac{f}{g}\right)' = \frac{f'g - fg'}{g^2} .$$

With this we obtain

$$\sigma'(u) = \frac{e^{-u}}{(1 + e^{-u})^2} = \frac{e^{-u}}{1 + e^{-u}} \cdot \frac{1}{1 + e^{-u}} = \frac{e^{-u}}{1 + e^{-u}} \cdot \sigma(u) .$$

We can see that

$$\frac{e^{-u}}{1 + e^{-u}} = \frac{1 + e^{-u} - 1}{1 + e^{-u}} = 1 - \frac{1}{1 + e^{-u}} = 1 - \sigma(u) .$$

And thus

$$\sigma'(u) = (1 - \sigma(u)) \cdot \sigma(u) .$$

#### 2. Backpropagation

- a) The goal of the backpropagation algorithm is to compute the error gradient of the neural network (by applying the chain rule several times).

- b) Let  $y$  be the true label,  $v^{(2)} = \sigma(u^{(2)})$  the output of the neural network and  $u^{(2)}$  the input into the output neuron. By applying the chain rule once and using the result from exercise 1 the error derivative with respect to  $u^{(2)}$  is given by

$$\begin{aligned}\frac{\partial E}{\partial u^{(2)}} &= \frac{\partial \frac{1}{2} \|\sigma(u^{(2)}) - y\|^2}{\partial u^{(2)}} \\ &= (v^{(2)} - y) \frac{\partial \sigma(u^{(2)})}{\partial u^{(2)}} \\ &= (v^{(2)} - y)(1 - \sigma(u^{(2)}))\sigma(u^{(2)}) .\end{aligned}$$

Let  $w_3$  denote the weight between the first hidden neuron and the output neuron and  $w_4$  the weight between the second hidden neuron and the output neuron. Let us further recall that  $u^{(2)} = v_1^{(1)} \cdot w_3 + v_2^{(1)} \cdot w_4$  where  $v_1^{(1)}$  and  $v_2^{(1)}$  are the output of the first and second hidden neuron, respectively. By applying the chain rule once more we get

$$\begin{aligned}\frac{\partial E}{\partial w_3} &= \frac{\partial E}{\partial u^{(2)}} \cdot \frac{\partial u^{(2)}}{\partial w_3} \\ &= \frac{\partial E}{\partial u^{(2)}} \cdot v_1^{(1)}\end{aligned}$$

and analogously

$$\begin{aligned}\frac{\partial E}{\partial w_4} &= \frac{\partial E}{\partial u^{(2)}} \cdot \frac{\partial u^{(2)}}{\partial w_4} \\ &= \frac{\partial E}{\partial u^{(2)}} \cdot v_2^{(1)} .\end{aligned}$$

The output of the hidden neurons is given by  $v_1^{(1)} = \sigma(u_1^{(1)})$  and  $v_2^{(1)} = \sigma(u_2^{(1)})$  with  $u_1^{(1)} = w_1 \cdot x$  and  $u_2^{(1)} = w_2 \cdot x$  where  $x$  is the input to the network and  $w_1$  and  $w_2$  the weight between input neuron and first and second hidden neuron, respectively. Therefore, (applying the chain rule 3 times in total) we have

$$\begin{aligned}\frac{\partial E}{\partial u_1^{(1)}} &= \frac{\partial E}{\partial u^{(2)}} \cdot \frac{\partial u^{(2)}}{\partial v_1^{(1)}} \cdot \frac{\partial v_1^{(1)}}{\partial u_1^{(1)}} \\ &= \frac{\partial E}{\partial u^{(2)}} \cdot w_3 \cdot (1 - \sigma(u_1^{(1)}))\sigma(u_1^{(1)})\end{aligned}$$

and analogously

$$\begin{aligned}\frac{\partial E}{\partial u_2^{(1)}} &= \frac{\partial E}{\partial u^{(2)}} \cdot \frac{\partial u^{(2)}}{\partial v_2^{(1)}} \cdot \frac{\partial v_2^{(1)}}{\partial u_2^{(1)}} \\ &= \frac{\partial E}{\partial u^{(2)}} \cdot w_4 \cdot (1 - \sigma(u_2^{(1)}))\sigma(u_2^{(1)}) .\end{aligned}$$

Finally we get (by applying the chain rule once more and 4 times in total)

$$\begin{aligned}\frac{\partial E}{\partial w_1} &= \frac{\partial E}{\partial u^{(2)}} \cdot \frac{\partial u^{(2)}}{\partial v_1^{(1)}} \cdot \frac{\partial v_1^{(1)}}{\partial u_1^{(1)}} \cdot \frac{\partial u_1^{(1)}}{\partial w_1} \\ &= \frac{\partial E}{\partial u_1^{(1)}} * x\end{aligned}$$

and analogously

$$\begin{aligned}\frac{\partial E}{\partial w_2} &= \frac{\partial E}{\partial u^{(2)}} \cdot \frac{\partial u^{(2)}}{\partial v_2^{(1)}} \cdot \frac{\partial v_2^{(1)}}{\partial u_2^{(1)}} \cdot \frac{\partial u_2^{(1)}}{\partial w_2} \\ &= \frac{\partial E}{\partial u_2^{(1)}} * x\end{aligned}$$

### 3. MLP-based SRL models

- a) In the E-MLP only the entities are presented by latent feature vectors which serve as inputs to the neural network and relation specific weights  $W_k$  and  $r_k$  are learned. In ER-MLP also the relations are mapped to a latent representation, so that the latent presentation of the two entities and the relation serve as neural network input and general weights  $W$  and  $r$  (which are the same for all relations) are learned .
- b) Let us use the following notations:

#entities =  $N_e$   
#relations =  $N_r$   
#latent features of entities =  $H_e$   
#latent features of relations =  $H_r$   
#hidden neurons in the MLP/NT=  $H_a$   
#additional hidden neurons in the NT (resulting from the tensor term) =  $H_b$

For E-MLP each of the  $N_e$  entities is represented by  $H_e$  latent features, and for each of the  $N_r$  relations one earns a wight matrix  $\mathbf{W}_k$  which has  $H_a \cdot 2H_e$  parameters and a vector  $\mathbf{r}_k$  of dimension  $H_a$ . Thus, we get

$$N_e \cdot H_e + N_r \cdot (H_a \cdot 2H_e + H_a)$$

parameters in total.

Following a similar argumentation for the ER-MLP one gets

$$N_e \cdot H_e + N_r \cdot H_r + H_a + H_a \cdot (2H_e + H_r) ,$$

where one has  $H_r$  latent features for each of the  $N_r$  relations and just one weight matrix  $\mathbf{W}$  of dimension  $H_a \times 2H_e + H_a$  and a vector  $\mathbf{r}$  of dimension  $H_a$ .

For the Neural Tensor model one has

$$N_r \cdot H_e^2 \cdot H_b + N_r(H_b + H_a) + 2N_r \cdot H_e \cdot H_a + N_e \cdot H_e$$

parameters, because there are  $N_r$  tensors with  $H_b$  slices of dimension  $H_e \times H_e$ , a weight vector  $\mathbf{r}_k$  of dimension  $H_b + H_a$  and a weight matrix  $\mathbf{W}_k$  which has  $H_a \cdot 2H_e$  parameters for each of the  $N_r$  relations, and  $H_e$  latent features for each of the  $N_e$  entities.