Deep Learning for Natural Language Processing

Exercise 3 – Backpropagation

Niraj Dev Pandey

Ubiquitous Knowledge Processing Lab
Technische Universität Darmstadt

Today

- Backpropagation Example
- Regular exercise
- Tensorflow Introduction

Today

- **Backpropagation Example**
- Regular exercise
- Tensorflow Introduction

Backpropagation

- Purpose:
  - Obtaining a gradient for each weight in each layer of the network
  - This is necessary to train multilayer perceptrons (MLPs) – and other more complex networks
- Principle:
  - The network error is propagated from the output layer back to the input layer

Backpropagation: Steps

Given a neural network and a training input $(\boldsymbol{x}, \boldsymbol{t})$:

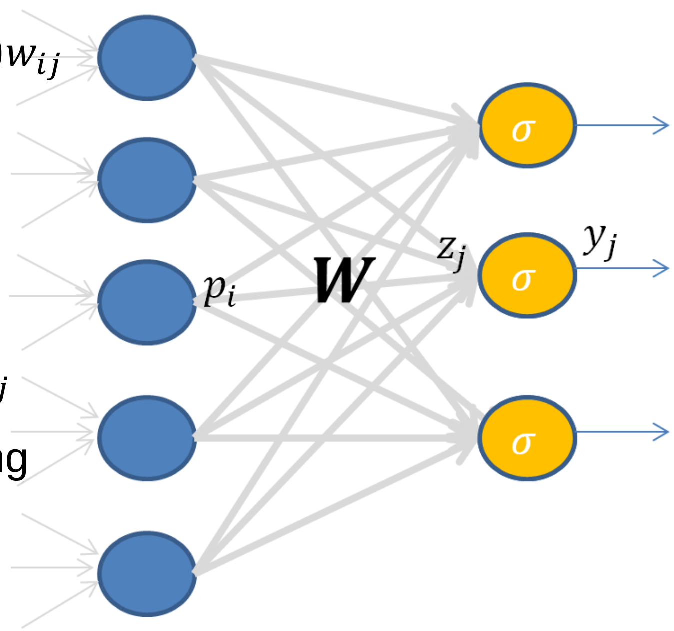
1. Forward propagation
   - Compute activation (the output of the activation function) and pre-activation for each neuron in the network
2. Backward propagation
   - Compute the error derivative $\frac{\partial E}{\partial p_i}$ at each neuron p_i
   - Compute the error derivative $\frac{\partial E}{\partial u_{ik}}$ for each weight u_{ik} in the weight matrices $\boldsymbol{U} = \boldsymbol{W}^{(\mathbf{1})}, \boldsymbol{W}^{(\mathbf{2})}, \boldsymbol{W}^{(\mathbf{3})}, \ldots$

Backpropagation: Gradient at Neurons

TECHNISCHE UNIVERSITÄT DARMSTADT

2. Backward propagation

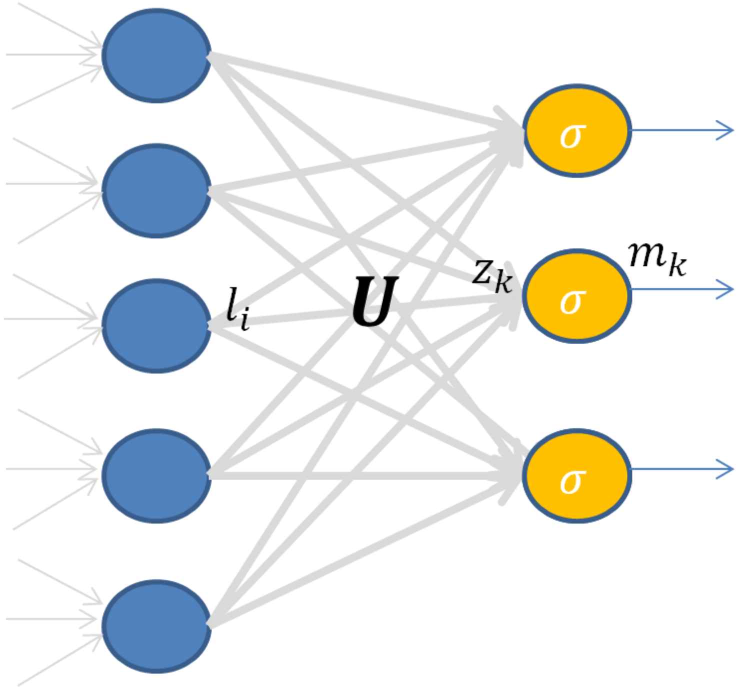
- Compute the error derivative $\frac{\partial E}{\partial p_i}$ at each neuron p_i
  - For the output layer: E is the network's loss function
  - For the hidden layers, use: $\frac{\partial E}{\partial p_i} = \sum_j \frac{\partial E}{\partial y_j} \sigma'(z_j) w_{ij}$
    - p_i: neuron for which we compute the error
    - y_j: neurons from which the error is propagated backwards
    - σ': derivative of the activation function at y_j
    - z_j: pre-activation at y_j (argument of σ during forward propagation)



Backpropagation: Gradient at Weights

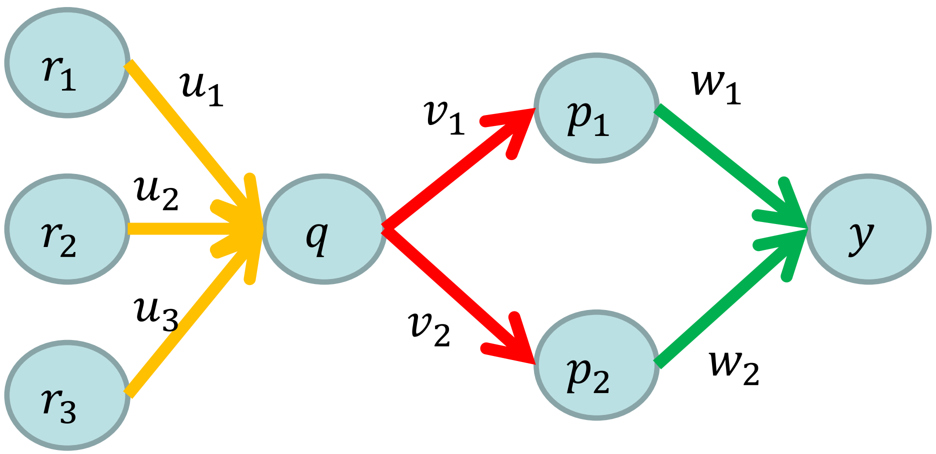
2. Backward propagation

- Compute the error derivative $\frac{\partial E}{\partial u_{ik}}$ for each weight u_{ik} in the weight matrices $\boldsymbol{U} = \boldsymbol{W}^{(\mathbf{1})}, \boldsymbol{W}^{(\mathbf{2})}, \boldsymbol{W}^{(\mathbf{3})}, \ldots$
  - $\frac{\partial E}{\partial u_{ik}} = \frac{\partial E}{\partial m_k} \sigma'(z_k) l_i$
    - m_k: the output/activation at the layer corresponding to matrix $\boldsymbol{U}$
    - l_i: the output/activation at the previous layer
    - z_k: pre-activation at m_k(argument of σ during forward propagation)



Backprop Example

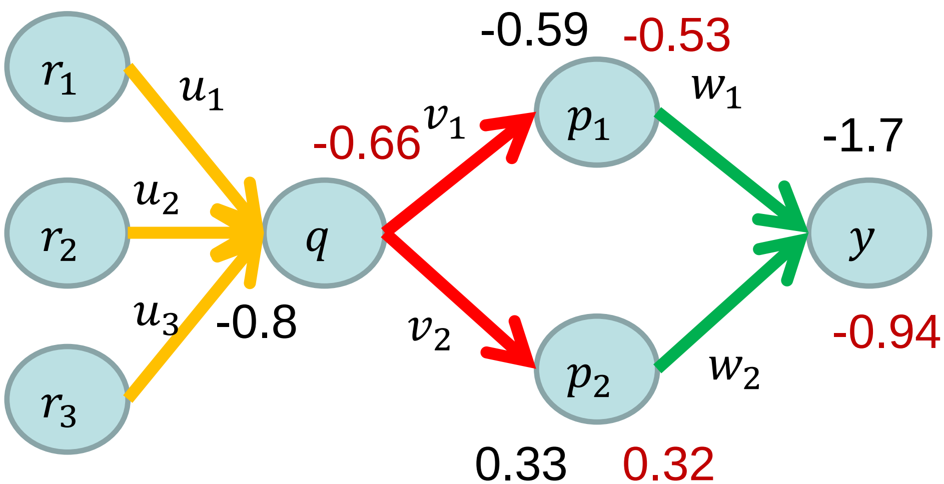
- Color coding: **preactivation**, activation, error deriv. at neurons, weights
- All activation functions are $\tanh$, loss is square loss
- Initialization: $\boldsymbol{u} = (0.2, 0.5, -1), \boldsymbol{v} = (0.9, -0.5), \boldsymbol{w} = (0.2, -5)$
- Round to two decimal places after each computation



- Assume that $\boldsymbol{x} = (1,0,1)$ and $t = 1$

Backprop Example: Forward Pass

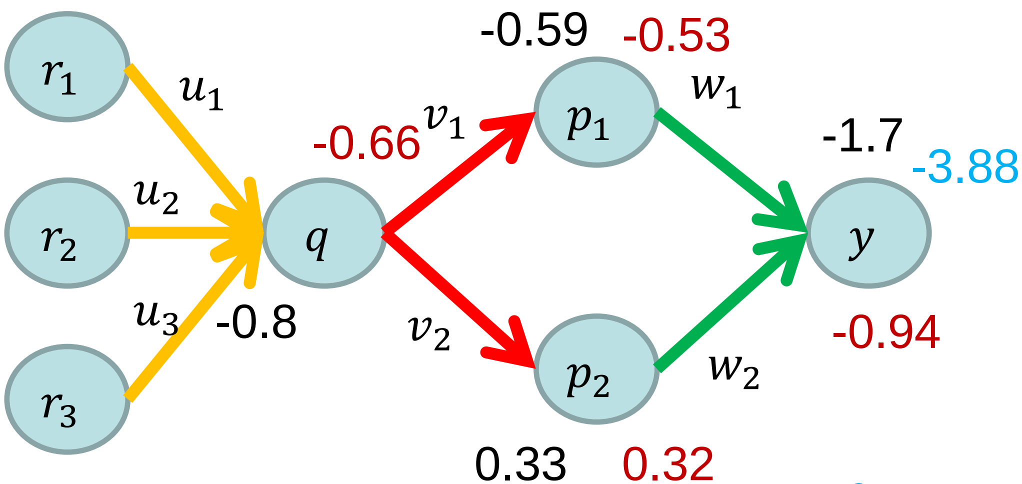
- Color coding: **preactivation**, activation, error deriv. at neurons, weights
- All activation functions are tanh, loss is square loss
- Initialization: $\boldsymbol{u} = (0.2, 0.5, -1), \boldsymbol{v} = (0.9, -0.5), \boldsymbol{w} = (0.2, -5)$
- Round to two decimal places after each computation



- Assume that $\boldsymbol{x} = (1,0,1)$ and $t = 1$

Backprop Example: Gradient at Neurons

- Color coding: **preactivation**, **activation**, **error deriv. at neurons**, **weights**
- All activation functions are tanh, loss is square loss
- Initialization: $\boldsymbol{u} = (0.2, 0.5, -1), \boldsymbol{v} = (0.9, -0.5), \boldsymbol{w} = (0.2, -5)$
- Round to two decimal places after each computation

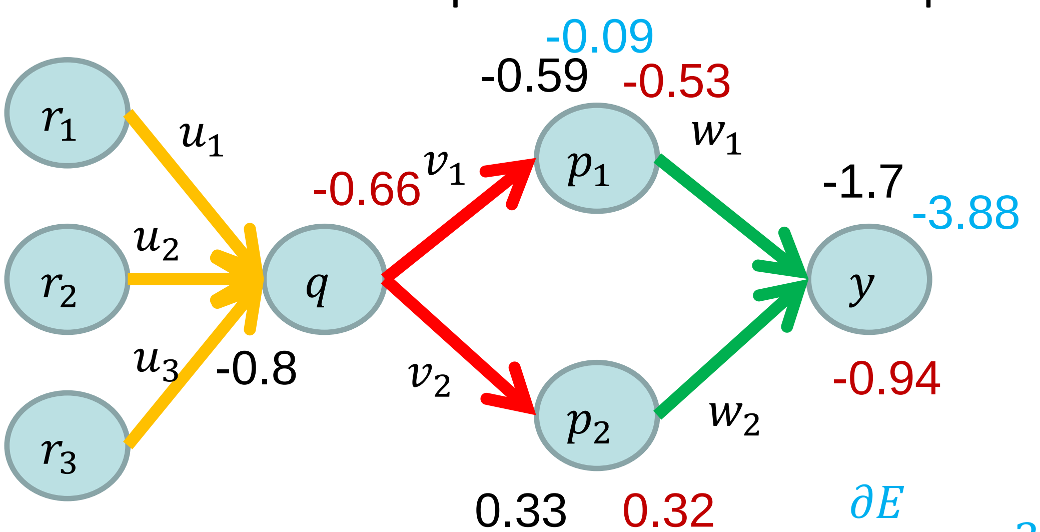


- Assume that $\boldsymbol{x} = (1,0,1)$ and $t = 1$
- **Square loss:** $(t - y)^2$

$$\frac{\partial E}{\partial y} = 2(y - t) = 2(-0.94 - 1) = -3.88$$

Backprop Example: Gradient at Neurons (cont.)

- Color coding: **preactivation**, activation, error deriv. at neurons, weights
- All activation functions are tanh, loss is square loss
- Initialization: $\boldsymbol{u} = (0.2, 0.5, -1), \boldsymbol{v} = (0.9, -0.5), \boldsymbol{w} = (0.2, -5)$
- Round to two decimal places after each computation



- Assume that $\boldsymbol{x} = (1,0,1)$ and $t = 1$
- Square loss: $(t-y)^2$

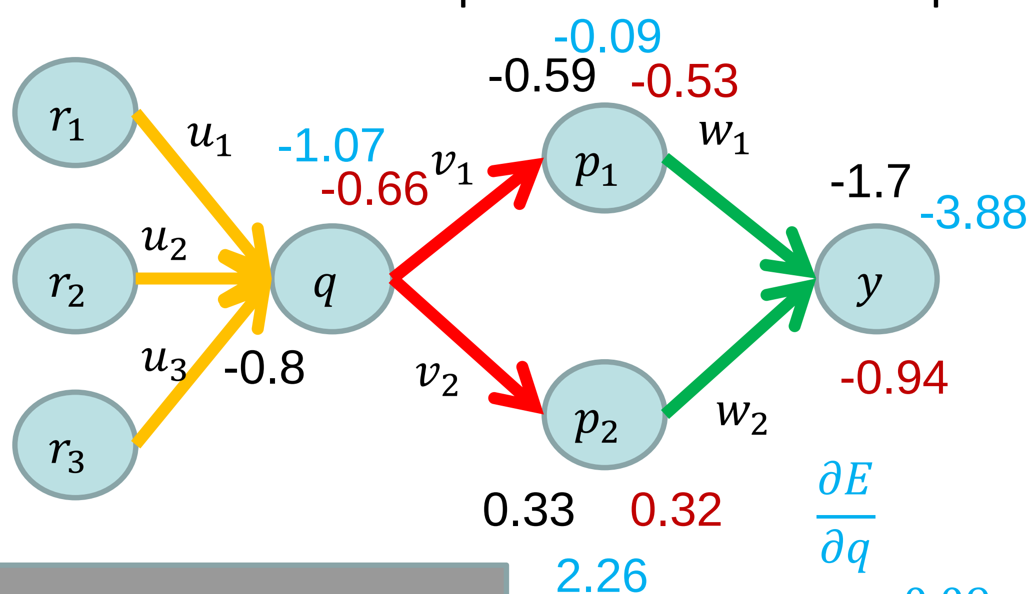
$$\frac{\partial E}{\partial p_1} = -3.88 * (1 - \tanh^2(-1.7)) * 0.2$$
$$= -3.88 * (1 - (-0.94)^2) * 0.2 = -0.09$$

- $\frac{\partial E}{\partial p_i} = \sum_j \frac{\partial E}{\partial y_j} \sigma'(z_j) w_{ij}$

$\tanh' = 1 - \tanh^2$

Backprop Example: Gradient at Neurons (cont.)

- Color coding: **preactivation**, activation, error deriv. at neurons, weights
- All activation functions are tanh, loss is square loss
- Initialization: $\boldsymbol{u} = (0.2, 0.5, -1), \boldsymbol{v} = (0.9, -0.5), \boldsymbol{w} = (0.2, -5)$
- Round to two decimal places after each computation



- Assume that $\boldsymbol{x} = (1,0,1)$ and $t = 1$
- Square loss: $(t - y)^2$

$$\frac{\partial E}{\partial q} = -0.09 * (1 - tanh^2(-0.59)) * 0.9 + 2.26 * (1 - tanh^2(0.33)) * -0.5$$

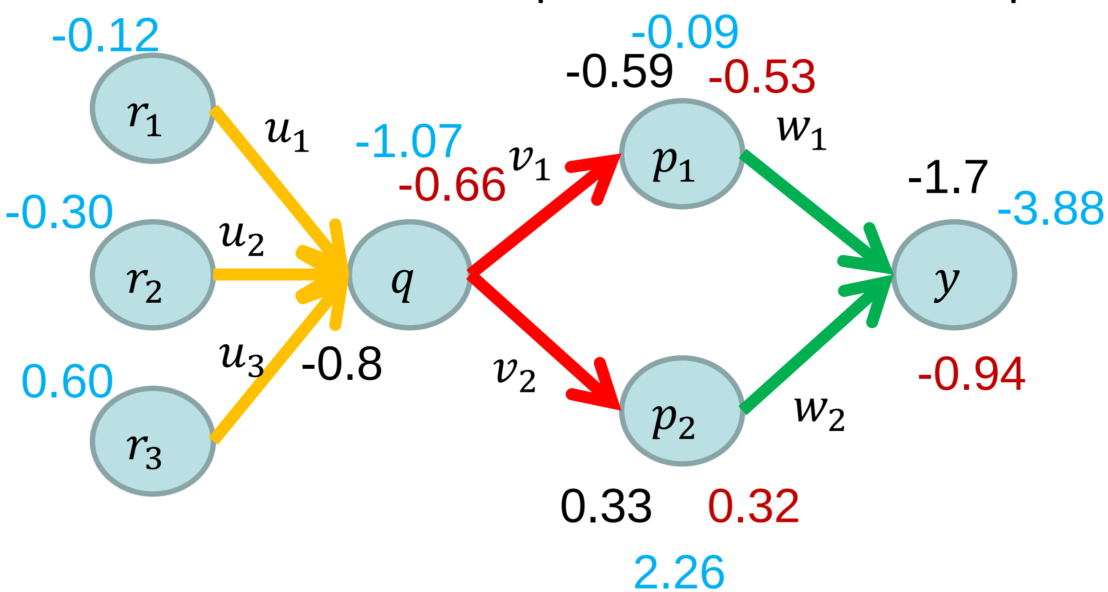
$$= -0.06 - 1.01 = -1.07$$

- $\frac{\partial E}{\partial p_i} = \sum_j \frac{\partial E}{\partial y_j} \sigma'(z_j) w_{ij}$

$\tanh' = 1 - \tanh^2$

Backprop Example: Gradient at Neurons finished

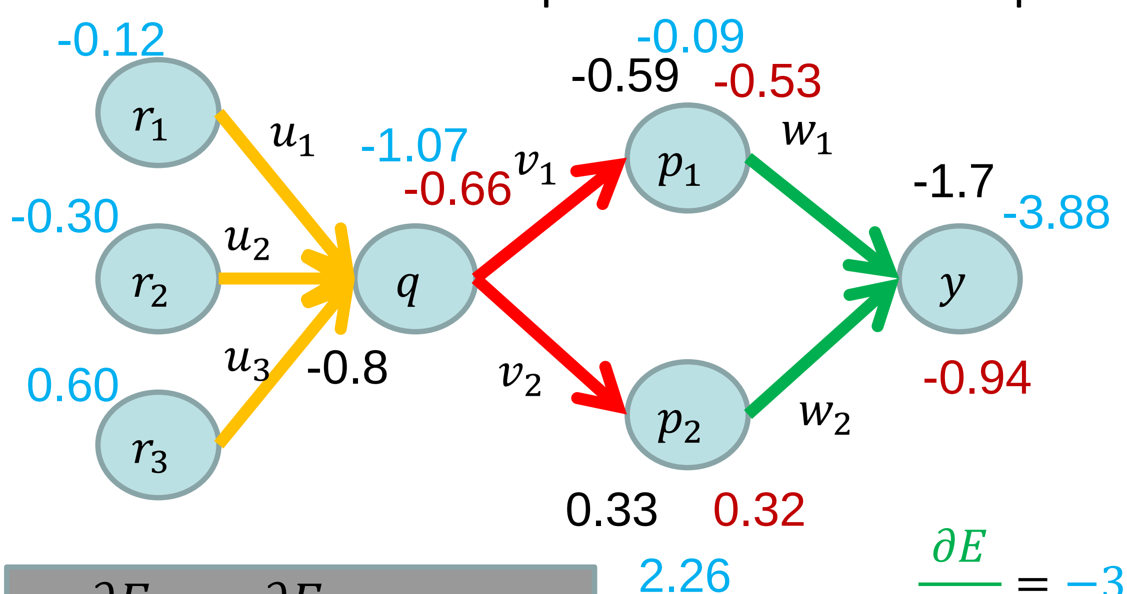
- Color coding: **preactivation**, activation, error deriv. at neurons, weights
- All activation functions are tanh, loss is square loss
- Initialization: $\boldsymbol{u} = (0.2, 0.5, -1), \boldsymbol{v} = (0.9, -0.5), \boldsymbol{w} = (0.2, -5)$
- Round to two decimal places after each computation



- Assume that $\boldsymbol{x} = (1,0,1)$ and $t = 1$
- Square loss: $(t - y)^2$

Backprop Example: Gradient at Weights

- Color coding: **preactivation**, activation, error deriv. at neurons, weights
- All activation functions are tanh, loss is square loss
- Initialization: $\boldsymbol{u} = (0.2, 0.5, -1), \boldsymbol{v} = (0.9, -0.5), \boldsymbol{w} = (0.2, -5)$
- Round to two decimal places after each computation



- Assume that $\boldsymbol{x} = (1,0,1)$ and $t = 1$
- Square loss: $(t - y)^2$

$$\frac{\partial E}{\partial u_{ik}} = \frac{\partial E}{\partial m_k} \sigma'(z_k) l_i$$

$$\frac{\partial E}{\partial w_1} = -3.88 \cdot (1 - tanh^2(-1.7)) \cdot (-0.53)$$

$$= 0.24$$

Today

- ✓ Backpropagation Example
- **Regular exercise**
- Tensorflow Introduction

Today

- ✓ Backpropagation Example
- ✓ Regular exercise
- **Tensorflow Introduction**