Deep Learning for Speech and Language Processing
Exercise Sheet 4: Backpropagation, Gradient Descent

Maximilian Schmidt, Pascal Tilli, Dirk Väth, Ngoc Thang Vu

November 24th, 2020

Deadline

Please submit your solutions until Monday, December 7th, 2020, noon. You may also see the deadline at any time on the website under *Exercises* → *Upload*.

Notation

Additional notation conventions for this exercise:

Symbol	Example	Description
d	$\frac{df}{dz}$	total derivative of f with respect to z
∂	$\frac{\partial f}{\partial z}$	partial derivative of f with respect to z (f directly depends on z)
Greek delta	$\delta^l = \frac{\mathrm{d}C}{\mathrm{d}z^l}^\top$	total derivative of the cost function C with respect to the inputs to the activation function in layer l of a neural network

For the total derivative you have to use the chain rule: $\frac{\mathrm{d}f(x)}{\mathrm{d}z} = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i} \frac{\mathrm{d}x_i}{\mathrm{d}z}$ for $x \in \mathbb{R}^n$

Submission

Calculation Tasks

Upload (only solutions!) to https://dlcourse.ims.uni-stuttgart.de/ using your account.

Theory Tasks

You don't have to submit this part, but it will help you with the contents of this course.

Theory Tasks

Theory Task 1: Activation derivatives

In the lecture you were shown the tanh function as alternative activation function in neural networks.

Exercise 1.

Compute the first derivative of $\tanh(z) = \frac{\sinh(z)}{\cosh(z)} = \frac{e^z - e^{-z}}{e^z + e^{-z}}$ with respect to z.

Hint: $\frac{d\sinh(z)}{dz} = \cosh(z)$, $\frac{d\cosh(z)}{dz} = \sinh(z)$. You can either first apply the product rule and then the chain rule or directly work with the quotient rule $f(x) = \frac{g(x)}{h(x)}; f'(x) = \frac{g'(x)h(x) - g(x)h'(x)}{[h(x)]^2}$.

Theory Task 2: Mean Squared Error vs Cross-Entropy Cost Function

In this exercise, we will look at the effect of the choice of the cost function on the gradients. Consider the network for multi-class classification with $N = 5$ classes, which only consists of an output layer, with f being the softmax function, depicted in Figure 1.

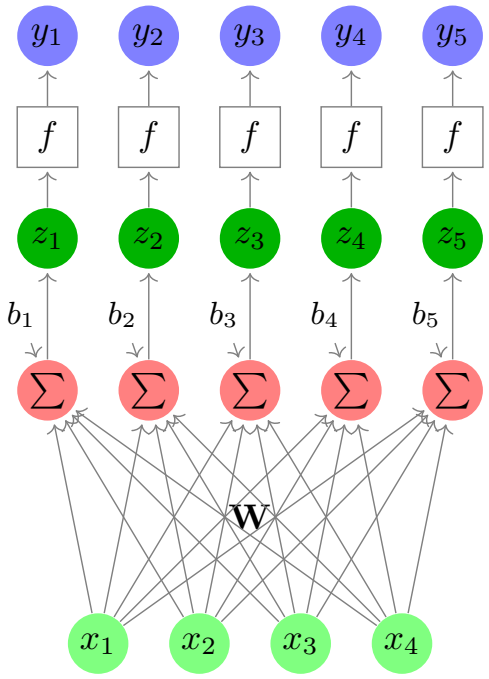

Figure 1: Network for multi-class classification.

In order to train the parameters of this network, i.e. W and b, we need a cost function, which evaluates the predictions of the network y against a ground truth $\hat{y}$. You already know two cost functions for neural networks:

Mean Squared Error (MSE) $c_{\text{MSE}}(\hat{y}, y) = \sum_{i=0}^{N-1}(\hat{y}_i - y_i)^2$
Note: We omit the scaling factor here since it's not important for optimization.

Cross Entropy (CE) $c_{\text{CE}}(\hat{y}, y) = -\sum_{i=1}^{N} \hat{y}_i \ln y_i = -\ln y_t$ for $\hat{y}_t = 1$ (i.e. t being the correct class) in the case of classification

Exercise 2.

(1) Give a general formula on how to compute the error c of the network given an input x. Then using the chain rule, derive the general formula to compute the derivative of c with respect to a weight w_{ij}, which connects input x_j to z_i. Hint: Use the derivative of the softmax function: $\frac{\partial y_i}{\partial z_j} = y_j(1 - y_j)$ for $i = j$ and $= -y_i y_j$ for $i \neq j$.

(2) What do you need to change in 1) to compute the derivative of c with respect to a bias entry b_i?

(3) Compute the derivative of c_{CE} with respect to a weight w_{ij} $\dfrac{\mathrm{d}c_{\text{CE}}}{\mathrm{d}w_{ij}}$.

(4) Compute the derivative of c_{MSE} with respect to a weight w_{ij} $\dfrac{\mathrm{d}c_{\text{MSE}}}{\mathrm{d}w_{ij}}$. What happens with the derivative for values of y_i close to 0 or 1?

(5) Why is it important to shuffle the dataset after each training pass through the dataset (= epoch) when training with stochastic or mini-batch gradient descent?

Calculation Tasks

Stochastic Gradient Descent

In this exercise, we will compute a complete step of the stochastic gradient descent algorithm by hand. This includes a forward pass to compute the output and backward pass to compute the gradients. We will use the multi-class banana network from exercise sheet 2 with some modifications as depicted in Figure 2. The task of the network is still to predict $y = \begin{bmatrix} \text{p(banana)} \\ \text{p(cookie)} \\ \text{p(other)} \end{bmatrix}$ given a tweet in bag-of-words representation as input.

The network has non-linear activation in the two hidden layers, specifically $f^1 = f^2 = \text{sigmoid} = \sigma$. The output layer uses softmax activation. There is only a bias vector in the first hidden layer initialized to 0. The second hidden layer and the output layer do not have biases. The network has the following initial weight matrices:

$$W^1 \in \mathbb{R}^{2\times 4} = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 \\ 1 & 0.5 & 0 & 1 \end{bmatrix}, W^2 \in \mathbb{R}^{2\times 2} = \begin{bmatrix} 0 & -1 \\ 1 & -1 \end{bmatrix}, W^3 \in \mathbb{R}^{3\times 2} = \begin{bmatrix} 1 & 0.5 \\ 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}.$$

We will refer to the set of all parameters of the network with $\theta = \{W^1, W^2, W^3, b^1\}$. In the forward pass, we will first compute the output of the network given some input and score the network output using CE. Subsequently, in the backward pass, we will compute the gradients of the cost function $\Delta c_{CE}(\theta)$ with respect to the parameters and make a gradient update such that the likelihood of the ground truth increases: $\theta^1 \leftarrow \theta - \eta \nabla c_{CE}(\theta)$.

Exercise 3.

(1) Forward pass: Compute the output y of the network for the tweet 'A banana a day keeps the doctor away.' Represent the tweet as a 4-dimensional bag-of-words vectors given the 4-word vocabulary $V = \{v_0 : banana, v_1 : chocolate, v_2 : sweet, v_3 : cookie\}$.
Hint: Write down the intermediate results a^l, z^l as you will need them from Question (3) onwards.

(2) Compute the cost c_{CE} (using ln) for the output y, given the correct label $\hat{y} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$.

(3) Backward pass: You need to successively compute the vectors δ^l in each layer, starting with the output layer. Use the original weights which were valid during the forward pass (the cost depends on these). Then you can obtain $\frac{\mathrm{d}c_{CE}}{\mathrm{d}w^l_{ij}} = \frac{\mathrm{d}c_{CE}}{\mathrm{d}z^l_i}\frac{\partial z^l_i}{\partial {w^l}_{ij}} = \delta^l_i a^{l-1}_j$. The computation for the $\frac{\mathrm{d}c_{CE}}{\mathrm{d}b^l_j}$ works analogously with a small change. You will need all derivatives in the following tasks.

(4) For the parameters W^3

(1) compute the gradient of W^3 with the result being the matrix $\begin{bmatrix} \frac{\mathrm{d}c_{CE}}{\mathrm{d}w^3_{11}} & \frac{\mathrm{d}c_{CE}}{\mathrm{d}w^3_{12}} \\ \frac{\mathrm{d}c_{CE}}{\mathrm{d}w^3_{21}} & \frac{\mathrm{d}c_{CE}}{\mathrm{d}w^3_{22}} \\ \frac{\mathrm{d}c_{CE}}{\mathrm{d}w^3_{31}} & \frac{\mathrm{d}c_{CE}}{\mathrm{d}w^3_{32}} \end{bmatrix}$

(2) perform a gradient update step using these derivatives with $\eta = 0.1$ (provide the updated weight matrix).

(5) For the parameters W^2

(1) compute the gradient of W^2 analog to the previous task.

(2) perform a gradient update step using these derivatives with $\eta = 0.1$ (provide the updated weight matrix).

(6) For the parameters W^1

(1) compute the gradient of W^1 analog to the previous task.

(2) perform a gradient update step using these derivatives with $\eta = 0.1$ (provide the updated weight matrix).

(7) For the parameters b^1

(1) compute the gradient of b^1 analog to the previous task.

(2) perform a gradient update step using these derivatives with $\eta = 0.1$ (provide the updated bias).

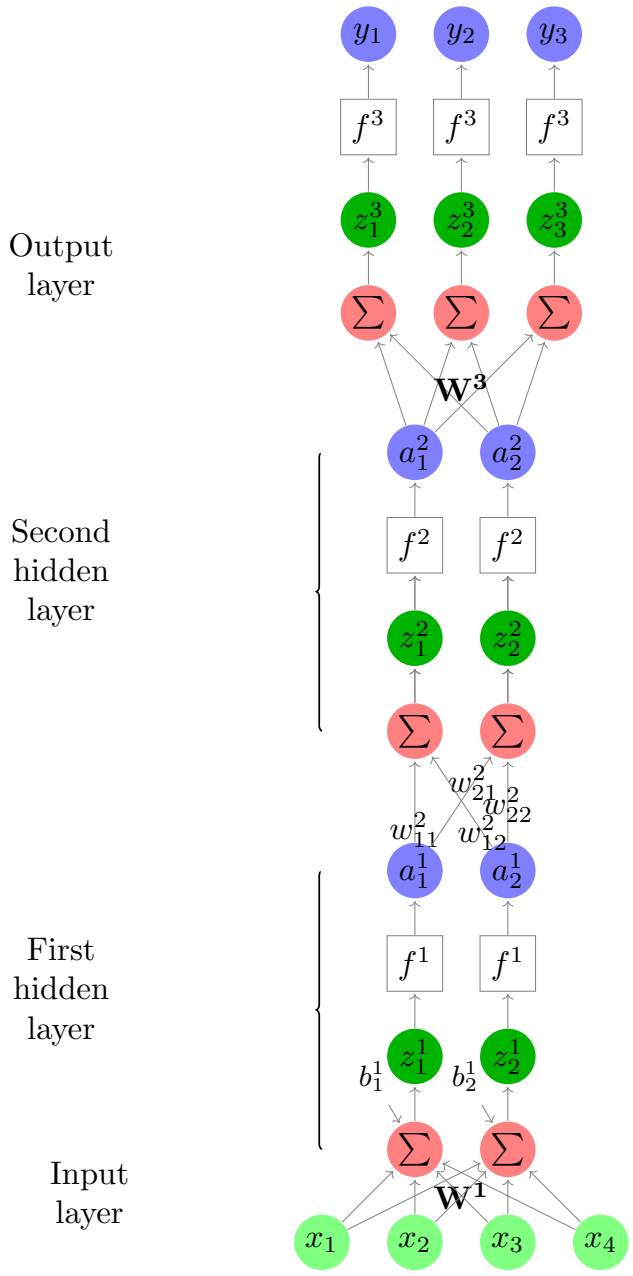

Figure 2: Revised banana network.