

Homework 4: Deep Learning

Due June 9, beginning of exercise session*

Kristian Kersting, Alejandro Molina

{kersting, molina}@cs.tu-darmstadt.de upload link:

<https://www.dropbox.com/request/OEyNaA11EGyiexfyadSa>

1. Implement the following network in PyTorch.

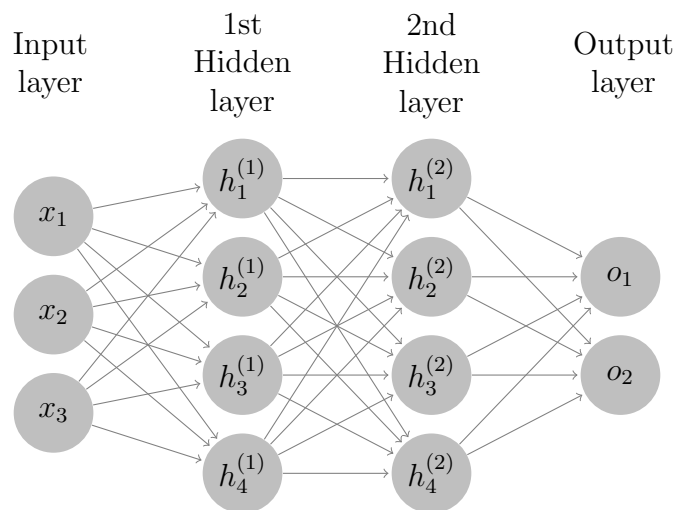


Figure 1: A feed-forward neural network architecture consisting of two hidden layers.

- The sigmoid activation function is used for computing hidden activations.
- No activation function in the output layer.
- In this homework, please use the following code snippet to initialize all parameters, inputs and their ground truth in the FIRST cell of your notebook.

```
# Python CODE  
import numpy as np  
# DO NOT FORGET TO SPECIFY THE SAME SEED  
np.random.seed(12345)
```

*We will discuss the solutions in the exercise session. It is my suggestion that you try to address at least 50% of the exercise questions. Simply try hard to solve them. This way, you will get familiar with the technical terms and with the underlying ideas of the lecture.

```

def initialize(input_dim, hidden_dim, output_dim, batch_size):
    W1 = np.random.randn(hidden_dim, input_dim) * 0.01
    b1 = np.zeros((hidden_dim,))
    W2 = np.random.randn(hidden_dim, hidden_dim) * 0.01
    b2 = np.zeros((hidden_dim,))
    W3 = np.random.randn(output_dim, hidden_dim) * 0.01
    b3 = np.zeros((output_dim,))

    # list of all network parameters
    parameters = [W1, b1, W2, b2, W3, b3]

    # minibatch of input instances
    x = np.random.rand(input_dim, batch_size)

    # ground truths
    y = np.random.randn(output_dim, batch_size)

    return parameters, x, y

# initialize parameters, inputs and targets
parameters, x, y = initialize(3, 4, 2, 5)

```

And the squared loss function:

$$\mathcal{L}(\theta; \mathbf{x}, \mathbf{y}) = \frac{1}{M} \sum_{n=1}^M \frac{1}{2} (f_{\theta}(\mathbf{x}_n) - \mathbf{y}_n)^2 \quad (1)$$

where f_{θ} denotes the outputs of the network given inputs \mathbf{x} , \mathbf{y} is the targets, and θ denotes a set of the parameters.

2. Compute the forward pass using PyTorch, print the output values of every layer.
3. Compute the gradients using PyTorch, i.e., `loss.backward()`, `param.grad` and show:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_1}, \frac{\partial \mathcal{L}}{\partial \mathbf{b}_1}, \frac{\partial \mathcal{L}}{\partial \mathbf{W}_2}, \frac{\partial \mathcal{L}}{\partial \mathbf{b}_2}, \frac{\partial \mathcal{L}}{\partial \mathbf{W}_3}, \frac{\partial \mathcal{L}}{\partial \mathbf{b}_3}$$

4. Did you get the same results as in your previous homeworks?

Some references:

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

<https://github.com/yunjey/pytorch-tutorial>