You can use the lecture slides and the resources in the book at http://neuralnetworksanddeeplearning.com/ to help with your implementation.

Submit a report of up to 3 pages (can be shorter) in pdf form. The report should contain answers to sub-tasks in Task 6. For each task list the parameters of the networks that you used (number of layers, number of neurons on each layer, learning rate, optimizer type, regularization used, learning rate schedule used) and show the results achieved. You can use the LaTeX template posted on Ucilnica.

1 Backpropagation

You are given a simple neural network with weights $w^l_{j,k}$ and biases b^l_k shown in Figure 1. For the input $x = [0.5, 0.5, 0.1]$, the ground truth $\hat{y} = [0.9, 0.1]$, the loss function $L = \frac{1}{2}\sum_{i=1}^{n_{out}}(y_i - \hat{y_i})^2$ and the learning rate $\eta = 0.1$ simulate the forward and the backward pass. All neurons use the sigmoid activation function $\sigma(x) = \frac{1}{1+e^{-x}}$. You do not have to simulate the backward pass for the entire network. Only show how the weight $w^1_{1,1}$ is updated after backpropagation i.e. derive $\frac{\partial L}{\partial w^1_{1,1}}$. Do the task using pen and paper, take a picture of the solution and submit it with the rest of the homework.

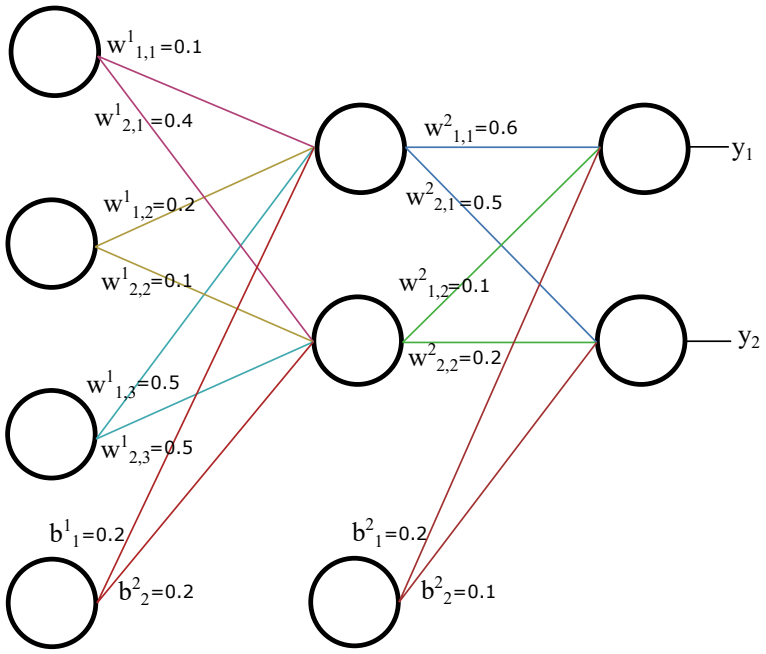

Figure 1: Architecture of a two-layer neural network.

2 Network implementation

Implement a basic neural network. If you wish you can use the template network_template.py added in the supplementary material of this exercise, where you can implement the missing $forward\_pass$, $backward\_pass$ and $update\_network$ functions. The activation function on the final layer should be the softmax function. The loss function used should be the cross entropy loss as the network will be used for classification in Exercise 6.

3 Regularization

Implement L2 regularization presented during lectures. Alternatively you can also implement a different regularization method presented during lectures.

4 Optimizer

Implement the Adam optimizer presented during lectures and add it to your network training.

5 Learning rate decay

Implement the exponential learning rate decay and add it to your network training. The learning rate for epoch t, decay rate k with a starting learning rate η is then calculated by:

$$\eta_t = \eta e^{(-kt)} \tag{1}$$

6 Network training

Train your network on the CIFAR-10 dataset. Experiment with network parameters (number, of layers, number of neurons, learning rate etc.) and examine the effect of the utilized regularization, optimizer and learning rate schedule. Note that not all of these features will directly improve your result, the goal is to familiarize yourself with some of the tools available for training neural networks. Examine the impact of removing regularization, learning rate schedule and training using SGD instead of Adam. You can use the *network_template.py* as a starting point for this homework as it contains the data loader for the CIFAR-10 data. It is provided in the supplementary material.

Tasks:

- Achieve a classification accuracy of at least 42%. If your implementation is correct, this is achievable with only a basic network implementation without using the features in tasks 3,4 and 5. With the addition of these features and with carefully set network parameters you can go quite a bit higher (last year some students got 50%+ classification accuracy). Save the network settings (learning rate, optimizer, number of epochs, number of layers and neurons of the network etc.) and use them for the remaining tasks.
- Train the network once with Adam and once with SGD. Adjust the learning rate but keep all other training parameters the same. Comment on the results achieved after each training?
- Train the network with and without L2 regularization. Keep all other training parameters the same. What are the results of the training?
- Train the network with and without a learning rate schedule. All other parameters are the same. What are the results of the training?