“Introduction to Deep Learning” course, 2018.

Lecturer: Artem Chernodub, Ph.D.

a.chernodub@gmail.com

version 1.2 from 26 may 2018

Student homework must be presented in an electronic form and sent by “Data Science MSc program” slack or by email. Deadlines for the first and the second homework are 18/06/2018, 9:00 AM CET and 2/07/2018, 9:00 AM CET, respectively. The penalty for the passing deadline: up to one week it is 50%, more than one week it is 100% of the homework’s scores.

Using LaTeX[1] or MS Word equation editors for the first homework and python & numpy or MATLAB (GNU Octave) frameworks for the second homework are highly recommended (but are not limited to). However, any equation editor or programming language is accepted[2]. At the same time, presentation of the first homework as a scanned document is also accepted, but the penalty for this is 25% for the first homework.

The first homework must be done individually. For the second homework, students can create the teams and present a single joint solution. Maximum size of a team is two cool neuroscientists.

[1] Please, take a look at http://sharelatex.com. Who knows, maybe you will find it useful for your needs.
[2] If you will provide the detailed instructions how to see it or how to run it.

Homework I (theoretical, 50% of scores)

Consider a feedforward multilayer perceptron with two hidden layers and ResNet-like shortcut connections. Neural network's structure is shown on Fig. 1. The notations below are the same as in Lecture 1, "Training the Multilayer Perceptrons".

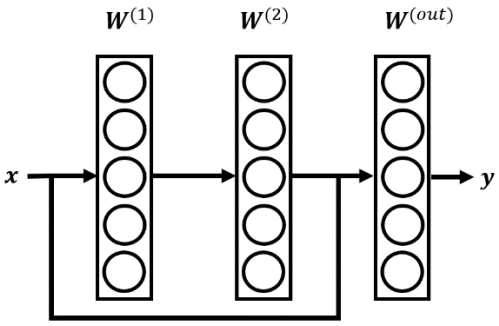

Figure 1. Neural network's structure. Neural network receives input vector $\boldsymbol{x}$ and produces output vector $\boldsymbol{y}$. Weight matrices are defined as $\boldsymbol{W}^{(1)}$, $\boldsymbol{W}^{(2)}$, $\boldsymbol{W}^{(out)}$ for the first hidden, the second hidden an the output layer, respectively.

We suppose that this neural network has D inputs, M hidden units at the first hidden layer, L hidden units at the second hidden layer and P outputs. Forward pass for (k)-th layer is:

$$a_j^{(k)} = \sum_i w_{ji}^{(k)} z_i^{(k-1)} + b_j^{(k)}, \quad (1)$$

$$z_j^{(k)} = f^{(k)}\left(a_j^{(k)}\right), \quad (2)$$

where $a_j^{(k)}$ are pre-synaptic values, $f^{(k)}(\cdot)$ is a non-linear activation function , $z_j^{(k)}$ are post-synaptic values, $w_{ji}^{(k)}$ and $b_j^{(k)}$ are tunable parameters ($b_j^{(k)} \equiv w_{j0}^{(k)}$). $w_{ji}^{(k)}$ is a weight responsible for connection from neuron i at layer $(k-1)$ to neuron j at layer (k). By the definition, $\boldsymbol{z}^{(0)} = \boldsymbol{x}$.

The neural network is fully connected: all units from the layer (k) has connections with all units at the layer $(k+1)$. Additionally, inputs $\boldsymbol{x}$ are fully connected to the output layer by shortcut connections as it is shown on Fig. 1. For the first hidden layer the activation function is *hyperbolic tangent* $f(a_i) = \tanh(a_i)$ for the second hidden layer it is *ReLU* $f(a_i) = \begin{cases} a_i, if\ a_i \geq 0 \\ 0, if\ a_i < 0 \end{cases}$ and for the

output layer it is *softmax* $f(a_i) = \frac{\exp(a_i)}{\sum_j \exp(a_j)}$. We assume that this neural network will futher be used for classification task in homework. Cross-entropy loss function and one-hot-encoding will be used for this purpose.

Tasks:

1. Write down a neural network's forward pass in scalar and vector forms for single input sample and for minibatch of size *B* using the notations above. You must show the size of each vector or matrix (20% of scores for the homework I).

2. Derive the backpropagation equations for the neural network taking into account different activation functions for each layer and shortcut connections between the input and output layer. You must present deltas in the explicit form for each layer. Please, note that *softmax* is a vector function which has multiple inputs and multiple outputs. Provide the detailed protocol of differentiation process step-by-step and comment it where it is required (70% of scores for the homework I).

3. Write down the backward pass in vector form for single input sample and for minibatch of size *B* using the notations above. You must show the size of each vector or matrix (10% of scores for the homework I).

The homework 1 should be presented as a PDF document.

Homework II (practical, 50% of scores)

The goal of the second (practical) homework is to code and run the derived equations from the first (theoretical) homework. As a result, you have to build the working modified Multilayer Perceptron from scratch with the structure shown on the Fig.1 at the previous Homework. The successful solution must contain a workable example that implements forward pass, backward pass (calculation of derivatives of error function) for the elements of the dataset. It must be trained by the vanilla gradient descent and be evaluated by calculation of values of loss functions for train part of dataset and accuracy for test part in the dataset.

Tasks:

1. Create the neural network as a class (or any structure) which has layers and weight matrices. Implement the weight matrices initialization using the Xavier's trick (10% of scores for the homework II).

3. Implement the forward pass. Taking the minibatch of size B as input, neural network must produce the output $\boldsymbol{y}$ for this minibatch (20% of scores for the homework II).

4. Implement the backward pass in order to calculate the derivatives of loss function for each layer $\frac{\partial E}{\partial \boldsymbol{W}^{(k)}}$. For P -class classification function, cross-entropy loss function is defined as:

$$E(\boldsymbol{w}) = -\sum_{p=1}^{P} t_{np} \ln y_{np}, \tag{3}$$

where n is number of element in dataset, p is current component's number. Vectors $\{\boldsymbol{t}_n, \boldsymbol{y}_n\}$ are encoded using one-hot-encoding.

Implement the correction of weights according to the vanilla gradient descent, $\boldsymbol{W}^{(k+1)}(i+1) = \boldsymbol{W}^{(k)}(i) + \eta \frac{\partial E}{\partial \boldsymbol{W}^{(k)}}$ for each minibatch (20% of scores for the homework II).

5. Prepare the pipeline for training and evaluation of the neural network. The pipeline must have: (i) iterating over T epochs and iterating over all minibatches of size B in the training dataset for each epoch; (ii) also, for each epoch correct the neural network's weights using the train part of dataset; (iv) then, calculate the accuracy in % of correct answers using the test part of dataset (10% of scores for the homework II).

6. Make the evaluation of the built framework. Set the training speed $\eta = 0.01 \ldots 0.001$, minibatch size $B = 10 \ldots 20$, number of epochs $T \sim 10$. Number of units in the hidden layers and all hyperparameters in general is your choice and your luck (40% of scores for the homework II).

6.1. Run the training process and plot the graphs of training loss function (cross-entropy) and test accuracy (% of the correct classification answers) conditioned on epoch number.

6.2. Replace the deltas (local gradients) calculated for the last *softmax* layer according to the derived equations in Homework I with simple resiudals,

$\delta_i^{(3)} = t_i^{(3)} - y_i^{(3)}$, all the rest leave the same as it was in the previous experiment (i.e., ignore derivatives $\frac{\partial softmax(a_i)}{\partial a_j}$ in the backpropagation's chain rule). Ignore the knowledge that the output layer's activation function Is *softmax*, think about it as it was an identity function. Run the training process using this modification, save the training time and loss/accuracy values for train/test datasets. Plot the accuracy graphs and make the comparison.

The approximate script to develop is shown as a text below. You simply need to transform it to python[3] (or another) code.

```
load dataset {x, t}
create the neural network

for epoch = 1:T
   for each minibatch xb_train, tb_train in train dataset
      yb_train = network's forward pass (xb_train)
      loss_function_for_batch = cross_entropy(tb_train, yb_train)
      dw = backpropagate_gradients(tb_train, yb_train)
      neural_networks's w = neural_networks's w + dw
   accuracy = get_accuracy(t_test, y_test)

plot(all loss functions of the final network for all epochs)
plot(all accuracies of the final network for all epochs)
```

[3] For this purpose, using software developed by Anatolii Stehnii or similar automatic text-to-code translators is strongly prohibited.

Datasets for the second homework will be given for each team individually after completing the Homework I.