

## Machine Learning 2 - HW1

Submission date: 09/12/2022

### Theoretical Part (20 points):

1. The Softmax function is used to normalize the output of a neural network  $f(\cdot, w)$  to a probability distribution over predicted output classes.  
For  $n$  classes, denote by  $f(\cdot, w)_i$  the unnormalized output corresponding to the  $i^{th}$  class:

$$\hat{y}_i = \text{Softmax}(x)_i = \frac{\exp^{f(x;w)_i}}{\sum_{j=0}^n \exp^{f(x;w)_j}}.$$

Denote by  $L(y, \hat{y})$  to be the loss function.

Show the derivative of the loss w.r.t the weights. I.e.,  $\frac{\partial L(y, \hat{y})}{\partial w}$ .

### Practical Part:

1. (40 points) In the following exercise, you will create a classifier for the MNIST dataset. You should write your own training and evaluation code and meet the following constraints:
  - You are only allowed to use torch tensor manipulations.
  - You are NOT allowed to use:
    - Auto-differentiation - *backward()*
    - Built-in loss functions
    - Built-in activations
    - Built-in optimization
    - Built-in layers (torch.nn)

The neural network you build should:

- Have at least one hidden layer
  - Obtain at least 75% accuracy on the test set
2. (40 points) In this exercise, we will demonstrate overfitting to random labels. The settings are the following:
    - Use the MNIST dataset.
    - Work on the first 128 samples from the training dataset.
    - Fix the following parameters:
      - Shuffle to False.
      - Batch size to 128.
    - Generate random labels from Bernoulli distribution with a probability of  $\frac{1}{2}$ . I.e., each sample is assigned a random label which is zero or one.

Show that using the network (architecture) from Ex.1 and cross-entropy loss you are able to achieve a loss value of  $\sim 0$  (the lower the better).

Plot the loss convergence for this data and the test data as a function of epochs.

What is the mean loss value of the test data? Explain.

Submission instructions:

Submission **must be individual** and will contain a short (two pages) pdf report containing:

- Model architecture description, training procedure (hyperparameters, optimization details, etc.).
- Convergence plot of accuracy as a function of time (epochs). The plot should depict both training and test performance (i.e. two curves, one for the train, and one for the test).
- A short summary of your attempts and conclusions.

In addition, you should also supply:

- Code (python file) able to reproduce your results - we might test it on different variants on these datasets.
- The trained network with trained weights (.pkl file).  
[the weights tensors can be saved with `torch.save({'w1':w1, 'w2':w2 }, 'path_to_w.pkl')` and load with `torch.load('path_to_w.pkl')`]
- A function called `"evaluate_hw1()`". The function should load the MNIST test-set, load your trained network (you can assume that the data and model files are located in the script folder), and return the average accuracy over the test-set. This function should be written in a separate script.

Moodle submission:

You should submit a Zip file containing:

- Python files for each practical question:
  - Training procedure, file name: `hw1_id_train.py`
  - Evaluation procedure, file name: `hw1_id_eval.py`
- 1 pdf file with
  - Your full name and ID
  - Typed answers for the theoretical part
  - A summary of the practical part
- Pickle file (If the file is too big for the Moodle, upload it to your Google-Drive and copy the link to your pdf report)

Good Luck!