Deep Learning for NLP 2019
Home Exercise 01

TECHNISCHE UNIVERSITÄT DARMSTADT

Due until Tuesday, 23.04. at 13:00

Submission Guidelines for all Home Exercises

- When submitting multiple files, submit one **zip-archive**.
- Submit python code as plain python scripts (**.py**). **Must** be runnable in the given Docker container.
- Submit answers to non-code assignments in **one PDF** file. Scans are permitted, if readable.
- Guidelines specific to neural network code:
  - Please submit your training/testing results (a copy of your console output is fine). Reasoning: Your network might train much slower on the tutor's system than on yours.
  - If you are aware that your network never stops training, please be honest and add a short statement saying so. Thank you!

Problem 1 Setup (1P)

Download the Docker container from Moodle[1] and follow the instructions in the readme file to set it up. Verify that your setup works by running the Keras MNIST example[2]. You don't need to provide evidence to receive points for this task.

Problem 2 Sigmoid Activation Function (2P)

When optimizing functions, first or higher-order derivatives (gradients, Hessians) are of major importance. In neural network learning, we typically want to minimize weight parameters so that the difference between the net output and the true labels is minimized, e.g.:

$$\min_{\mathbf{w}} \sum_{j=1}^{N} \Big(\sigma(\mathbf{x}_j \cdot \mathbf{w}) - y_j\Big)^2$$

Here, σ is an activation function. A frequently used activation function is the *sigmoid* function, defined as:

$$\text{sig}(x) = \frac{1}{1+\exp(-x)}$$

Show that:

$$\text{sig}'(x) = \text{sig}(x) \cdot \big(1 - \text{sig}(x)\big)$$

You may find the chain rule useful: $f(g(x))' = f'(g(x)) \cdot g'(x)$

[1] https://moodle.informatik.tu-darmstadt.de/mod/resource/view.php?id=18931
[2] https://github.com/keras-team/keras/blob/master/examples/mnist_mlp.py

Problem 3 Perceptron Learning by Hand (2P)

A simple perceptron learning algorithm was introduced in the lecture. Here is the weight update rule again for reference:

$$w' \leftarrow w - \alpha \sum_{(\mathbf{x},y)\in\mathcal{T}'} \Big(\sigma(\mathbf{x}\cdot\mathbf{w}) - y\Big)\cdot\sigma'(\mathbf{x}\cdot\mathbf{w})\cdot x^T$$

The weight update rule is designed to minimize the square loss between the perceptron output and the target labels.

Problem 3.1 Training (1P)

Using pen and paper, train a perceptron using the abovementioned learning algorithm on the following training data T:

j	x_1	x_2	y
1	−1.28	0.09	0
2	0.17	0.39	1
3	1.36	0.46	1
4	−0.51	−0.32	0

Perform one training epoch (one iteration over the training data). Apply one weight vector update per data point $(\mathbf{x}, y)$ and report the new weight $w^{(j)}$ after each step. No bias value should be used.
Use the following parameters:

- activation function $\sigma = \text{sig}$ (see Problem 2)
- initial weight vector $w^{(0)} = (-1, 1)^T$
- learning rate $\alpha = 1$

Problem 3.2 Evaluation (1P)

Compute the square loss L before training (using $w^{(0)}$) and after training (using $w^{(4)}$) on the following test data:

j	x_1	x_2	y
1	−0.50	−1.00	0
2	0.75	0.25	1

Square loss is defined as:

$$L = \sum_{j=1}^{N} \ell(\mathbf{x}_j, y) = \sum_{j=1}^{N} (\sigma(\mathbf{x}_j\cdot\mathbf{w}) - y_j)^2$$