Deep Learning **Summer Term 2024**

`http://ml3.leuphana.de/lectures/summer24/DL`
Machine Learning Group, Leuphana University of Lüneburg

Soham Majumder (soham.majumder@leuphana.de)

Exercise 1

Due on: Thursday, 25.04.2024

Task 1 Tensors

Generate the following matrix without using Python loops:

$$\begin{pmatrix}
4 & 5 & 4 & 4 & 4 & 4 & 5 & 4 & 4 & 4 & 4 & 5 & 4 \\
5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\
4 & 5 & 4 & 4 & 4 & 4 & 5 & 4 & 4 & 4 & 4 & 5 & 4 \\
4 & 5 & 4 & 6 & 6 & 4 & 5 & 4 & 6 & 6 & 4 & 5 & 4 \\
4 & 5 & 4 & 6 & 6 & 4 & 5 & 4 & 6 & 6 & 4 & 5 & 4 \\
4 & 5 & 4 & 4 & 4 & 4 & 5 & 4 & 4 & 4 & 4 & 5 & 4 \\
5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\
4 & 5 & 4 & 4 & 4 & 4 & 5 & 4 & 4 & 4 & 4 & 5 & 4 \\
4 & 5 & 4 & 6 & 6 & 4 & 5 & 4 & 6 & 6 & 4 & 5 & 4 \\
4 & 5 & 4 & 6 & 6 & 4 & 5 & 4 & 6 & 6 & 4 & 5 & 4 \\
4 & 5 & 4 & 4 & 4 & 4 & 5 & 4 & 4 & 4 & 4 & 5 & 4 \\
5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\
4 & 5 & 4 & 4 & 4 & 4 & 5 & 4 & 4 & 4 & 4 & 5 & 4
\end{pmatrix}$$

Hint: use `torch.full` and the slicing operator

Task 2 Softmax Function

Let $\mathbf{x} \in \mathbb{R}^d$ be a vector. The softmax function $\sigma : \mathbb{R}^d \to [0,1]^d$ is given by

$$\sigma(\mathbf{x})_j = \frac{\exp(x_j)}{\sum_{k=1}^{d} \exp(x_k)} \quad \text{for } j = 1, \ldots, d.$$

Implement the sigmoid and the softmax functions in PyTorch and compare their outputs for a random vector. What can you say about the highest values, respectively?

Task 3 Empirical Verification of Perceptron Convergence

Consider a perceptron $f(x) = \text{sign}(w \cdot x + b)$ and the following training algorithm:

(1) Initialize w^0, i.e., via $w^0 = \mathbf{0}$ (or randomly).

(2) While there is some $n_k \in \{1, \ldots, n\}$ such that $y_{n_k}(w^k \cdot x_{n_k}) \leq 0$, update $w^{k+1} = w^k + y_{n_k} x_{n_k}$.

Note that this assumes that b is added into w and that x is augmented with a 1.

Assume for every n that:

(1) $\|x_n\|_2 < R$, with some $R > 0$.

(2) There exists a margin $\gamma > 0$ and some non-zero w^* (with $\|w^*\|_2 = 1$) such that $y_n(w^* \cdot x_n) \geq \frac{\gamma}{2}$.

Confirm empirically that training this model will converge in $k \leq \frac{4R^2}{\gamma^2}$ updates by following these steps:

(i) Create a linearly separable toy data set.

(ii) Compute the R and γ values for that data set.

(iii) Implement the perceptron algorithm.

(iv) Initialize the perceptron at least 250 times with random weights, track the number of updates, and confirm that those numbers are smaller than the bound.

Hint: Take a look into `sklearn.datasets` for creating a toy data set. You can compute the margin using an SVM, i.e., `sklearn.svm.LinearSVC`. Experiment with tighter and wider margins. There is a template `code1.py`.

Task 4 The Perceptron on MNIST

(i) Load the MNIST data set, i.e., via `torchvision.datasets.MNIST`

(ii) Visualize some of the numbers.

(iii) Restrict the multi-class classification problem to a binary classification of digit 0 vs digit 1.

(iv) Run the perceptron algorithm on this data set.

(v) Compute the train and test accuracies.

(vi) Show the misclassified input instances.

Hint: Take a look into `code1.py`.