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

Solution 1

See `solution1.py`

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?

Solution 2

See `solution1.py`

Now, regarding the highest values:

- For the sigmoid function, the highest value will be close to 1 because the sigmoid function squashes its input to the range [0, 1]. So, the highest value will be the closest to 1 among all the elements of the sigmoid output.
- For the softmax function, the highest value will depend on the input vector. Softmax function calculates the probability distribution over multiple classes, ensuring that the sum of probabilities is equal to 1. Therefore, the highest value in the softmax output will correspond to the element with the highest score in the input vector relative to the other elements. The softmax function essentially amplifies the differences between the input values, emphasizing the highest value and suppressing the lower ones. So, the highest value in the softmax output will be the one with the highest score in the input vector.

In summary, the highest values in the outputs of the sigmoid and softmax functions reflect different aspects of the input vector: for sigmoid, it's the closest value to 1, while for softmax, it's the element with the highest score relative to others.

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`.

Solution 3

The code is provided as `solution1.py`.

Note that the experimental setup of this task is not really confirming the bound as we broke some assumptions, i.e., the initialization of w.

To get a hyperplane with upper and down boundary, what is the role of `a`? The separating hyperplane is where $w^T x + b = w_1 x_1 + w_2 x_2 + b = 0$. For plotting reasons, we specify x_1 positions and want to know where the corresponding x_2 positions are. Hence, we compute

$$\begin{aligned} & w_1 x_1 + w_2 x_2 + b = 0 \\ \iff & w_2 x_2 = -w_1 x_1 - b \\ \iff & x_2 = -\frac{w_1}{w_2} x_1 - \frac{b}{w_2} \end{aligned}$$

Within the code, `a = -w[0] / w[1]` (which corresponds to $a = -\frac{w_1}{w_2}$) is the slope of the linear function we draw.

What is the purpose of the 1d array `xx` containing 50 numbers from -5 to 5? The variable `xx` contains all x_1 positions for which we compute the x_2 positions derived in the previous question for plotting the separating hyperplane. The limits -5 and 5 are used for drawing purposes. You can see that the hyperplane is only drawn for x values between -5 and 5. Since it is a linear hyperplane, 2 instead of 50 points would have been enough.

What do 0,1 and 1,2 mean after `y==+1` and `y==-1` within `plot()`? The function `plot()` wants to have a x and y value, or lists thereof. Our dataset is contained in `X`, where we have n rows and d columns. To index the points belonging to the positive class we call `y==+1` and obtain a boolean array that selects a subset of points of `X`. From this subset of points, we index the columns with 0 and 1 for the x (x_1) and y (x_2) values, respectively. Later in the code the indices change from 0 and 1 to 1 and 2. This is due to the fact that we added a static 1 column as the first column. Hence, the x_1 and x_2 values are accessible one column further.

How can we interpret the graph of histogram? The histogram shows quickly the distribution of the number of iterations that were needed until the perceptron algorithm converges.

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`.

Solution 4

The code is provided as `solution1.py`.