Assignment 6

Authenticity Verification of Banknotes

Banks often need to verify the authenticity of banknotes. These checks can be viewed as a classification problem: feature vectors represent banknotes and class labels indicate whether the banknote is genuine or counterfeit. In this task, you are to implement and test a perceptron algorithm for verifying the authenticity of banknotes.

Note. Although the task can be easily solved with `numpy` and `scikit-learn`, the goal of the task is to practice the concepts learned in the lecture. Therefore, structure your program using functions and use the methods and techniques we have covered so far.

Data. Use the dataset `data/banknote.csv` for this task. This dataset consists of $1,372$ rows, each row representing a banknote. Each row contains five comma-separated values: The first four values are features of the banknote and the last value indicates the class (authenticity) of the banknote. The class labels are 0 and 1. Use

```
with open(filename, 'r') as f:
    data = f.readlines()
```

to read the file `banknote.csv`, where `filename` is the path to the file.

Exercise 1: Inspecting the Data

Write a notebook that consists of three functions:

1. `read_csv`: the function takes a file name as parameter and optional additional parameters. It reads the data table from the file and returns it values.

2. `head`: the function accepts the data table returned by `read_csv` and a maximum number n of rows as parameters. It displays the top n rows in the following format (for $n = 5$):

```
    x1      x2      x3      x4      y
 3.622   8.666  -2.807  -0.447  0.000
 4.546   8.167  -2.459  -1.462  0.000
 3.866  -2.638   1.924   0.106  0.000
 3.457   9.523  -4.011  -3.594  0.000
 0.329  -4.455   4.572  -0.989  0.000
```

3. `info`: the function accepts the data table returned by `read_csv` as parameter. It displays the minimum, maximum and average of each column in the following format:

```
feature       min     max     avg
x1      :   -7.042   6.825   0.434
x2      :  -13.773  12.952   1.922
x3      :   -5.286  17.927   1.398
x4      :   -8.548   2.450  -1.192
y       :    0.000   1.000   0.445
```

For fancy output formatting, please see the documentation on Input and Output.

Exercise 1: Data Inspection

Create a notebook with three functions:

1. `read_csv`: Accepts a file name (and optional parameters), reads the data table from the file, and returns its values.

2. `head`: Accepts the data table from `read_csv` and a number n , and displays the top n rows in the given format (for n=5):

   ```
   x1       x2       x3       x4        y
   3.622    8.666   -2.807   -0.447    0.000
   4.546    8.167   -2.459   -1.462    0.000
   3.866   -2.638    1.924    0.106    0.000
   3.457    9.523   -4.011   -3.594    0.000
   0.329   -4.455    4.572   -0.989    0.000
   ```

3. `info`: Accepts the data table from `read_csv`, and displays the min, max, and avg of each column in the given format:

   ```
   feature       min     max     avg
   x1      :   -7.042   6.825   0.434
   x2      :  -13.773  12.952   1.922
   x3      :   -5.286  17.927   1.398
   x4      :   -8.548   2.450  -1.192
   y       :    0.000   1.000   0.445
   ```

For output formatting, refer to the Python documentation on Input and Output.

Exercise 2: Perceptron Algorithm

Perceptron. The perceptron algorithm described here assumes class labels ± 1. A feature vector is denoted by $\boldsymbol{x} \in \mathbb{R}^d$ and the associated class label by $y \in \{\pm 1\}$.

A perceptron with weight vector $\boldsymbol{w} \in \mathbb{R}^d$ and bias $b \in \mathbb{R}$ classifies a point $\boldsymbol{x} \in \mathbb{R}^d$ to class y according to the rule

$$f_{\boldsymbol{w},b}(\boldsymbol{x}) = \begin{cases} +1 & : \quad \boldsymbol{w}^{\mathsf{T}}\boldsymbol{x} + b \geq 0 \\ -1 & : \quad \text{otherwise} \end{cases} ,$$

where $\boldsymbol{w}^\mathsf{T}\boldsymbol{x} = w_1x_1 + \cdots + w_dx_d$ is the scalar product of $\boldsymbol{w}$ and $\boldsymbol{x}$. The prediction $f_{\boldsymbol{w},b}(\boldsymbol{x})$ is correct if $f_{\boldsymbol{w},b}(\boldsymbol{x}) = y$. Otherwise, there is a misclassification.

The goal of learning is to find suitable parameters $\boldsymbol{w}$ and b so that the learned function $f_{\boldsymbol{w},b}$ misclassifies as few training examples as possible. The perceptron algorithm describes how suitable parameters $\boldsymbol{w}$ and b can be learned.

Perceptron Algorithm

1. Initialize the weight vector $\boldsymbol{w} \in \mathbb{R}^d$ and bias $b \in \mathbb{R}$
2. Repeat `max_iter` times
   (a) For each training example $(\boldsymbol{x}, y)$:
      i. calculate $\hat{y} = f_{\boldsymbol{w},b}(\boldsymbol{x})$
      ii. update $\boldsymbol{w}$ and b according to the rule

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta(y - \hat{y})\boldsymbol{x}$$
$$b \leftarrow b + \eta(y - \hat{y})$$

   where $\eta \in [0, 1]$ is the learning rate (step size).
   (b) Output the classification accuracy over all training examples
3. Return $\boldsymbol{w}$ and b.

In **Step 1**, the weights $w_1, \ldots, w_d$ and the bias b are initialized with random values between $[-\sigma, +\sigma]$. Use the function

```
random.uniform(a, b)
```

of the `random` module with $a = -\sigma$ and $b = \sigma$. In **Step 2**, the algorithm is repeated for a given number of iterations (`max_iter`). In each run, all training examples are visited, the prediction is calculated, and the weight vector and bias are adjusted. After each run, the classification accuracy on the training set is calculated and output. In **Step 3**, the learned weight vector and bias are returned.

Task: Write a notebook to implement the perceptron algorithm. The Jupyter notebook

```
perceptron.ipynb
```

contains some imports that may be useful. Structure your program into suitable functions. Test your program with the banknote dataset `banknote.csv`. To do this, partition the dataset randomly into a training and test set, with the test set comprising 20% of all example data.
You can use the following parameter values:

```
sigma = 0.1
num_epochs = 10
eta = 0.1
```