Assignment 4

Prof.	Panos Trahanias	trahania@csd.uoc.gr
T.A.	Despina - Ekaterini Argiropoulos	despargy@csd.uoc.gr
T.A.	Myrto Villia	mvillia@ics.forth.gr
T.A.	Michalis Savorianakis	csdp1387@csd.uoc.gr
Deadline	**Friday 23:59, 16th of May 2025**	

Note: This is a personal assignment and should be pursued individually and without use of any computerised AI facilities. Note that this will be automatically checked for every delivered assignment and for cases of relevant detections the whole assignment will be dropped. The assignment should be implemented entirely in Google Colaboratory following the delivered instructions below. **Comments regarding the submitted code are mandatory.**

Note: The pseudo-code and the theory for the following algorithms are written in the book's fifth chapter uploaded on the course's website.

Exercise 1: Single-sample Perceptron (50/100)

Data: For this exercise, you will use the dataset in the *dataset.csv* file that consists of 2000 2-dimensional data samples and their labels.

Equation:

$$learning\_rate_i = 1/\sqrt{k_i} \tag{1}$$

where $k_i = k_{i-1} + (i \mod n)$, $k_0 = 0, i >= 1$, i = iteration index, n = number of samples.

Notice: You are not allowed to use library functions; except for matrix multiplication and plotting, everything else should be developed from scratch.

Questions:

1. Create a scatter plot of the dataset. Is the dataset linearly separable? Justify your answer in 1-2 sentences.
2. Create a function called 'train_single_sample' that implements the Fixed-Increment Single-Sample Perceptron algorithm. The arguments of the function should be:
   (a) a := weights + bias (bias trick)
   (b) y := data + '1's (bias trick)
   (c) labels
   (d) n_iterations := the number of iterations or updates
   (e) lr := learning rate
   (f) variable_lr := boolean

   and the function should return:
   (a) a := trained model
   (b) acc_history := list of accuracy values at every iteration

   Call the function 'train_single_sample' to train a linear model and print the trained model. Also, print the model's accuracy every *n_samples* iterations. Use the following hyperparameters:
   (a) n_iterations = 30001
   (b) lr = 1
   (c) variable_lr = False (Set it False for now, you will set it True in Question 5).

Bonus 7% added to the total assignment grade: To implement the above function, you used a convention: **score** $>= 0$ **→ predicted class a** and **score** < 0 **→ predicted class b**. There are four possible (prediction, true label) combinations. List all of them. Identify which of these combinations require an update. For the cases where an update is needed, describe the update rule and explain why it is applied. Explain your answer. Could any label values be used instead of a and b?

3. Plot the history of the accuracy during training.
4. Create a function called 'plot_model' that takes as input the trained weights (+ bias), the data, and the labels and returns a scatter plot with the decision boundaries of the model (**Hint**: you can use plt.contour()). Call the function you implemented to plot the data along with the trained model.
5. Now we are going to retrain our model but with a variable learning rate. Create a 'Scheduler' class that implements a function 'get_next_lr' that every time it is called returns the next learning rate. The object should work according to the Equation at the beginning of the assignment. Now configure the training function 'train_single_sample' to use this object when 'variable_lr = True'. Retrain the model with a variable learning rate. Plot the dataset and the trained model as before using the function 'plot_model' . What is the main difference compared to training with a fixed learning rate? What method do you think is better? Justify your answer.

Exercise 2: Batch Perceptron (50/100)

The **data**, the **equation**, and the **notice** from Exercise 1 apply to this one too.

Question:

1. Create a function called 'train_batch' that implements the Batch Perceptron algorithm. The arguments of the function should be:
   (a) a := weights + bias (bias trick)
   (b) y := data + '1's (bias trick)
   (c) labels
   (d) theta := the value for the theta criterion
   (e) batch_size
   (f) lr := learning rate
   (g) variable_lr := boolean

   and the function should return:
   (a) a := trained model
   (b) acc_history := list of accuracy values
   (c) error_history := list of error values

   The function should print the model's accuracy and the error at every iteration. The error here is the sum of the absolute values of the updates.
2. Train a linear model using the function 'train_batch' you have implemented. Use the following hyperparameters: theta = 0.01, batch_size = 16, lr = 1, variable_lr = False. Print the trained model.
3. Plot the history of the accuracy and the error during training.
4. Plot the data and the trained model. Use the 'plot_model' function you implemented in Exercise 1.
5. Retrain the model with a variable learning rate. Use the Scheduler you implemented in Exercise 1. Plot the dataset and the trained model using 'plot_model'. What is the main difference compared to training with a fixed learning rate? What method do you think is better? Justify your answer. Discuss the expected behavior of the two algorithms: the Fixed-Increment Single-Sample Perceptron and the Batch Perceptron algorithm.

Bonus 3% added to the total assignment grade: Use only two for/while loops throughout the whole assignment.

Deliverable

This assignment should be implemented entirely in Google Colaboratory. Google's notebook allows you to combine executable Python scripts with rich text in a single document. Your deliverable should be a single '.ipynb' file along with its corresponding .py file (both can be easily exported from Google Colaboratory). Every single question should be implemented in a single code block. Code blocks should be clearly and shortly explained (you may use the text boxes for that goal). Use **only** library functions for matrix operations and plots.