The Intro to Computer Vision labs will be run in Google Colaboratory, a Jupyter notebook environment that runs entirely in the cloud, you don't need to download anything. To run these labs, you must have a Google account.

Step 1: click on the assignment invite link -> **Accept this assignment**. Refresh page -> individual repo for the specific assignment is created automatically

Project 2: 2 weeks -> https://classroom.github.com/a/8-LkZQJb

Step 2: Navigate to http://colab.research.google.com/github -> Click the **Include Private Repos** checkbox -> **select the correct repo** (SistemeDeVedereArtificiala/assignment_name-student_name) -> Click on the jupyter notebook of the current assignment

Step 3: [GitHub sign-in window] In the popup window, sign-in to your Github account and authorize Colab to read the private files.

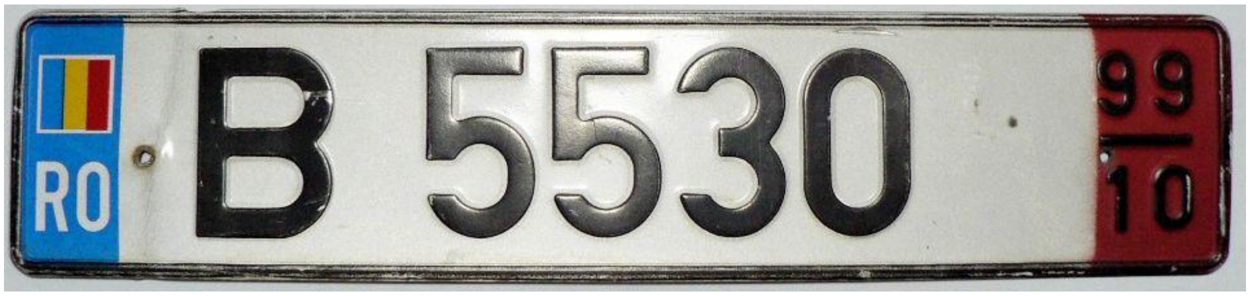
Step 4: [in colab] **File** -> **Save a copy to GitHub**. Select the correct repository for the SPECIFIC assignment -> Click the **Include Colab Link** -> Click **OK**

Step 5: [in colab] Navigate to the **Runtime** tab --> **Change runtime type**, under **Hardware accelerator** select **GPU/TPU** (tensor processing unit) according to your needs.

Read the suggestions and accomplish all tasks marked with **#TODO**.

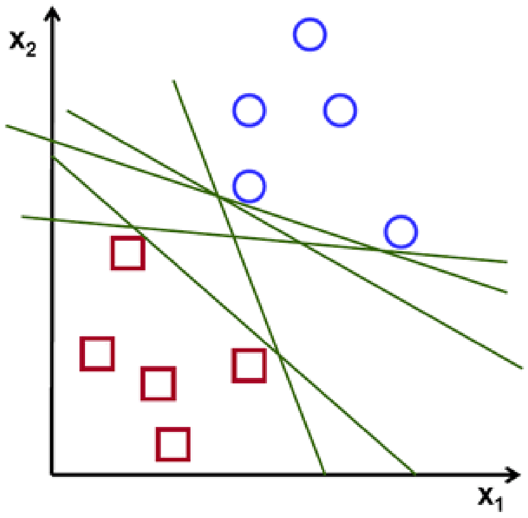
!!! At the end of each laboratory **REPEAT step 4 in order to SAVE** the answers to your private repository (individual for each assignment)

Project 2: License plate text recognition



Label:B
Prediction output: B

1. Classification using SVM



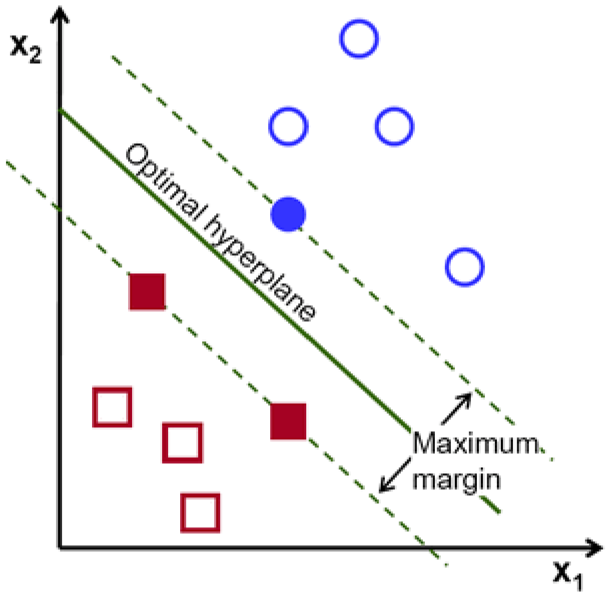
A **Support Vector Machine (SVM)** is a discriminative classifier formally defined by a separating hyperplane. In other words, given **labeled training data (supervised learning)**, the algorithm outputs an optimal hyperplane which categorizes new examples.

In which sense is the hyperplane obtained optimal? Let's consider the following simple problem: For a linearly separable set of 2D-points which belong to one of two classes, find a separating straight line.

In the above example we deal with lines and points in the Cartesian plane instead of hyperplanes and vectors in a high dimensional space. This is a simplification of the problem. It is important to understand that this is done only because our intuition is better built from examples that are easy to imagine. However, the same concepts apply to tasks where the examples to classify lie in a space whose dimension is higher than two

In the above picture you can see that there exists multiple lines that offer a solution to the problem. Is any of them better than the others? We can intuitively define a criterion to estimate the worth of the lines: A line is bad if it passes too close to the points because it will be noise sensitive and it will not generalize correctly. Therefore, our goal should be to find the line passing as far as possible from all points.

Then, the operation of the SVM algorithm is based on finding the hyperplane that gives the largest minimum distance to the training examples. Twice, this distance receives the important name of margin within SVM's theory. Therefore, the optimal separating hyperplane maximizes the margin of the training data.



How is the optimal hyperplane computed? Let's introduce the notation used to define formally a hyperplane:

$$f(x)=\beta_0+\beta^T x,$$

where β is known as the *weight vector* β0 as *the bias*.

The optimal hyperplane can be represented in an infinite number of different ways by scaling of β and β_0. As a matter of convention, among all the possible representations of the hyperplane, the one chosen is:

$$|\beta_0+\beta^T x| = 1,$$

where ***x*** symbolizes the training examples closest to the hyperplane. In general, the training examples that are closest to the hyperplane are called support vectors. This representation is known as the canonical hyperplane.

Now, we use the result of geometry that gives the distance between a point x and a hyperplane (β, β0):

$$distance = \frac{|\beta_0+\beta^T x|}{||\beta||}.$$

In particular, for the canonical hyperplane, the numerator is equal to one and the distance to the support vectors is:

$$distance_{support\ vectors} = \frac{|\beta_0+\beta^T x|}{||\beta||} = \frac{1}{||\beta||}.$$

Further study on the topic at:

1. OpenCV Documentation: https://docs.opencv.org/4.1.2/d1/d73/tutorial_introduction_to_svm.html
2. Elements of Statistical Learning, Section 4.5 (Separating Hyperplanes): https://web.stanford.edu/~hastie/Papers/ESLII.pdf