Ex2 Report

Eliran Darshan - 311451322

Afik Aharon - 301494670

In this report we will explain how we chose our hyper – parameters, how did we choose the number of epochs for the algorithms and which value we chose for the lambda variable in the SVM algorithm.

First and foremost, because checking the number of iterations is quiet a challenging job (running all the algorithms in a loop and finding the best value that gives us the best result – very long and complex operation) we broke down the iterations by segments of '5'.
We concluded that when the algorithms receive the next values, the loss function is minimized (and the runtime is quiet fast). Perceptron : 15 epochs, SVM: 10 epochs, PA: 30 epochs.

- **Perceptron:**

  The default values , if no parameters are given are:
  Number of iterations(epochs) – 100.
  Learning rate (eta) – 0.01

  After testing the algorithm with various values , we concluded that the best values should have the learning rate of 0.01 and epochs (iterations number) should be 13

- **SVM**

  The default values, if no parameters are given are:
  Number of iterations(epochs) – 100
  lambda – 0.01
  learning rate – 0.01

  Here we noticed that the best fit for the lambda and the learning rate should both be 0.01 and the epochs number should be 10

- **PA**

  The default values, if no parameters are given are:
  Number of iterations(epochs) – 100

  For the PA algorithm, we implemented as it was presented at class, as for updating the weights we first calculated the loss function, then the theta value. The number of epochs is 22

- **Note**

  The most challenging values we had to choose were the MAX and MIN of the normalization function, as we had to guess for the values which could optimize the normalization.
  We used the min-max normalization for the PA and Perceptron as explained in at class:

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A$$

  At the moment we found all the other values (which are noted above) ,we started playing with the MAX and MIN values, first with small range (initial range was between 1 and -1), and got to the point the best values were:

  MAX : 3, MIN : -2

  As for the SVM algorithm, we used the Z-Score normalization function: (better results for lower number of iterations)

$$v' = \frac{v - mean_A}{stand\_dev_A}$$

  As for the the seed array (which will help the random function in the np), we tried serveral ways to generate it properly, (for example, giving it values from 1 to 1000). The best fit in the end was multiplying each iteration by 53, and that helped us lower the loss function.