Report Lab01

Emanuele Falci, Simone Varriale

27/11/2023

1 Part I - Implementing the perceptron

The perceptron is a model used to perform binary classification over a dataset. It relies on two assumptions:

- Binary classification: $y_i \in \{-1, 1\}$
- The dataset is linearly separable

The decision boundary is mathematically defined by the hyperplane $H = \{x : w^T x + b = 0\}$ where the vector w contains the weight of each feature and b is the bias. It is possible to simplify that expression including the bias in the weight vector and adding one fake feature to the dataset.

The training phase consists of finding the hyperplane that divides the samples in the space, basically we find the weigths that determin the normal vector of the plane. Then the classification is made in base of the position of the data with respect to the hyperplane.

Regarding the implementation of the model, we need two functions: one to train the model which takes as a parameter the learning rate α, and one to predict $\hat{y}$ given a dataset.
To train the model we initialize our weigths w, this could be done either using a vector of all zeros or initializing it randomly. Then, we check if every point of the dataset is correctly classified through the perceptron criteria. If a point is not correctly classified, we apply a learning rule to update the weights accordingly to the error and we try again until the plane correctly divides the two cluster.
Mathematically speaking the learning rule is $w^{(t+1)} = w^{(t)} - \alpha \nabla E_p$ where $E_p(w) = -\sum w^T x_i y_i$ is the perceptron criterion and calculates the amount of misclassified data we would have using the hyperplane defined by w.
N.B. we did not implement a maximum number of iteration since we assumed to work only with linearly separable dataset (it is prone to infinite loop in case of wrong input).

To predict the class of a dataset, we simply check on which side of the hyperspace the data is. This is done exaclty as explained before to obtain $y_i = h(x_i) = sign(w^T x + b)$.

2 Part II - Using the perceptron

We used the perceptron model to classify a dataset made of images of handwritten digits (0 and 1). It is taken starting from a subset of the dataset given by the "*load_digits*" function in scikit-learn. We utilized the parameters "*returnXy = true*" and "*n_class = 2*" to get only the X and y vectors including data of the two first digits only.

Then, we split the dataset into training, validation and test ($60\% - 20\% - 20\%$) through the *train-test-split* function, as seen in the first lab.

We firstly trained the model with alpha $\alpha = 0.0001$ and with "*initial_weights*= 0". In that way we were able to confront our model checking that the prediction on the validation set gives the same result of the scikit-learn's model.

Then we train different models with different alphas and a random initial vector, and we check the results on the prediction of the validation set. Since different models give the same accuracy, we computed the margin with a normalized vector and took the model with the smallest margin in order to be less prone to error, however it is not a good way to choose the model since it still depends on the specific validation set you have. This is possible because the dataset is not too big, otherwise it would be computationally heavy.

Lastly we checked the accuracy of our model on the test set, and confronted it with the accuracy of the scikit model. The chosen model has an $\alpha = 0.001$ and the weights are saved in *experiment.ipynb* We obtained in most cases an accuracy of 100%.