Machine Learning - Exercise 2

The Perceptron algorithm

Goal of the exercise

The goal of this exercise is to implement the perceptron algorithm for binary linear classification on two artificial datasets `linear.data` and `nonlinear.data` which are provided in the archive.

The file `Perceptron.py` imports numpy and matplotlib, loads a dataset (by default `linear.data`) and separates it into positive and negatives classes.

```
from numpy import *
import matplotlib.pyplot as plt

# Load the data set
data = loadtxt('linear.data')

# Separate the input from the output
X = data[:, 0:-1]
Y = data[:, -1]
N, d = X.shape

# Separate the positive from the negative class
positive_class = X[Y==1., :]
negative_class = X[Y==-1., :]
```

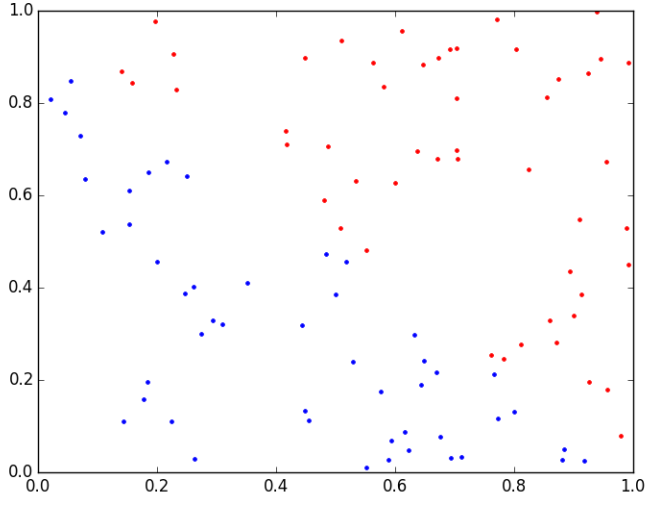
1. Print the different data to understand their structure.

Visualizing the data

The input space has two dimensions, so it can be visualized easily.

2. Write a visualization method using matplotlib to display the positive and negative classes in two different colors.

3. Check if the two datasets are linearly separable or not.



Reminder: `plt.plot(X, Y, '.r')` allows to plot two array against each other with red dots. The color can be `r` for red, `b` for blue, `g` for green, `k` for black, etc. Don't forget to call `plt.show()` at the end.

Learning for one epoch

We can now apply the online version of the perceptron algorithm on a single iteration over the training set.

4. Initialize the weight vector and bias to 0 (using a numpy array for w). The learning rate η is 0.1 at first.

5. Apply the perceptron algorithm for a single epoch:

    - for each example $(\mathbf{x}_i, y_i)$ do:

        1. compute the prediction $f_{\mathbf{w}}(\mathbf{x}_i) = \mathrm{sign}(\langle \mathbf{w}. \mathbf{x}_i \rangle + b)$

        2. update the weight vector $\Delta \mathbf{w} = \eta \cdot (y_i - f_{\mathbf{w}}(\mathbf{x}_i)) \cdot \mathbf{x}_i$

        3. update the bias (find the rule!)

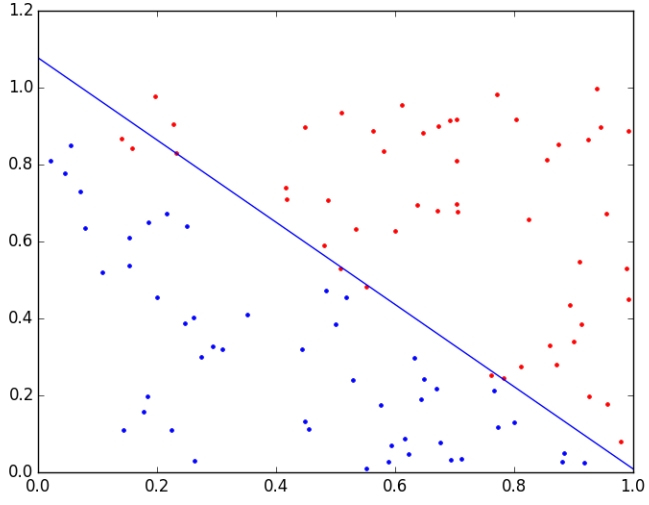
    Print everything during the loop to check you have the correct dimensions. Use `np.dot()` for the scalar product.

6. What is the weight vector at the end?

Visualizing the hyperplane

7. Modifying the visualization method so that it displays the hyperplane corresponding to the weight vector and bias together with the data.

8. Display the hyperplane at the end of the epoch. Is learning correct?



Measure the performance

9. Write a method that returns for a particular hyperplane $(\mathbf{w}, b)$:

    1. The error rate (proportion of misclassified examples) on the training set.
    2. The functional margin of the training set:

$$\hat{\gamma} = \min_i \hat{\gamma}_i = \min_i y_i \cdot (\langle \mathbf{w}. \mathbf{x}_i \rangle + b)$$

Hint: `min(a, b)` returns the minimum between two values `a` and `b`.

10. What is the error rate and functional margin after one epoch of learning?

Complete Perceptron algorithm

11. Modify your algorithm so that it iterates until the functional margin is positive.

```
while True:
    # Do something...
    if condition:
        break # exit the infinite loop
```

12. How many epochs do you need to correctly classify the data? Print all the relevant information for each epoch.

13. Vary the learning rate of the algorithm between extreme values (e.g. 0.00001, 0.1 and 0.9). Does it change the number of epochs needed to converge? Does it change the norm of the weight vector (print it after each epoch)? What do you conclude on the role of the norm of the weight vector in binary classification?

14. Use a different initialization for the weight vector (e.g. (1, 1)). Vary again the learning rate of the algorithm. Does it change anything? Why?

Non-linear classification

15. Apply your algorithm on `nonlinear.data`. Does your algorithm converge? Why?

16. Modify your algorithm so that it stops when either:

    1. The error rate falls below a threshold.

    2. The geometric margin is higher than a fixed threshold:

$$\gamma = \frac{\hat{\gamma}}{||\mathbf{w}||}$$

    3. The variation of the weight vector is smaller than a percentage:

$$||\Delta\mathbf{w}|| < \alpha \cdot ||\mathbf{w}||$$

    Implement only one of these methods. You have to find sensible values for the thresholds and percentages by yourself.

17. Does the result of linear classification on this non-linearly separable dataset make sense?

Logistic regression

In logistic regression, the output of the classifier uses the logistic function instead of the sign function:

$$f_{\mathbf{w}}(\mathbf{x}_i) = \sigma(\langle \mathbf{w}. \mathbf{x}_i \rangle + b)$$

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

The learning rule stays the same (see the course):

$$\Delta \mathbf{w} = \eta \cdot (y_i - f_{\mathbf{w}}(\mathbf{x}_i)) \cdot \mathbf{x}_i$$

18. Modify your algorithm to perform logistic regression on the linear and non-linear datasets.

    Beware: The output y_i of the negative class used in the learning rule should be 0, not -1 as previously.

19. What does it change to the speed of convergence? Why?