TÉCNICO LISBOA

MEIC 2020/2021
Aprendizagem - Machine Learning
Homework II
Prof. Andreas Wichert
Deadline 03/05/2021
Submit on Fenix as pdf

Submission instructions:

Submit a single .zip file on Fenix containing:

- 1 pdf with a 2 page report
- 1 with the working code for Exercise 2

Every file name must be the group number

You can install all the libraries using https://www.anaconda.com, or you can use google colab, https://colab.research.google.com/notebooks/intro.ipynb

Exercise 1 (2 pts)

Use the famous dataset that contains 150 iris flowers to detect Iris-Virginica or Not Iris-Virginica (**two classes**). Perform Logistic Regression with Scikit-learn using the code snippet provided below. Indicate the training and test accuracy (fraction of correctly classified examples).

Example of code using NumPy and Scikit-learn (https://scikit-learn.org/stable/):

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np

iris = datasets.load_iris()
print("Iris Data\n",list(iris.keys()))

X = iris.data[:, :2]  # we only take the first two features.

y = (iris["target"] == 2).astype(np.int) # 1 if Iris-Virginica, else 0

print(y)

# Split into a train and test partition.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=0)

from sklearn.linear_model import LogisticRegression

log_reg = LogisticRegression()

log_reg.fit(X_train, y_train)

print("Predict (train):", log_reg.predict(X_train[:len(y_train), :]))
print("Predict (test):", log_reg.predict(X_test[:len(y_test), :]))
```

TÉCNICO LISBOA

MEIC 2020/2021
Aprendizagem - Machine Learning
Homework II
Prof. Andreas Wichert
Deadline 03/05/2021
Submit on Fenix as pdf

Exercise 2 (18 pts)

Use the same Iris dataset (**two classes**) to investigate **gradient descent** of one unit (perceptron). Implement the perceptron in Python and NumPy (without Scikit-learn). You can still use the code snippet of the previous exercise **but only to load and split the data**. You will try three configurations:

i. Squared error and linear unit (Linear Regression)
ii. Squared error and sigmoid unit
iii. Cross-entropy error and sigmoid unit (Logistic Regression)

$$\sigma(net) = \frac{1}{1+e^{(-net)}} = \frac{e^{(net)}}{1+e^{(net)}}$$

The sigmoid (logistic) activation function.

a) (3 pts) Describe the learning rule for each example i, ii, iii, implement it using Python and NumPy (not Scikit-learn).

b) (4 pts) For each model perform three experiments with different random initialization of weights with a fixed learning rate (does it have any consequences?)

c) (4 pts) Which of the three models i, ii, iii converge faster to a near-optimal solution? Indicate the number of steps/epochs related to the learning rate (three experiments for each model). For the best choice of learning rate for each model, plot the error curves as a function of the number of epochs. You can use matplotlib for plotting, to do it store the data into an array A:

```
import matplotlib.pyplot as plt

plt.plot(A)

plt.show()
```

d) (4 pts) Which model is the most accurate? Report the training and test accuracies. Are the models equivalent or not and why do you think so?

e) (3 pts) Do you get the same results as Logistic Regression of Scikit-learn (Exercise 1)?