Advanced Techniques in Machine Learning

Experience 1
Due Date: 29/11/2018
Yossi = {Keshet, Adi}

In this exercise you will implement and compare several different multiclass classification techniques with Error-Correcting Output Codes.
You will implement binary SVM models with hinge-loss (*detailed explanation of the objective and pseudo code can be found in the Appendix page) using:

i) One vs. All
ii) All Pairs
iii) Your own code Matrix (can be random)

Using two different decoding functions:

i) Hamming Distance
ii) The loss value

In both settings you will classify new examples based on the row with the minimum value.
Over all you will need to report results for **six** different experiments.

Notice, when implementing One vs. All you are likely to encounter ties between classes. In that case, choose the class with the lowest index, e.g. let's say we have a tie between classes 0 and 2, we will classify the example as 0.

Dataset:
For this exercise you will use MNIST dataset with classes 0, 1, 2, and 3.
Code for getting the data:

```
from sklearn.datasets import fetch_mldata
from sklearn.utils import shuffle

# read data
mnist = fetch_mldata("MNIST original", data_home="./data")
eta = 0.1
X, Y = mnist.data[:60000] / 255., mnist.target[:60000]
x = [ex for ex, ey in zip(X, Y) if ey in [0, 1, 2, 3]]
y = [ey for ey in Y if ey in [0, 1, 2, 3]]

# suffle examples
x, y = shuffle(x, y, random_state=1)
```

What to submit?

1. You will be provided with a test file called test.txt (you can load is using numpy package with the following code: x_test = np.loadtxt(path)), you are required to provide **six** different prediction files named:
    a. test.onevall.ham.pred
    b. test.allpairs.ham.pred
    c. test.randm.ham.pred
    d. test.onevall.loss.pred
    e. test.allpairs.loss.pred
    f. test.randm.loss.pred

2. A brief PDF report with your conclusion from the comparison between the methods. i.e. who performed best, what is the prefer decoding function and on what scenario, what is the role of the distance in rows for each code matrix.
3. Python3.6 code of the project.

Appendix

SVM Objective:

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \ \frac{1}{m}\sum_{i=1}^{m} \max\{0, 1 - y_i\, \boldsymbol{w}\cdot\boldsymbol{x}_i\} + \frac{\lambda}{2}\|\boldsymbol{w}\|^2.$$

Pseudo code:

```
INPUT: training set $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$, parameter $\lambda$, $\eta_0$
INITIALIZE: $\boldsymbol{w}_0 = \mathbf{0}$
FOR: $t = 1, \ldots, T$
    • choose example: $(\boldsymbol{x}_i, y_i)$ uniformly at random
    • set $\eta_t = \eta_0/\sqrt{t}$
    • if $1 - y_i\, \boldsymbol{w}\cdot\boldsymbol{x}_i \geq 0$
          $\boldsymbol{w}_t = (1 - \eta_t\lambda)\, \boldsymbol{w}_{t-1} + \eta_t y_i\, \boldsymbol{x}_i$
      else
          $\boldsymbol{w}_t = (1 - \eta_t\lambda)\, \boldsymbol{w}_{t-1}$
OUTPUT: $\boldsymbol{w} = \sum_{t=1}^{T} \boldsymbol{w}_t$
```