EPFL
ENAC TRANSP-OR
Prof. M. Bierlaire
Gael Lederrey & Nikola Obrenovic

Decision Aids
Spring 2018

DATA MINING I - CLASSIFICATION - EXERCISES

For the classification exercises, you will work with two artificial datasets. These two datasets have been created so that they have a linear separation and a non-linear separation. You goal is to implement two algorithms (Logisitic Regression and k-NN) and test them on these two datasets. You can change the datasets by commenting lines 13-14 and uncommenting lines 16-17 in the files `exercise1.m` and `exercise2.m`.

Exercise 1

In this exercise, you will implement the logistic classification algorithm. Here are the steps you need to follow:

1. Run the file `exercise0.m` to have a look at the artificial data you will be using for this exercise set.

2. In the file `computeCostAndGrad.m`, you need to compute the cost and the gradient. The cost is given by:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(h_\theta(x_i)) + (1 - y_i) \cdot \log(1 - h_\theta(x_i))$$

   where x_i is the vector of feature for sample i, y_i is the label for sample i, N is the number of samples, and $h_\theta(x)$ is the hypothesis function. In our case, we have a sigmoid function σ and $h_\theta(x) \equiv \sigma(\theta \cdot x)$ where θ corresponds to the weights vector.

   The k-th element of the gradient is defined by:

$$(\nabla\mathcal{L})_k = \frac{\partial\mathcal{L}}{\partial\theta_k}$$

   You need to compute the derivation by yourself.

   Hints:

   - *The derivation of the sigmoid function is given by:*

$$\frac{d\sigma(x)}{dx} = \sigma(x) \cdot (1 - \sigma(x))$$

   - *You can already run the file `exercise1.m` at this point to verify if your implementation is correct. However, it will throw an error at line 49. Nevertheless, you should still be able to see the separation between the data.*
   - *You can implement it with 5 lines of code.*

3. The final step is to fill the file `predict.m` with the function prediction that will be used. Once it is done, you can run the file `exercise1.m` and get the accuracy of your classification.

   Hint:

   - *Do not forget to use the sigmoid function σ for the prediction and round the values.*
   - *You can implement it with 3 lines of code.*

Exercise 2

In this exercise, you will implement the k-Nearest Neighbours algorithm. You can find the pseudo-code for this algorithm in Algorithm **??**.

Algorithm 1: k-NN algorithm

Result: Returns the predicted labels on the test data based on the train data
Input : *train, labels, test, k*
Output: *predicted_labels*

```
1 Prepare the arrays to return the results
  /* Your implementation starts here                                    */
2 foreach i ← 1 to #test_samples do
3 |   Compute distance between test sample i with all train samples
4 |   Sort the distances from smallest to largest
5 |   Retrieve the k closest train samples from test sample i
6 |   For each unique label, compute the number of times it appears in the k neighbours
7 |   Set the predicted label as the label that is the most represented amongst the k neighbours
8 end
```

You have to implement this algorithm in the file `knn.m`. Once it is done, you can test the algorithm by using the file `exercise2.m`. Do not forget to add (and change) the number of neighbours on line 28 of the file `exercise2.m`.

Hint:

- *The function* `hist` *from Matlab is useful to know which labels appear the most.*
- *You can implement this with 6 lines of code.*