TP 2 Data Mining: k-Nearest Neighbors
Obligatory

Tuesday 30th March, 2021
deadline: Monday 19th April, 2021, 23:59

The goals of this TP are to understand the importance of a vectorized code with numpy, to implement and apply a k-Nearest Neighbor (kNN) classifier, to understand the difference among the Euclidean, the Machalanobis, and the Manhattan distances and to see the effect of a transformation in the unit circle.

In this TP you are going to fill a few missing functions in the python scripts [1] to implement the exercises that we ask. So first of all read and understand the given python scripts. To run your code you have to run the `main_knn_linear_transformations.ipynb` notebook. Here you have to write only a short code (it is mentioned where) to run the kNN algorithm for different k and distances (exercise 2). For the rest exercises the code is given and it works if the missing functions are correctly implemented
You are going to use the *CIFAR-10* and the *iris* data set.

Exercise 1

In this exercise you are going to see how important is to vectorize your code using numpy. To do that you will compute the Euclidean distance with 3 different ways. Using 2 for loops, 1 for loop and without for loops and you will run your algorithm for each case using the *CIFAR-10* data set to compare the time performance (using Euclidean distance and $k = 5$).

Exercise 2

Now you will check and comment on how the performance changes with respect to the values of k. How the number of the neighbors influences the classification? Run your algorithm using the *iris* data set for $k = [1, 3, 5, 10, 25, 50]$. For

[1]Part of the given code is based on Stanford's repository

each k use the Euclidean, the Mahalanobis, and the Manhattan distance.

The *Euclidean* distance between two learning instances $\mathbf{x}_i \in \mathbf{R}^d$ and $\mathbf{x}_j \in \mathbf{R}^d$, where d is the feature (attribute) dimension, is defined as:

$$d_2(\mathbf{x}_i, \mathbf{x}_j) = \left[\sum_{k=1}^{d}(x_{ik} - x_{jk}^2\right]^{1/2} = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T(\mathbf{x}_i - \mathbf{x}_j)}$$

The *Manhattan* distance between two learning instances is defined as:

$$d_1(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{d} |\mathbf{x}_{ik} - \mathbf{x}_{jk}|$$

The *Mahalanobis* distance, is parametrized by a $d \times d$ covariance matrix Σ, and is defined as:

$$d(\mathbf{x}_i, \mathbf{x}_j; \Sigma) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T \Sigma^{-1} (\mathbf{x}_i - \mathbf{x}_j)}$$

It is easy to see that the euclidean distance is simply the Mahalanobis distance with the identity matrix I as covariance matrix.

In the case of the Mahalanobis distance you are going to explore three different approaches to compute it.

1. Define Σ as a diagonal matrix that has at its diagonal the average variance of the different features, i.e. all diagonal entries Σ_{ii} will be the same ($\Sigma = I\sigma$, where $\sigma = \frac{1}{d}\sum_{k=1}^{d} \sigma_k$).
2. Define Σ as a diagonal matrix that has at its diagonal the variance of each feature, i.e. σ_k.
3. Define Σ as the full covariance matrix between all pairs of features.

- For a fixed number of neighbors ($k = 10$) comment on the differences among the three distances and on the differences among the three different versions of the Mahalanobis distances. Comment how these affect the performance of the classification, when we should prefer the one over the other, etc..
- Chose the distance which you believe it performs better and explain how the performance changes with respect to the values of k. How the number of the neighbors in influences the classification?
- Verify that the euclidean distance is the Mahalanobis distance with the identity matrix I as covariance matrix.

Exercise 3

Visualize, study, and discuss the decision surfaces that kNN algorithm produces for the different values of k using the Euclidean distance and an other distance of your choice. Explain how the performance changes with respect to the values of k. How the number of the neighbors in influences the classification?

To do so, you will work only in two attributes.

- Testing will be done on an artificially generated dataset that covers in a regular manner all possible values for the two chosen attributes. To do so we need to divide the space into a grid by discretizing the space into n values between the minimum and maximum value of an attribute. Each of these values must be compared with the n discrete values of the second attribute. The resulting array will be of shape ($n * n$, 2)
- Using your training set classify your test instances and visualize the results of the classification

Exercise 4

Linear Transformations:
In this exercise you will visualize the effect of a linear transformation in the unit circle. First you draw a unit circle, then you transform your unit circle with a given matrix A and comment your result. Finally apply a linear transformation on an unknown set of points Z such that the result of the linear transformation with the matrix A gives you the unit circle (comment the result).

General instructions

You have to send **code** and a **formal report**, saved in a zip file using as name this format: TP2_LASTNAME_Firstname. The report should be in **.pdf** format or you can use the jupyter notebook (in both cases should be a formal report). Explain the problem and discuss the results, it should be a summary of the main findings, factual, clear, and concise. Please write your name in both the code (as comment in all all the python scripts and notebook) and the report. If you prefer to work using python script instead of the jupyter notebook you can download the .py file from the notebook (open the notebook and then: File → Download as → Python (.py))