HW Brief – Learning Algorithms

Responsible T.A.: Yar Mazal, mazaly@post.bgu.ac.il
Submission date: see Moodle

Intro

In this exercise, you will implement two basic algorithms for learning. One is called *k-nearest-neighbors* and is used for classification. The other is called *k-means* and is used for clustering.

This brief will not go into the details of the algorithms, and will not teach you how to implement them. Instead, here are several sources which you are advised to go over. Note that they cover the same topics, and therefore there is no need to go over all of them, choose the ones you're most comfortable with. Alternatively, there are many other online sources.

Keep in mind that most likely some of you will have some questions about this exercise. Good questions will be added to an FAQ file that will be added and updated to Moodle as needed.

KNN:

- Wiki
- https://www.ibm.com/uk-en/topics/knn
- https://sebastianraschka.com/pdf/lecture-notes/stat479fs18/02_knn_notes.pdf

K-Means

- https://en.wikipedia.org/wiki/K-means_clustering
- https://towardsdatascience.com/machine-learning-algorithms-part-9-k-means-example-in-python-f2ad05ed5203

Allowed Imports

- Currently, the only import allowed is NumPy.
- If you wish to use other imports, a written pre-approval is needed. A list of allowed imports will be updated in Moodle.

Clarifications

- KNN can be used both for classification and regression. In this work, we consider classification, not regression.
- Both KNN and K-Means require a distance metric. There are many such metrics. We will use the most popular one, the Euclidean distance for both algorithms.
- K-Means usually use some degree of randomness for initialization. We won't do this. See details below.

What will you do?

You will implement both algorithms according to the template files given. Each algorithm will be implemented in a separate class in a separate file.

KNN

- Implement in knn.py according to the template given.
- Constructor – The constructor receives a single argument named k, which dictates the number of neighbors considered.
- A “fit” method.
    - The method receives a NumPy array of samples to be used as train data (called x_train in the template file). In the array, each row is a data sample whose dimensions equal the number of columns.
    - The method receives a second 1-d NumPy array of labels per training sample to be used as train data (called y_train in the template file).
    - It doesn’t return anything.
- A “predict” method.
    - The method receives a NumPy array of samples whose class is to be predicted. In the array, each row is a data sample whose dimensions equal the number of columns.
    - The method returns a 1-d NumPy array of labels, one per training sample.
- The underlying data type which will be used both for features and labels will be integers.

K-Means

- Implement in k_means.py according to the template given.
- Constructor – The constructor receives an argument named k, which dictates the number of clusters considered, and max_iter which dictates the maximal number of iterations done during the fit stage.
- An “initialize” method.
    - As you might have read, the k-means algorithm performance depends on the initialization of the model. Even though it is customary to use random initialization, this would make it hard for us to grade your work.
    - Therefore, you are required to implement an “initialize” method, which will accept a single argument. The argument is a 2-d NumPy array. The number of rows is the same as the number of clusters, and each row will represent the initial coordinates of some centroid. You are **required** to use this initialization for the clusters.
    - Our test code will call this method before the “fit” method to provide some initialization for clusters.
- A “fit” method.
    - The method receives a NumPy array of samples and uses it as train data (called x_train in the template file). In the array, each row is a data sample whose dimensions equal the number of columns. The method performs the k-means algorithm on the train data to find centroids and train the model.

    - The method returns a dictionary. Its keys are clusters IDs that you assign during training (unique integers of your choice). The values are 1-d NumPy arrays, each of which represents a centroid.
- A “predict” method.
    - The method receives a NumPy array of samples to be assigned to clusters. Each row is a data sample whose dimensions equal the number of columns.
    - The method returns a 1-d NumPy array of cluster IDs. Each ID correlates to the cluster to which the sample was assigned. Cluster IDs are the same ones returned during training.
    - Note that the centroids are calculated only during training, and aren’t changed by the predict method.
- A “wcss” method – The method will return the WCSS of the model, as calculated during the “fit” stage. During grading, it won’t be called before the “fit” method is called.
- The underlying data type which will be used both for features is floats.

What you’re allowed to do?

- You may add methods and classes (but you must submit the required py files).
- You may import NumPy and all other approved modules (A list will be updated in Moodle per your requests).

What you’re not allowed to do?

Doing any of these will fail your code. As a result, grades could be as low as zero.

- Don’t try to import any modules other than approved. They won’t be installed in the testing system, and your code will crash upon import.
- Don’t change the prototype of provided methods, our testing code will only call these and will assume the specified signature.
- Please do not add to submitted files global variables, or any testing code. Doing this could trigger all sorts of issues and is a bad practice. We cannot guarantee reliable execution of your code if you do this. For your tests, use the supplied main.py file.

What should you submit?

- You should submit two files: knn.py and k_means.py.
- In these files, you should implement the required classes. **Do not put testing code in these files.**
- The main.py file was uploaded for your testing needs. **Do not upload it as part of your solution.**

How will we test your code?

Testing will be similar to tests provided prior to submission. Basically, initialize models using the constructor, and call fit and predict methods. For the K-Means, also the initialize and wcss methods will be called. Grading will be based on comparing the method's output with expected outputs.