The *k*-Means Project

COMP9318 (Fall 2024)

Introduction

k-means and sometimes referred to as c-means was first proposed by Stuart Lloyd at Bell Labs in 1957. Despite its age, k-means is one of the most powerful and widely used clustering algorithms despite its simplicity. Many variations came out from k-means such as Gustafson and Kessel k-means (GK k-means), probabilistic k-means, fuzzy k-means, and relational k-means, to name a few.

In this project, your goal is to implement the original k-means from scratch without relying on existing k-means implementations such as scikit-learn. To do that, you have to fully understand the mathematical formulation of k-means. In addition to the implementation, you are required to train your implementation of k-means on a small dataset to verify the validity of your algorithm. This exercise will test your ability to transfer what you have learned into practice as one day you may encounter algorithms and ideas that require implementation from the ground up.

Recall that k-means is an iterative algorithm that has two major steps, update assignments to the clusters and update the cluster centers until the algorithm converges. You will work with data of different dimensions. Some may be 2-dimensional, which are easily visualized, while others dataset may come with higher dimensions, which require additional processing for visualization.

It is highly recommended to use Python for this project. As much as possible, your code should be optimized, and you should take advantage of Python functionalities such as list comprehension and broadcasting when performing mathematical operations. You are allowed to use Numpy, Scipy and matplotlib throughout your code.

Matplotlib lets you plot the 2-dimensional data easily, however for higher dimensional data (d > 2) you need to use some kind of dimensionality reduction. We will talk about dimensionality reduction later in the course, but you should be able to use existing Python packages for dimensionality reduction such as PCA and t-SNE that are available in scikit-learn (explain the details of any dimensionality reduction technique you used in your report).

Note that in some cases and on some datasets, some features may have very large values compared to other features. In such cases, you need to normalize your data.

You will observe that k-means will not cluster all datasets accurately. But there are other algorithms that can. Two well-known clustering algorithms are DBSCAN and Spherical Clustering. You need to try those two algorithms in addition to your k-means implementation and report the results in the report. You do not need to implement DBSCAN and Spherical clustering, instead use existing packages.

Deliverables

Your deliverables from this project are the following:

1. Source code: should be well-documented and modular.
2. Report: which will include the following
    a. Description of the k-means algorithm, including pseudo-code
    b. The mathematical formulation of k-means (you need to write the equations) and describe each step
    c. Explain how DBSCAN and Spherical Clustering works.
    d. Explain the details of any dimensionality reduction techniques you use.
    e. Description of the datasets we provided, you need to know and ins and outs of the each dataset
    f. Description of any additional datasets you like to add
    g. Results, which includes:
        i. Your own explanation and any findings for all clustering algorithms (your own implantation of k-means, DBSCAN and Spherical Clustering).
        ii. Visualization of the clusters and loss function curves
        iii. Feel free to add any additional results if you like)
3. Use Latex (Overleaf) for your report, not MS word.

Python Packages

You can use the following packages and functions:

- Numpy: this will you optimize your matrix operations
- Scipy: you may or may not need this package
- Matplotlib: for graphs, figures and data visualization
- Scikit-learn: from this package you will need:
    - PCA or TSNE functions for dimensionality reduction
    - DBSCAN: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html
- sphere cluster: https://pypi.org/project/sphere cluster/0.1.2/#description

Datasets

Iris

URL: https://archive.ics.uci.edu/ml/datasets/iris
classes: 3
data points: 150

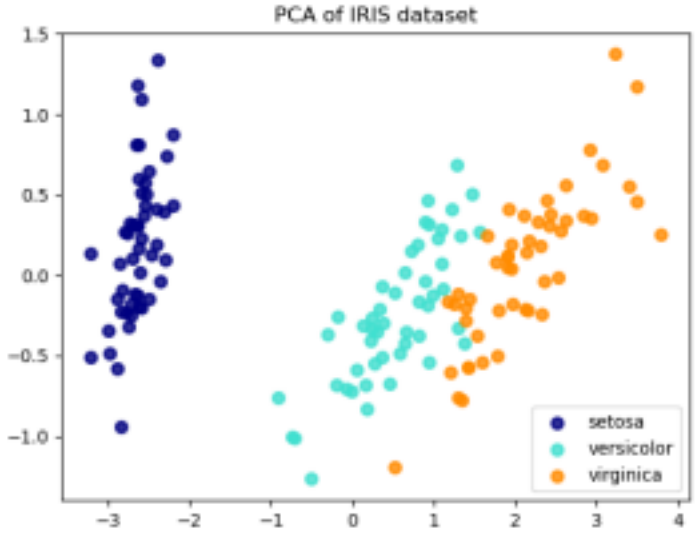

Fig. 1. Iris in 2D after performing PCA (https://scikit-learn.org/stable/modules/decomposition.html)

Three separable gaussians

URL:
https://github.com/mohammedkhalilia/birzeit/blob/master/comp9318/data/3gaussians-std0.6.csv
classes: 3
data points: 300

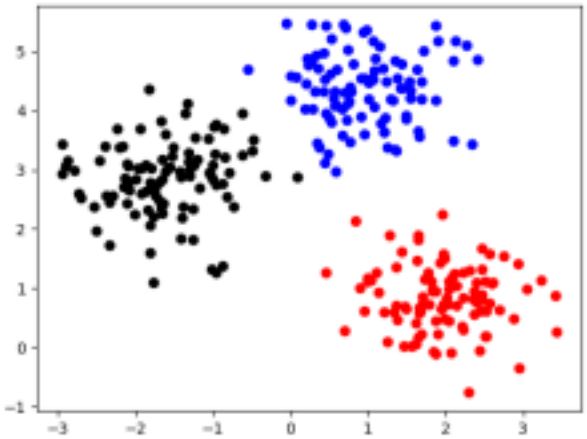

Fig. 2. Three separable gaussians with 0.6 standard deviation

Slightly overlapping three gaussians

URL:
https://github.com/mohammedkhalilia/birzeit/blob/master/comp9318/data/3gaussians-std0.9.csv
classes: 3
data points: 300 Circles

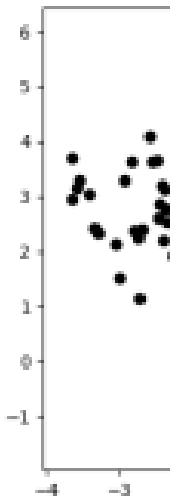

Fig. 3. Three separable gaussians with 0.9 standard deviation

URL: https://github.com/mohammedkhalilia/birzeit/blob/master/comp9318/data/circles.csv

classes: 2
data points: 1000

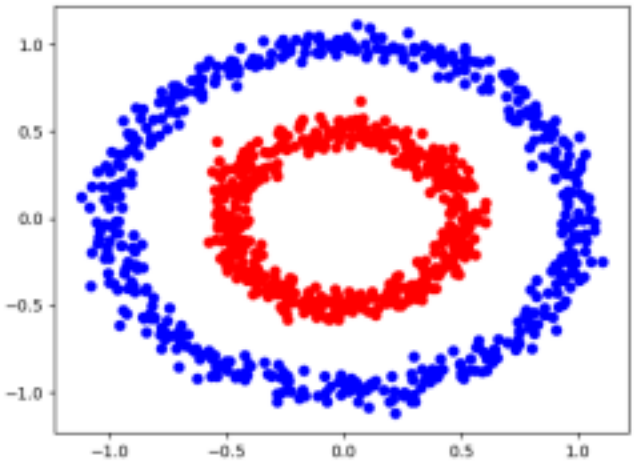

Fig. 4. Circles dataset

Moons

URL: https://github.com/mohammedkhalilia/birzeit/blob/master/comp9318/data/moons.csv
classes: 2
data points: 1000

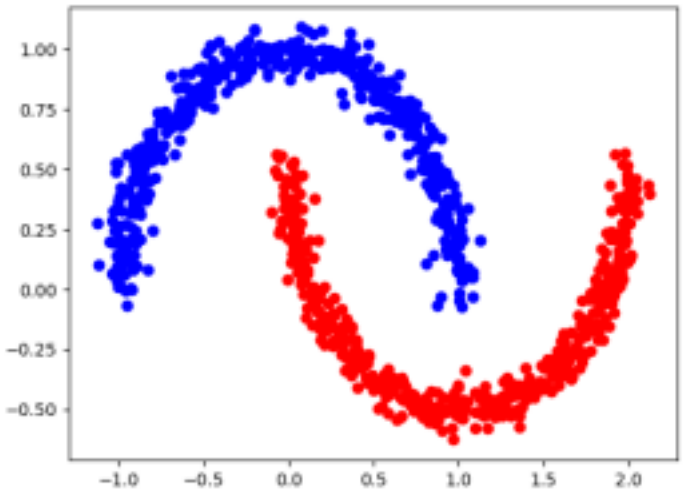

Fig. 5. Moons dataset