SFU SCHOOL OF MECHATRONIC SYSTEMS ENGINEERING

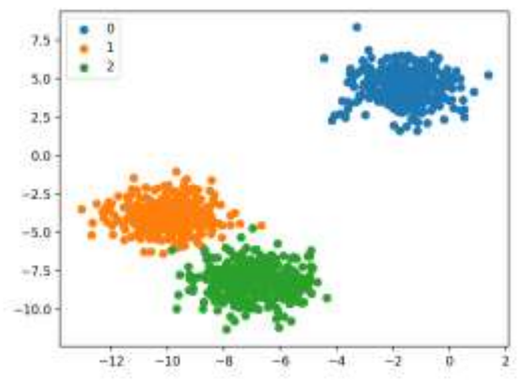
Simon Fraser University
School of Mechatronic Systems Engineering
MSE491 - Application of Machine Learning in Mechatronic Systems

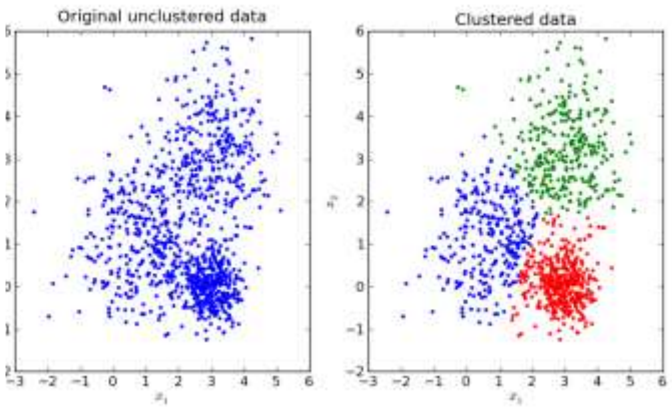
Lab 4 – Clustering

Due Date: April 11th, 11:59pm

Background

Clustering technique is a popular method, mainly used to get an intuition about the data structure. The main idea behind the clustering algorithms is to identify sub-groups in the data. Similar to the classification techniques (e.g. Fig.1), clustering algorithms put "similar" instances together into "clusters". However, unlike classification, clustering is an unsupervised learning method. This means that the data is not labelled in clustering (e.g. Fig. 2). The learning algorithm itself assigns un-labelled data into sub-groups. Clustering is a great tool for data analysis, customer segmentation, recommender systems, search engines, image segmentation, semi-supervised learning, dimensionality reduction, and more.





© https://machinelearningmastery.com

© https://towardsdatascience.com

Fig1. An example for classified data with labels

Fig2. An example for clustering data with no labels

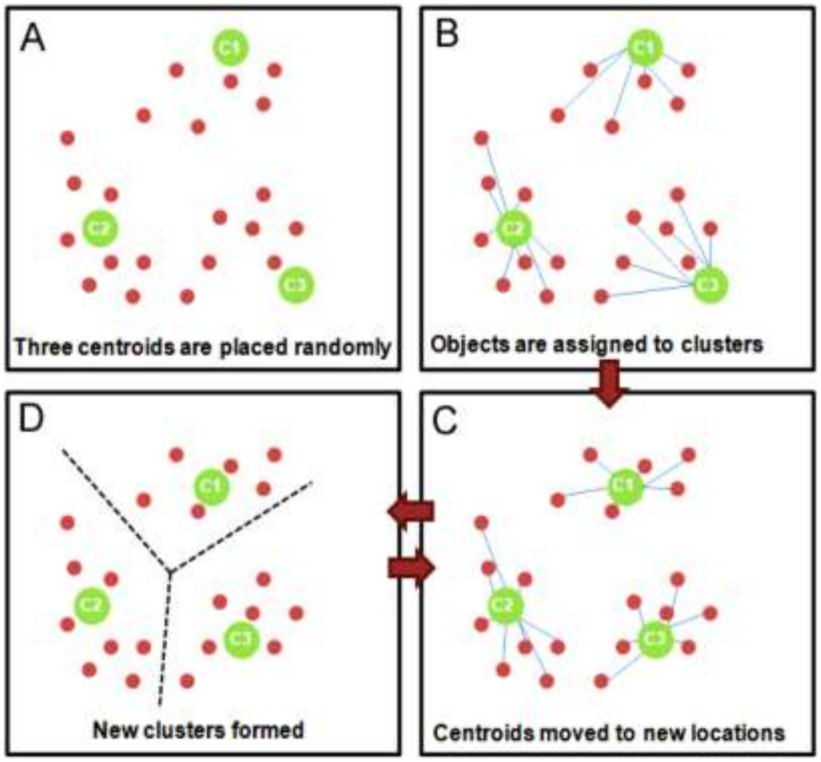
In this assignment, we will look at a popular clustering algorithm, K-means, and explore some of its applications, such as dimensionality reduction, image segmentation, and semi-supervised learning.

K-means clustering

The K-Means algorithm, also referred to as Lloyd-Forgy is a simple iterative algorithm capable of clustering the dataset into K sub-groups or clusters, often in just a few iterations. The K-means algorithm put similar datapoints into clusters while keeping the clusters as far as possible. The algorithm calculates the sum of squared distance between the datapoints and the centroids of the clusters (i.e. mean of all datapoints in each of the clusters). It then assigns datapoints to a cluster such that this summation is minimized.

The K-means algorithm can be summarized as below:

1. Initialization (Fig. 3A):
    a. Specify the number of clusters (i.e. K).
    b. Specify random centroids.
2. Compute the sum of the squared distance between data points and all centroids.
3. Assign each data point to the closest centroid (Fig. 3B).
4. Calculate the centroids for the clusters (Fig. 3C).
5. Repeat step 3 and 4 until the location of the centroids does not change. This means that no datapoints is moved between the clusters (Fig. 3D).


A
C1
C2
C3
Three centroids are placed randomly
B
Objects are assigned to clusters
C
Centroids moved to new locations
D
New clusters formed

© Microarray data analysis: Gaining biological insights

Fig 3. K-means clustering algorithm

MSE 491- Assignment 4 (Spring 2021)

For this assignment, please follow the instructions given in this document to complete and run different sections in the .ipynb file provided along with the assignment . You can open and run the .ipynb file either in Jupyter notebook or Google Colab.

To get started with this assignment, open Assignment4.ipynb in Jupyter notebook or Google Colab.

1. Simple K-Means Clustering

For the first section of this assignment, the Iris dataset is used. This is a multivariate dataset consisting of samples from three species of Iris. For each sample in this dataset, four features (i.e. length, width, sepals and petals in centimeters) were measured. Take the following steps to first explore this dataset and perform simple K-means clustering:

1. Complete the "import Iris dataset" section to import and load this dataset. Separate the features and targets in two separate variables.
   a. What are the target variable names? **(1 points)**
2. Generate a scatter plot for the petal width vs. petal length in the iris dataset where each instance's species (i.e. each class) is represented by a different marker **(2 points)**.
3. Generate a scatter plot for the petal width vs. petal length in the iris dataset without using the labels (i.e. all datapoints are plotted with the same marker and are not distinguishable) **(2 points)**.
4. Perform K-means clustering on the dataset not using the labels:
   a. With K=2. Using different markers, show the clusters obtained **(3 points)**.
   b. With K=3. Using different markers, show the clusters obtained **(3 points)**.
   c. Discuss about the results of section 4.a and 4.b. Using clusters generated, explain how selecting different number of K values affects the clustering? **(4 points)**.

2. Inertia and Silhouette Score

The main goal of this section is to understand the meaning behind inertia and silhouette score in K-means clustering. To get started with this section, run the first part of the code to create five blobs.

1. Plot the blobs in a single figure **(2 points)**.
2. Import Kmeans from sklearn.cluster. Perform K-means clustering with K=5 and random state= your group number **(3 points)**.
3. Print out the centroids of the clusters on the figure generated in section 1 **(2 points)**.
4. KMeans instance preserves the labels of the instances it was trained on. Find the label of an instance (or the index of the cluster) that instance is assigned to **(3 points)**.
5. In clustering, it is possible to measure the distance between each instance and the cluster's centroid. This is the idea behind the inertia metric. Calculate the inertia to evaluate the performance of the model generated in section 2 **(5 points)**.
6. Perform K-means clustering again with a K value in the range of 5-10 **(1 points)**.
7. Perform K-means clustering again with a K value lower than 5 **(1 points)**.
8. Computing inertia, evaluate and compare the performance of the scenarios described in section 6 and 7 **(3 points)**.
9. a. Plot inerta as a function of 1<K<11 **(2 points)**.

   b. Discuss which K value is the best to choose. Bring reasons to support your idea **(3 points)**.
10. A more precise but more computationally expensive approach is to use the Silhouette score[1] for the purpose of choosing the best value for the number of clusters in K-means clustering. You can use Scikit-Learn's silhouette_score() function to calculate this score.
    a. Plot Silhouette score as a function of 1<K<11 **(2 points)**.
    b. Discuss which K value is the best to choose. Bring reasons to support your idea **(3 points)**.

[1] The silhouette score's range is between -1 and +1. A score of +1 means that the sample is well inside its own cluster and has a larger distance to the other clusters. A score of 0 means that the sample is on the boundary of a cluster. A score of -1 means that the sample may have been assigned to the wrong cluster.

3. Using Clustering for Image Segmentation

The main goal of this section is to perform image segmentation using K-means clustering and understand the advantages and disadvantages of K-means clustering for image segmentation.

1. Download the "ladybug.jpg" image provided along with this addignment. Load and plot this image in your notebook **(2 points)**.
2. What is the shape of the image? **(2 points)**.
3. In the next section, the first line of code reshapes the array of the image to get a long list of RGB colours. Perform K-means clustering with number of colours=8 (i.e. K=8) and random state=42 **(2 points)**.
4. On the same image, perform K-means clustering with K=10, 8, 6, 4, and 2 with random state= 42. You will end up with five segmented images for each K value **(2 points)**.
5. Plot all the five segmented images you obtained in the last step along with the original image (i.e. 6 images in total) **(2 points)**.
6. Comparing the figures obtained in section 5, discuss how changing the K value affects the results **(5 points)**.
7. Do you recommend using K=2 to segment this specific image? Why? **(5 points)**

4. Using Clustering for Pre-Processing

Clustering is also commonly used as an efficient pre-processing step prior to the supervised learning algorithm. The goal of this section is to show how this can be done in a simple task.

For this section of the assignment, we tackle the digits dataset, which is a simple dataset, similar to MNIST. This dataset contains 1797 grayscale 8 × 8 images. These images represent the digits 0 to 9.

1. Run the first section to load the data and get access to the train and test splits.
2. From Scikit-Learn, import LogisticRegression.
3. Fit a logistic regression model with the following parameters: multiclass="ovr", solver="lbfgs", max_iter=3000, random_state=42 **(4 points)**.
4. Evaluate the logistic regression model on the test set using score (here is a link for how to obtain the score) **(2 points)**.
5. In order to see whether the score can be improved by K-means clustering as a pre-processing step, a pipeline is created. This pipeline first clusters the training set into 50 clusters[2]. It then replace the images with their distances to these 50 clusters and finally applies a Logistic Regression model (The code to create the pipeline is given in the .ipynb file).
6. Fit the logistic regression model created in the pipeline to the train set **(2 points)**.
7. Print out the score of the pipeline on test set and compare it to the score you obtained in section 4 **(2 points)**.
8. How much did the score in section 7 change compared to the score obtained in section 4? What is the reason behind this? **(5 points)**.

[2] Since there are 10 different digits, it might be implied that the number of clusters need to be 10. However, since each digit can be written in different ways, it is usually preferable to use a larger number of clusters. This is why 50 clusters are chosen.

5. Semi-Supervised Learning

Another application of clustering is semi-supervised learning. This is also one of the hot topics in the field and it is employed when there is plenty of un-labelled instances and very few labelled ones. The main goal of this section is to give you an idea how semi-supervised learning can be useful in an example application.

For this section of the assignment, we again use the digits dataset.

1. Run the first section to see the performance of a logistic regression model having only 50 labelled instances. Print out the score for this logistic regression model.
2. Using kmeans clustering, cluster the training set into 50 clusters **(5 points)**.
3. For each cluster, find the image closest to the centroid (representative image) **(5 points)**.
4. Plot the representative images (i.e. 50 images representing each of the clusters) **(4 points)**.
5. Label these representative images manually.
6. Perform logistic regression with the following parameters: multiclass=”ovr”, solver=”lbfgs”, max_iter=3000, random_state=42. Fit for the representative images and labels **(2 points)**.
7. Print out the score of the new logistic regression model on the test set **(2 points)**.
8. How much did the score change compared to the first section? What is the reason behind this? **(2 points).**

Submitting your Assignment

The assignment must be submitted online to Canvas with the following structure:

For each group, you must submit **one zipped folder** that contains:

1. **A report**- The assignment report must be in **PDF format**, called **report.pdf**. This report must contain all the figures and discussions/explanations for the questions.
2. **An .ipynb file that contains all your codes**- This file must be called **Assignment4.ipynb**. Only **a single file** is acceptable. There is space provided for each section. **Please do not add/remove sections.** Please fill out the sections with you code.