K-means and hierarchical clustering with MATLAB

Objective: The objective of today's exercise is to understand how the unsupervised learning methods k-means clustering and hierarchical clustering work. Upon completing the exercise you should also understand how the choice of number of clusters, distance metrics and linkage functions can impact the solutions obtained and further be able to interpret dendrograms and measures of cluster validity.

Material: Lecture notes *"Introduction to Machine Learning and Data Mining"* as well as the files in the exercise 10 folder available from Campusnet.

Preparation: Exercises 1-9

Part 1: Group discussion (max 15 min)

For the group discussion, each group should have selected a *discussion leader* at the previous exercise session. The purpose of the discussion leader is to ensure all team members understands the answers to the following two questions:

Multiple-Choice question: Solve and discuss **problem 16.1** from chapter 16 of the lecture notes. Ensure all group members understand the reason why one of the options is true and why the other options can be ruled out. (After today's exercises make sure to complete the remaining multiple-choice problems listed as part of the preparation for week 10 on the course homepage).

Discussion question: Discuss the following question in the group

- Explain the K-means algorithm.

Part 2: Programming exercises

Piazza discussion forum: You can get help by asking questions on Piazza: https://piazza.com/dtu.dk/fall2018/02450

Software installation: Extract the Matlab toolbox from DTU Inside. Start Matlab and go to the `<base-dir>/02450Toolbox_Matlab/` directory using the command `cd('<base-dir>/02450Toolbox_Matlab/')` and run `setup.m`. Remember the purpose of the exercises is not to re-write the code from scratch but to work with the scripts provided in the directory `<base-dir>/02450Toolbox_Matlab/Scripts/`

Representation of data in Matlab:

	Matlab var.	Type	Size	Description
	`X`	Numeric	$N \times M$	Data matrix: The rows correspond to N data objects, each of which contains M attributes.
	`attributeNames`	Cell array	$M \times 1$	Attribute names: Name (string) for each of the M attributes.
	`N`	Numeric	Scalar	Number of data objects.
	`M`	Numeric	Scalar	Number of attributes.
Regression	`y`	Numeric	$N \times 1$	Dependent variable (output): For each data object, `y` contains an output value that we wish to predict.
Classification	`y`	Numeric	$N \times 1$	Class index: For each data object, `y` contains a class index, $\texttt{y}_n \in \{0, 1, \ldots, C-1\}$, where C is the total number of classes.
	`classNames`	Cell array	$C \times 1$	Class names: Name (string) for each of the C classes.
	`C`	Numeric	Scalar	Number of classes.
Cross-validation				All variables mentioned above appended with `_train` or `_test` represent the corresponding variable for the training or test set.
	`⋆_train`	—	—	Training data.
	`⋆_test`	—	—	Test data.

In previous exercises we considered supervised learning, i.e., we were given both input data $\boldsymbol{X}$ and output values $\boldsymbol{y}$. In classification, the outputs were discrete variables, and in regression, the outputs were continuous.

We now move on to unsupervised learning where we are only provided input data $\boldsymbol{X}$. The aim is here to find common patterns in the data such as groups of observations that are similar in some sense. In this exercise we will consider two clustering approaches for unsupervised learning: k-means clustering and hierarchical clustering.

10.1 k-means clustering

In this part of the exercise we will investigate k-means clustering. In k-means each of the data points are assigned to the cluster in closest proximity according to some measure of distance between cluster centers and data points. When the distance is given by the squared Euclidean distance the centers are also denoted centroids. Once the data points have been assigned, each cluster center is updated to be placed at the center of the data points that are assigned to the cluster. This continues iteratively, usually until the assignment of data points to centers no longer change or until a maximal number of iterations is reached.

10.1.1 You can load the `Data/synth1.mat` data set file into Matlab using the `load` function. The script `ex10_1_1.m` clusters the data into $K = 4$ clusters using the k-means algorithm. Notice how the script makes a scatter plot of the data and the clustering using the `clusterplot` function in the toolbox.

Script details:

- *In Matlab, you can use the function* `kmeans` *to compute a k-means clustering. Type* `help kmeans` *to learn how to use the function.*
- *The function can be called as* `[i,Xc]=kmeans(X,K);` *where K is the number of clusters.*
- *You can choose the distance measure used in the algorithm by setting the* `'Distance'` *property.*
- *Type* `clusterplot(X,y,i,Xc)` *to plot the data and the clustering.*
- *Type* `help clusterplot` *to learn more about how to use the clustering plot tool in the toolbox.*
- *Sometimes,* `kmeans` *will fail, because an empty cluster has been generated. You can choose how to handle empty clusters by setting the* `'EmptyAction'` *property.*
- *To be more robust against different initial conditions, you can set the* `'Replicates'` *property.*

Does the clustering coincide with the true classes? Try running your code several times (with `'Replicates'` set to 1) to show that the algorithm can fail if the initial conditions are poor. Try also the data sets `synth2`, `synth3`, and `synth4`.

For supervised learning we evaluated the model performance in terms of the error rate and accuracy. This however requires that we can match the estimated outcome to the true underlying classes, provided they are known. Even when we know the true underlying classes, we do not in general know which cluster corresponds to which class. Thus, by some means the true classes and the estimated clusters must be related to each other. One way of relating the two is to find out which cluster best matches each class such that each class is assigned one of the clusters and then calculate the error rate based on these clusters.

10.1.2 Is the above definition of the classification error rate for clustering reasonable if we extract the same number of clusters as classes in the data? Does the definition of the classification error make sense when the number of extracted clusters are different from the true underlying classes?

Rather than using the error rate we will consider the supervised measures of cluster validity, in particular Rand Statistic, Jaccard coefficient and normalized mutual information (NMI). Carefully review these measures in the book and make sure you understand how they are calculated.

10.1.3 The script `ex10_1_3.m` repeats exercise 10.1.1, but this time perform k-means clustering for $K = 1, \ldots, 10$ clusters. For each value of K the three cluster validity measures mentioned above are computed. Notice how the script plots the cluster validity measures as a function of K.

Script details:

- *You can either write your own code for computing the cluster validity measures, or you can use the function* `clusterval` *in the toolbox. Type* `help clusterval` *to learn how to use the function.*

How can the cluster validity measures be used to select the best number of clusters?

What happens when more than four clusters are used to model the data?

10.1.4 For supervised learning we used cross-validation to evaluate performance and estimate the number of parameters in our models, i.e., the number of clusters.

Let us assume that we split the data into a training and a test set, train the k-means model on the training set and evaluate how well the model accounts for the test data. Consider evaluating the clustering by summing the distance of test points to the closest estimated cluster center obtained. What will happen with this training and test error as we increase the number of clusters?

K-means clustering has many different applications, one of which is data compression. A data set can be compressed by performing k-means clustering and then representing each data object by its cluster center. Thus, the only data that need to be stored is the K cluster centers and the N cluster indices.

10.1.5 We will consider a subset of the wild faces data described in [2]. You can load the wildfaces data set from the `Data/wildfaces.mat` file with the `load` function, see the script `ex10_1_5.m`. Each data object is a $40 \cdot 40 \cdot 3 = 4800$ dimensional vector, corresponding to a 3-color 40×40 pixels image. The script computes a k-means clustering of the data with $K = 10$ clusters. Plot a few random images from the data set as well as their corresponding cluster centroids to see how they are represented.

Script details:

- *You can plot an image by the command* `imagesc(reshape(X(i,:),40,40,3))` *which reshapes an image vector to a 3-dimensional array and uses the* `image` *function to plot it. Similarly, you can plot the cluster centroids.*
- *Running k-means on a large data set can be slow. If you type* `[i,Xc]=kmeans(X,K,'Display','iter','OnlinePhase','off');` *it will run faster and provide information about the iterations as they run. Type* `help kmeans` *to read more about these options.*

How well is the data represented by the cluster centroids? Are you able to recognize the faces in the compressed representation? What happens if you increase or decrease the number of clusters?

10.1.6 Modify the script `ex10_1_5.m` to repeat the previous exercise but with the digits data set. You can load the digits data set from the file `Data/digits`. Each data object is a $16 \cdot 16 = 256$ dimensional vector, corresponding to a gray scale 16×16 pixels image.

Script details:

- *You can change the color map to black-on-white grey-scale by the command* `colormap(1-gray);`

Why does running k-means with $K = 10$ not give you 10 clusters corresponding to the 10 digits 0–9? How many clusters do you need to visually represent the 10 different digits? Are there any digits that the clustering algorithm seems to confuse more than others?

10.2 Hierarchical clustering

We will in this part of the exercise consider hierarchical clustering based on the built-in Matlab functions `pdist` that forms a sample to sample distance matrix according to a given distance metric, `linkage` that creates the linkages between data points forming the hierarchical cluster tree and `dendrogram` that creates a plot of the generated tree. Use `help` for the three function and see how they are used and inspect what distance metrics and linkage functions are implemented.

10.2.1 Inspect and run the script `ex10_2_1.m`. The script loads the data set from the file `Data/synth1` and partitions dthe data using hierarchical clustering with single linkage using the Euclidean distance measure. Notice how the script is used to cluster the data into 4 clusters by cutting off dendrogram at a threshold and plots a dendrogram and a scatter plot of the clusters.

Script details:

- *The function* `linkage` *computes the hierarchical clustering, resulting in a matrix representing the hierarchy of clusterings. Type* `help linkage` *to learn how to use it.*
- *The second parameter of* `linkage` *can be used to select the method used in the hierarchical clustering procedure.*

- *The function* `linkage` *calls the function* `pdist` *to compute distances. You can pass parameters to* `pdist` *through the third parameter of* `linkage`*. Use this e.g. to change the distance measure.*
- *You can e.g. type* `Z=linkage(X,'single','euclidean');` *to use single linkage with the Euclidean distance measure.*
- *To compute a clustering, you can use the function* `cluster`*. For example, typing* `i=cluster(Z,'Maxclust',4)` *to return a maximum of 4 clusters. Type* `help cluster` *to learn more about what this function does.*
- *To plot a dendrogram, you can use the* `dendrogram` *function.*
- *Again, you can use the function* `clusterplot` *to plot a scatter plot of the clustering.*

Try changing the linkage method and see how it changes the dendrogram. Try running your code several times to see if it generates exactly the same dendrogram each time. Try also the data sets `synth2`, `synth3`, and `synth4`, and choose suitable distance measures for these data sets.

10.3 Old Faithful geyser data

Old Faithful is a famous geyser located in Yellow Stone national park in the US. The Old Faithful geyser dataset described in [1,3] consists of $N = 272$ observations of two variables, namely the duration of each eruption in minutes (duration) and the waiting time between eruptions also in minutes (waiting).

10.3.1 Load the Old Faithful data in the file `load Data/faithful.mat`. Analyze the data by `k-means` visually inspect the labeling of the data points.

What happens if you increase K beyond the obvious $K = 2$? Try to run the program a couple of times (resulting in different initial conditions). Do you see the same solution every time? The scaling of the variables can seriously affect the results we get in clustering. Discuss whether it is more reasonable to normalize the Old faithful data set? Try normalizing the data and see whether the results of running `k-means` change.

10.3.2 Run hierarchical clustering of the Old Faithful data using the "single" and "ward" linkage, with and without normalizing the data. Do you find support for a two cluster model from the structure of the dendrograms?

10.4 Extra Challenge

10.4.1 Try clustering some of the data sets you have analyzed in the previous exercises such as the Iris data and the wine data using k-means and hierarchical clustering.

10.5 Tasks for the report

Analyze your data by hierarchical clustering and try interpret the generated dendrogram. Use the cluster validity measures to evaluate how well the clusters reflect your labeled information at one of the levels of the dendrogram. If your data form a regression problem you can generate class labels by suitable thresholds of your data as you did when you analyzed the data as a classification problem.

References

[1] A Azzalini and AW Bowman. A look at some data on the old faithful geyser. *Applied Statistics*, pages 357–365, 1990.

[2] Tamara L Berg, Alexander C Berg, Jaety Edwards, and DA Forsyth. Who's in the picture. *Advances in neural information processing systems*, 17:137–144, 2005.

[3] Wolfgang Härdle. *Smoothing techniques: with implementation in S*. Springer, 1991.