CMPT 741 Fall 2019
Data Mining
Martin Ester
TA: Zahra Zohrevand

Assignment 1

Total marks: 100 Due date: October 1, 2019
Submission: Submit to CourSys as Assignment1.pdf

Assignment 1.1 (40 marks)

We want to develop a Divisive (top-down) Hierarchical Clustering algorithm. This algorithm starts with the whole dataset as one cluster. In each iteration, it (1) chooses one of the clusters to split and (2) splits the chosen cluster into two clusters.

a) How to choose the cluster to be split? Present your solution and a rationale for your choice.

b) To determine how to split the chosen cluster into two clusters, first consider an optimum split that is the opposite (inverse) of the merge in agglomerative (bottom-up) hierarchical clustering. How is this optimum split defined? Present an (inefficient) algorithm to determine the optimum split. What is the runtime complexity of this algorithm in terms of the number *n* of elements of the cluster to be split? Present a short argument for the runtime complexity.

c) Now suggest a more efficient algorithm to find a "good" split approximating the optimum split. What is the runtime complexity of this algorithm in terms of the number *n* of elements of the cluster to be split?

Assignment 1.2 (30 marks)

The DBSCAN algorithm is based on the following theorem of density-based clusters:

For any core object *o* in cluster *C*, the set of objects density-reachable from *o* is equal to *C*.

Prove this Theorem.

Assignment 1.3 (30 marks)

In semi-supervised clustering, a user can provide some domain knowledge in the form of so-called must-link and cannot-link constraints on pairs of data points. A *must-link* constraint MUST(o_1,o_2) specifies that o_1 and o_2 must belong to the same cluster, a *cannot-link* constraint CANNOT(o_1,o_2) specifies that o_1 and o_2 may not belong to the same cluster. Design a variant of the *k*-means algorithm that optimizes the standard objective function under the condition that all specified must-link and cannot-link constraints are satisfied.

a) Explain your design.

b) Provide some pseudo-code for your algorithm.

Simon Fraser University

Assignment 1 - CMPT 741 — Data Mining, Fall 2019

Date**: October 1st, 2019**

Name: **Anurag Bejju**
Student ID: **301369375**

---

1. Proof:

1.1. The first approach I would use to choose the cluster to split would be to select the one with maximum number of objects in it. This way we would be ensuring we have more sizable clusters in most cases. The second approach would be to find the cluster with maximum number of dissimilar objects and split it. This approach uses the logic that objects that are not similar should not be part of the same cluster.

1.2. The optimum way to split the chosen cluster $[C]$ into two clusters would be to iteratively find the distance between all possible subsets of the cluster C. This distance can be calculated using any of the distance functions like *Single-Link, Complete-Link or Average-Link*. Once we have determined all the distances, we split the subset from cluster C having the maximum distance value.

The above mentioned approach is pretty inefficient as it follows a brute force approach by iterating through all possibilities. Due to this, the runtime would be $O(2^n)$ with n being the size of the cluster. We get $O(2^n)$ as there are 2^n partition possibilities in total (and all of them have to be visited before an optimum split can be determined)

1.3. There are two sub tasks present for this problem. The first is to determine which cluster needs to be split and second is actually splitting it. A good approximation method would be to find the cluster with maximum number of dissimilar objects (suggested in 1st part of the question) and then run k means with size = 2 on it. Iteratively do these two steps till we get n single node clusters in total. The run time for the two intermediate steps in total would be $O(n^2)$ and we do it for n times. So the total runtime is $O(n^3)$

2. *Proposition*: For any core object o in cluster C, the set of objects density-reachable from o is equal to C.
Assumption: Let's assume C' is a cluster with objects p that are *density-reachable* from core object o
To prove: $C = C' \; or \; [\, C \subset C' \; and \; C' \subset C]$

From the proposition, we know that core object o belongs to cluster C, and based on our assumption we know that p is density-reachable from core object o , using the *property of maximality*, we can say that $p \in C$. Since p belongs to C' as well, $C' \subset C$

Formal definition of property of maximality:
$$\forall \; \boldsymbol{p, q \; if \; p \in C \; and \; if \; q \; is \; density\text{-}reachable \; from \; p \; then \; also \; q \in C}$$

Now lets choose a random object p' where $p' \neq o$ from cluster C. Using the property of connectivity, p' is *density-connected* to o.

Formal definition of property of density-connected:

$$\forall\, \boldsymbol{p, q} \in \boldsymbol{C, p \text{ and } q \text{ are density-connected}}$$

This implies that both p' and o are *density-reachable* from a core object o'. Which means that p' is *density reachable* from the core object and is a member of C'. Since p' is also a member to C, $C \subset C'$

With $C \subset C' \; and \; C' \subset C$, we can say C' and C are the same. Which proves for any core object o in cluster C, the set of objects density-reachable from o is equal to C.

3.

3.1. Based on the question we have two constraints for semi-supervised clustering on a pair of data points:

a. **must-link constraint** $\boldsymbol{MUST(o_1, o_2)}$: o_1 and $o_2$must belong to the same cluster
b. **cannot-link constraint** $\boldsymbol{CANNOT(o_1, o_2)}$: o_1 and o_2 may not belong to the same cluster

With a small tweak in k-means algorithm, we can optimize the standard objective function under the condition that all specified must-link and cannot-link constraints are satisfied. In the algorithm, before we assign an object to any the closest k clusters, we check if any of the must-link or cannot-link constraints are violated. In case one of the conditions are violated, we assign it to the second best (provided it doesn't violate any of the must-link and cannot-link constraints). This process is iteratively conducted till a legally valid cluster is found. In case no legally valid cluster is found, them an empty partition is returned.

3.2. **Proposed Algorithm** *(Based on ClusteringByVarianceMinimization algorithm in slides)*

SSKMeansClustering(dataset D, integer k, ML constraints , CL constraints)

Initialize the set $C' = \{C_1, \ldots, C_k\}$ of the centroids of the k clusters;
$C = \{\}$;
repeat until $C = C'$
    $C = C'$;
    For each object o_i in D, assign it to the closest cluster C_j such that check_constraints (o_i, C_j, ML constraints , CL constraints) is false.
        If no such cluster exists return {}
    re-calculate the set $C' = \{C_1, \ldots, C_k\}$ of the centroids for the newly determined clusters;
return C;

check_constraints (o_i, C_j, must-link constraints , cannot-link constraints):

IF must-link constraint $MUST(o_i, C_j)$ satisfied and
cannot-link constraint $CANNOT(o_i, C_j)$ satisfied:
    return True
ELSE
    return False