OR Project Documentation

Implement linear programming model and network flow for k-means clustering method

Fereshteh Baradaran - Helia Shariati
spring 2023

Introduction

Clustering is a widely used technique in machine learning for grouping similar data points together based on their inherent characteristics. The K-means algorithm is a popular clustering method that partitions data into K distinct clusters. In this project, we aim to implement the K-means algorithm in **two different ways** to explore its effectiveness and compare the results. This project aims to provide a comprehensive analysis of clustering techniques and their applications. In Phase 1, we will design the K-means algorithm using a linear programming model in MiniZinc, leveraging its optimization capabilities to find optimal cluster assignments. Phase 2 involves implementing the K-means algorithm using network design techniques and the NetworkX library, utilizing graph representations to capture relationships between data points and clusters. Furthermore, as a bonus phase, we will explore methods to estimate the optimal number of clusters for datasets with an unknown number of clusters.

Phase 1

In this phase, the goal is to design and implement the K-means clustering algorithm using a linear programming model in MiniZinc. By leveraging the optimization capabilities of linear programming, we can find the optimal cluster assignments for our dataset.

To implement the K-means clustering algorithm in MiniZinc, we need to define the objective function, constraints, and decision variables. The objective function should capture the goal of minimizing the distance between data points and their assigned cluster centroids. The constraints should ensure that each data point is assigned to exactly one cluster, and that the distances between data points and their assigned centroids are within certain limits.

Model

Our implementation works for 2 clusters. We eliminate absolute (|x|) and use x^+ , x^- to have a linear model. Also the considered distance between two nodes is manhattan distance.

- objective function

$$min \sum_{i}^{n} (x_{1i}^{+} + x_{1i}^{-} + y_{1i}^{+} + y_{1i}^{+})a_i + (x_{2i}^{+} + x_{2i}^{-} + y_{2i}^{+} + y_{2i}^{-})(1 - a_i)$$

- Constraints

$$s.t \quad d_{i1} - c_{11} = x_{1i}^{+} + x_{1i}^{-}$$

$$d_{i2} - c_{12} = y_{1i}^{+} + y_{1i}^{-}$$

$$d_{i1} - c_{21} = x_{2i}^{+} + x_{2i}^{-}$$

$$d_{ii} - c_{22} = y_{2i}^{+} + y_{2i}^{-}$$

$$x , y \geq 0$$

$$a_i \in \{0 , 1\}$$

- decision variables

$d_{i1} : x\ of\ point\ i$

$d_{i2} : y\ of\ point\ i$

$c_i : location\ of\ i(th)\ centroid$

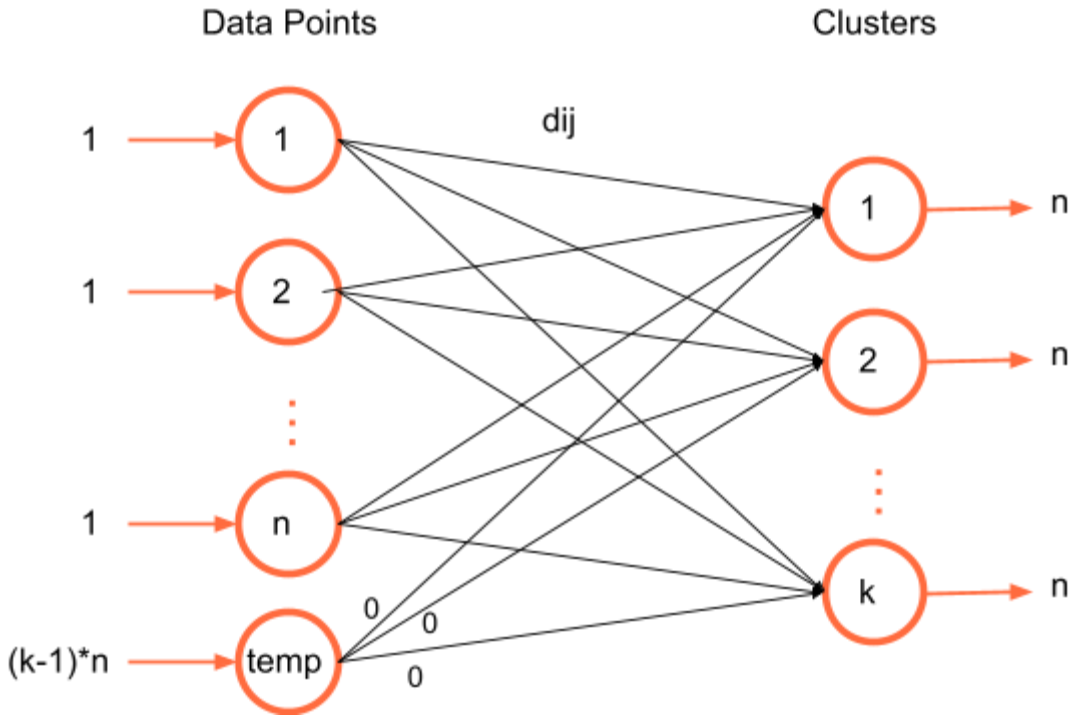
```
var float: obj = sum(i in n)(min((abs(D[i, 1] - x1) + abs(D[i, 2] - y1)), (abs(D[i, 1] - x2) + abs(D[i,
2] - y2))));

set of int: n; % number of data points
set of int: k;  % number of clusters

array[n, 1..2] of float: D;

var -1000.0..1000.0: x1;
var -1000.0..1000.0: y1;

var -1000.0..1000.0: x2;
var -1000.0..1000.0: y2;

array[n] of var bool: a;
constraint forall(i in n)(a[i] = ((abs(D[i, 1] - x1) + abs(D[i, 2] - y1)) < (abs(D[i, 1] - x2) +
abs(D[i, 2] - y2))));

solve minimize obj;
```

```
----------
y1 = -9.71299979845935;
x1 = 5.047842817658729;
y2 = 6.245405085572656;
x2 = -6.92617395128309;
a = [true, true, true, false, true, true, false, false, false, true, true, false, true, true, true];
----------
y1 = -9.71299979845935;
x1 = 5.047842817658729;
y2 = 6.245405085572656;
x2 = -7.230295326603881;
a = [true, true, true, false, true, true, false, false, false, true, true, false, true, true, true];
----------
y1 = -9.909055847281024;
x1 = 5.047842817658729;
y2 = 6.245405085572656;
x2 = -7.230295326603937;
a = [true, true, true, false, true, true, false, false, false, true, true, false, true, true, true];
----------
y1 = -10.10437245490341;
x1 = 5.047842817658729;
y2 = 6.245405085572656;
x2 = -7.230295326603937;
a = [true, true, true, false, true, true, false, false, false, true, true, false, true, true, true];
----------
y1 = -10.13546421907586;
x1 = 5.047842817658729;
y2 = 6.245405085572656;
x2 = -7.230295326603937;
a = [true, true, true, false, true, true, false, false, false, true, true, false, true, true, true];
----------
==========
Finished in 2s 1msec.
```

Phase 2

In the second phase of this project, we explore an alternative approach to implementing the K-means clustering algorithm using **network design techniques** and the NetworkX library.

To begin, we represent the clustering task as a graph using NetworkX. Each data point will be represented as a node in the graph, and the relationships between data points, such as distance, will be captured through edges in the graph. This graph representation allows us to leverage network design techniques for clustering.

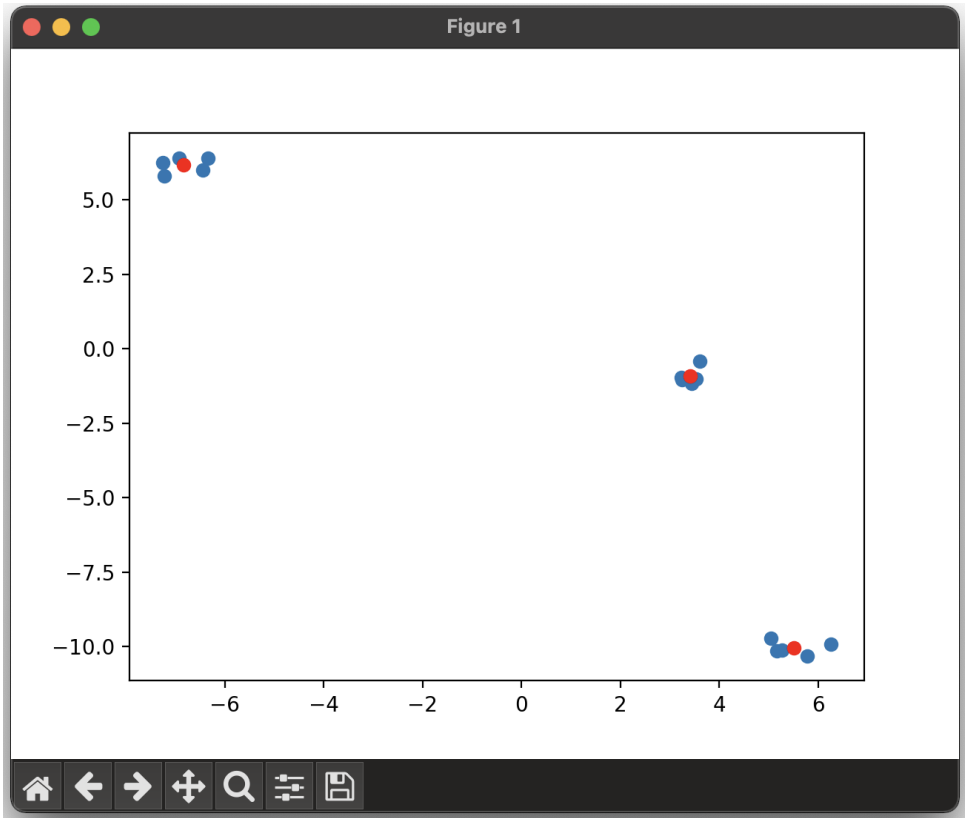


Model

This graph shows a **transport** network model, each data point is a node with enter value 1 (n node), and for each cluster we have a centroid node with demand of n (k node). There is an edge between each node of data points and each centroid which represents the distance between them. In other words the cost of edges between points and centroids is their distance. Also To balance the graph we have a temp node, which input is (k-1)*n so that we have a balanced graph and this equation n*1 + (k-1)*n = k*n . the cost of all edges between temp node and other nodes (centroids) is zero.

Initial position of the centroids at the first iteration is a random number between -10 to 10 . Then we run simplex method on the graph until we find a solution which is not improving anymore. During solving the graph we also update the position of centroids each time.

Result



Phase 3

At this phase to estimate the optimal number of clusters for datasets with an unknown number of clusters we use the built in k-means algorithm of sklearn library and run k-means for clusters 1 to n and plot it. We use the **elbow method** on the final result and find the best number of clusters (the best number is going to be found on the elbow of the plot). Then we run our built in k-means algorithm with this predicted number of clusters.

The optimal number of clusters significantly impacts the interpretability and usefulness of the clustering algorithm. Therefore, it is essential to develop an approach or metric that helps estimate the appropriate number of clusters for a given dataset.