Presentation stage

Multiple Constrained Clustering

Mathieu GUILBERT

Université d'Orléans

Introduction

Cette présentation évoque une partie du travail réalisé durant un stage au LIFO dans le cadre du Projet Involvd depuis février 2021.

On s'intéresse au clustering ensemble sous contraintes afin de pouvoir fusionner plusieurs partitions intéressantes tout en respectant des contraintes données par un expert.

Ce stage a comme objectifs de:

- Produire un état de l'art sur les méthodes de clustering ensemble et de clustering ensemble sous contraintes.
- Proposer et tester une nouvelle méthode de clustering ensemble sous contrainte.

Sommaire

Clustering

Clustering: trouver une structure sous-jacente d'un jeu de données en regroupant les données en un nombre k de clusters.

On considère ici le cas où l'ensemble des clusters forme une **partition** (*crisp clustering*) : chaque objet doit être affecté à un unique cluster.

Le but est d'obtenir une partition où les instances similaires sont dans un même cluster et les dissimilaires dans des clusters différents.

Clustering - formalisation du problème

Soit $X = \{x_1, x_2, ..., x_n\}$ l'ensemble des objets et un nombre $k \in \{2, \ldots, n-1\}$, une partition P de X en k clusters est définie comme $P = \{c_1, c_2 ... c_k\}$ tel que:

- $c_i \neq \emptyset, i = 1, ..., k,$
- $\cup_{i=1}^{k}, c_i = X,$
- $c_i \cap c_j = \emptyset,\ i, j = 1, ..., k$ et $i \neq j$

Clustering - Critère d'optimisation

Lors de la réalisation d'un clustering, on peut souhaiter optimiser un critère d'optimisation. On peut par exemple:

- Minimiser le diamètre maximum de clusters. Le **diamètre** d'une partition P est la plus grande dissimilarité entre deux objets dans le même cluster.

$$D(P) = \max_{c \in [1,k], o_i, o_j \in C_c} (d(o_i, o_j)).$$

- Maximiser la séparation minimum entre clusters. La **séparation minimum** entre clusters d'une partition P est la plus petite dissimilarité entre deux objets de différents clusters.

$$S(P) = \min_{c < c' \in [1,k], o_i \in C_c, o_j \in C_{c'}} (d(o_i, o_j)).$$

Comparaison de partitions

Il existe différentes manières d'évaluer la similarité entre deux partitions.

- **Rand Index** RI (ou indice de Rand) : mesure le taux d'accord entre deux partitions d'un même ensemble de données.
  Soient $P1$ et $P2$ des partitions d'un ensemble de données X.

$$RI(P1, P2) = \frac{a + d}{a + b + c + d}$$

avec :

-a: nombre de paires de X groupées dans $P1$ et dans $P2$.

-b: nombre de paires de X groupées dans $P1$ et séparées dans $P2$.

-c: nombre de paires de X groupées dans $P2$ et séparées dans $P1$.

-d: nombre de paires de X séparées dans $P1$ et dans $P2$.

- ARI : version ajustée du Rand Index, utilisée dans nos expérimentations

$$ARI(P1, P2) = \frac{2(ab - cd)}{(a + d)(d + b) + (a + c)(c + b)}$$

Plus il est élevé, et plus on considère les partitions similaires.

Sommaire

Clustering sous contraintes

L'expert dispose de connaissances préalables et on souhaite pouvoir réaliser un clustering satisfaisant ces connaissances. On les représente sous forme de contraintes.

- contraintes à échelle d'instance: must-link (ML) et cannot-link(CL).
- contraintes à l'échelle du cluster : diamètre, séparation, ...
- ...

Clustering sous contraintes - formalisation

Soient $X = \{x_1, x_2 ... x_n\}$ l'ensemble des objets, un nombre $k \in \{2, \ldots, n-1\}$, un ensemble de contraintes $\mathbb{C}$ et un critère d'optimisation Opt, rechercher une partition P de X en k clusters, $P = \{c_1, c_2 ... c_k\}$ tel que:

- $c_j \neq \emptyset, j = 1, ..., k$,
- $\cup_{j=1}^{k}, c_j = X$,
- $c_j \cap c_l = \emptyset$, $j, l = 1, ..., k$ et $j \neq k$,
- P satisfait toute contrainte $C \in \mathbb{C}$
- P est optimal (par rapport à Opt)

Sommaire

Clustering ensemble - Principe

Nous utiliserons les notations suivantes.

- $X = \{x_1, x_2 ... x_n\}$ l'ensemble des objets.
- $\mathbb{P} = \{P_1, P_2, ..., P_m\}$ un ensemble de partitions ou chaque $P_i = \{C_1^i, C_2^i, ..., C_{d_i}^i\}$ est une partition de X avec d_i clusters.
- C_j^i le j-ème cluster de la i-ème partition, pour tout $i = 1, ..., m$.
- $\mathbb{P}_X$ l'ensemble de toutes les partitions de X possibles, $(\mathbb{P} \subset \mathbb{P}_X)$.

Clustering ensemble - Principe

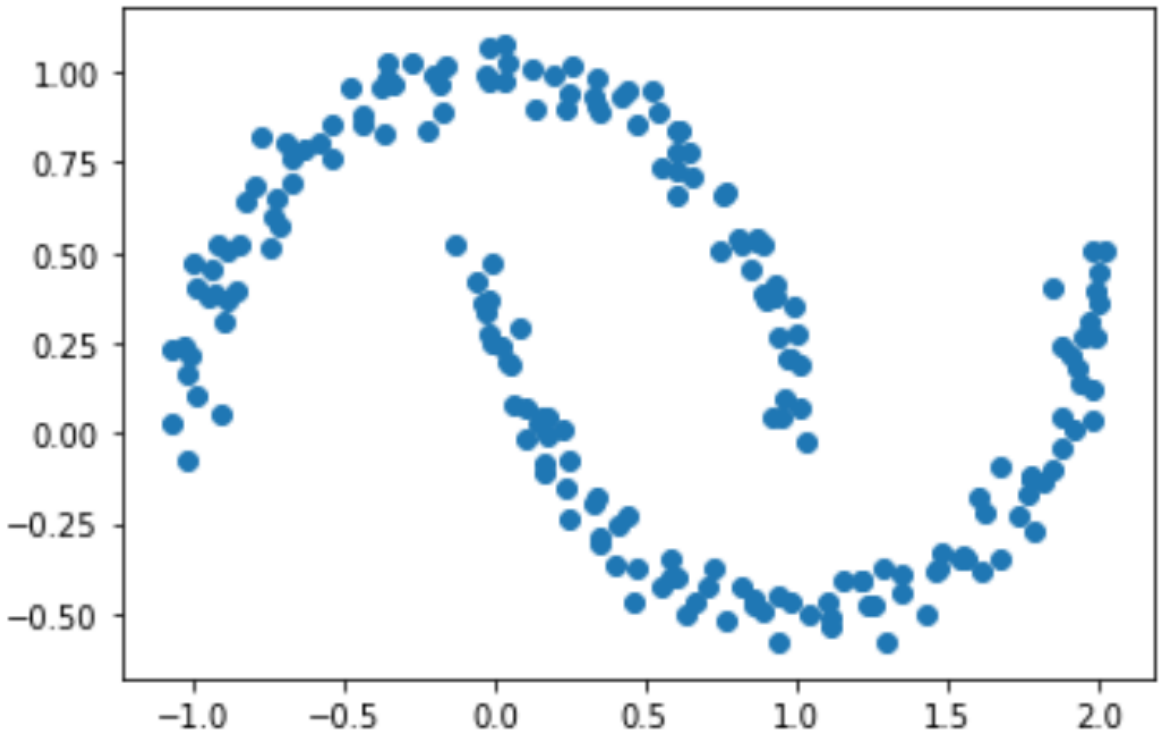
Nous utiliserons les notations suivantes.

- $X = \{x_1, x_2 ... x_n\}$ l'ensemble des objets.
- $\mathbb{P} = \{P_1, P_2, ..., P_m\}$ un ensemble de partitions ou chaque $P_i = \{C_1^i, C_2^i, ..., C_{d_i}^i\}$ est une partition de X avec d_i clusters.
- C_j^i le j-ème cluster de la i-ème partition, pour tout $i = 1, ..., m$.
- $\mathbb{P}_X$ l'ensemble de toutes les partitions de X possibles, $(\mathbb{P} \subset \mathbb{P}_X)$.

But: trouver une *partition consensus* $P* \in \mathbb{P}_X$ qui représente au mieux les propriétés de chaque partition de $\mathbb{P}$ en minimisant la somme des dissimilarités entre la partition consensus et chacune des partitions bases.

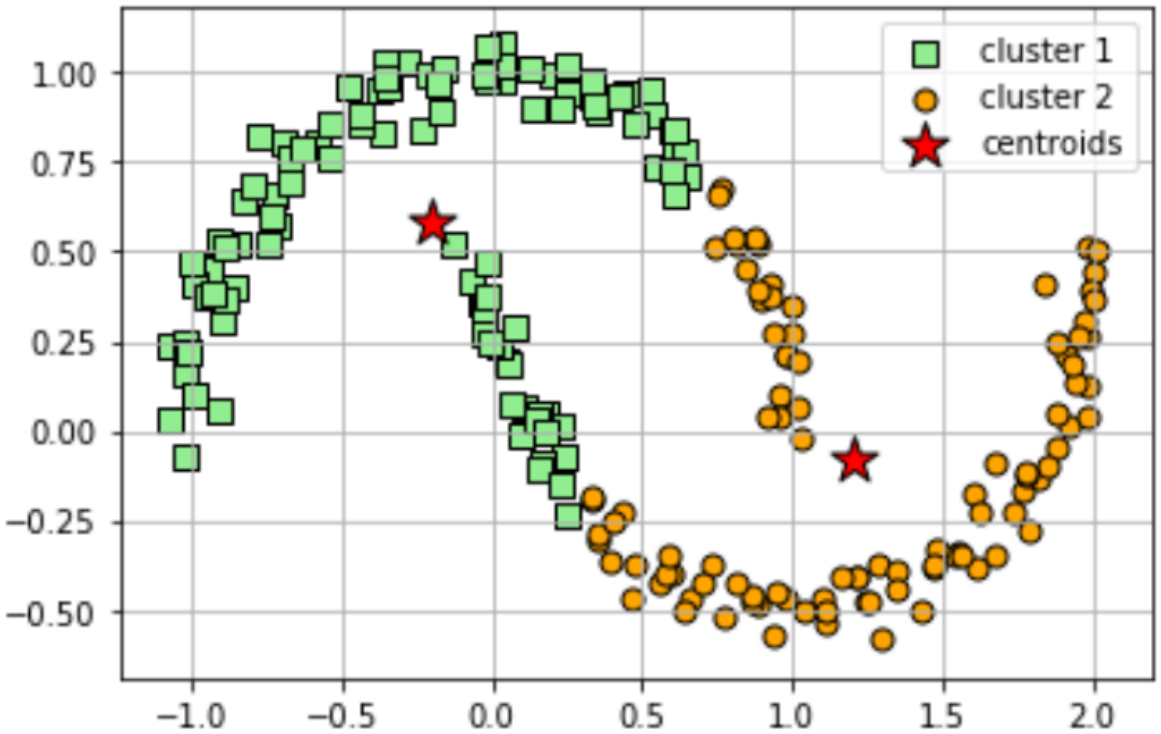
Clustering ensemble - exemple halfmoon

Dataset original



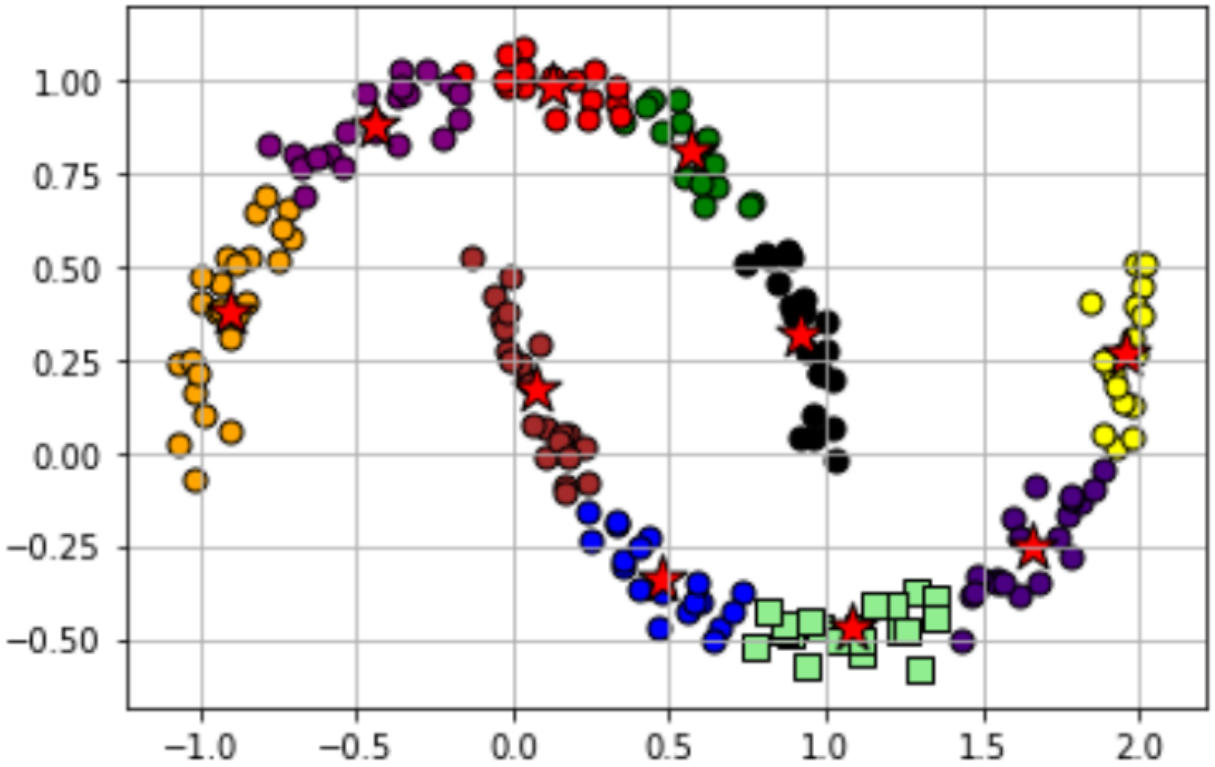
Clustering ensemble - exemple halfmoon

Clustering avec Kmeans, k=2

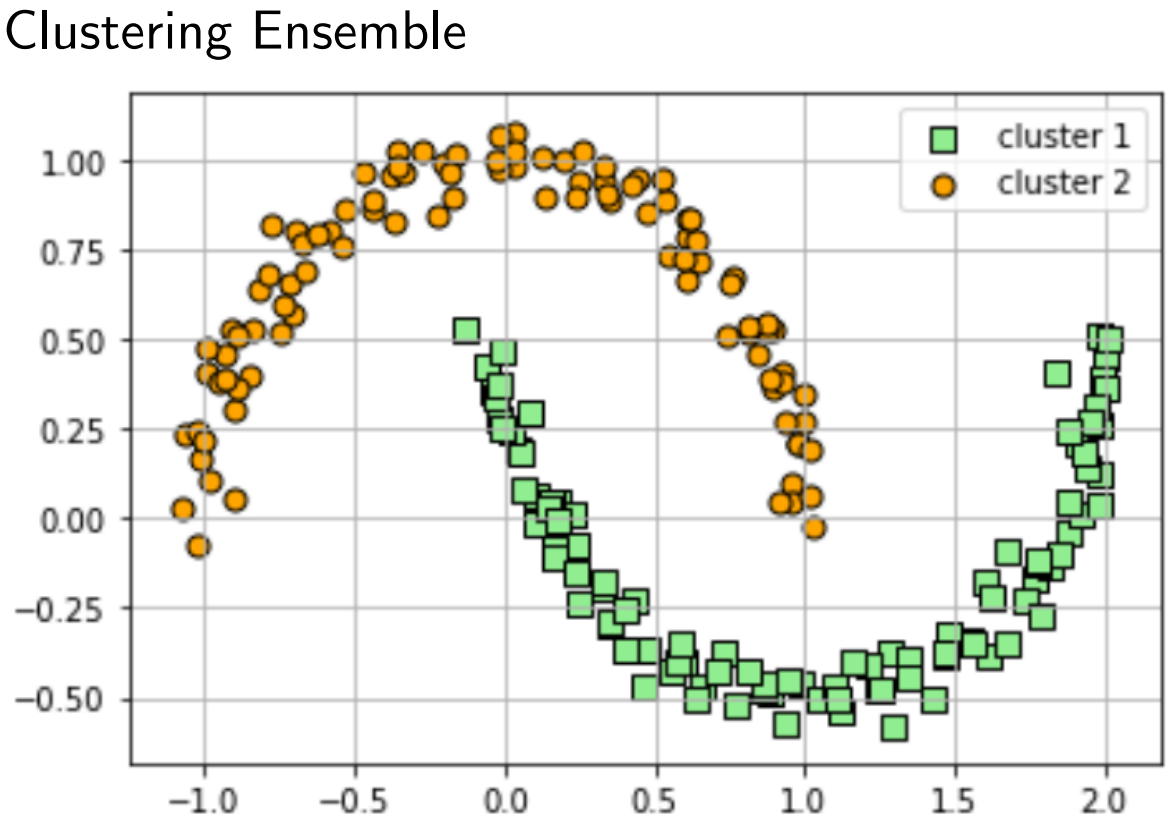


Clustering ensemble - exemple halfmoon

Clustering avec Kmeans, k=10

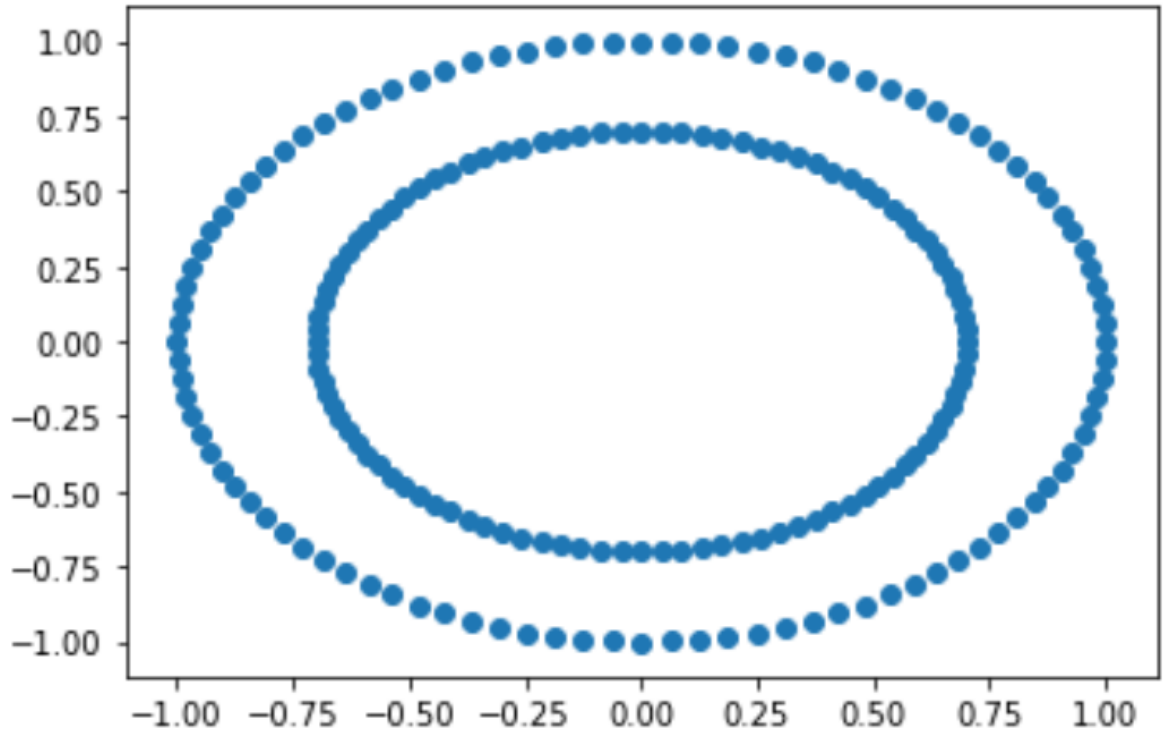


Clustering ensemble - exemple halfmoon



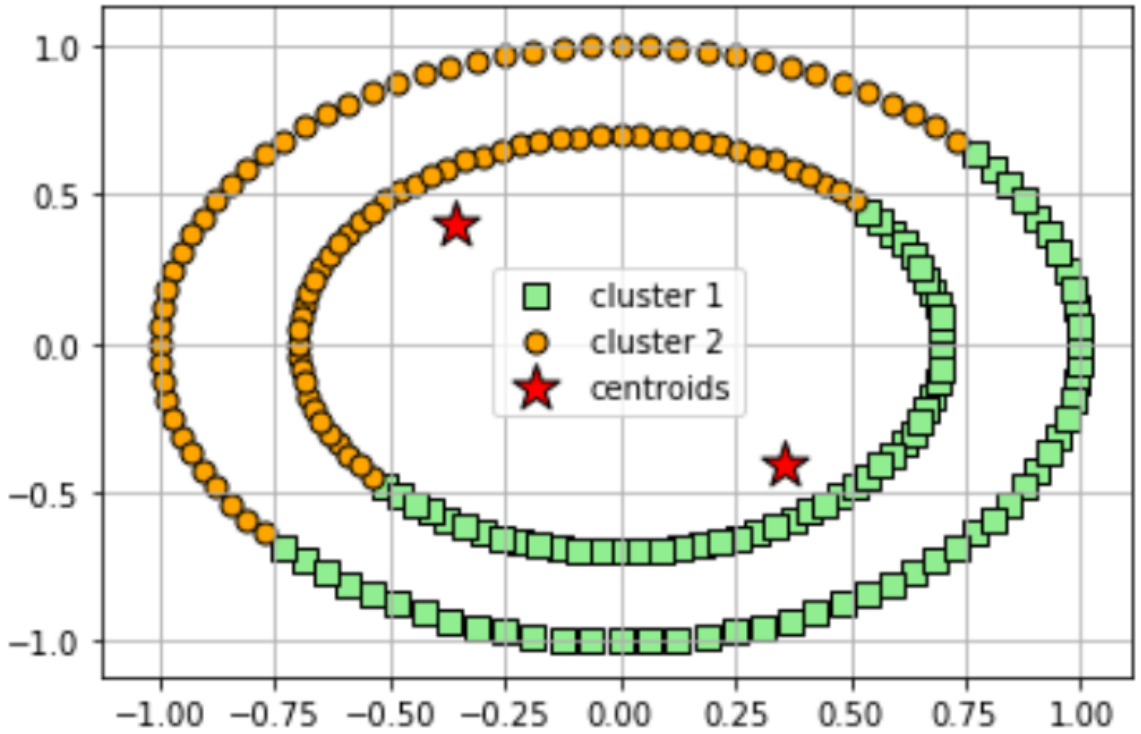
Clustering ensemble - exemple circle

Dataset original



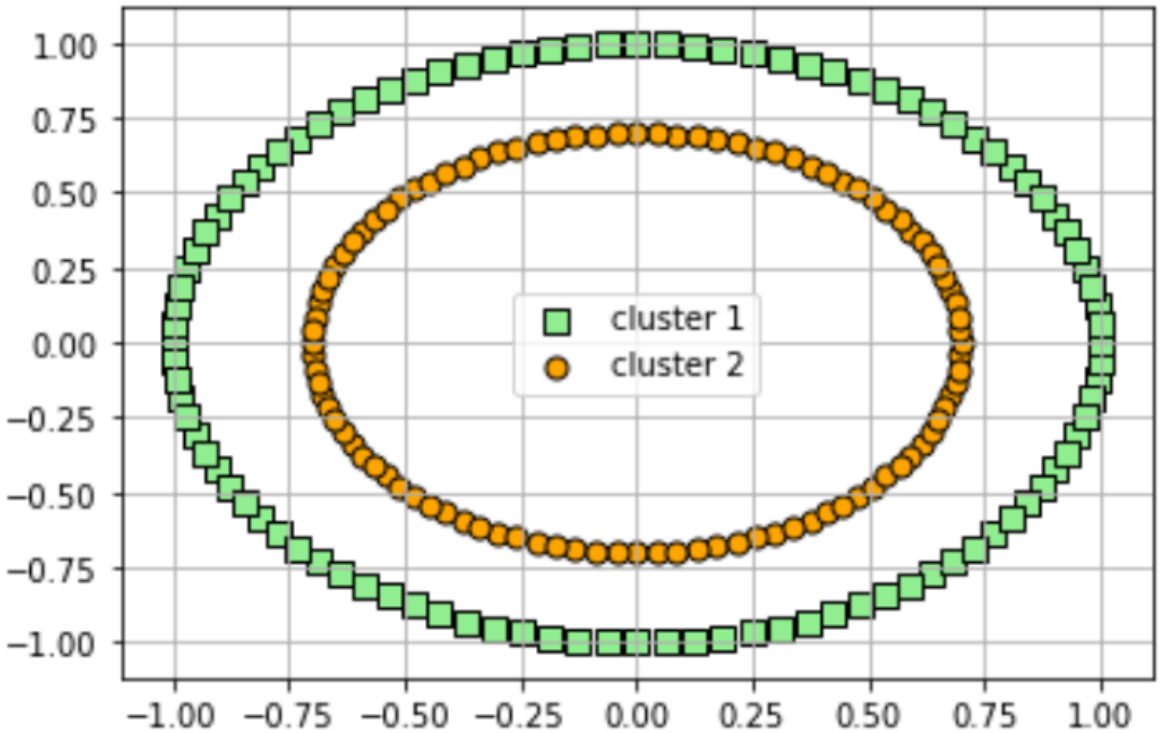
Clustering ensemble - exemple circle

Clustering avec Kmeans, k=2



Clustering ensemble - exemple circle

Clustering Ensemble



Clustering ensemble - Etapes

Toute méthode de clustering ensemble possède deux étapes: Génération et Fonction Consensus.

Clustering ensemble - Etapes

Toute méthode de clustering ensemble possède deux étapes: Génération et Fonction Consensus.

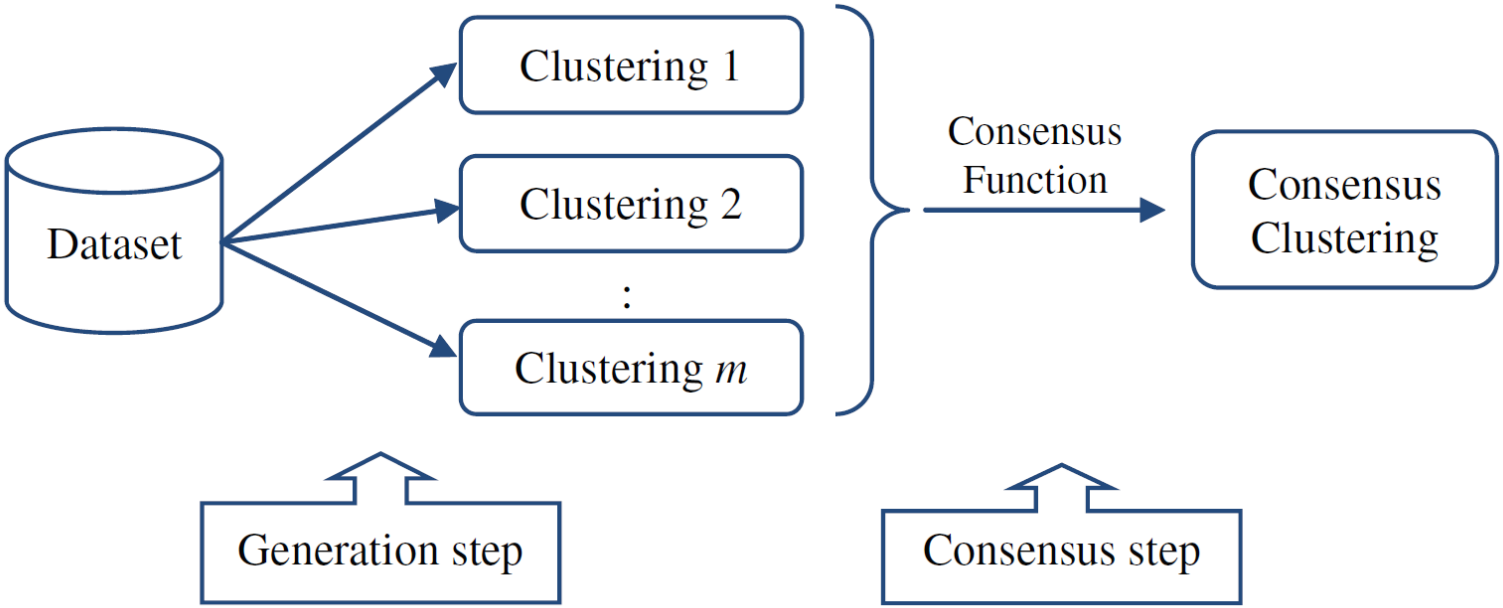


Diagramme du processus général du clustering ensemble

Tiré de l'article *A survey of clustering ensemble algorithms* publié en 2011 pas José Ruitz-Shulcloper et Sandro Vega-Pons.

Clustering ensemble - Base Partition Generation

Il existe plusieurs méthodes pour générer les partitions de base pour un ensemble de données X:

- Utiliser un algorithme différent pour chaque partition.
- Utiliser le même algorithme pour chaque partition mais avec des paramètres différents.
- Utiliser différentes représentations des objets de X, différents sous-ensemble d'objets ou encore des projections des objets sur différents sous-espaces.

Clustering ensemble - Fonction Consensus

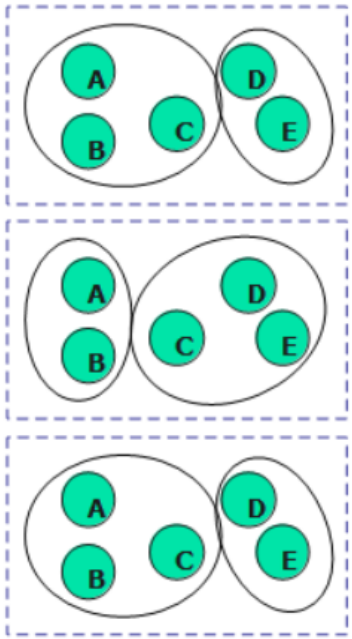
La fonction consensus: étape la plus importante d'un algorithme de clustering ensemble. Elle permet de calculer La partition consensus $P*$.

Il existe deux principales approches de fonction consensus :

- Co-occurrence
- Partition médiane

Co-occurrence

Partitions base

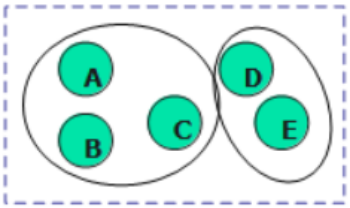


Matrice de Co-association

	A	*B*	*C*	*D*	*E*
A	-	3/3	2/3	0	0
B	3/3	-	2/3	0	0
C	2/3	2/3	-	1/3	1/3
D	0	0	1/3	-	3/3
E	0	0	1/3	3/3	-

Fonction consensus

Final partition



Clustering ensemble - Co-occurrence

Déterminer quel doit être le cluster associé à chaque objet dans la consensus partition. Pour cela, on analyse combien de fois un objet appartient à un cluster particulier ou le nombre de fois où deux objets appartiennent tous les deux au même cluster.

Voir par exemple *Convergence analysis of semi-supervised clustering ensemble.* de D. Chen, Y. Yang, H. Wang et A. Mahmood (2013) la technique de *evidence accumulation clustering* utilisant une matrice de co-association est introduit.

Clustering ensemble - evidence accumulation

Combiner les résultats de plusieurs clusterings en une seule partition en considérant chaque partition de base comme une indication sur l'organisation des données.

Clustering ensemble - evidence accumulation

Combiner les résultats de plusieurs clusterings en une seule partition en considérant chaque partition de base comme une indication sur l'organisation des données.

Mécanisme de vote pour combiner les partitions;

Matrice de co-association CAM:

$$CAM_{ij} = \frac{n_{ij}}{n}$$

avec n_{ij} le nombre de fois ou i et j sont dans le même cluster et n le nombre de points dans les données.

Partition consensus obtenue en appliquant un algorithme de clustering sur CAM.

Clustering ensemble - Median partition

La partition consensus est obtenue en résolvant un problème d'otimisation. La *median partition* = partition qui maximise la simililarité avec toutes les base partition.

Formellement, la partition médiane est définie comme:

$$P^* = arg\max_{P \in \mathbb{P}} \sum_{j=1}^{m} \Gamma(P, P_j)$$

, où Γ est une mesure de similarité entre partitions.

Sommaire

Clustering ensemble sous contraintes

Quasiment toutes les approches existantes utilisent des contraintes ML and CL.

Introduction des contraintes dans le processus:

- Lors de la création des partitions bases.
- Sélection des partitions bases à prendre en compte.
- Lors de la création de la matrice de co-association.
- Dans la fonction consensus.

Utilisation de critères d'optimisation

Sommaire

But du travail

- Intégrer les contraintes dans le calcul de la partition consensus, à partir de la matrice de co-association
- Appliquer le modèle développé dans [Dao, Vrain, Duong, AIJ 17]
  étant donnée une matrice de distance de dimension $n * n$, fournir une partition G de taille n qui satisfait les contraintes et optimise un critère d'optimisation.
- Modèle optimisé pour le critère du diamètre avec une matrice de dissimilarité
  - $\rightarrow$ transformation de la matrice de co-association en une matrice de dissimilarité
  - $\rightarrow$ adaptation pour optimiser la marge (single link pour obtenir des formes complexes)

Modèle

- Données:
  - nombre d'éléments n
  - k_{min} et k_{max}
  - matrice de distance
- Variables: variables entières $G_1, ..., G_n$ dont le domaine est l'ensemble d'entiers $[1, k_{max}]$.
- Critère d'optimisation: maximiser la marge (séparation minimum entre clusters)
- Contraintes de partition:
  - imposer que le résultat soit un partition
  - éviter les symétries
- Contraintes utilisateur: Must-Link et Cannot-Link
  Exemple : constraint G[10]=G[18];

Critère d'optimisation

Critère d'optimisation: Séparation minimum

- var min(i,j in 1..n where $i < j$)(dist[i,j]) .. max(i,j in 1..n where $i < j$)(dist[i,j]) : S;
- constraint forall (i,j in 1..n where $i < j$) (dist[i,j] $<$ S $->$ G[i] $==$ G[j]);
- solve maximize S;

Sommaire

Superpoints

On souhaite réduire la taille de la matrice de distance. Pour cela, on regroupe les points liés par des contraintes ML en des **superpoints**.
On obtient après agrégation une matrice de distance plus petite que la matrice originale car les points appartenant aux même superpoints y ont été fusionnés.

superpoint = ensemble de points reliés par des contraintes ML

$$dist(S1, S2) = min_{p1 \in S1, p2 \in S2} dist(p1, p2))$$

Chaque contrainte CL(p1,p2) devient CL(S1,S2) avec $p1 \in S1, p2 \in S2$).

Exemple

Soit une liste de points $Y = [1, 2, 3, 4, 5, 6]$.

Soit un ensemble de contraintes Must-Link $ML = [(1, 2), (2, 3)]$

Soit un ensemble de contraintes Cannot-Link $CL = [(2, 4), (4, 5)]$

Après agrégation, on obtient:

- superpoints

$$S1 = (1, 2, 3), S2 = (4), S3 = (5), S4 = (6)$$

- liste de contraintes Cannot-Link mise à jour

$$CL' = [(S1, S2), (S2, S3)]$$

Sommaire

Stratégie de pré-traitement

Afin de réduire le temps d'exécution de MiniZinc, on développe deux stratégies de pré-traitement différentes dont le but est de réordonner les points (et donc la matrice distance) de telle sorte que Minizinc trouve plus rapidement la solution optimale.

Première stratégie

-Trouver les deux points $s0$ et $s1$ les plus éloignés et les mettre dans une liste *seeds*.

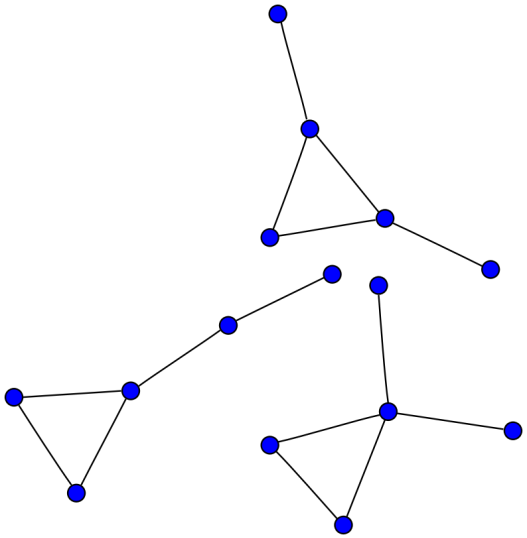
-Ajouter dans *seeds* le point avec la plus grande somme de distances avec les éléments déjà présent dans la liste. A répéter $k-2$ fois.

-Le nouvel ordre sera constitué d'abord des graines, puis des autres points tel que le point à la position z est le point le plus proche d'un des éléments entre les positions 0 et $z-1$.

Deuxième stratégie - définitions

Dans la seconde stratégie, on utilise quelques notions de théories des graphes.

Une **composante connexe** CC est un sous-graphe connexe maximal d'un graphe non orienté, c'est à dire que pour tous sommets u, v de CC, il existe un chemin reliant u à v.

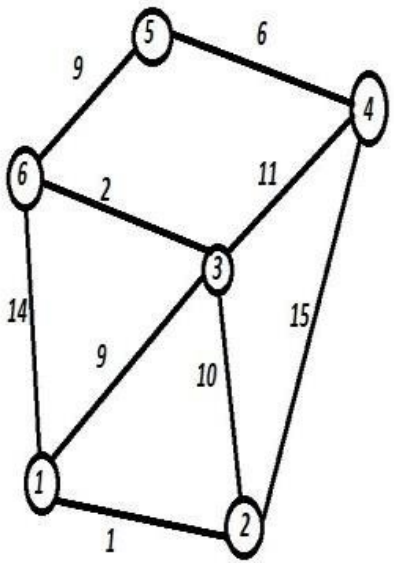


Ci dessus, un graphe comportant 3 composantes connexes.

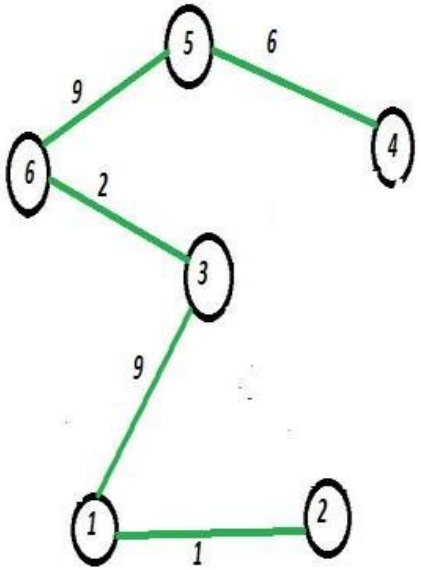
Source: Wikipedia

Deuxième stratégie - définitions

L'**arbre couvrant de poids minimum** (ou minimum spanning tree) d'un graphe G est un arbre qui connecte tous les sommets de G par des arêtes de G et dont la somme des poids des arêtes est minimale.



Undirected graph G = (V,E)



The minimal spanning tree
The tree cost is 33

Source: Optimization Algorithm's Problems: Comparison Study: R. N. Naby, R. Azad, S. Saeed, R. N. Naby

Deuxième stratégie - déroulement

- Créer l'arbre couvrant de poids minimum du graphe représentant la matrice de distance.

Deuxième stratégie - déroulement

- Créer l'arbre couvrant de poids minimum du graphe représentant la matrice de distance.

- Enlever les $k-1$ arrêtes les plus lourdes. On aura alors k composantes connexes.

Deuxième stratégie - déroulement

- Créer l'arbre couvrant de poids minimum du graphe représentant la matrice de distance.

- Enlever les $k-1$ arrêtes les plus lourdes. On aura alors k composantes connexes.

- Pour chaque composante connexe, on choisit un des points qu'elle contient en tant que "graine". Ces k points seront les premiers à apparaître dans le nouvel ordre.

Deuxième stratégie - déroulement

- Créer l'arbre couvrant de poids minimum du graphe représentant la matrice de distance.

- Enlever les $k-1$ arrêtes les plus lourdes. On aura alors k composantes connexes.

- Pour chaque composante connexe, on choisit un des points qu'elle contient en tant que "graine". Ces k points seront les premiers à apparaître dans le nouvel ordre.

- Ajouter dans le nouvel ordre tous les points apparaissant dans au moins une contrainte CL, en commençant par les points appartenant à la première composante connexe, puis à la deuxième ...

Deuxième stratégie - déroulement

- Créer l'arbre couvrant de poids minimum du graphe représentant la matrice de distance.

- Enlever les $k-1$ arrêtes les plus lourdes. On aura alors k composantes connexes.

- Pour chaque composante connexe, on choisit un des points qu'elle contient en tant que "graine". Ces k points seront les premiers à apparaître dans le nouvel ordre.

- Ajouter dans le nouvel ordre tous les points apparaissant dans au moins une contrainte CL, en commençant par les points appartenant à la première composante connexe, puis à la deuxième ...

- Ajouter le reste des points également dans leur ordre de composante connexe.

Sommaire

Baseline

Avant d'utiliser un modèle de PPC, on utilise une baseline

Deux versions: une sans contraintes et une avec des contraintes ML et CL.

Baseline

Avant d'utiliser un modèle de PPC, on utilise une baseline

Deux versions: une sans contraintes et une avec des contraintes ML et CL.

Dans les deux cas, on commence par générer un nombre $nBP = 50$ de partitions de base avec l'algorithme Kmeans en sélectionnant pour chacune un nombre de clusters k aléatoirement entre 2 et la racine carrée du nombre d'éléments (limité à 50).

On répète les deux versions 10 fois.

Baseline - Sans contraintes

- Générer les nBP partitions de base
- Calculer l'ARI pour chacune des partitions de base et en extraire des statistiques (minimum, maximum, moyenne, écart type et médiane).
- Générer la matrice de co-assocation CAM, où $CAM(i,j) = nCO/nBP$ avec nCO le nombre de partitions dans lesquelles les points co-occurrent.
- Transformer CAM en une matrice de distance
- Utiliser Single Link sur la matrice de dissimilarité afin d'obtenir la partition consensus
- Calculer l'ARI et la séparation minimum de la partition consensus.

Baseline - Avec contraintes

- Générer 5 ensemble de contraintes ML et CL. On en crée un nombre $nContraintes = 50$.
- Pour les 10 lancements, on garde les mêmes matrices de distance que dans la version sans contraintes afin de pouvoir comparer les performances. De plus, on utilise les 5 mêmes ensembles de contraintes.

Baseline - Avec contraintes

- Récupérer les partitions de base de la version sans contraintes.
- Générer la matrice de co-assocation CAM, où $CAM(i,j) = nCO/nBP$ avec nCO le nombre de partitions dans lesquelles les points co-occurrent.

- Appliquer les contraintes :
  $\forall ML(x_1, x_2), CAM(x_1, x_2) = 1$ et $CAM(x_2, x_1) = 1$
  $\forall CL(x_1, x_2), CAM(x_1, x_2) = 0$ et $CAM(x_2, x_1) = 0$

- Transformer CAM en une matrice de distance
- Utiliser Single Link sur la matrice de distance afin d'obtenir la partition consensus
- Calculer l'ARI et la séparation minimum de la partition consensus.

Sommaire

Résultats expérimentaux - Baseline

Tableau d'ARI:

dataset	**Sans contraintes**	**Avec contraintes**
halfmoon	1	1
s2c2sc13	0.505	0.598
iris	0.581	0.698
glass	0.244	0.244

Résultat temps d'éxécution

Avec les données Halfmoon et 5 ensembles de contraintes $CS0 \ldots CS4$,

	$CS0$	$CS1$	$CS2$	$CS3$	$CS4$
default	15s	470s	670s	200s	99s
aggregated	20s	380s	1100s	180s	160s
reordered1	3.6s	2.3s	1.8s	3.8s	2.4s
reordered2	1.1s	1.8s	1.8s	2s	20s

Les réordonnancements débouchent sur de mauvais ARIs.

Sommaire

Conclusion

Les résultats expérimentaux doivent être vérifiés:

- Comprendre pourquoi la version avec Superpoints est parfois plus longue que la version par défaut.

- Chercher pourquoi l'ARI des réordonnancements est mauvais.

Une fois les vérifications faites, continuer à faire tous les tests.