Ensemble method for constrained clustering

ACCE: Anchored Constrained Clustering Ensemble

Mathieu GUILBERT

Université d'Orléans - LIFO

Introduction

Cette présentation évoque une partie du travail réalisé depuis le début de mon doctorat au LIFO depuis septembre 2021 dans le cadre du Projet Involvd.

Objectifs

- Poursuivre le travail commencé durant mon stage de février à juillet 2021.
- Pouvoir fusionner plusieurs partitions intéressantes tout en respectant des contraintes données par un expert.
- Proposer et tester une nouvelle méthode de clustering ensemble sous contrainte.
- A terme, utiliser cette nouvelle méthode sur données Involvd.

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

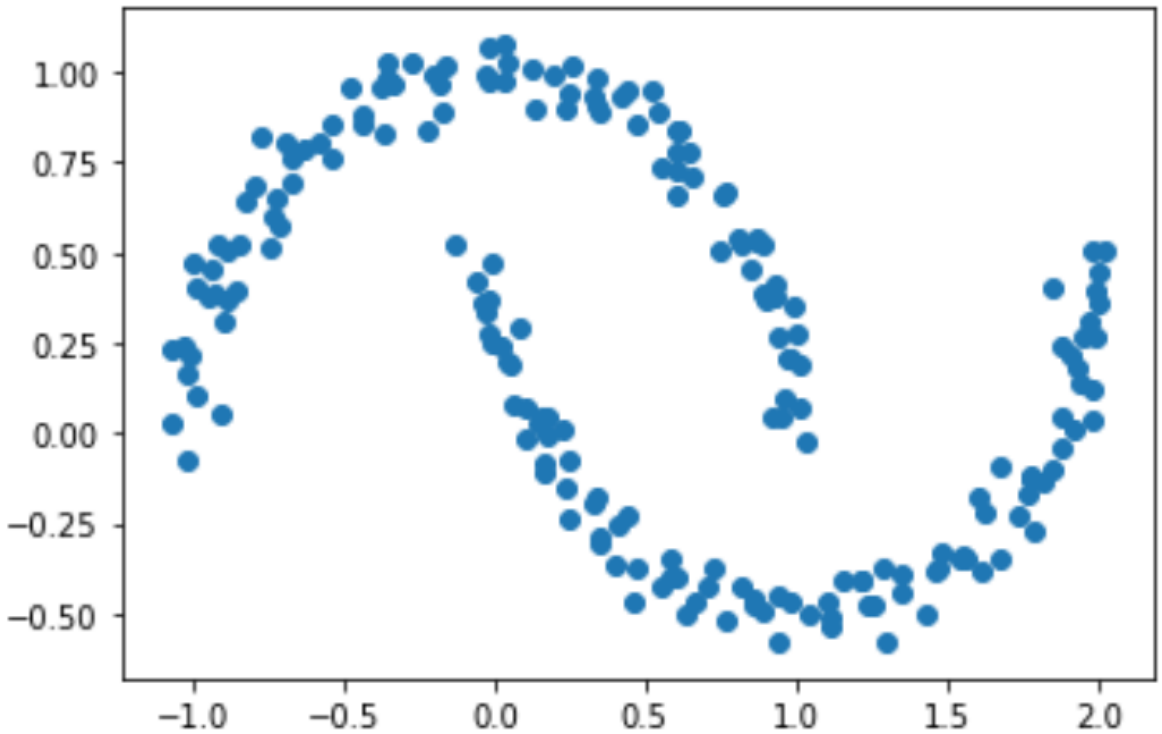
Nous utiliserons les notations suivantes.

- $X = \{x_1, x_2, ..., x_n\}$ 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, $(\mathbb{P} \subset \mathbb{P}_X)$.

A partir d'un ensemble de partitions de bases, on cherche à créer une *partition consensus* $P* \in \mathbb{P}_X$ contenant les propriétés récurrentes de ces partitions.

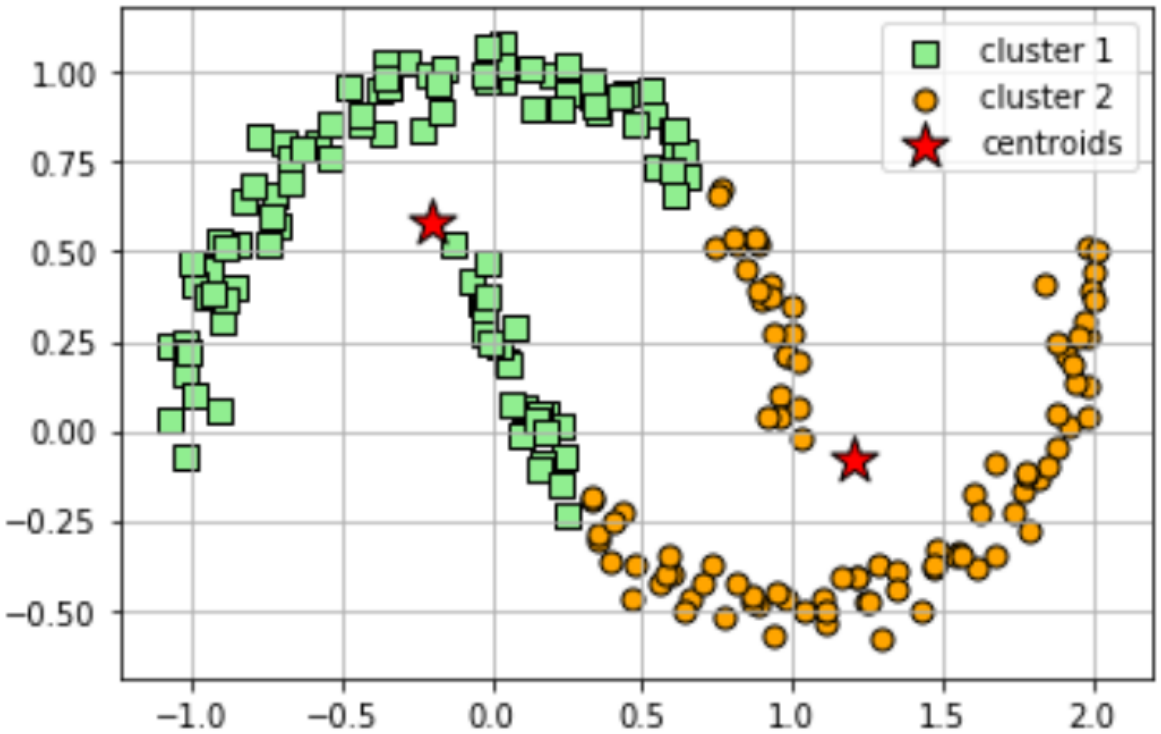
Clustering ensemble - exemple halfmoon

Dataset original



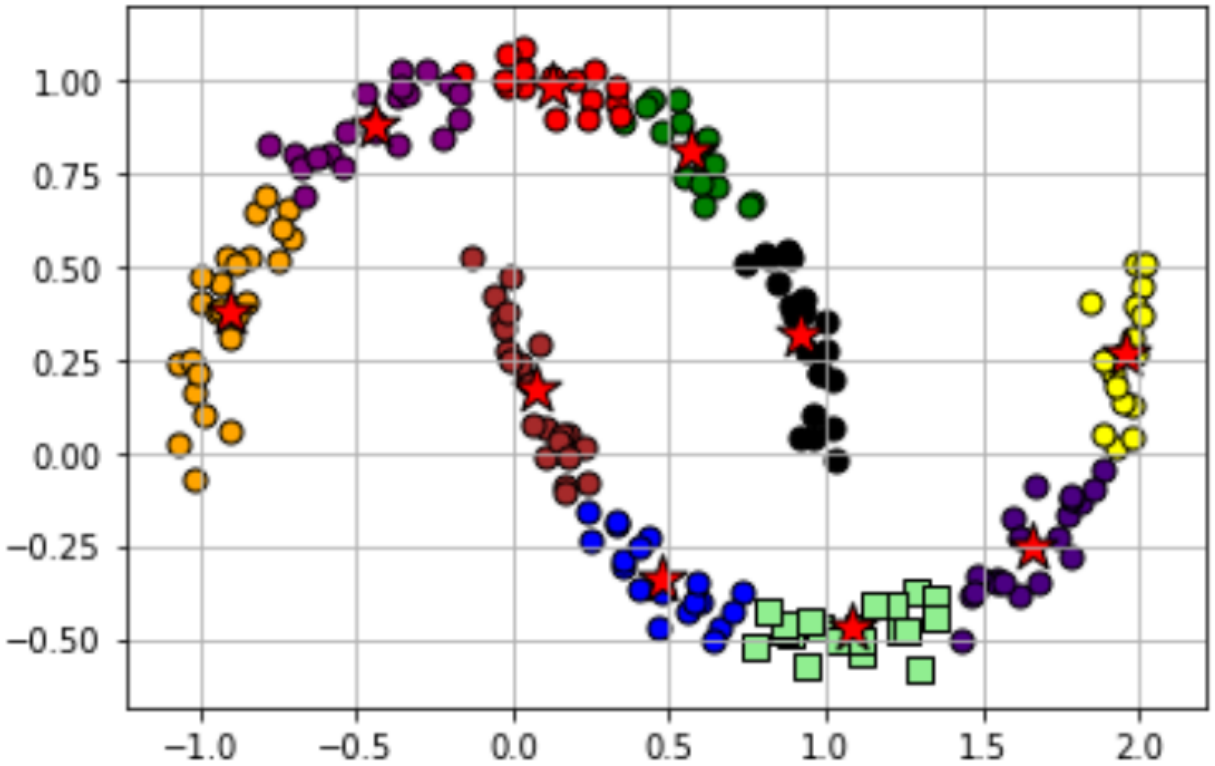
Clustering ensemble - exemple halfmoon

Clustering avec Kmeans, k=2



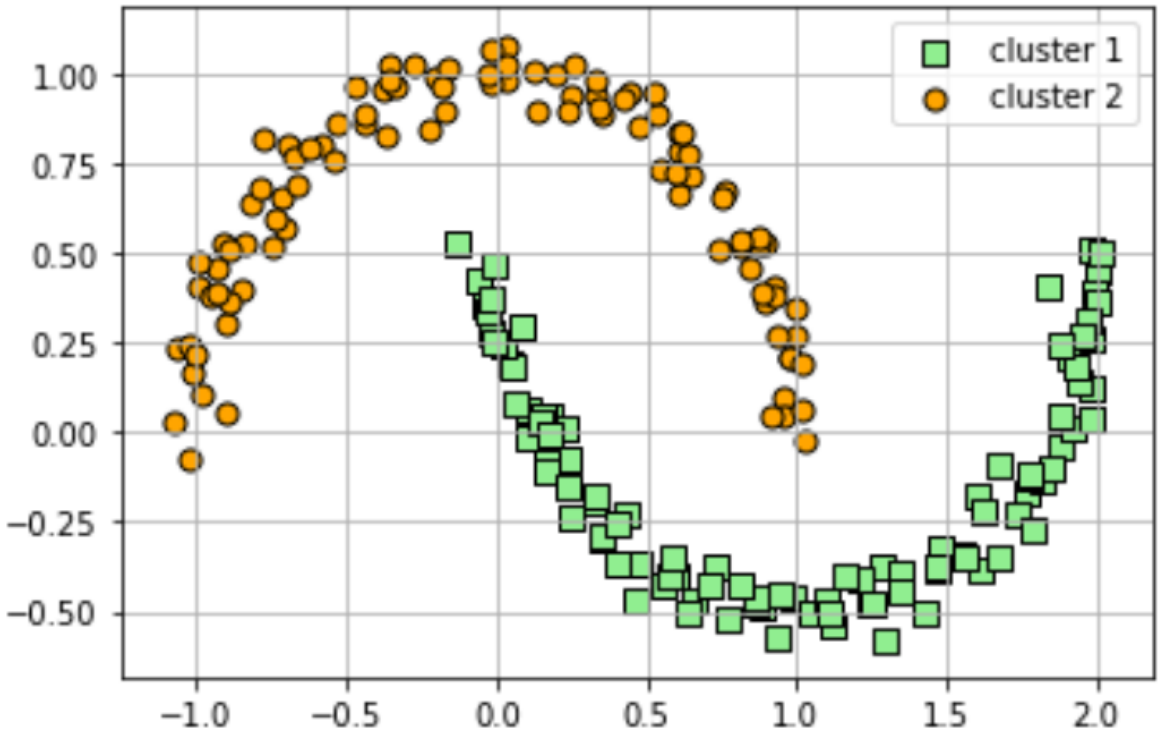
Clustering ensemble - exemple halfmoon

Clustering avec Kmeans, k=10



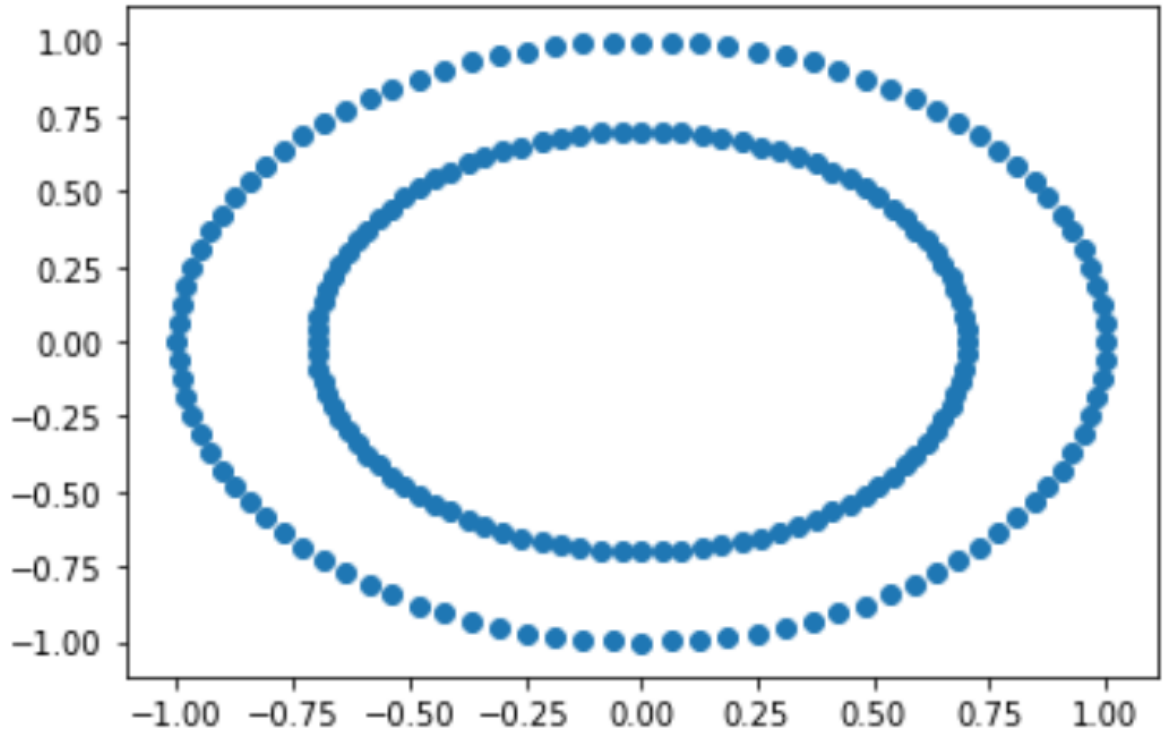
Clustering ensemble - exemple halfmoon

Clustering Ensemble



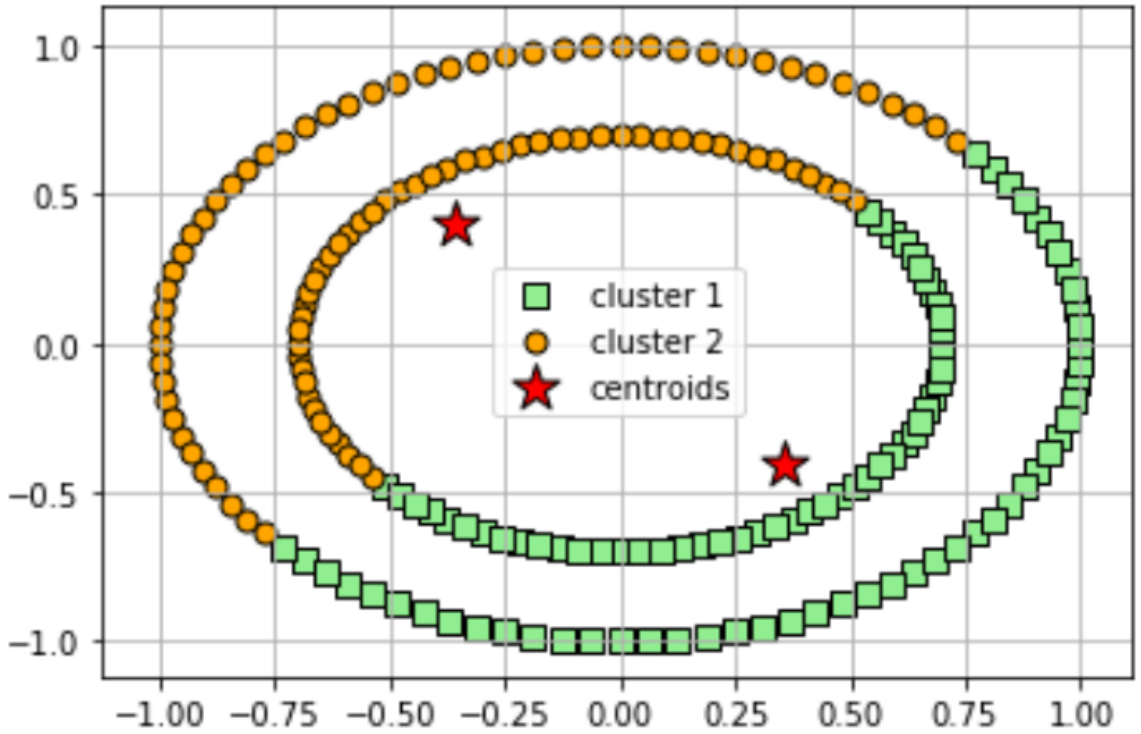
Clustering ensemble - exemple circle

Dataset original



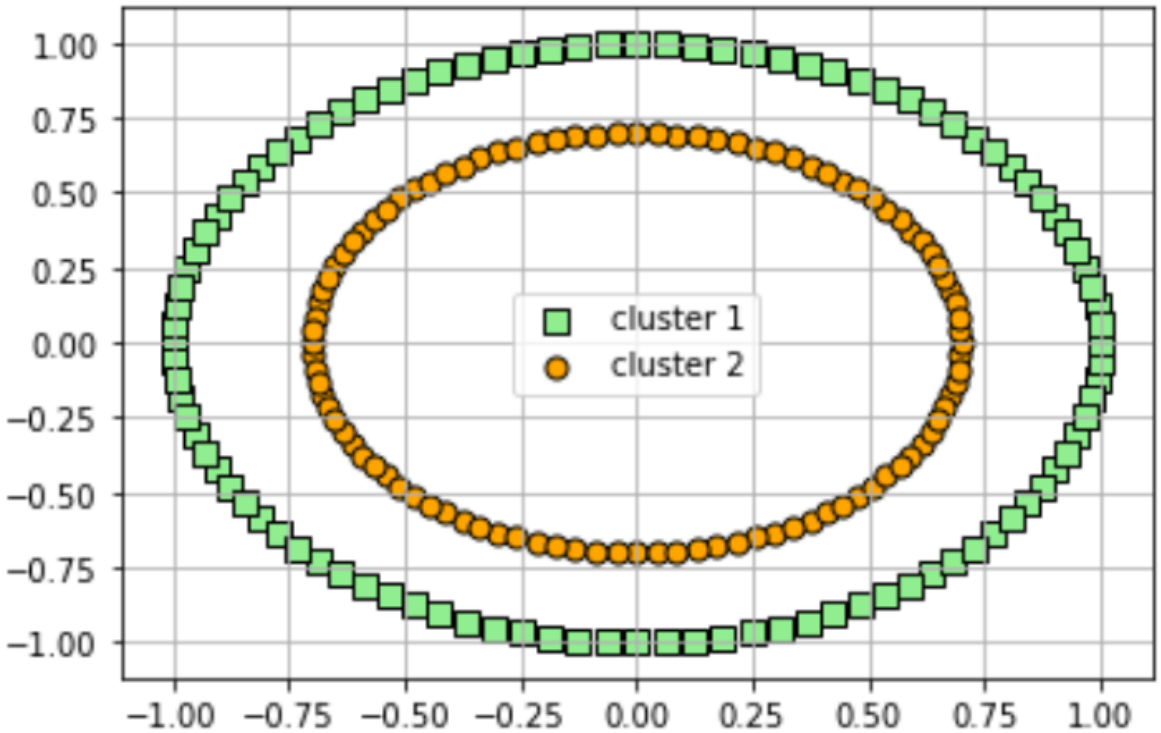
Clustering ensemble - exemple circle

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.

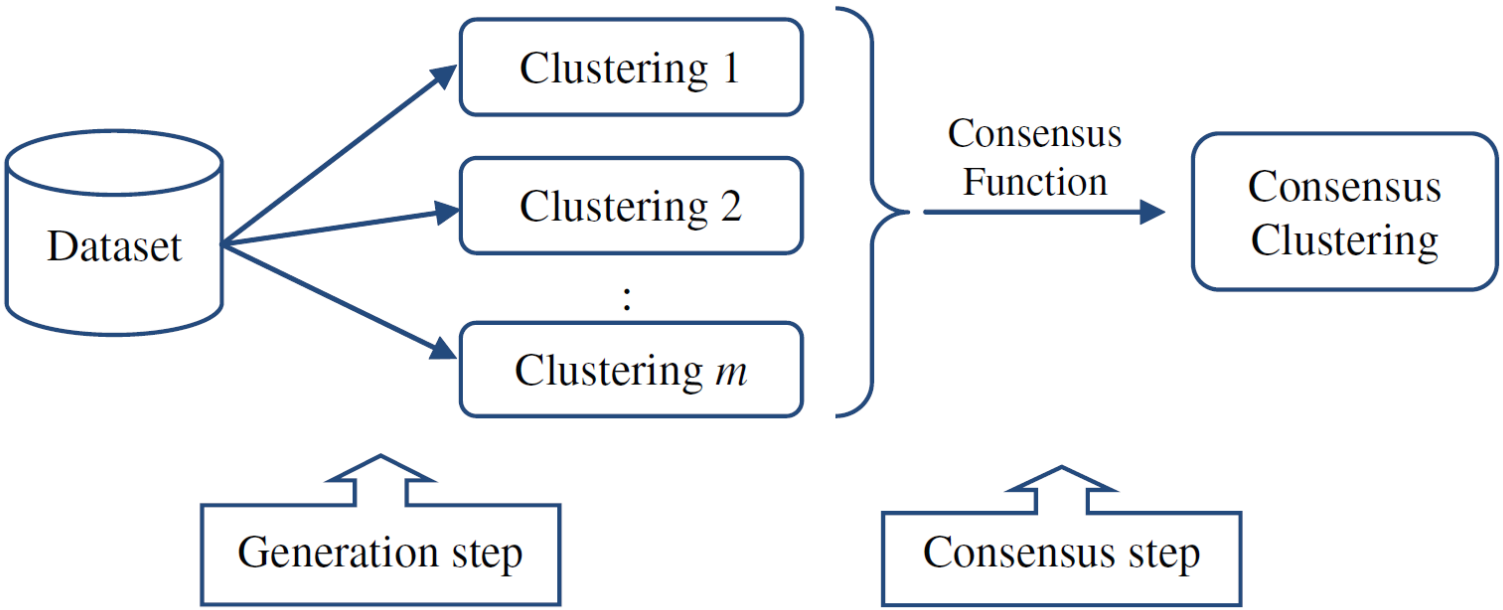


Diagramme du processus général du clustering ensemble

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

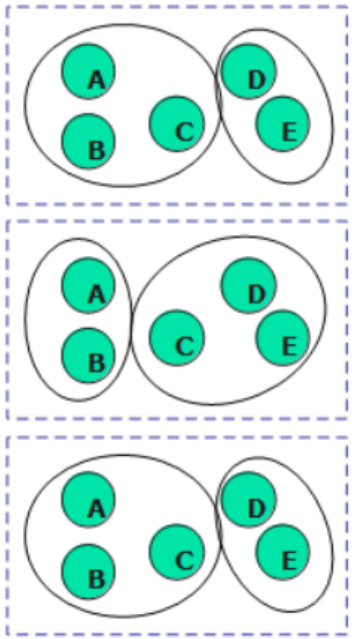
La fonction consensus: étape la plus importante d'un algorithme de clustering ensemble. Elle permet de calculer La partition consensus $P*$.

Il existe deux principales approches de fonction consensus :

- Co-occurrence
- Partition médiane

Co-occurrence

Partitions base

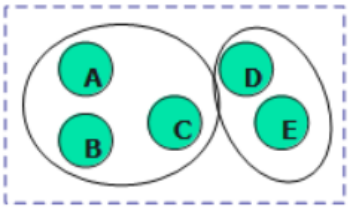


Matrice de Co-association

	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 *Combining multiple clusterings using evidence accumulation* de Fred, Ana LN and Jain, Anil K (2005): Evidence Accumulation

Clustering ensemble - evidence accumulation EAC

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:

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

avec n_{ij} le nombre de fois ou i et j sont dans le même cluster et nBP le nombre de partitions de base.

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'optimisation.
median partition = partition qui maximise la similarité 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.

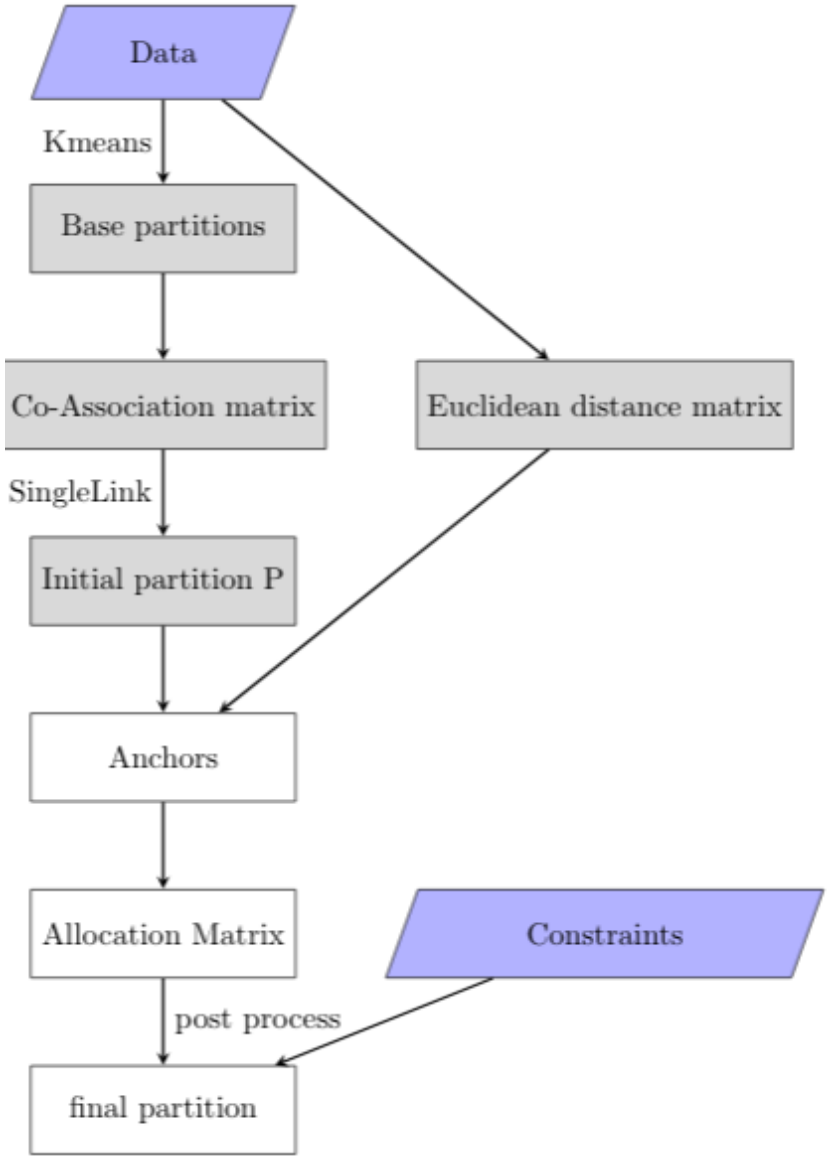
Sommaire

Proposition initiale

- 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 dis-similarité de dimension $n * n$, fournir une partition $\mathbf{G}$ 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/séparation minimum (s'apparente à single link pour obtenir des formes complexes)
- Problème rencontré à cause du critère d'optimisation

Sommaire

Diagramme

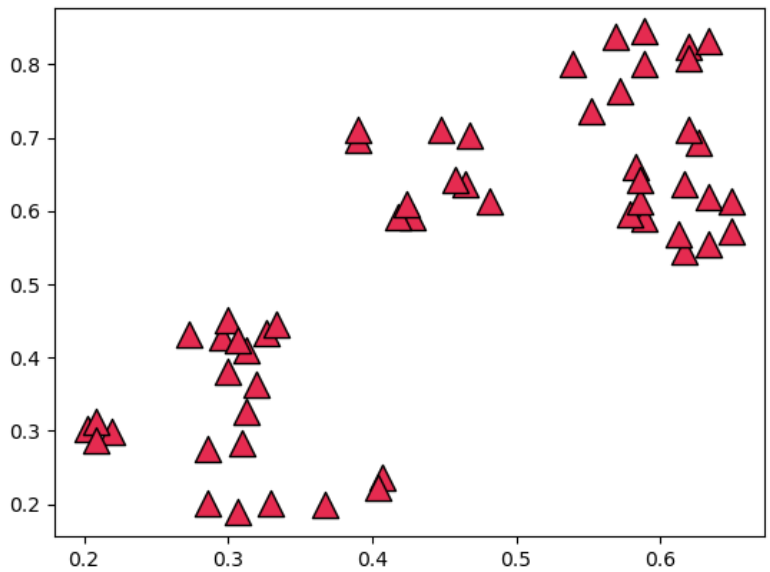

Data
Kmeans
Base partitions
Co-Association matrix
Euclidean distance matrix
SingleLink
Initial partition P
Anchors
Allocation Matrix
Constraints
post process
final partition

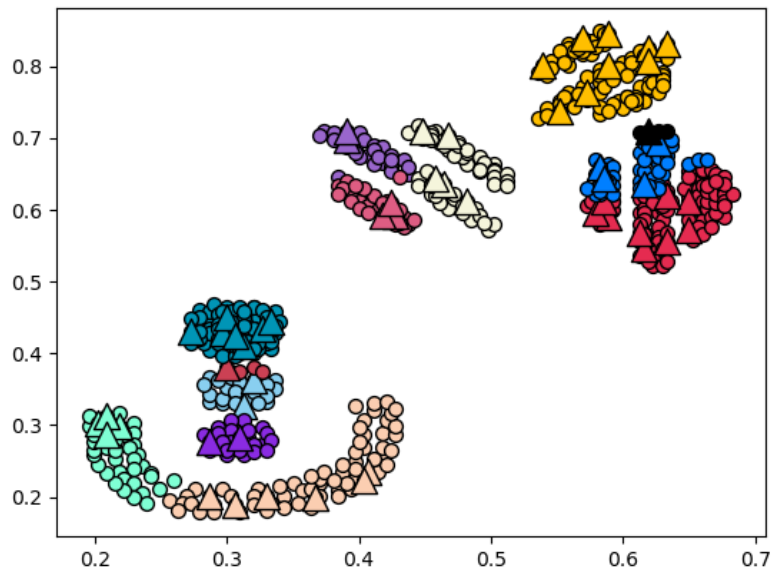
Ancres

Ancres = points jugés "représentatifs" de leur voisinage.

2 manières de calculer les ancres:

1. Total: Faire SingleLink sur les données pour obtenir H clusters, qui seront chacun représentés par une ancre.
2. Local: Pour chaque cluster *clust*, faire SingleLink sur les points lui appartenant en $|clust|/div$ clusters qui seront chacun représentés par une ancre,avec *div* un paramètre représentant la proportion de points qui deviendront des ancres.





Matrice d'Allocation

Matrice M de taille $N * K$ ou M_{ik} représente la probabilité que le point i appartiennent au cluster k, avec $k \in \{1..K\}$.

M_{ik} est la plus petite distance euclidienne entre i et une ancre appartenant au cluster k normalisée entre 0 et 1 et inversée.

Partition finale

3 critères utilisant la matrice d'allocation, avec P la partition original et $\mathbf{G}$ le résultat:

(1) maximize $\sum_{i=1}^{N} \sum_{k=1}^{K} M_{ik} * I(\mathbf{G}[i] = k)$

Maximiser somme des probabilités que les points soient dans leurs clusters dans $\mathbf{G}$.

(2) minimize $max(M_{ik} * I(P[i] \neq \mathbf{G}[i] \,\&\, P[i] = k))$

Changer l'affectation des points qui avaient la plus petite probabilité d'appartenir au bon cluster dans P.

(3) maximize $min(M_{ik} * I(P[i] \neq \mathbf{G}[i] \,\&\, \mathbf{G}[i] = k))$

Change l'affectation des points qui ont la plus grande probabilité d'appartenir à leurs nouveaux clusters.

Méthodes de Postprocess

Différentes méthodes de postprocess pour appliquer ces critères:

1. Programmation Linéaire (ILP)
2. Programmation Par Contrainte (PPC)

- ILP

Travaux de Nguyen-Viet-Dung Nghiem:

Utilisée pour critère d'optimisation **(1)**.

- PPC

Utilisée pour critères **(2)** et **(3)**.

Implémentés en Minizinc.

Sommaire

Baseline

Avant d'utiliser notre approche, on utilise une baseline

Deux axes: sans contraintes et avec contraintes ML et CL.

Dans les deux cas, on commence par générer un nombre $nBP = 50$ de partitions de base avec les algorithmes Kmeans ou COPKmeans 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 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 dissimilarité.
- Utiliser Single Link sur la matrice de dissimilarité afin d'obtenir la partition consensus.
- Calculer l'ARI 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

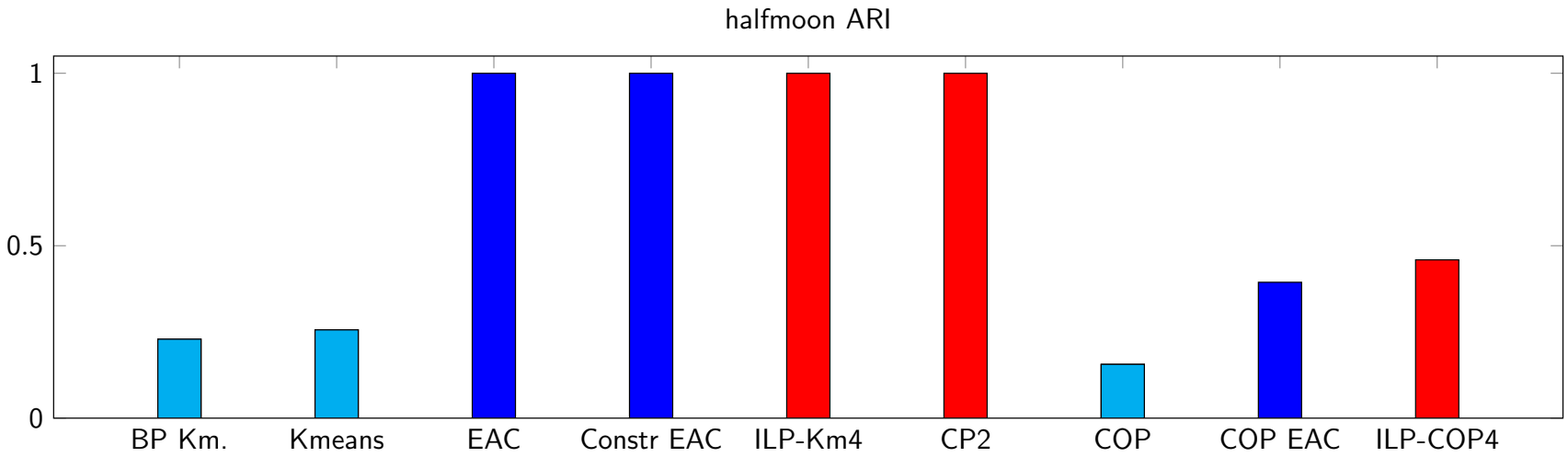
- Récupérer les partitions de base de la version sans contraintes.
- Générer la matrice de co-assocation CAM, où $CAM(i,j) = nCO/nBP$ avec nCO le nombre de partitions dans lesquelles les points co-occurrent.

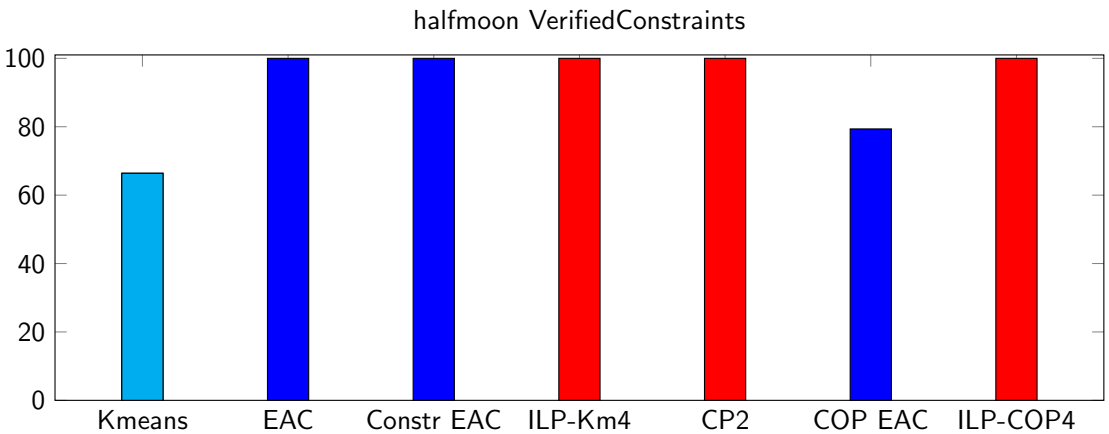
- Appliquer les contraintes :
  $\forall ML(x_1, x_2), CAM(x_1, x_2) = 1$ et $CAM(x_2, x_1) = 1$
  $\forall CL(x_1, x_2), CAM(x_1, x_2) = 0$ et $CAM(x_2, x_1) = 0$

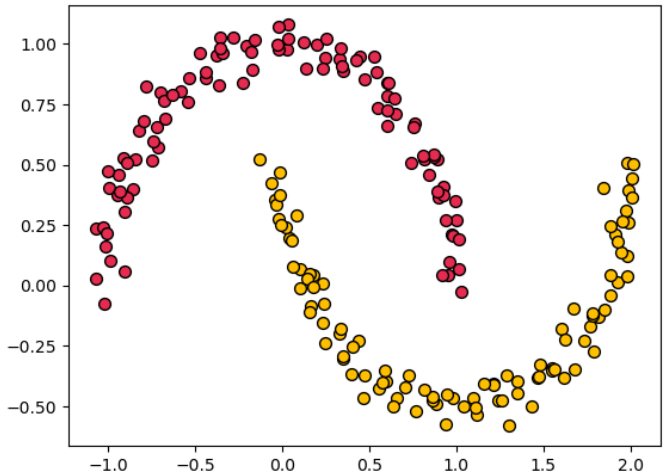
- Transformer CAM en une matrice de dis-similarité
- Utiliser Single Link sur la matrice de dis-similarité afin d'obtenir la partition consensus
- Calculer l'ARI de la partition consensus.

Sommaire

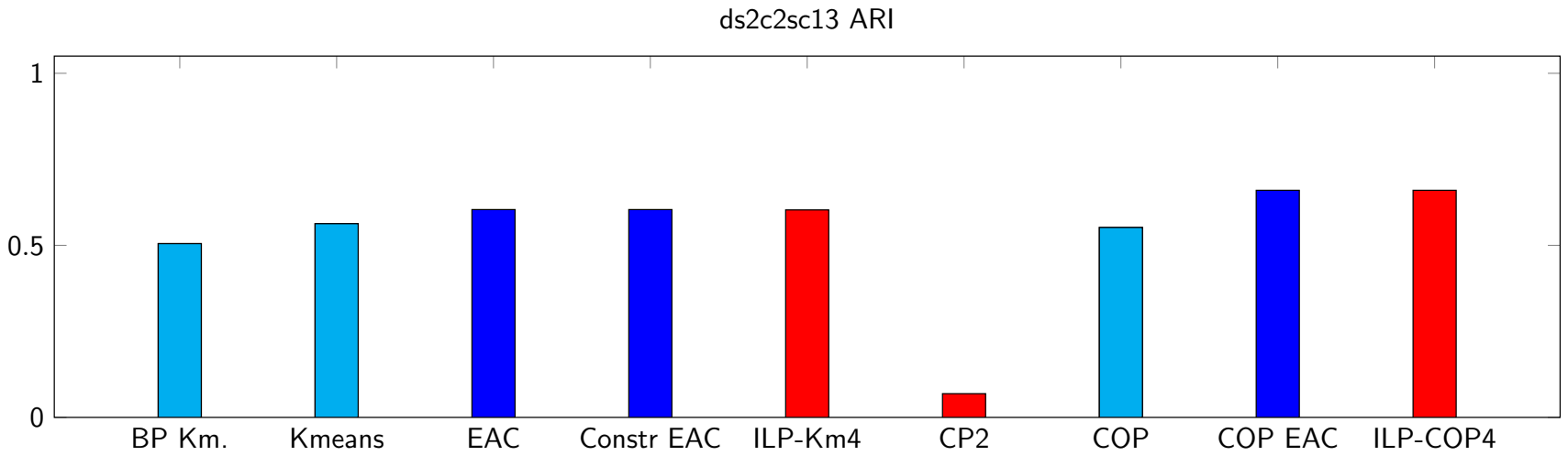
Résultats - halfmoon

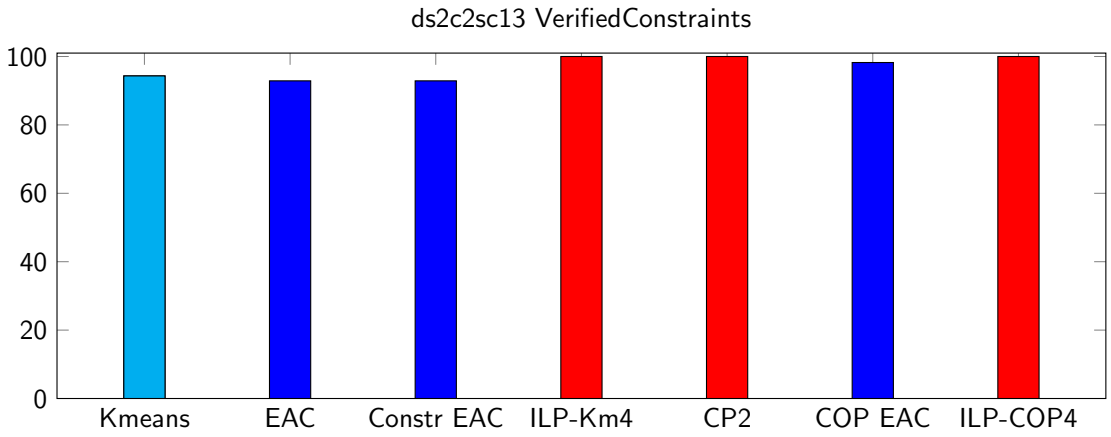


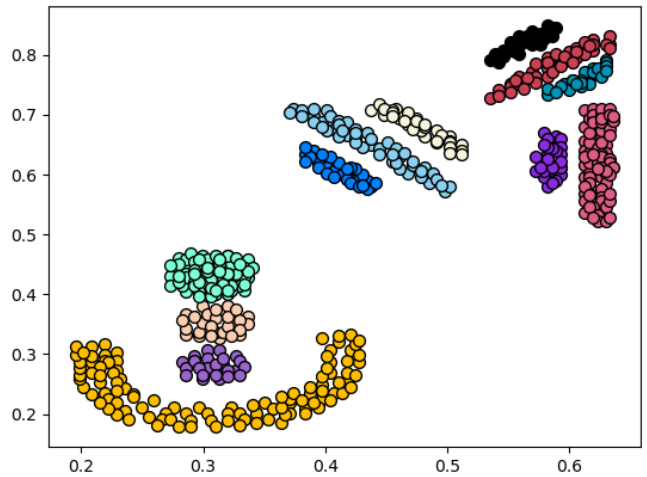




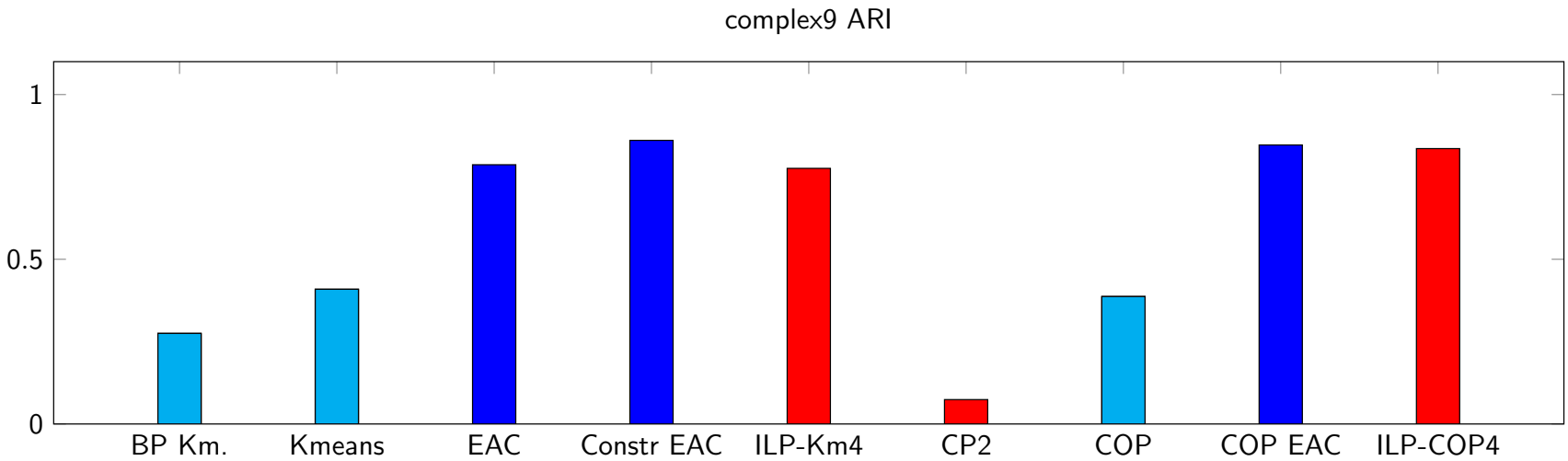
Résultats - ds2c2sc13

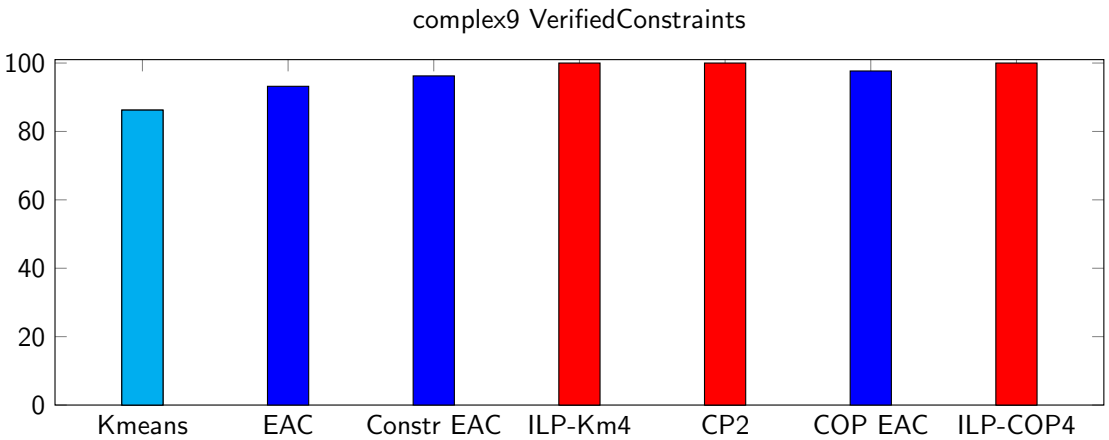


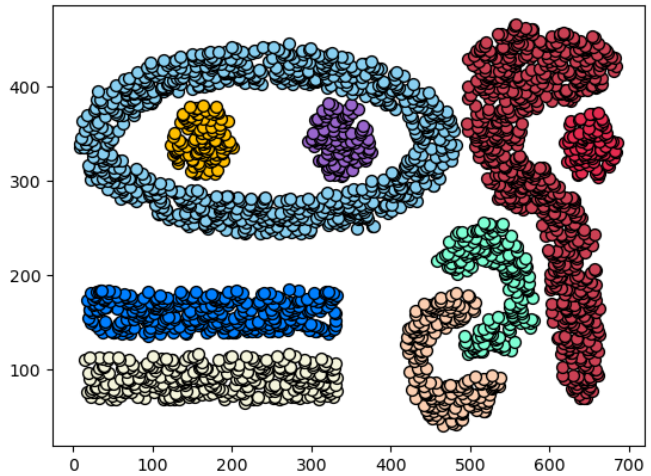




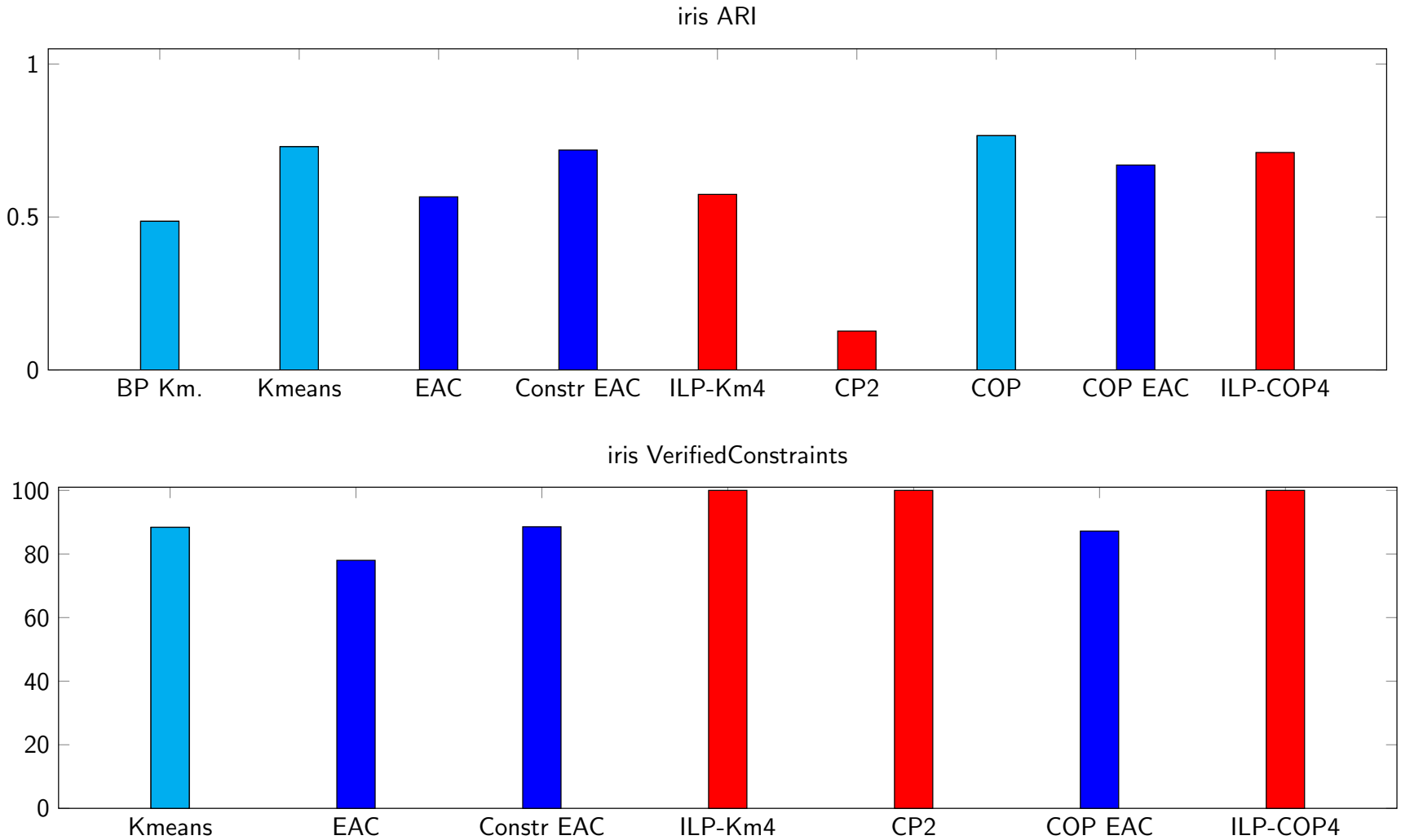
Résultats - complex9



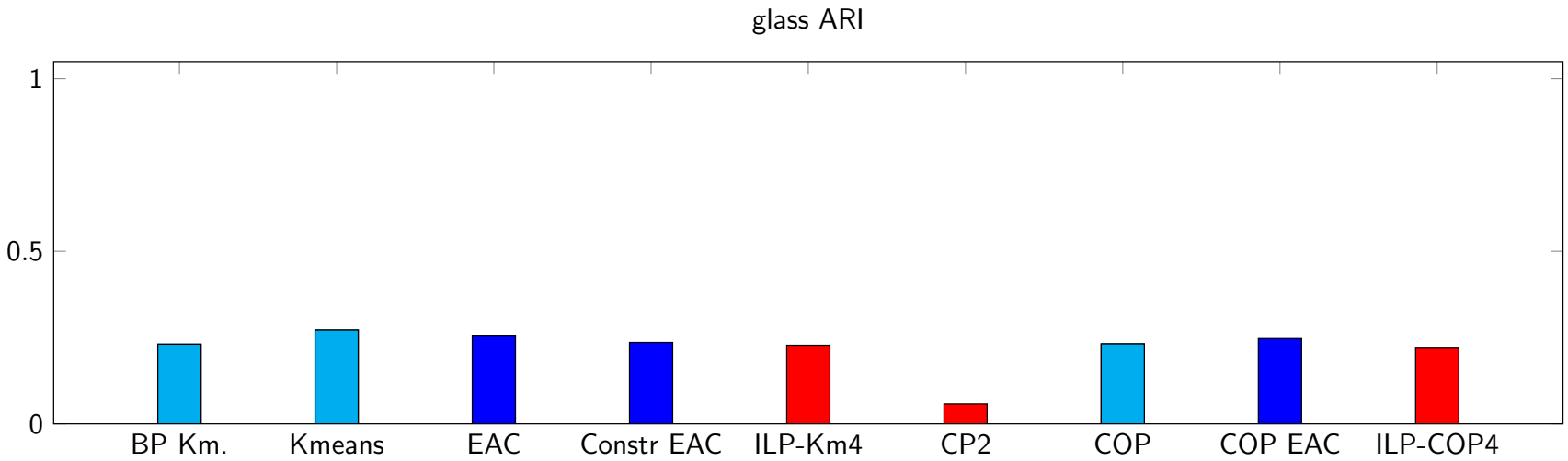


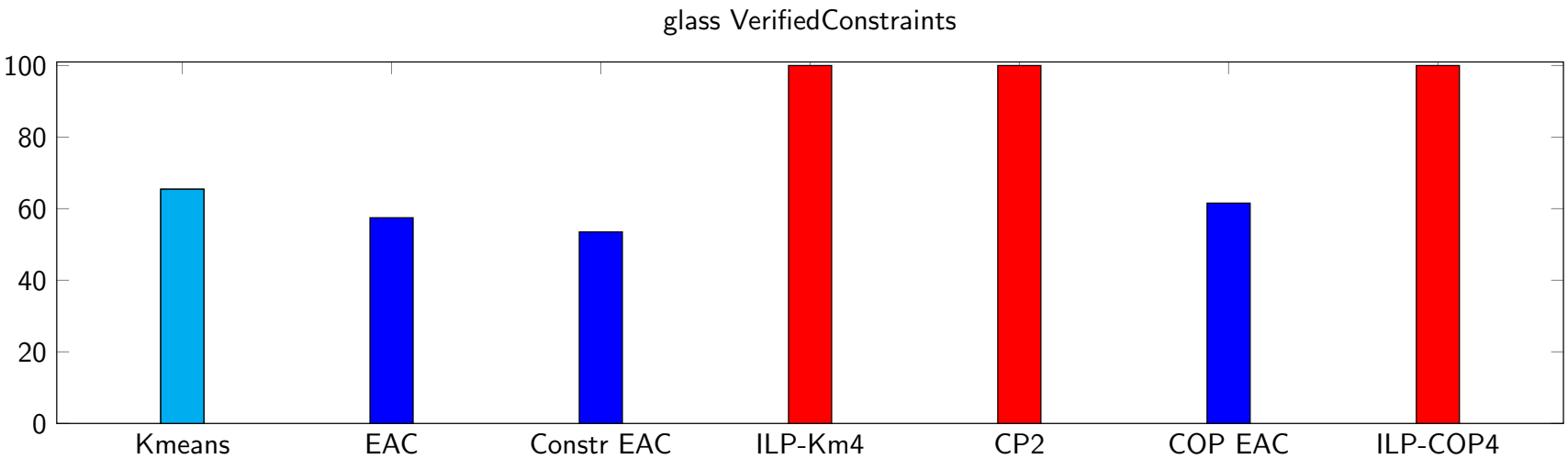


Résultats - Iris



Résultats - Glass





Sommaire

Conclusion

Observation sur les résultats:

1. ARI avec ACCE-ILP est équivalent ou un peu moins bon que baseline.
2. Satisfaction de contraintes avec ACCE-ILP toujours meilleure que baseline.

Travail futur:

1. Étude plus détaillée sur influence du nombre d'ancres par cluster.
2. Réduire temps exécution ACCE-CPP3 (changement d'environnement).
3. Varier le nombre de contraintes.
4. Changer attribution des points n'appartenant pas à des contraintes.
5. Essayer d'autres types de contraintes.
6. Sélection de clusters dans les base-partitions.
7. Travail sur les données Involvd.
8. Explicabilité

Merci de votre attention