CLUSTER CON VINCOLI

Potrebbe essere necessario tracciare dei vincoli per forzare l'attribuzione dei punti a determinati cluster, perché magari ci sono vincoli fisici o vincoli richiesti dall'utente (Una banca che vuole piazzare i suoi ATM in città deve tener conto degli ostacoli come fiumi e colline)

CONSTRAINTS ON INSTANCES

Questa tecnica consiste nello specificare quando una coppia o un set di oggetti dovrebbero essere messi insieme

1) MUST-LINK(x, y)
   x e y devono stare nello stesso cluster.
2) CANNOT-LINK(x, y) se $dist(x, y) > Threshold$

CONSTRAINTS ON CLUSTERS

Specificare dei requisiti per i cluster stessi, come per esempio il numero minimo di oggetti nel cluster, il diametro massimo, la forma, numero di cluster, ...

1) δ-constraints
   Per ogni coppia di cluster C_i e C_j e $\forall i, j$
   Per ogni coppia di istanze $x \in C_i$ e $y \in C_j$ $\forall x, y$

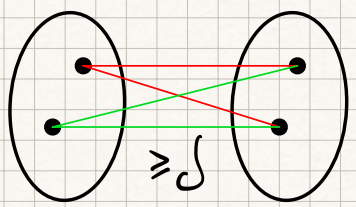
$$D(x, y) \geq \delta$$

2) ε-Constraints

Per ogni cluster C_i : $|C_i| > 1$

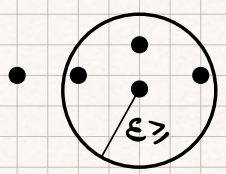
$\forall$ punto $x \in C_i$, $\exists\, y \in C_i$: $\varepsilon \geq D(x,y)$, $x \neq y$.

Il δ-constraint può essere convertito in un vincolo di tipo must-link:



Per ogni punto x applica must-link(x,y) & $dist(x,y) < \delta$.

Mentre l'ε-constraints può essere visto come un must-link del tipo:



Per ogni punto x must-link almeno un punto y : $dist(x,y) \leq \varepsilon$.

CONSTRAINTS ON SIMILARITY MEASURES

Specificare dei requisiti che il calcolo della similarità fra gli oggetti deve rispettare.

HARD vs SOFT

1) Hard Constraints: Un cluster che viola questo vincolo è inaccettabile

2) Soft constraints: Un cluster che viola questo vincolo è accettabile quando non ci sono soluzioni migliori.

Come capire se un vincolo è utile o meno?

1) INFORMATIVENESS

È la quantità di informazione che ha il vincolo e che va oltre il metodo di cluster usato. Si può vedere come, preso un modello di cluster A e un insieme di vincoli C, il numero di vincoli che non sono soddisfatti dal metodo A.

2) COHERENCE OF A SET OF CONSTRAINTS

Quanto i vincoli non si ostacolino fra loro e quindi siano coerenti.

COP K-MEANS : Hard Constraints

Praticamente crea delle SUPER-ISTANZE per i vari must-link:

1) Transitive closure dei must-link, dove se esistono due must-link : must-link(A,B) e must-link(B,C) di conseguenza esiste il must-link(A,C) implicito.

Sostituire gli oggetti della transitive closure con un oggetto dato dalla media degli oggetti nella transitive closure, questa è detta super-istance.
Questa super-istance è pesata col numero di oggetti che rappresenta.

Usare ora un k-means modificato per la gestione dei cannot-link. Quando bisogna assegnare il punto a un centroide bisogna vedere se c'è un cannot-link che lo impedisca.

```
def COPkmeansHC(Su = unlabeled_data, Sl = labeled_data, K = number_of_clusters, q = number_of_constraints):
    ML = []
    CL = []
    # Generazione dei q constraints
    for i in range(0, q):
        # Scegli due punti a caso dal set labeled.
        # Se appartengono alla stessa classe, aggiungi la coppia nei MustLink. Altrimenti, aggiungi la coppia nei CannotLink.
        x = randomly select point from Sl
        y = randomly select point from Sl
        if (label(x) == label(y)):
            ML = ML U (x, y)
        else:
            CL = CL U (x, y)

    # CC sono i punti uniti dalle varie transitive closure
    CC = transitive_closure(ML)

    # clacola il punto medio di ogni CC
    r = CC.size()
    for i in range(1, r):
        mean = calculate_mean(CC[i])

    # genera k centroidi iniziali a caso
    centroids = []
    for i in range(0, K):
        centroids = centroids U randomly select point from Su

    while convergence is not reached:
        # assegna ogni punto al centroide più vicino
        for i in range(0, Su.size()):
            assign_point_to_nearest_centroid(Su[i], centroids)
        # aggiorna i centroidi
        for i in range(0, centroids.size()):
            centroids[i] = calculate_mean(centroids[i])

    return centroids
```

Handling Soft-Constraints

Quando un cluster non rispetta un soft-constraints allora viene penalizzato.
L'obiettivo è massimizzare la qualità dei cluster e minimizzare il numero di vincoli violati.

Objective Function

Somma della distanza usata da K-means per calcolare la media dei punti di un cluster, per poi ricavare i nuovi centroidi:

$$\frac{1}{|C_i|} \sum_{p \in C_i} p + \text{penalità}$$

Così da spostare il centroide.

1) Penalizzare un must-link violato

Viene aggiunta la distanza $dist(C_1, C_2)$ alla funzione obiettivo, ovvero la distanza dei due cluster C_1 e C_2 che hanno oggetti che violano il must-link.

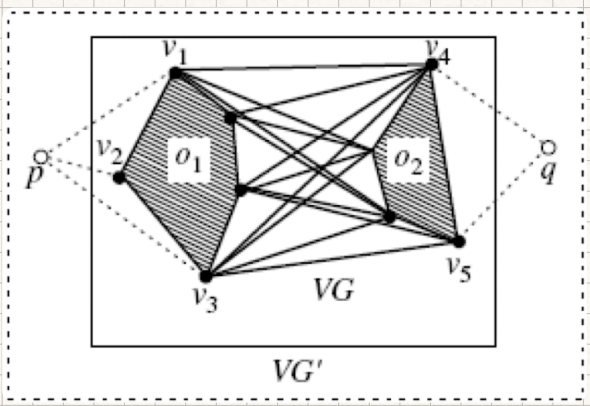
must-link(x, y) ma $x \in C_1$ e $y \in C_2$.

2) Penality of a cannot-link violation

Quando due oggetti x, y con vincolo cannot-link sono associati allo stesso centroide C, allora viene aggiunta alla funzione obiettivo la distanza fra C e il suo più vicino altro centroide C'.

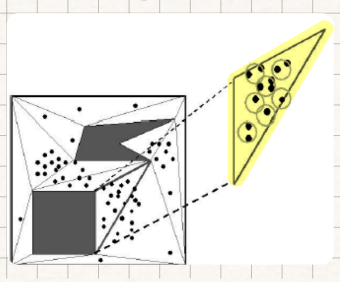
VELOCIZZARE LE PERFORMANCE

Si utilizza un grafo detto visibility graph che rappresenta gli ostacoli. Ogni vertice è un "angolo" di un ostacolo, e i nodi vengono uniti da archi se e solo se si possono "vedere" fra loro. Per esempio se vogliamo rappresentare una casa, allora colleghiamo solo i nodi corrispondenti agli angoli della casa che si vedono:



Creiamo il grafo VG' partendo da VG, aggiungendo i due nodi p e q. Collego i nodi mutuamente visibili fra loro.

Possiamo ora raggruppare i punti del dataset in MICRO-CLUSTER, ovvero gruppi di punti ottenuti dividendo la regione in triangoli fra i vertici del grafo, e poi utilizzare un qualsiasi algoritmo di cluster per cercare i micro-cluster



Usando i micro-cluster la complessità computazionale viene ridotta.