**Chap 7 – Clustering**

**Apprentissage non supervisé** : caractériser distribution des données et relations (distances) entre enregistrements

• Pas de connaissances a priori ni d’ensemble d’entraînement.

**Classification automatique** – clustering : Plus populaire. Étant donné un ensemble d’objets, la classification automatique a pour but de trouver des sous-ensembles (clusters) d’objets homogènes.**Applications** : Trouver communautés graphes, segmentation images, modélisation sur des ensembles résuits (construire un modèle prédictif distinct pour chaque cluster), réduction des données (remplacer/représenter chaque groupe d’éléments par son représentant), détection des valeurs aberrantes (objets éloignés ou dans clusters mini)

**Ingrédients** : • Une collection X = {X1, . . . , Xn} de n objets de dimension d à classifier.

• Une matrice de dissimilarité D = (dij) entre les objets de X est calculée, tel que dij pour i, j = 1, . . . , n satisfait :dij = dji ≥ 0 et dii = 0 (si instances sont les mêmes). Pas besoin de satisfaire inégalités triangulaires (être des distances)

**Mesures de dissimilarité** :

• mesures de similarité naturelles not distances : Coefficient de corrélation (-1 à 1), Produit scalaire

**Distances euclidiennes** : (x1,y1), (x2,y2). Dij =sqrt[ (x2-x1)2+(y2-x1)2**}**

**Distances de Minkowski** : k = 1 : distance de Manhattan (additionner composantes), k = ∞ : composante maximale

Une image contenant texte, antenne

Description générée automatiquementUne image contenant texte, antenne, jauge

Description générée automatiquement**Jaccard**

**Index** :

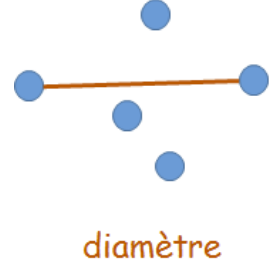
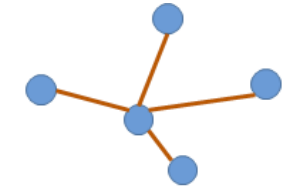
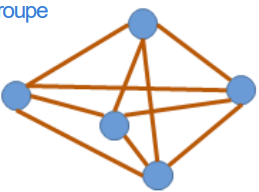
**Jaccard :**

Une image contenant texte, horloge, jauge

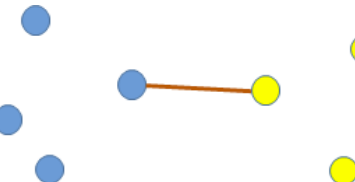
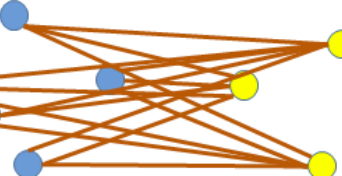
Description générée automatiquement**distance** :

= 1 – (1/3)

**Critères de clustering** :

Homogéinité : Diamètre :Étoile :Clique : 

Minimisation des mesures pour que chaque groupe soit le plus compact, homogènes, proches, similaires

Séparation : Split :  Coupe : 

Maximisation des mesures pour que chaque groupe soit le plus loin possible

Types de clustering :

i) Partition Pk = {C1, C2, . . . , Ck } de X en k classes :

a) Ci ≠ ∅ i = 1, 2, . . . , k ; b) Ci ∩ Cj = ∅ i, j = 1, 2, . . . , k et i ≠ j ; c)

Une image contenant texte, montre

Description générée automatiquementii) Hiérarchie : ensemble imbriqué de partitions de X

Partitionnement : Nb de partitions différentes de n enregistrements en k clusters donné par :

**Algorithmes** :

**Type 1 : Algorithme de k-moyennes.** Regroupement basé sur les centres. Réduit les distances intra-clusters (erreur). Objectif : partitionner data en k clusters.

• Données X = {x1, ...xn}• k est le nombre de clusters • Centres µi, avec i = 1, ..., k

Centre (moyenne de tt les elements ds le cluster. Ci =nb elem cluster somme tt pt ds cluster (somme x) -> new composante en x du centre. (meme chose pr y mais ak somme y)

• Simple, performant, minimise l’erreur au carré (clusters ronds), méthode d’optimisation locale, + populaire

• Sensible à initialisation centres, décider nb clusters pas évident (erreur - quand on ajoute, jusqu’à un certain point)

1)choisir aleatoirement k centres de clusters.

2)calculer dist pr tt les points ak les k centre. Distance : manhatan ou eucledienne etc.

3)placer chaque pt ds le cluster k ak la + petite distance [entre point et centre k]

4) recalculer les centres des k clusters (classes) 5) refaire 2 et 3 jusqua convergence ak new centres.

**Type 2 : Algo/Regroupement hiérarchique**. Construit clusters imbriqués en les fusionnant/divisant successivement

Approche de division (moins utilisée) : 1. Commence par un cluster initial avec toutes les données

2. Effectue des bipartitions successives d’un cluster à la fois 3. Répète jusqu’à l’obtention de singleton clusters.

Approche d’agglomération : 1) Commence avec une partition initiale avec n singletons 2)calculer la mat de distance 3) Regrouper 2 clusters ak distance la + petite ds la matrice en un nouveau cluster. If ++ clusters ak meme dist et c plus small, prendre une pair au hasard. 4) update la mat de distance. Et calculer new dist.ex : on join cluster x1 et cluster {x2,x5} 🡪 on aura d(x4,x1) et d(x4,{x2,x5}). 🡪 faut choisir un des 2 pr update mat apres le join. Pr single linkage -> dist la plus petite des 2.

5)Répète jusqu’à ce que toutes les données appartiennent au même cluster

Single linkage Algorithm : Fustionne 2 clusters Ci et Cj si **min** D(Ci, Cj) est la + petite parmi toutes les paires de clusters (2 points les plus proches d’un cluster à l’autre). Peut regrouper des objets très différents

Complete linkage Algorithm : Fusionne 2 clusters Ci et Cj si **max** D(Ci, Cj) est la + petite valeur parmi paires de clusters (2 points les plus loin d’un cluster à l’autre). Peut séparer des objets très similaires

**Type 3 : Regroupement basé sur la densité**. DBSCAN. Robuste au bruit, bon pour identifier groupes formes arbitraires

Motivation : 1. Chaque cluster est région dense de points. 2. Régions peu denses séparent les clusters. 3. Bon pour jeux de données avec structures non régulières, présence d’outliers ou de bruits

2 paramètres : • : si deux points Xi et Xj sont à une distance dij ≤ 🡪 voisins

• m : nombre minimal requis de points pour former un cluster

Calculer matrice de distance.

1) choisir Point au hasard Xi. 2) Regroupe Xi avec ses voisins (voisin si dist entre pt et autre pt < ) 3)Si un point est trouvé comme part d’un cluster, son -voisinnage est lui aussi part du même cluster (si pt est voisin a un pt déjà ds un cluster. Add ce pt la ds le cluster du voisin. X1 voisin x2. X2 ds cluster 1. Mettre x1 ds cluster x2.) 4) Si ce processus abouti à un cluster d’au moins m points, on le garde. Sinon,bruit. 5) On retourne à 1. jusqu’à ce que tous les points soient considérés.

+ Robuste aux bruits et outliers. Pas besoin de connaître le nb de clusters à priori

- Qualité clustering dépend bcp params. Mauvais pour jeux de données avec grandes différences densité

• En utilisant les conditions d’optimalité de premier ordre, on arrive aux expressions suivantes :

si on suppose que les γ(zij) sont fixes, alors on est capables de calculer les params de distribution. Par contre, changer µj, πj et Σj change aussi γ(zij)

**Évaluation de clusters**

• Mesures internes : Liées au critère de clustering choisi, à maximiser pour les critères de séparation / minimiser pour

les critères d’homogénéité

• Mesures externes : Utilisent une partition ground-truth pour calculer des mesures

tel quelles le F-score, recall, etc (attention à la symétrie !) Basée sur une connaissance a priori

**Chap 8 – Détection d’outliers**

• Un outlier (ou donnée aberrante) est un enregistrement très différent de la majorité des données : déviations d’intérêts (phénomène normal inconnu ou rare), bruit, anomalies (erreurs, artefacts, attaques, etc.)

• Leur identification est un problème complémentaire au clustering : (appartiennent à des clusters petits ou creux)

• Application : Nettoyage des données, Détection de fraudes (bancaires, assurances, etc.), Détection d’intrusions, Diagnostic médical, Détection de fautes dans les systèmes informatiques

**Modèles par clustering** : • Clustering et détections d’outliers partagent une relation complémentaire bien connue. Outlier score = distance entre pt et le centre du cluster le + proche.

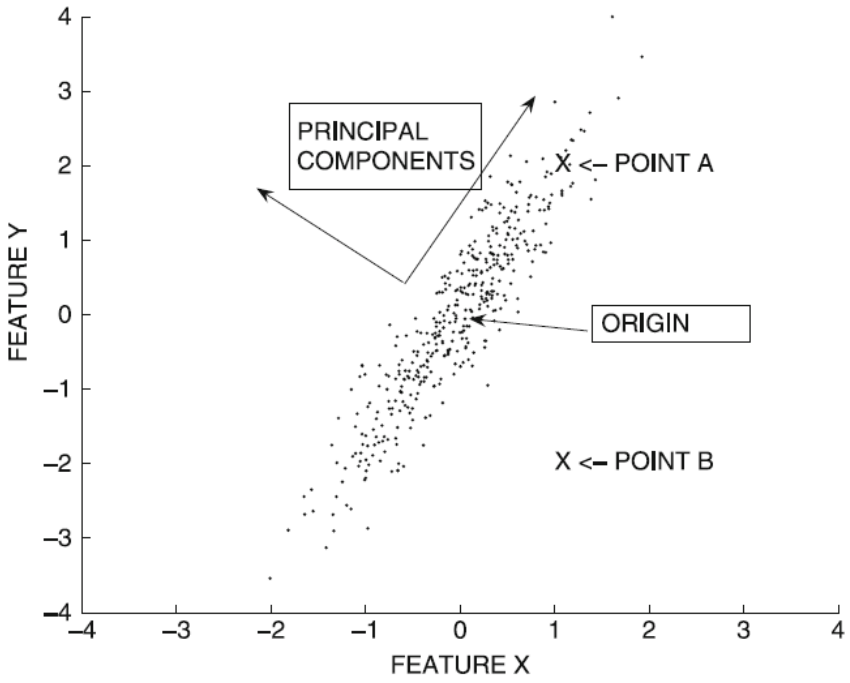
• Détection d’outliers en tant que produit secondaire des méthodes de clustering pas best way.

• Les algorithmes de clustering ne sont pas optimisés pour la détection d’outliers

• La manière simple de définir l’outlier score d’un enregistrement consiste à grouper d’abord l’ensemble de données, puis à utiliser la distance de chaque enregistrement vers son centre le plus proche. Mais on peut faire mieux.

• Ces distances doivent dépendre de la répartition des autres points dans l’espace

• La droite de O à A est alignée avec une direction de variance élevée, et statistiquement, il est plus probable que les points soient plus éloignés dans cette direction.

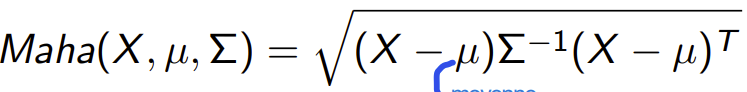
• D’autre part, le segment de O à B est faiblement peuplé.

• Statistiquement, il est beaucoup moins probable que B soit aussi loin de O dans cette direction

• Donc la distance de O à A devrait être inférieure à celle de O à B .Distance de Mahalanobis :

• Soit Σ la matrice de covariance d × d de X.

• L’entrée Σ[i][j] est égale à la covariance entre les dimensions i et j.



• Normalise données en se basant sur covariances entre dimensions

**Modèles par clustering** : Considèrent k clusters. • µr : centroïde du cluster r • Σr : matrice de covariance des données groupées dans le cluster r. L’outlier score d’un point Xi est calculé par rapport à son cluster r distance de Mahalanobis

• Basés sur une analyse globale : masse critique nécessaire pour avoir un cluster (plus d’un certain nombre des enregistrements - hyperparamètre)

• Ne distinguent pas très bien les données générées par du bruit de celles qui sont vraiment des anomalies.

• La distance d’un point à son centre le plus proche n’est pas très informative dans certains cas.

• On a besoin d’une analyse locale.

**Modèles basés sur des distances** : Basé sur les k-plus proches voisins. Outlier score d’un enregistrement donné par sa distance à son keme -plus proche voisin (sensible au choix de la distance). k est un hyperparamètre

• Normalement les données avec beaucoup de bruit n’ont pas de grands outliers scores selon ce modèle, mais les vrais outliers oui. Cette distinction est perdue dans les méthodes de clustering où la distance par rapport au centre le plus proche ne reflète pas précisément l’isolation d’un certain enregistrement.

• Get distance enregistrement à son keme plus proche voisin nécessite temps O(n), O(n2) pour tous les enregistrements

• On ne calcule pas les outliers scores de tous les enregistrements, seulement les plus aberrants (top-r outliers) aka sampling methode

Sampling : • Choisissez un échantillon S d’un ensemble de données D de taille s << n.

1) Calculer toutes les distances par paires entre les points dans S (ds echantillon choisi) et les points dans D (autres pts) : O(sn).(mat dist). Tt les dist pr les elem du sample.

• Les k-plus proches voisins sont connus pour les points dans S.

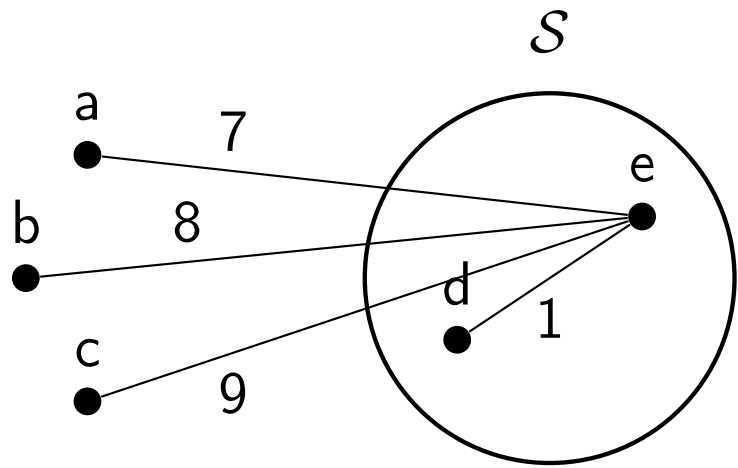
2) calculer lower bound. [on prend les elements ds l’echantillon et on choisi le ke plus proche voisin]

Ke dist la + petite. K= 1 (top-r c la dist la plus petite). Cest le vrai score.

• Il est un lower bound (L) pour le reme outlier score dans l’ensemble total D. si top r >1 🡪 on choisi les 2 plus petit. Top 1 => plus grande valeur parmi les top-r.

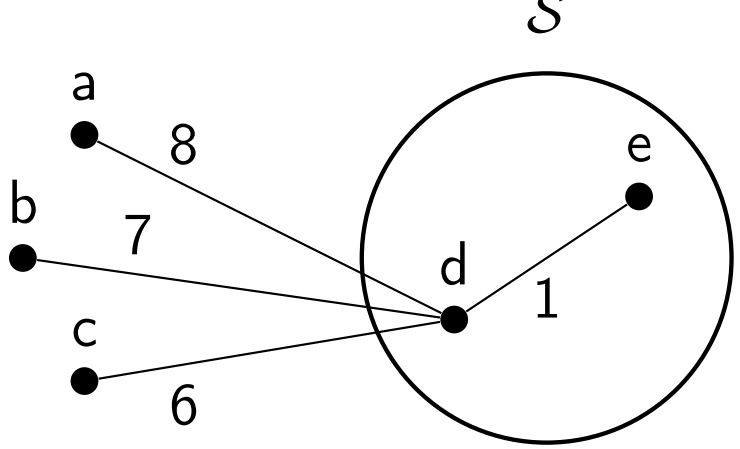
3) find upper bound. Prendre tt les elem hors de lechantillon. Score estime ici car on a pas tt les distances juste elle ak les pt choisi ds echantillon. If score estime elem ds ensemble <= lowest top-r. vag sur ce pt et choisir le next one ds lensemble totale(aka hors du sample). Si score estime >= lowest top-r : keep et calculer tt les distance pr ce point.

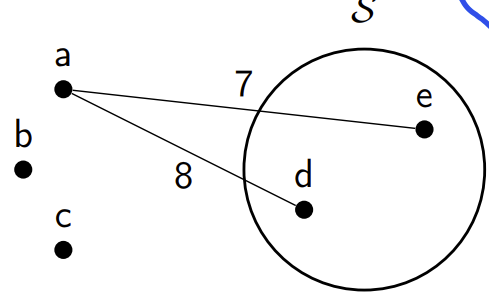
• Si ce upper bound est plus petit que L, alors le point de D − S peut-être exclu comme possible top-r outliers.

Une image contenant table

Description générée automatiquement4)L’algorithme reprend son exécution avec les points restants.Exemple : Ensemble de points D = {a, b, c, d, e} et S = {d, e} un échantillon. k=2 (2e plus proche voisin) et r=2 (top-2)

Une image contenant table

Description générée automatiquementOutlier score d s(d)=6

Les outlier scores des top-r sont des lower bound (on peut trouver une valeur plus grande, mais pas plus petite). Le score du top-1 outlier (point + isolé) vaut au moins 7. Il est possible de trouver un point avec un score plus élevé, mais un score plus petit ne peut pas remplacer celui déjà connu.Considérons le point a. Seules les distances d(a, e) et d(a, d) sont connues. Le score estimé de a est un upper bound. Trouver un score plus élevé signifie trouver un plus proche voisin plus éloigné que ceux déjà connus. L’outlier score estimé du point a est sˆ(a) = 7.

• Si le score estimé (upper bound) est inférieur ou égal au plus petit score des r-outliers (le plus petit des lower bound) alors le point n’est pas un candidat et peu être élagué. On continue en ordre alphabétique à vérifier scores estimés

• Si le score estimé est plus grand, alors il faut calculer son score exact. Cela met aussi à jour le score des autres points (pas encore considéré).

sˆ(a) = 7 > s(d) = 6 : le score estimé de a est supérieure au plus petit lower bound, il faut calculer la distance entre a et tous les autres points.**Chap 9 – Analyse de réseaux sociaux**

• Réseau social peut être structuré comme un graphe G = (N, A) (N : tt sommets et A : tt arêtes)

• Facebook : relation d’amitié (symétrique) • Twitter : relation de follower (asymétrique)Homophilie les sommets connectés les uns aux autres sont plus susceptibles d’avoir des propriétés similaires.

Fermeture triadique si 2 entités d’un réseau social ont 1 ami en commun, alors il est plus probable qu’elles soient connectées ou qu’elles vont éventuellement se connecter dans l’avenir (concept lié au coefficient de regroupement)

• **Coefficient de regroupement** : mesure de la tendance inhérente d’un réseau à se regrouper. • Soit Si ⊆ N l’ensemble de sommets connectés au sommet i ∈ N (voisins de i);

ni = |Si| (nb d arc connecter au sommet i) (voisin de i) (degre)

• Arêtes possibles entre sommets de Si :

A picture containing sketch, diagram

Description automatically generatedCombinaison  : C(n, k) = n! / (k! \* (n - k)!)

1) calculer ηi pour tt les sommet. ηi =nb liens entre les voisins de i / nb chemins possible entre les voisins ( ).

2)Coeff pr graph : moyenne : Somme ηi  pr tt sommet/ nb sommet total.

Nb lien entre voisin de i : ex : nœud 9 a 3 voisin (6,10,11). ya un arc en 6-10? 10-11? 6-11 ?

**Attachement préférentiel** Dans un réseau en croissance, la probabilité qu’un sommet reçoive des nouveaux liens augmente avec son dégrée. Composant connecté géant les nouveaux liens sont plus susceptibles de se joindre aux sommets densément connectés et de haut dégrée dans le réseau.**Mesures de centralité et de prestige** : « Les indicateurs de centralité sont des mesures censées capturer la notion d’importance dans un graphe, en identifiant les sommets les plus significatifs. Les applications de ces indicateurs incluent l’identification de la ou des personnes les plus influentes dans un réseau social » Les sommets centraux du réseau ont un impact significatif sur ses propriétés. Ces sommets sont souvent plus importants parce qu’ils ont des liens avec de nombreux sommets et sont dans une position de plus grande influence. Dans le cas des graphes dirigés, nous parlons du prestige d’un sommet (ex. Twitter)

**Centralité de degré** d’un sommet i : . Les sommets avec un dégré plus élevé sont souvent des sommets centraux qui ont tendance à rapprocher les parties éloignées du réseau. Le problème majeur avec la centralité de degré est qu’elle est plutôt myope.(n = nb sommet ds graph)

**Prestige de degré** (lorsque graphe orienté) : in\_degree nb d’arcs entrants du sommet i

• Plus court chemin moyen d’un sommet i mesuré sur des graphes connectés : où Dist(i, j) est la longueur du plus court chemin entre les sommets i et j. (cmb de nœud je dois parcourir pr aller i a j (j E tt les nœud ds le graph). I c le nœud quon veut calculer le degre de.

**Centralité de proximité**:.

Influence(i) composé par les sommets qui peuvent atteindre le sommet i. (pr graph diriger)

• Dist(j, i) est calculée du sommet j au sommet i (Dist(i,j) = nb arret je dois parcourir pr aller de i a j. pr tt les nœud j ds influence(i). si 5 (j change) nœud ds influence, alors somme de 5 dist(j,i)). Utiliser l’inverse de la distance moyenne ne serait plus juste, les sommets avec moins d’influence doivent être pénalisés.

Influence(i) = nb sommet qui peuvent atteindre (directement ou indirectement) le noeud i. Ca peut passer par un autre nœud. A🡪 B 🡪 g : infue(g) = 2. | A 🡪 B 🡨 G : influ(G) =0

• Facteur de pénalité (multiplicatif) inclus dans mesure prestige de proximité.

• Correspond à la taille relative de l’ensemble d’influence du sommet i :

**Prestige de proximité**: (find nœud + interessant)

A picture containing line, diagram, font, sketch

Description automatically generated**Centralité d’intermédiarité** CI(i): Considère la criticité d’un sommet en termes du nombre de chemins les plus courts qui le traversent. Crucial pour déterminer les sommets qui contrôlent le plus le flot d’informations entre les autres sommets d’un réseau social. 1) Faire une matrice F avec diagonale et remplir le haut seulement (j < k) pr tt les noeud. Si ya pas de lien entre j ->k on mettrera vide a la position fij ds la matrice. Pr nœud qon veut calculer la centralite on met rien a sa position ds la matrice. Si on a k =i > j 🡪 laisse blank ex : si ds img, on veut calculer la centralite pr nœud 3. Tt colonne 3 sera blank (check vert ds img)



• Soit qjk le nb de chemin (plus courts chemins) posible entre les sommets j et k. (sil ya 1 facon daller de k -> j = 1. Si on peut aller de 2 facon k->j = 2)

•qjk(i) le nombre de ces chemins possible qui passent par le sommet i (lui on veut calcul CI).

• fjk(i) = qjk(i)/qjk  : ce quil ya ds la matrice F pr tt les pos k > j

• fjk(i) : niveau de contrôle que le sommet i a sur les sommets j et k en termes de régulation du flot d’informations (On a une matrice pr chaque sommet i).|pr centralite du sommet i : 1 matrice et faire somme de tt elem en haut diag.

Une image contenant texte

Description générée automatiquement• 2) CI(i) est la valeur moyenne de cette fraction pour toutes les paires de sommets (en excluant le sommet i) :faire la somme de tt les elements dans la matrice(pr k>j ,donc les elem en haut diag).

• Contrairement à la centralité de proximité, la centralité d’intermédiarité peut également être définie pour les réseaux déconnectés. Peut être généralisé pour les arêtes aussi. Permet de co plusieurs clusters differents.

**Détection de communautés** : • Clustering pour les réseaux sociaux. Pour chaque cluster, ses éléments ont plusieurs liens entre eux, et très peu de liens pour le reste des éléments du réseau social.

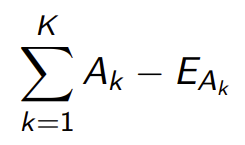
**Algorithme de Girvan-Newman** : Basé sur l’intuition que les arêtes ayant une grande centralité d’intermédiarité CI ont tendance à connecter les différents clusters. A : ensemble des arrets. Permet de split en k clusters

Tant qu’il reste des arrêtes, **1**) Calculez CI(a) pour tout a A **2**) Enlevez les arrêtes avec les plus grandes valeurs de CI.

• Algorithme de regroupement hiérarchique descendant (division).• Composants connectés sont les communautés.

• 2 défis : - le calcul de la centralité d’intermediarité pour chaque itération de l’algorithme de Girvan-Newman

- la définition du bon nombre de clusters.

**Modularité** : Critère de clustering pour les réseaux. Étant donné une partition du réseau en K clusters, la modularité Q est directement proportionnelle à :

où : • Ak est le nombre d’arêtes présentes dans le cluster k • EAk la valeur attendue de Ak dans un graphe aléatoire. • EAk demande la construction d’un modèle nul.

Une image contenant texte

Description générée automatiquement**Modèle nul** : • Étant G = (N, A), on construit un nouveau réseau aléatoire G’ préservant la distribution des degrés des sommets de G (conserve les meme sommets et nene distrib des degres que G). • Nb attendu d’arêtes entre les sommets i et j est égal à :



• Le nombre attendu d’arêtes attendues pour tout le graphe G’ :

Montrer Nb attendu aretes pr multigraph G’ =|A| :



A picture containing text, font, handwriting, screenshot

Description automatically generated

• Modularité Q(G,P) d’une partition P = {C1, . . . , CK } des sommets d’un graphe G (voir formule en dessous)

où δij = 1 si les sommets i et j sont liés, 0 sinon.• Q ∈ [−1, 1]

• Q > 0 si le nb d’A dans les clusters dépasse le nb attendu

• Q > 0.7 indique une structure communautaire

significative.• On peut utiliser la modularité pour sélectionner le nombre de clusters dans l’algorithme de Girvan-Newman.

• On calcule la modularité pour chaque partition de la hiérarchie. On garde celle avec la plus grande valeur de Q. |A| = nb arret ds tt graph

Une image contenant texte, horloge, montre

Description générée automatiquementSomme tt les nœuds ds le cluster k. i et j pcq on veut pas le meme on veut decaler. clusters.

**Classification collective** : Dans plusieurs apps, étiquettes peuvent être associées à des sommets d’un réseau social.

Étiquettes disponibles ⇒ classer les étiquettes inconnues. Les sommets ayant des propriétés similaires sont généralement connectés. Il est raisonnable de supposer que ceci est également vrai pour les étiquettes.

• Solution simple : examiner l’étiquette majoritaire trouvée parmi les voisins d’un sommet (similaire à k-NN).

• Parfois impossible pour la classification collective en raison de la rareté des étiquettes.

Une image contenant texte

Description générée automatiquement**Algorithme de classification itérative** :

• Le nombre total de itérations dépend du nombre d’étiquettes fixées par itération.

• Les attributs de contenu sont ceux relatifs aux caractéristiques du sommet, ex. Xi = {43ans, 74kg, 80000$}.

• Les attributs de liens sont structurels et mis à jour à chaque itération : distribution de classes dans le voisinage d’un sommet, différentes mesures de centralité, etc.

• L’algorithme A doit fournir la proba qu’un sommet appartienne à une classe. • Algo de classification semi-supervisé.

**Marches aléatoires** : Pour classifier un sommet non étiqueté i, des marches aléatoires peuvent être exécutées à partir du sommet i. • Une marche se termine lorsqu’on trouve le premier sommet étiqueté. • La classe dont la probabilité de terminaison de la marche aléatoire est la plus élevée est indiquée comme classe prédite du sommet i. • L’idée est que la marche a plus de chance d’être terminée dans un des sommets étiquetés aux alentours du sommet i.Hypothèse Graphe est label connected : il existe un chemin entre tous sommets non étiquetés et un sommet étiqueté.

Condition Une marche aléatoire se termine toujours aux premiers sommets étiquetés atteints .**1)** : Graphe non-dirigé -> graphe dirigé (i 🡪 j et i 🡨 j)**.**

**2)** : Remplacer arcs sortants des *sommets étiquetés* par self-loops (sommet etiquete on enleve arcs sortant et mettre self-loop)

**3) : soit** M la matrice (n × n) de transition du graphe après l’étape 2.

• Rappel : Mij est la probabilité de que le prochain sommet parcouru après le sommet j soit le sommet i.

• Mij = 1/out\_degree(j), s’il existe l’arc j → i; else Mij = 0. • Pour sommet j avec un self-loop (j 🡪 j), Mjj = 1.• Si le point du départ est fixé, il n’y a qu’une distribution de probas π qui découle des itérations successives de :

4) πk = M · πk−1 processus markovien



à partir de π0 avec (π0)i = 1 dans le cas où le sommet de départ est i, else 0. Choir le nœud quon veut start ak,on met sa val = 1 ds vect et les autres = 0.

• La classe du sommet i (proba + élevée) est alors donnée par (image) :

πk = [x,y,z,…]T: vecteur colonne de dimension n (nb sommet) itera k. x : proba finir au sommet 1. Y : proba finir sommet 2… Pr get les proba de finir a quel classe on prend les valeurs ds le vect colonne qui represente les sommets qui ont des classes. 🡪 take + grande proba on classe le nœud ds celle la.

**Chap 10 – Fouille de graphes** Nombreux petits graphes

**Taxonomie des graphes** : Non-dirigé vs dirigé (flèches sur arêtes), Simple vs non-simple (loop, doubles liens), Embedded vs topological (arêtes suivent un tracé précis), non-pondéré vs pondéré (chiffre sur les arêtes), sparse vs dense (plus de liens parmi ceux possibles), non-étiquetté vs étiquetté (nom sur les sommets)

**Isomorphisme de graphes** : • Le problème de savoir si deux graphes sont isomorphes est NP-complet. Le problème devient encore plus difficile lorsque les étiquettes de sommets se répètent.

• Le problème de savoir si un graphe est sous-graphe isomorphe à un autre est aussi NP-complet.

Isomorphe : 2 graph ak meme structure. Facon dont nœud connecter est identique. Meme nb nœud. Meme nb arret.

**Maximum commun sous-graphe (MCS)**paire graphes G1 = (N1, E1) et G2 = (N2, E2) en un graphe G0 = (N0, E0) qui est sous-graphe isomorphe à G1 et G2 pour lequel la taille de l’ensemble de noeuds N0 est le + grand possible (garder + nœud et darcs). On veut keep la meme structure [nœud 1 conecter a 3 nœud.] le MCS keep ces lien. Pysiquement pas besoin detre pareil. Mais structure is the same (nœud relier a 3 nœud 🡪 conservred meme si c pas le same geospacialement aka a-b-c = a-b ).



d(G1, G2) = |G1| + |G2| - 2 |MCS(G1, G2)|



Mesures de similarité : ; -> NP-difficile

|G1| : nb sommet graph 1. ; |MCS(g1,g2)| : nb de nœud en commun ds le sous graph isomorphe a G1 et G2. (des 2 graph init on veut find un sous graph qui a le max nb de nœud et les meme arcs en les nœuds)

**Distance d’édition** Edit(G1, G2) est égal au coût minimum des opérations de modif à appliquer au graphe G1 pour le transformer en G2 (node/edge deletion, label substitution, edge insertion) -> NP-difficile. Pas besoin de move le nœud pr que se soit pareil. Move c ps une edition(modif) qui compte ds edit. meme connect entre sommet et meme nb de sommet pas pas necessairement meme structure.

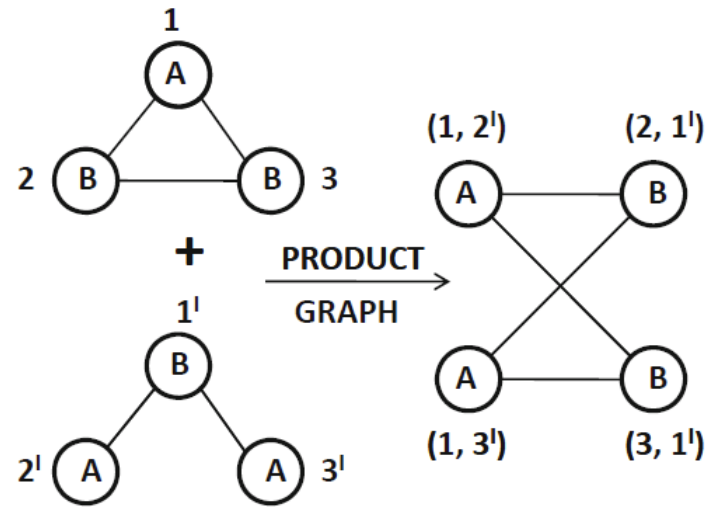
So far, ces mesure peuvent etre used pr petit graph car MCS et Edit -> NP - difficile

**Transformations basées sur un noyau :** Peuvent être utiliséespour un calcul de similarité plus rapide, directement avec les SVMs.• La similarité de noyau K(Gi, Gj) entre une paire de graphes Gi et Gj est le produit scalaire des deux graphes après leurs transformations hypothétiques dans un nouvel espace défini par la fonction φ(·) :

K(Gi, Gj) = φ(Gi) · φ(Gj)

• La fonction φ(·) n’est pas définie directement. Plusieurs façons de définir une similarité de noyau pour les graphes

**Marches aléatoires** : Compter marches communes dans G1 et G2. Marches : séquences de sommets avec répétition

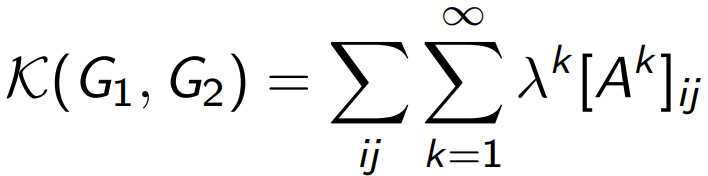
• Calcul : Construction du graphe produit de G1 et G2

• Le nombre de marches de longueur k peut être calculé en



regardant la k-ième puissance de la matrice d’adjacence A du graphe produit.





où λ ∈ (0, 1) choisi pour garantir la convergence de la série.



Graphe produit : • Les sommets du graphe produit de G1 et G2 correspondent aux paires de sommets de G1 et de G2 qui ont la même étiquette (toutes les combinaisons) (faire la marche aleatorie): VX = {(v1, v2) : v1 ∈ G1 ∧ v2 ∈ G2 ∧ label(v1) = label(v2)}

V1 (sommet du 1er graph ds img): label(v1) = A; v1= sommet 1 ; v2: (sommet du 2e graph en bas ds img) label(v2) = A, v2=2’. 🡺 A (1.2’) .

1)Pour chaque etiquette(label) pareil dans les 2 graph on créer un sommet dans le graph produit avec le label commun et comme indice de nœud (v1,v2). les labels pareil. (ds img : on met A de G1 ak les A de G2.)

V1 = le sommet i ds le graph 1. Soit 1 ak label =A, sommet 2 : label = B et sommet 3 : label = B.

• Les arêtes du graphe produit correspondent aux arêtes communes à G1 et à G2 : EX = {(u1, u2),(v1, v2) : (u1, v1) ∈ G1 ∧ (u2, v2) ∈ G2}

2)Ak les sommets créer ak 1 label et 2 sommets (1 sommet de chaque graph) on va tester pr voir si on doit mettre un arc ds le product graph. ex : sommet graph produit : (1,2’) et (2,1’) -> arc entre (1,2) et (2’,1’). Faut que arc existe ds les 2 graph pr quil soit ds le graph produit. Si existe juste ds 1 🡪 pas darc ds graph prod.

**Chemins plus courts** : Les marches aléatoires permettent de répéter les sommets des séquences. Une marche peut visiter le même cycle de sommets plusieurs fois. Le noyau basé sur les marches aléatoires mesure la similarité en termes de marches communes. Par conséquent, une petite similarité structurelle peut provoquer une énorme valeur de noyau. Solution : noyau basé sur les plus courts chemins.

• La fonction k(i1, j1, i2, j2) est définie par paires de sommets avec i1, j1 ∈ G1 et i2, j2 ∈ G2.

Une image contenant texte, montre

Description générée automatiquement• k(i1, j1, i2, j2) = 1 si le plus court chemin entre i1 et j1 dans G1 est de la même taille que le plus court chemin entre i2 et j2 dans G2. Ainsi, la fonction noyau est définie comme :

**Clustering de graphes** : • Partitionne la base de données de n graphes G1, . . . , Gn en k clusters.

• out-of-the-box : méthodes basées sur des dissimilarités.

• Une deuxième méthodologie utilisée est celle des méthodes spectrales.

• Les graphes de données G1, . . . , Gn sont utilisés pour construire un seul graphe global G.

• Chaque graphe Gi correspond à un sommet dans G. (chaque graph espace original -> 1 sommet ds graph G)

• Chaque sommet de G est lié à ces plus proches voisins selon les distances calculées.

• Donc, le problème de regrouper G1, . . . , Gn devient le problème de regrouper les sommets d’un seul graphe G.

• Possible algorithme : transformation spectrale sur G + k-means.**Classification de graphes** : • On suppose qu’un ensemble de n graphes G1, . . . , Gn est disponible, mais seul un sous-ensemble de ces graphes est étiqueté (avec des étiquettes 1, . . . , k). • out-of-the-box : KNN, chaque nouveau graphe non-étiqueté prend l’étiquette de la classe majoritaire parmi ses k-plus proches voisins (regarder graph + proche check class majoritaire. C new class du graph).

**Descripteurs topologiques** : convertissent les graphes en données multidimensionnelles où chaque attribut mesure une caractéristique structurelle importante. Une fois la conversion effectuée, des algorithmes d’exploration de données multidimensionnels peuvent être utilisés sur la représentation transformée. L’inconvénient de cette approche est qu’elle implique une perte information. Ex :

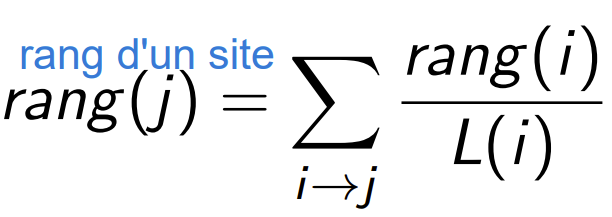
• Morgan Index égal à un vecteur de taille |G| (nb noeud) où chaque composante i est égale au nombre de sommets accessibles depuis le sommet i à une distance d’au plus t. faire vect de taille nb sommet et dedant on met le nb de sommet quon peut atteindre a partir du sommet dindice du vect.• Wiener Index égal à la somme des distances les plus courtes entre toutes les paires de sommets du graphe.

• Hosoya index égal au nombre de matchings valides dans le graphe.

**Chap 11 – Page Rank** Un seul grand graphe

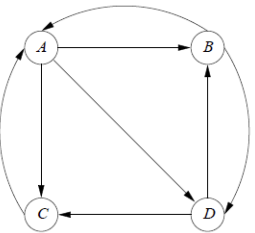
Une manière intuitive de savoir si une page est populaire consiste à compter le nombre d’hyperliens entrants (c.-à-d. qui pointent vers cette page). Sauce secrète originale derrière Google (ignore le contenu des pages Web, se concentre uniquement sur la structure des hyperliens entre les pages.

• Supposons 2 pages A et B, chacune avec trois hyperliens entrants. A est pointé par les sites de Pierre, Paul, et Jack, B est pointé par LeMonde, LaPresse, et LeDevoir. Doit-on considérer que A est aussi “populaire” que B ?

Première amélioration : Si une page Y liée à la page X a des millions de liens sortants, cette connexion est moins importante que celle venant d’une page W liée à X avec seulement quelques liens sortants. L(i) est le nb d’arcs sortants de i

Une image contenant texte, horloge

Description générée automatiquement• Soit M la matrice (n × n) de transition. n = nb sommets

• Mij est la probabilité de que la prochaine page visitée après la page j soit la page i.

• Alors Mij = 1/L(j) s’il existe un arc j → i, Mij = 0 sinon. L(j) : nb d’arc sortant de j.

• Le PageRank initiale de toutes les pages (PR0) vaut 1/n (peut commencer à partir de n’importe quelle page du web) vect colone dim= nb de noeud

• L’algorithme (Markovian process) converge alors à condition que le graphe soit fortement connexe (il existe un chemin entre chaque sommet) et M soit stochastique : (pas de cul de sac)

**Structure du Web** :

• Sites fortement connexes (SFC) : pages + intéressantes

• Composantes entrantes (CE) : pages qui peuvent atteindre les SFC.

• Composantes sortantes (CS) : pages qui peuvent être atteints par SFC mais ne peuvent les atteindre.

• Attaches sortantes : pages atteints des CE qui ne peuvent pas atteindre les SFC.

• Attaches entrantes : pages qui atteignent les CS mais ne peuvent pas atteindre les SFC.

• Tubes : pages liant les CE et les CS qui skip les SFC.

• Composantes isolées : ne peuvent pas atteindre ni être atteintes du reste du réseau

• Plusieurs de ces structures violent hypothèses nécessaires pour que l’itération de PageRank converge vers limite : • Usagers peuvent pas sortir des CS (dead-end). • Les usagers vont certainement finir dans les CS ou les attaches sortantes.

• Conséquence : les PageRanks des SFC ou des CE seront nuls à la fin.**Dead-end** : Proba internaute finisse de surfer à une page quelconque passe à 0 au fil du temps. M pas stochastique

**Spider Traps** : La probabilité qu’un internaute finisse de surfer à la page C (self-loop) est de 100% au fil du temps ! M est stocastique par contre

**Solution : Téléportation**. à chaque pas de temps, l’internaute a deux options : avec prob. β, il suit un lien au hasard.

avec prob. 1 – β (teleportation), il saute à une page quelconque du web.

**Nouveau PageRank** : PRk = βM · PRk−1 + (1 − β)e/n, où e est le vecteur unitaire (e/n -> vect 1/nb de sommet. Dimention : nb de sommet). L(i) est le nombre d’arcs sortants de i. e/n et pr0 : vecteur de n colonne [1/n 1/n ….]T

**A picture containing circle, diagram, line, sketch

Description automatically generated**Spider trap peut aussi etre entre plusieurs noeud. Pr fix. On diminue beta. Ce qui augmente 1- beta (aka teleportation) ce qui permet internaute de escape le trap.

**Chap 12 – Fouille de flots de données**

Les données ne sont pas nécessairement stockées pour toujours... ou même du tout ! Dans certaines applications, il peut être avantageux de calculer des statistiques on the fly, au fur et à mesure que les données arrivent pour que nous puissions les jeter en suite. Dans un algorithme de fouille d’un flot de données, nous n’avons qu’une seule chance de voir chaque donnée. Alors, nous devons décider quoi faire avec chaque donnée à ce moment-là.

**Caractéristiques des flots de données** : Entrée continue et rapide des données (Big Data : dimension vélocité). Mémoire limitée (moins que linéaire dans la taille d’entrée). Temps limité pour traiter chaque donnée. Accès séquentiel (pas d’accès aléatoire). 1 seule chance (ou pt très peu de chances) de voir chaque donnée du flot.

**Exemple** : Calcul de la **moyenne** d’un flot de nombres : Deux variables : sum qui accumule la somme de nombres à date, et n la quantité de nombres qu’on a vu jusqu’à présent. Pour chaque nouveau nombre Xi, on ajoute cela à sum et on incrémente n. On renvoie µ = sum/n à chaque fois que la moyenne est demandée.

**Exemple** : Calcul de la **variance** d’un flot de nombres :

• De nombreuses quantités ne peuvent pas être calculées exactement selon le modèle de flots (ex : médiane)

• Mais même si pas exact, on peut avoir une bonne estimation (qualité dépendant de la quantité de mémoire dispo)

**Fouille de flots de données** : Maintient sketch (représentation) du flot de données pour répondre aux requêtes

**Échantillonage de flots** : Puisque nous ne pouvons pas stocker tout un flot de données, une option consiste à stocker un échantillon de ses données Deux approches différentes :

1) Conserver un échantillon aléatoire de taille fixe sur un flot potentiellement infini

2) Échantillonner une proportion fixe de données dans le flot (disons 1 sur 10)

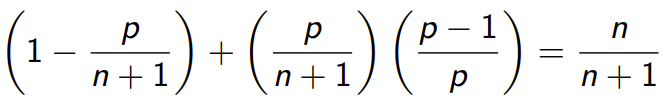
Une image contenant texte

Description générée automatiquement• À chaque unité de temps, t, nous gardons un échantillon aléatoire de données P.Échantillon de taille fixe : Supposons que nous devons maintenir un échantillon P de taille exactement p (ex : contrainte RAM)

Taille du reservoir = p = taille echantillon.

• Claim : Cet algorithme maintient un échantillon P avec la propriété que chacune des n données vues jusqu’ici a la même probabilité p/n d’être dans l’échantillon. [n > p] la n donne arrive et notre reservoir est plein 🡪 n/p proba keep.

• Preuve par induction : Supposons qu’après n données, l’échantillon contienne chaque donnée vue jusqu’ici avec probabilité p/n. Nous devons montrer qu’après avoir vu la donnée n + 1, l’échantillon maintient la propriété, c.-à-d., l’échantillon contient chaque donnée vue jusqu’ici avec probabilité p/(n + 1).• Case de base : Après avoir vu n = p données l’échantillon P a la propriété désirée. Chaque donnée parmi n = p est dans l’échantillon avec une probabilité p/p = 1

• Hypothèse inductive : Après avoir vu n = p données l’échantillon P contient chaque donnée avec probabilité p/n. Maintenant, il arrive la donnée n + 1. Pour une donnée déjà dans P, la probabilité que l’algorithme la conserve dans P est : (proba de conserver une donnee déjà ds le reservoir.)



• Conclusion : À l’instant n (n donnee) arrive, les données sont dans P(reservoir) avec prob. p/n (hypothèse inductive). À l’instant n + 1, la donnée demeure dans P avec prob. n/(n + 1). Donc, prob. que la donnée soit dans P à l’instant n + 1 est {proba que donne est ds le reserv \* proba donne stay ds rersv}

Échantillon de proportion fixe : Scénario : flot de requêtes d’un moteur de recherche. Flot de tuples : (user, query,time). On a de l’espace pour stocker 1/10 du flot de requêtes.

• Solution simple : Générez entier aléatoire [0..9] pour chaque requête. Stockez requête si l’entier est 0, sinon rejeter.

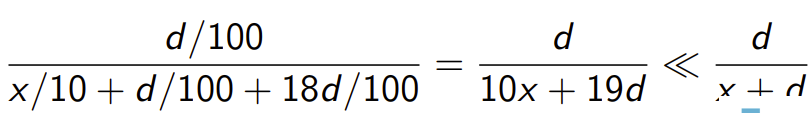
• Question : Chaque usager fait x requêtes une (1) fois et d requêtes deux (2) fois, des requêtes sont répétées.

Problème avec approche simple : L’échantillon va contenir x/10 des requêtes uniques. (nb x requettes simples)

• Seules d/100 des requêtes doublées seront aussi doublées dans l’échantillon : (d nb de requette doublees)

Une image contenant texte, horloge, jauge, montre

Description générée automatiquement• Des d requêtes originalement doublées 18d/100 apparaîtront exactement une fois

• Alors la réponse (incorrecte) à la question de départ à partir de l’échantillon sera :

Solution : échantillonner des requêtes par usager. • Prenez 1/10 des usagers ainsi que toutes leurs requêtes pour l’échantillon.

• Utilisez une fonction hash qui hache le id de l’usager de façon uniforme en 10 buckets. (rappel : (user, query, time))

• Solution généralisée : Pour obtenir un échantillon de taille a/b du flot, hachez chaque clé uniformément en b buckets et retenez le tuple si la valeur de hachage de sa clé est au plus a.

**Filtrage de flots** : • Considérons que chaque élément du flot de données est un tuple. • Input : liste de clés S • But : Filtrer les tuples du flot selon S • Applications : Filtrage de spam si nous connaissons une liste des « bonnes »adresses mail. Si un mail provient de l’une d’elles, ce n’est pas un spam, Web crawler : doit filtrer les URL déjà visitées

• On veut filtrer le flot et conserver uniquement éléments intéressants (certains usagers, catégories, attributs, etc)

• Solution triviale : table de hachage (hash table) qui stocke toutes les clés de S. Mais nécessite bcp mémoire…

Première solution : Soit un ensemble de clés S que nous voulons filtrer :

1) Créez un tableau de bits B de n bits, initialement tous à 0. 2) Choisissez une fonction de hachage h avec plage [0, n]

3. Hachez chaque **s ∈ S à l’un des n buckets**, et mettez ce bit à 1, c’est-à-dire, B[h(s)] = 1

4. Hachez chaque élément a du flot et gardez seulement ceux dont les résultats de B[h(a)] = 1

Bucket : une position dans le tableau de Bits (B). on a n buckets si B est de dim n. B[h(a)] -> bucket.

• Si a ∈ S alors B[h(a)] = 1 par déf. • Si a ∈/ S alors B[h(a)] = 0 ou 1 (faux positifs mais pas de faux négatifs)

Exemple : • |S| = 1 milliard d’adresses email.• |B| = 1 Go = 8 milliards de bits.

• Si l’adresse e-mail est en S, alors il hache à un bucket dont le bit vaut 1 B[h(a)]=1 (pas de faux négatifs)

• Environ 1/8 des bits mis à 1, donc environ 1/8 des adresses qui ne sont pas en S sont erronément filtrées (faux +)

• Alors, le probabilité de faux positifs est S/B = nombre approx. de buckets dont le bit vaut 1

Analyse de faux positifs : Mais nous pouvons faire une analyse plus approfondie. Si nous lançons |S| fléchettes dans |B| cibles également probables, quelle est la probabilité qu’une cible soit atteinte ?

• Cibles : buckets / fléchette : code hash h(s) des différentes clés s ∈ S | B[h(s)]

• Les faux positifs arrivent lorsqu’un bucket (cible) est associé au code hash d’une clé quelconque de S (2 data 🡪 meme hash aka collision).

associer le mauvais element a filter avec la mauvais cle. Ex : Cle : bob h(bob) =1. Ds flow data : alice. H(alice) = 1. 🡪 faux pos.

• Probabilité qu’un bucket soit marqué par un élément de S (vrai pos): 1/|B|

•Probabilité qu’aucun élément de S marquent un bucket en particulier (vrai pos) [le faux pos : 1-… ]

|S| : nb element ds lensemble des cle. |B| : dimension du tableau(nb bucket)

• Donc, la probabilité qu’un bucket soit marqué (faux positif aka prop de get 1) = 1 − e−|S|/|B|

• exemple : (|S| = 109 fléchettes , |B| = 8 · 109 cibles)

• Notre estimation précédente de faux positifs était 1/8 = 0.125, la nouvelle est de 1 − e−1/8 = 0.1175

Pr diminuer le nb de faux positif : on peut augmenter le nb de hash fcts

Filtres de Bloom : Considérez : |S| = m, |B| = n. Utilisez k fonctions hash h1, . . . , hk

• Initialisation : Faites B ← 0. Hachez chaque élément s ∈ S en utilisant chaque fonction de

hachage hi, c.-à-d. faite B[hi(s)] = 1 pour i = 1, . . . , k (pour permettre multiples vérifications de collisions). On met tt les positions obtenu par le hash de tt les cle de S (c basically le dictionnaire de quoi est chill) = 1

• Run-time : Quand élément avec clé x arrive : Si B[hi(x)] = 1 pour tout i = 1, . . . , k, alors déclarez x ∈ S. Sinon, jeter x

Analyse de faux positifs : • Nous avons maintenant k fonctions de hachage.

• On lance k · m fléchettes sur n cibles (m = |S| :nb de cle a filtrer), donc la fraction du vecteur binaire de bits à 1 : (1 − e−km/n)

• Nous filtrons l’élément x si B[hi(x)] = 1 pour tout i = 1, . . . , k .

• Les fonctions de hachage sont indépendantes, alors, la probabilité d’un faux positif est (1 − e−km/n)kValeur optimale de k =n/m\*ln(2) n : nb de bucket-taille mem.

Filtres de Blooms : • Les filtres Bloom garantissent l’absence de faux négatifs et utilisent une mémoire limitée :

• Approprié pour implémentation sur hardware (Les calculs de fonction de hachage peuvent être parallélisés)

• Est-il préférable d’avoir 1 gros B ou k petits B ?- C’est pareil : (1 – e−km/n)k contre (1 − e−m/(n/k))k

Lorsque la fct de hash , hash les cle uniformement. Prop faux pos = nb elem a filtre/taille bucket(mémoire limite).

Taux de faux positif vaut 1 🡪 aucun elem sera filtre.

**Chap 10 b -Apprentissage de representation des graphes**

Pipeline fouille donnee : data prepared before. Raw data 🡪 structured data 🡪 learning algo 🡪 model.

Incorporation des sommets : passer dun graph de d sommet a un espace de dimension d. conserve tt les relations. Relation geo ds espace appris sont les meme que ds le graph original. (conservation features et espace).

->embedding graph… peut predire classe pr des nœuds manquants, arret manquant.

Graph G =(V,A). V: ensemble sommet, A : mat adjacence. [juste base sur la topologie].

Apprentissage incorporation de sommet :

1)definir encodeur (mappring des sommets a espace incorporation) 2) definir fct de similarite entre sommet (c une mesure similarite ds graph origine ie. Nb chemin entre 2 nœud, + courte dist,etc) 3) optimisation param encodeur pr

Similarite(u,v) . 🡪 encodeur son but c estimer la similarite en 2 vecteur.

Codage simple : Attrib chaqie sommet un vect incorporation unique. (node2vect,…). Distinction entre methode codage simple : cmt la similarite est definie entre les sommets ds le graph origine (aka info quon veut preserver ds espace origine de embedding).

2 sommet devrait-il avoir incorporation similaire si -sont connecter ? si ya des arrets, it means sont connecter. – ont voisin en commun ? oui. – ont des roles structurels similaires? Oui car si on enleve les arretes entres ses sommets structurel -> graph deconnecter.

Similarite base sur matrice adjacence : fct similarite = juste poids de arrete entre sommet u et v ds le graph origine.

-> Prod scalaire entre incorporations de sommets rapporche existance arretes.

Fct perte : 2  L : perte, Zut,Zv -> similarite ds espace incorpo. A : mat adjacence. But : minimiser la fct de perte pr tt paire sommet ds graph si ya un arrete entre 1 (=1 ds la amt A).

Inconveniens : - temps O(|V|2) V : nb arret. 🡪 possible optimiser a 0(|E|). – optimisation considere connect locale et directe. (nœud non connecnter directement mais + proche sont + similaire que 2 nœud non connect. Et dist() >)

Similarite multi-hop : fix ce last issue

A picture containing text, diagram, font, line

Description automatically generated

Incorporation par marche aleatoire :

1)calculer proba (ak marche aleatoire) de visiter sommet v lors marche aleatoire a partir sommet u.

2) optimiser incorporation pr encoder stats de marche.

A picture containing line, diagram

Description automatically generatedoptimiser angle teta entre 2 vect. Teta inversement proport au proba des vect v et u. si proba atteindre v par u est petite 🡪 angle grand.

Expresivite : definition stochastique flexible de similarite entre sommet inegre a la fois info de voisinage locales ordre supperieur.

Efficience : pas prendre compte tt paires de sommets lors entrainement. Considerr paires qui coexiste lors marche aleatoires. (faire marche aleatoire les sommet pas atteignable on vag)

Optimisation marche aleatorie

1)faire courte marche aleatoire a partir each sommet ak strargie R.

A picture containing text, font, screenshot, line

Description automatically generated2) chaque sommet u collect NR(u) (ensemble de sommet visites pr marche aleatoire debutant sommet u. {suite a ++ marche mettre tt sommet visited}

3) zu : proba atteindre sommet v a partir nœud u. on veut optimiser

A picture containing font, handwriting, text, line

Description automatically generatedA picture containing font, text, graphics, white

Description automatically generatedMinimiser fct



With softmax : cauz on veut sommet v soit + similaire sommet u parmis tt les sommets.

A picture containing text, font, screenshot

Description automatically generated

Quel methode use? Aucune est meilleur que lautre. Node2vec work mieux ak classification.mult-hop work pr prediction lien.

Prob en commun : - methode codage simple partage aucun param entre sommets. – exploitent pas attrib sommets – sln : graph neural networks.

**Exercices**

**1. a) Supposez que l’algorithme k-moyennes a été exécuté sur X. Est-ce que la partition obtenue à la fin**

**correspond au minimum global pour (1) ? )** L’algorithme de k-moyennes est un algorithme d’ optimisation locale. Par conséquent, il n’est pas garanti qu’il trouve la solution optimale globale suite à une exécution.

**b) Supposez que l’algorithme k-moyennes a été exécuté sur X pour k clusters, et qu’après convergence,**

**une partition avec k − 1 clusters est retournée par l’algorithme. Comment pourriez-vous améliorer**

**de façon triviale l’inertie de cette partition ?** Nous pouvons choisir un point au hasard pour composer le k-ème cluster. Cette partition est garantie d’être mieux que la partition obtenue par k-moyennes ayant k-1 clusters.

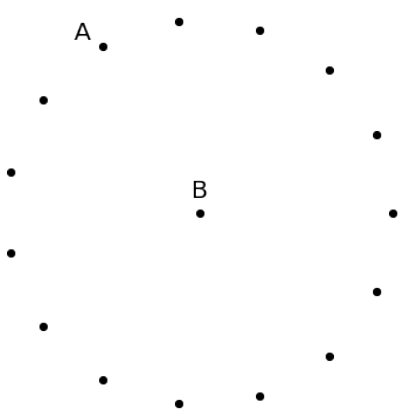
2. **Comment utiliser les k-plus proches voisins pour détecter les données aberrantes ? Quels sont les avantages**

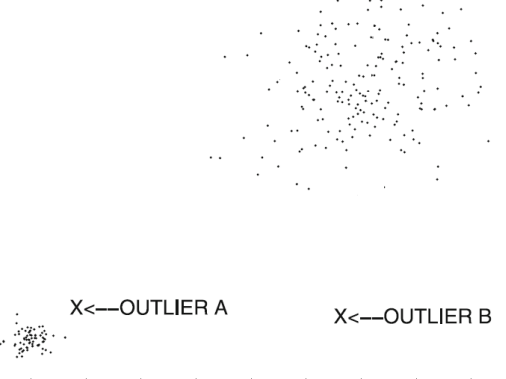
**et les inconvénients de cette approche ?** L’outlier score d’un enregistrement est donné par sa distance avec son k-plus proche voisin. Des variantes considèrent la moyenne des k plus proches voisins. Un enregistrement est considéré outlier si son score est supérieur à un seuil, ou si son score est parmi les r-plus grand.

+ Les données avec beaucoup de bruit n’ont pas de grands outliers scores selon ce modèle, sauf les

vrais outliers. + L’analyse est plus fine que celle utilisant le clustering. + La méthode est applicable pour n’importe quel type de donnée si la distance entre deux enregistrements est définie. - complexité : déterminer la distance d’un enregistrement à son k-plus proche voisin nécessite un temps O(n), soit O(n2) pour l’ensemble des données

**3. La distance euclidienne est-elle généralement bien adaptée pour détecter des données aberrantes ?** Non, la distance euclidienne n’est pas adaptée, car elle ne prend pas en compte la distribution des données. Une meilleure alternative est la distance de Mahalanobis. Aussi vrai pour méthodes basées sur les distances (k-plus proche voisin).

**4. Quel point de A ou B a le plus grand outlier score : a) Selon un modèle de clustering. Considérez que tous les points appartiennent au même cluster et que la méthode utilise la distance euclidienne.** A. L’outlier score d’un point est sa distance avec le centroïde du cluster auquel il appartient (généralement le plus proche). Le centroïde du cluster est le point B donc l’outlier score de B est nul et celui de A est égal à la distance entre A et B. **b) Selon un modèle basé sur les distances. Considérez k=2 et que la méthode utilise la distance euclidienne.** B. L’outlier score d’un point est sa distance avec son k-plus proche voisin. La distance entre A est son deuxième plus proche voisin (point directement à gauche ou à droite de A) est inférieure à la distance entre B et son deuxième plus proche voisin (n’importe quel point puisqu’ils sont tous à la même distance de B).

**5. a) Lequel de deux outliers est plus difficile à détecter automatiquement ? Pourquoi ?** L’outlier A parce que sa distance au centre de la classe est petite par rapport aux distances des points du cluster de droite à leur centre.

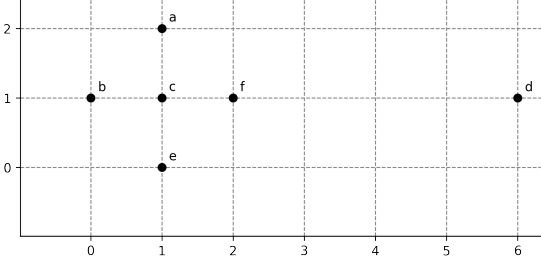
**b) Est-ce que l’outlier indiqué en (a) peut être identifié par son outlier score obtenu en utilisant un modèle de k plus proches voisins basé sur des distances euclidiennes ?** Justifiez. Cette méthode n’est pas appropriée pour détecter l’outlier A parce que les deux clusters présentent de densités très différentes.

**6. À quoi sert le graphe produit?** Le graphe produit sert à compter les marches communes dans G3 et G4 à l’aide du noyau K(G3, G4) =. Chaque marche dans le graphe produit correspond à une séquence appariée en termes de sommets dans G3 et G4.**7. Plutôt que d’utiliser la téléportation, le problème des dead-ends est résolu en les supprimant. Pour**

**supprimer une dead-end, le sommet et toutes ses arrêtes entrantes sont retirés du graphe. En se faisant,**

**d’autres dead-ends sont créées et doivent être aussi supprimées récursivement.**

**Quel va être l’impact sur la structure du web considérée dans le calcul de PageRank ?** — Sites fortement connexes (SFC) : il existe un chemin entre chaque sommet (définition de fortement connexe), donc chaque sommet à au moins une arête sortante et aucun sommet n’est supprimé. Remarquez qu’un sommet sans d’arête sortante ne peut atteindre aucun autre sommet. — Composantes entrantes (CE) : il existe un chemin entre chaque sommet des CE et les SFC, donc chaque sommet à au moins une arête sortante et aucun sommet n’est supprimé. — Composantes sortantes (CS) : certains sommets sont des dead-ends et sont supprimés. Cela va créer de nouvelles dead-ends qui vont être supprimées à leur tour. Récursivement, une grande partie des CS vont être supprimés. Seuls les (petits) ensembles de sommets qui sont fortement connexes ou contenant un spider trap sont conservés sous la forme de composantes isolés. — Attaches sortantes : même analyse que pour les CS. — Attaches entrantes : en supprimant récursivement les CS, les attaches entrantes qui étaient liées à un sommet des CS qui a été supprimé vont devenir des dead-ends, et vont être supprimées récursivement à leur tour. Seuls les (petits) ensembles de sommets qui sont fortement connexes ou contenant un spider trap sont conservés sous la forme de composantes isolés. — Tubes : même analyse que pour que les attaches entrantes. — Composantes isolées : si les composantes isolées contiennent des dead-ends, alors ils vont être supprimés partiellement ou entièrement. Seuls les (petits) ensembles de sommets qui sont fortement connexes ou contenant un spider trap sont conservés sous la forme de composantes isolées.

**8. Considérez le jeu de données suivant par ordre alphabétique et utilisez la distance euclidienne. Nous souhaitons détecter le point avec le plus grand outlier score tel que mesuré par la distance à son plus proche voisin. Autrement dit, k = 1 et r = 1. Appliquez la méthode du Sampling avec l’échantillon S = {a, b}. Quelles distances n’ont pas été calculées ?**

Il faut calculer toutes les distances par paires entre les points dans S et tous les points. Soit D la matrice des distances connues (la diagonale est indiquée pour améliorer la lisibilité).Le score de a est s(a) = d(a, c) = 1 (top-1). Le score de b est s(b) = d(b, c) = 1. En cas d’égalité, point déjà trouvé conservé

Le score estimé de c est sˆ(c) = d(a, c) = 1. Puisque sˆ(c) est un upper bound, le score réel de c est inférieur ou égal à 1. Le top-1 outlier déjà trouvé (a) à un score de 1 donc c n’est pas candidat.

Le score estimé de d est sˆ(d) = d(a, d) = 5.1 > s(a) donc d est candidat et il faut calculer son score réel.

Le score réel de d (plus petite distance) est s(d) = d(d, f) = 4 > s(a), donc d remplace a (top-1).

Le score estimé de e est sˆ(e) = d(b, e) = 1.41 ≤ s(d) donc non candidat.

Le score estimé de f est sˆ(f) = d(a, f) = 1.41 ≤ s(d) donc non candidat.

Le point avec le plus grand outlier score tel que mesuré par la distance à son plus proche voisin est d. L’utilisation de la méthode du Sampling à permis de ne pas calculer les distances d(c, e), d(c, f), et d(e, f).

Expliquer ce qu’est une mesure de centralité et de prestige d’1 sommet? indicateur centralite = mesure cense capturer la notion import ds graphe, en identifiant les sommet + significatifs. Application des indicateurs incluent identifi personne la + influente ds reseau social.

A quoi sert le graph produit (graph resultant de la marche aleatoire) ? Graph produit sert a compter les marches communes ds G3 et G4 a laide du noyeau k(G3,G4) = formule ak lambda. Chaque marche ds le graph produit corresp a une sequence appariee en termes de sommets ds G3 et G4.

Reseau social Facebook? Graph simple (pas de boucle ni arret multiple). Pas possible detre son propre ami ou ami plusieurs fois avec la meme eprsonne.(simple). Si t ami ak qln, il est ton ami aussi (non oriente). Tt le monde a petit gr dami (non dense). Pas de poids ds un lien amitie. Pas de relation topologique. Reseau social comme twitter ? oriente et simple. Si tu suit qln nimplique pas quil va te suivre (oriente) pas possible de se suivre ou de suivre qln ++ fois.(simple) majorite des gens suivent pas tt les utilisateurs (non complet). Pas de poids ds un lien. Pas relation topologique. Reseau routier canadien? Oriente, et topologique. Route ont sens de circulation (oriente, non simple). Tt villes pas connecte (non complet). Graph peut etre ponderer (poids = taille ou nb utilisateur sur route).

3 contraintes de la fouille de flow de data. Quel caracteristique du big data les flots de donnees corresp? Flot de donnees corrsp a aspect vitesse (velocite). Contraintes : - mémoire limite pr store data – temps limite pr traiter chaque element – acces sequentiels (pas acces aleatorie) – 1 chance de voir chaque element du flow.. 🡪 connaissance a priori des info interet necessaire. – bcp quantite ne peuveut pas etre calculer exactement (ex : mediane).