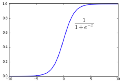
**Cours 8 – Classification**

• Droite de régression coupe ces classes, même s’il existe un séparateur minimisation de l’erreur carrée.

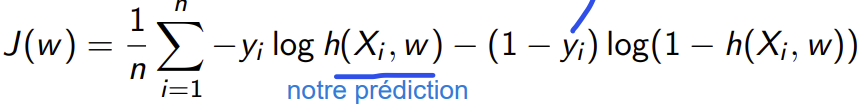
• Régr. logistique: 1ère métho pour trouver f() de séparation de 2 classes

**Fonction sigmoide :**

• Régression logistique vise à convertir une valeur continue en valeur de proba ∈ [0, 1].

• Fonction sigmoïde : donne la probabilité de Xi d’appartenir à une classe particulière. • f (0) = ½ • f (∞) = 1 • f (−∞) = 0

• Pour étendre la fonction sigmoïde à d attributs, on fait : et ensuite

**Fonction d’erreur – entropie croisée :**

• La régression logistique utilise une fonction d’erreur différente de celle de la régression linéaire : l’entropie croisée. y est binaire, alors 1 seul des termes de la somme est actif pour chaque enregistrement

• L’entropie croisée = fonction convexe ! On peut trouver le meilleur séparateur entre 2 classes avec méth du gradient

• L’erreur est nulle seulement si les classes sont linéairement séparables

• Afin d’utiliser la régression logistique pour séparer des classes non linéairement séparables, il faut ajouter des colonnes non linéaires (ex : x2, √x) dans X

**Classification – Problèmes :**

1. Ensemble d’entraînement déséquilibré (ex. identification d’un terroriste)

• Considérons la droite de séparation optimale pour les classes grossièrement déséquilibrées, disons 1 exemple positif contre 1,000,000 exemples négatifs.

• La meilleure droite trouvée par la régression logistique essaiera d’être très loin du grand groupe des gens non terroristes au lieu de se placer entre les classes.• Présence des faux négatives !

• Même classer tout le monde comme non terroriste aura peu d’impact pour l’entropie croisée.

• Solution (si possible) utilisez le même nombre d’exemples + et -.

- Supprimer des éléments de la classe majoritaire.

- Pesez plus lourdement les données de la classe minoritaire, mais méfiez-vous du surapprentissage ou overfitting.

- Répliquer les membres de la classe minoritaire, idéalement avec une perturbation aléatoire (valeurs gaussiennes)

2. Classification multi-classe

• Les tâches de classification ne sont pas toujours binaires (film est-il comédie, drame, sci-fi, documentaire, etc.)

• Sélectionnez la classe de probabilité la plus élevée comme étiquette prédite

• La classification multi-classe devient beaucoup plus difficile à mesure que le nombre de classes augmente.3. Fonctions de partition

• Les classifieurs binaires indépendants ne produisent pas de vraies probabilités (la somme ne vaut pas 1).

• Une possible solution : Une image contenant horloge, montre

Description générée automatiquementoù k est le nombre de classes



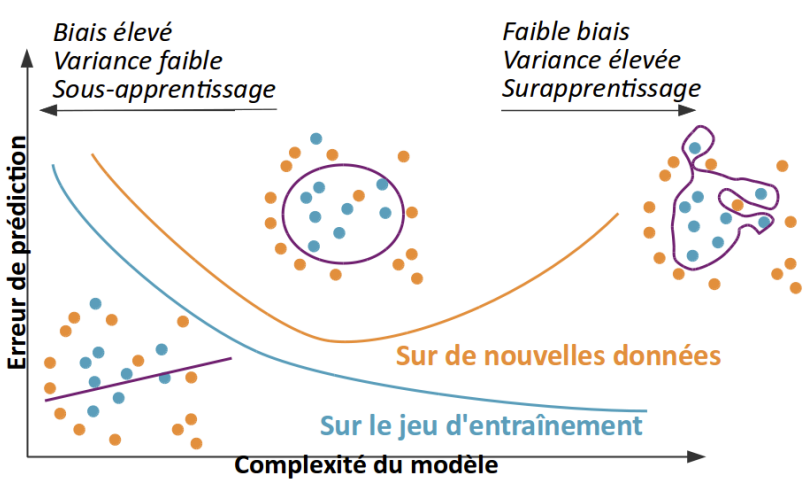
• La régression multinomiale (softmax) combine les classifieurs au niveau de l’entraînement.**Limites de la régression :** On a besoin d’attributs non linéaires (ex. x1x2) pour que la régression puisse bien séparer des données non linéairement séparables (implique énormité combinaisons d’attributs à tester)

**Apprentissage profond** : • Le domaine le plus en vogue de l’apprentissage machine implique aujourd’hui de grandes architectures de réseaux neuronaux profonds.

• Les couches cachées créent de compositions de fonctions non linéaires ⇒ un plus grand pouvoir de représentation.• Faire apprendre le réseau signifie définir les valeurs des coefficients w (pour résoudre problème : minimiser erreur)

• Plus il y a de connexions entre les neurones, plus il y a des paramètres à apprendre.

• L’apprentissage consiste à analyser l’ensemble de données d’entraînement étiquetées et à ajuster les coefficients de telle sorte que les nœuds de sortie génèrent quelque chose proche de yi lorsqu’ils le réseau est alimenté avec Xi

• Les réseaux de neurones semblent fonctionner par surapprentissage, trouvant un moyen d’utiliser des millions

d’exemples pour s’adapter à des millions de paramètres

+ Flexibilité en termes de paramètres.

+ Grand nb de librairies et d’aides techniques disponibles.

+ Mise en échelle plus naturelle que d’autres modèles

Surapprentissage**:** Lorsqu’un modèle est suffisamment flexible ou complexe, il est capable de prédire parfaitement les données de l’ensemble d’apprentissage. Cependant, il n’est alors plus capable de généraliser, c-à-d de prédire correctement la sortie de nouvelles données. Le surapprentissage est caractérisé par un faible biais et une variance élevée. La fonction apprise doit avoir une bonne performance de généralisation. Lié à la complexité du modèle (nb itérations descente gradient) et non au temps d’entrainement ou qualité/quantité de données. En surapprentissage, on est bons pour un jeu de données, mais on se trompe si on change les données. En sous-apprentissage, le modèle n’est pas assez complexe pour résoudre le jeu ou n’importe quelle autre donnée

• L’apprentissage profond généralement évite le pire comportement du surapprentissage, peut-être en utilisant des moyens moins précis d’encoder la connaissance.

• Chaque neurone j du réseau calcule une fonction non linéaire φ(vj) où v est donné par :

où : tij est l’input au neurone j avenant du neurone i et wij est le poids donné à cet input.

**Backproportion :** La sortie dépend de la couche d’avant, qui dépend de la couche d’avant…

• Les réseaux de neurones sont entraînés par un algorithme de type descente de gradient (préf. stochastique).

• Les changements pour chaque exemple d’entraînement sont ramenés à des niveaux inférieurs.

• Les fonctions d’activation non linéaires aboutissent à une fonction d’erreur non convexe, mais son optimisation produit généralement de bons résultats.

**Machines à vecteurs de support** (SVM) : cherchent des séparateurs linéaires de marge max entre 2 classes

• Si données linéairement séparables -> nombre infini d’hyperplans séparateur linéaire entre les classes

• Instance de test placée dans une région limite entre 2 classes. Classifieur avec + grande marge préféré ⇒ + robuste

• Vecteurs de support : points de données d’entraînement sur hyperplans // à l’hyperplan séparateur classifieur

• Hyperplan séparateur précisément au milieu de ces deux hyperplans afin d’atteindre la classification la plus précise

• L’hyperplan séparateur est donné par : wTX + b = 0

• Les d + 1 coefficients de w et b doivent être appris à partir des données d’apprentissage pour maximiser la marge

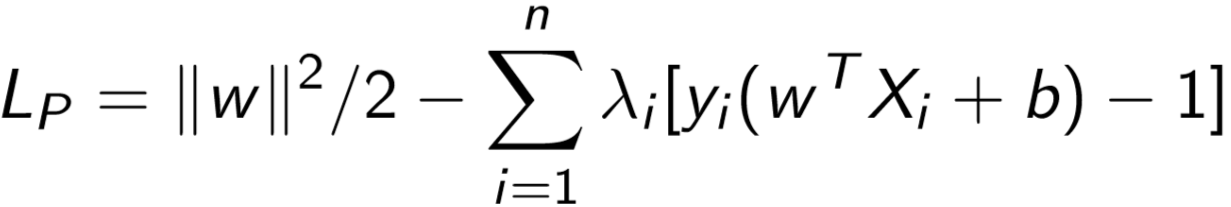
• On assume que : 1. données d’entraînement sont linéairement séparables 2. les classes sont étiquetées +1 et -1

Une image contenant texte

Description générée automatiquement• Problème d’optimisation qu’on veut maximiser exprimé comme :

• Distance M entre 2 hyperplans est 2/∥w∥. Maximiser cela = minimiser ∥w∥2/2

• Chaque donnée d’entraînement ajoute une contrainte au modèle

• Utiliser relaxation Lagrangienne : Associer ensemble n de multiplicateurs non-négatifs λ = (λ1 . . . λn) ≥ 0 pour les différentes inégalités de la marge. Les contraintes sont ensuite relaxées et la fonction objectif est augmentée en incorporant une pénalité lagrangienne

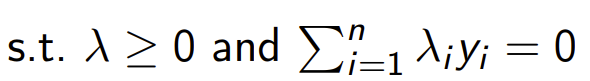
• Si LP est minimisé par rapport à w et b pour un λ particulier, puis maximisée par rapport aux multiplicateurs λ, la solution duale résultante LD est une borne inférieure de la fonction objectif optimale O de la SVM linéaire

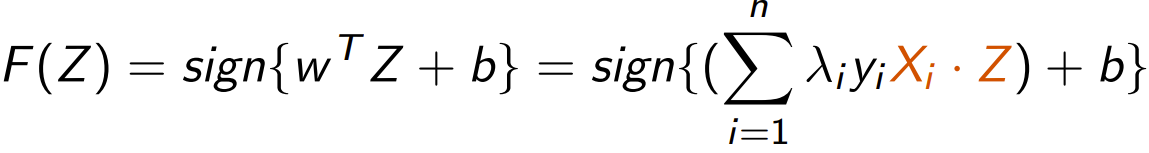
Une image contenant texte

Description générée automatiquement• SVM linéaire : résultat est encore + fort puisque f() convexe (O=LD)

• En utilisant les conditions d’optimalité de premier ordre, c’est possible prouver que :

Une image contenant objet, horloge

Description générée automatiquement

Ainsi que Une image contenant texte, horloge

Description générée automatiquement • Pour une instance de test Z, sa classe F(Z) est défini par :

Une image contenant texte

Description générée automatiquement• Remarque que la classification ainsi que la valeur de la f.o. de la SVM sont obtenus sans qu’on ait besoin de connaître la matrice de données X directement • Tout est exprimé par des produits scalaires

• Cela est crucial pour l’application de la SVM pour des données pas linéairement séparables (kernel trick)

**SVMs non-linéaires** • la formulation SVM peut être entièrement résolue en termes de produits scalaires (ou similarités) entre des paires de points de données

• la clé est de définir le produit scalaire (ou la fonction de similarité) directement dans la représentation transformée Φ(X) avec l’utilisation d’une fonction noyau K(Xi, Xj) K(Xi, Xj) = Φ(Xi) · Φ(Xj).

• Cela implique : *Remplacer la partie orange des formules en haut par* K(Xi, Xj)

• Tous les calculs sont effectués dans l’espace d’origine avec la fonction noyau choisie

• La transformation réelle Φ n’a pas besoin d’être connue tant que la fonction noyau K soit connue

• En utilisant la similarité basée sur des noyaux choisis avec soin, on obtient des SVMs non-linéaires

• Il y a différentes manières de modéliser la similarité entre les données avec des noyaux

• Chacune a des avantages sur certains ensembles de données, il est donc nécessaire de jouer avec les options disponibles dans les librairies pour obtenir les meilleures performances

**k plus proches voisins**

1. Trouver les k enregistrements de X qui sont les plus proches de la nouvelle donnée Z que l’on veut classifier

2. Faire voter chacun de ces enregistrements selon leur classes associées y 3. Retourner la classe majoritaire

• Succès dépend de 2 facteurs: qté données d’entraînement et qualité mesure de distance (2 enregistrements similaires doivent faire partie de la même classe)

• Grand pouvoir de représentation (peut classifier des données non linéairement séparables)

• Par contre, performance liée à notre capacité de stockage

**Cours 8 – Arbres de décision** méthode simple, facile, rapide bons résultats, séquence décisions pour classer

Une image contenant texte

Description générée automatiquementAlgorithme exhaustive optimal créer tous les arbres, tester avec données et choisir l’arbre avec erreur minimale

Désavantage temps de calcul prohibitive pour données possédant bcp de features

Solution heuristique gloutonne (au lieu de maximiser le gain final, on maximise le gain de la prochaine étape)

**Choisir meilleur feature?**

• Entropie mesure incertitude d’une distribution de proba

• Soit N le nombre d’exemples de notre jeu de données, l’entropie est :

Une image contenant texte, horloge, montre

Description générée automatiquementoù n est le nombre de possibilités

• Notez que ni/N est une approximation de la probabilité p(A = ai)

• On appele gain (G) la différence entre l’entropie initiale et l’espérance de l’entropie après que feature A est choisie

• Critère glouton dans l’algorithme de construction d’une arbre de décision choisit le feature qui maximise le gain

• Mieux vaut s’arrêter lorsque le gain d’information est faible, au lieu de zéro. On peut aussi construire tout l’arbre et éliminer quelques ramifications de gain faible

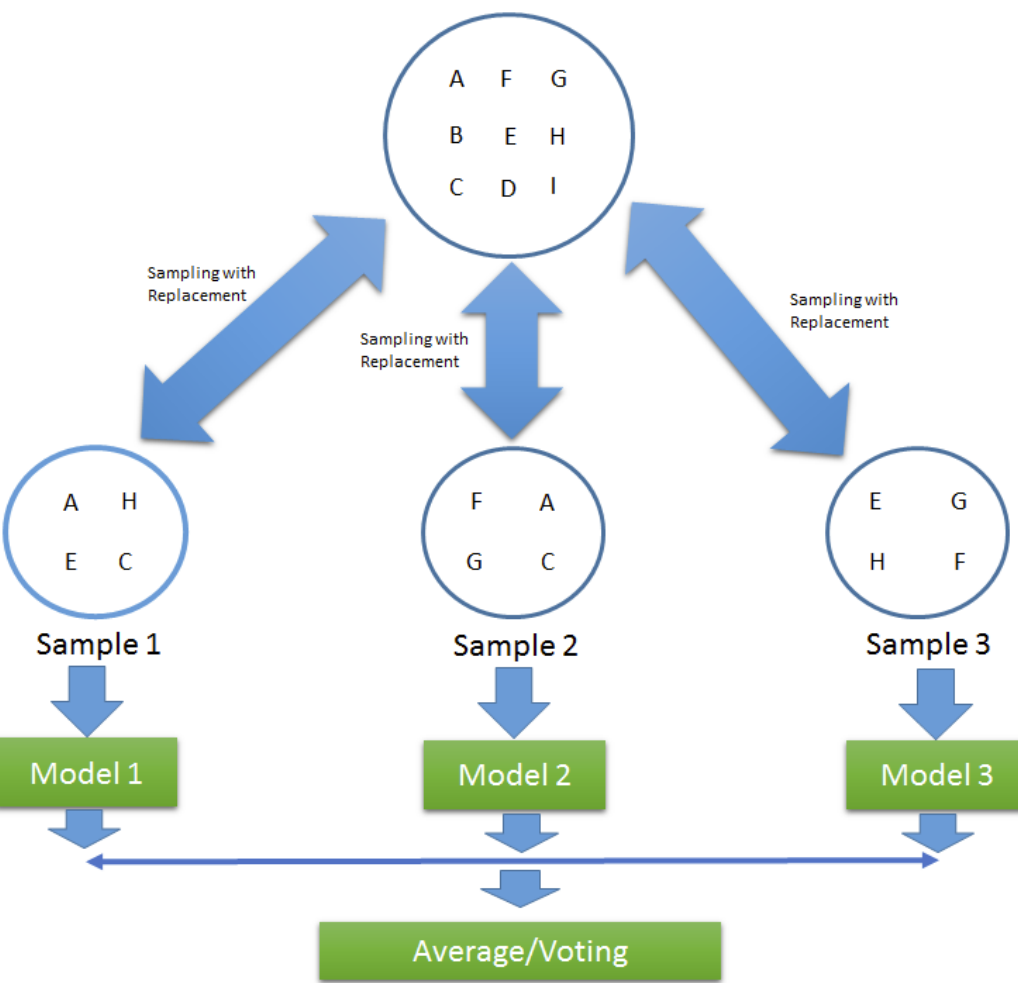
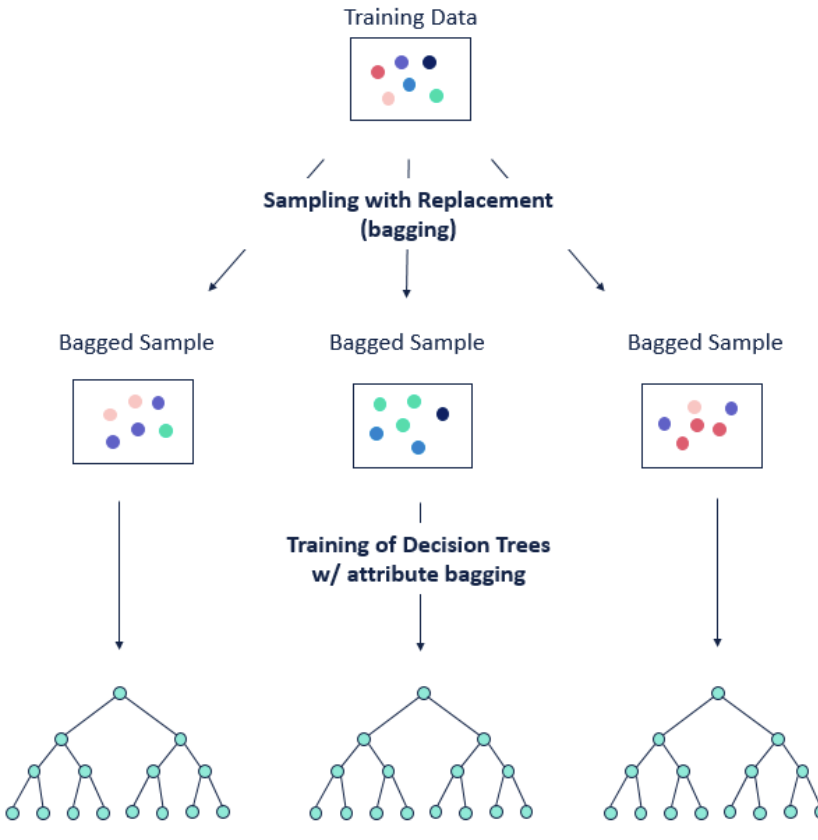
**Ensemble d’arbres de décision :** • Combine plusieurs arbres de décision pour produire meilleure prédiction

• Principe de base du modèle d’ensemble : un groupe de modèles faibles se réunit pour former un modèle fort

• Vote majoritaire entre plusieurs classificateurs ++ robustesse et permet de quantifier notre niveau de confiance

**Bagging :** • Choisit sous-ensembles de données choisis au hasard pour entraîner chaque arbre• -- surapprentissage

**Foret d’arbres de décision :** • Extension de la technique de bagging • Effectue également sélection aléatoire des feature au lieu de toutes les utiliser pour construire les arbres. • Diminue corrélation entre arbres

• Un feature très dominant oblige chaque arbre de décision à le choisir pour les premiers splits, ce qui fait que tous les arbres se comportent de façon similaire • La forêt d’arbres est composée par des arbres de décision non corrélés.

**Cours 8 – Évaluation de modèles**

**Procédure d’évaluation d’un algorithme de classification**

• Souvent, nous n’avons pas assez de données pour les séparer en trois sous-ensembles. La validation croisée partitionne les données en k blocs de taille égale. Puis, on entraîne k modèles distincts

• Le modèle i est entraîné sur l’union de touts les blocs sauf 1 (le bloc d’index i), et validé sur le bloc i

• Bagging combine tout les k modèles entraînés dans un seul modèle de classification

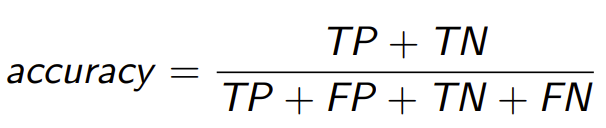
**Évaluation :** 4 résultats possibles : True Positive (TP), False Positive (FP), False Negative (FN), True Negative (TN)

Accuracy : Rapport entre les prédictions correctes et les prédictions totales. Classificateur randomisé : espérance d’accuracy de 50%. Classificateur gourmand (classe + nombreuse comme prédiction) : accuracy ≥ 50%. Serait de 95% si

Quand |P| << |N|(une classe sur-représenté) accuracy inutile.

**Précision** : Instances positives prédites. Plus sensible à l’obtention de la classe positive. Impossible d’avoir une précision élevée pour un classificateur gourmand ou randomisé. + bien prédits / tous les + prédits

**Recall** : Capacité d’avoir raison sur les instances positives. On peut être plus tolérants aux faux positifs qu’aux faux négatifs. Dire que tout le monde a le cancer donne un recall parfait, mais une mauvaise précision.

Une image contenant texte

Description générée automatiquementUne image contenant texte, horloge

Description générée automatiquement

**F-score** : Mesure équilibrée de score unique. 2 \* (precision \* recall) / (precision + recall)

**Exercice**

Une image contenant horloge

Description générée automatiquement• Précision > recall : Classificateur classifie peu des données comme de la classe positive. Augmenter le poids de l’erreur sur l’entraînement des données de la classe positive

• Recall > précision : Classificateur est trop agressif pour classer les données comme appartenant à la classe positive. Augmenter le seuil de classification de la classe positive

Note : Si classe + majoritaire, p > N – p. Si classe – majoritaire, p < N - p

**Receiver-Operator Curves (ROC)** : Faire varier le seuil d’un modèle modifie le recall et la precision (seuil = 1, tout est -, si seul = 0, tout est +).

• (TP) Rate =TP/P et (FP) Rate =FP/N La surface sous une ROC est une bonne mesure d’évaluation d’un modèle.

**Évaluation de systèmes multiclasses :** Classification + difficile avec + de classes

• On note C[i, j] le nombre d’objets de la classe i classés comme de la classe j

• precisioni est la fraction de tous les objets déclarés de la classe i qui étaient en fait de la classe i

• recalli est la fraction de tous les membres de la classe i qui ont été correctement identifiés comme tel :

Une image contenant texte, horloge, antenne

Description générée automatiquement Une image contenant texte, horloge

Description générée automatiquement

• Taux de réussite du top-K vous donne du crédit si la bonne étiquette aurait été l’une de vos premières suppositions

• Il est important de choisir K afin que de réelles améliorations puissent être reconnues

• Si K = au nombre de classes, le taux de réussite est 100% • Si K = 1 on a simplement l’accuracy

• Normalement, on choisi K tel que la performance du classificateur est supérieure à celle d’un classificateur aléatoire

• Mais pas trop mieux, de façon qu’on ait encore de l’espace d’amélioration (K pas trop élevé ou trop bas)

**Évaluation pour la régression**

• Pour les valeurs numériques, l’erreur est une fonction de la différence entre la prévision f (X) et l’observation y :

• |f (X) − y| • |f (X) − y|/y (normalement meilleur) • (f (X) − y)2 (toujours non-négatif)• Peuvent être agrégés sur de nombreuses données, ex : et RMSE =

**Cours 9 – 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** :

• Certaines mesures de similarité naturelles pas des distances : Coefficient de corrélation (-1 à 1), Produit scalaire

• Tandis que d’autres le sont sans que cela soit évident : edit distance (delete, substitute, insert entre 2 string)

Une image contenant texte

Description générée automatiquement**Distances euclidiennes** : Métrique de distance euclidienne traditionnelle pèse toutes les dimensions pareil : • Coefficients pour donner poids différents à chaque dim • Normaliser pour rendre les dimensions comparables.

**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** :

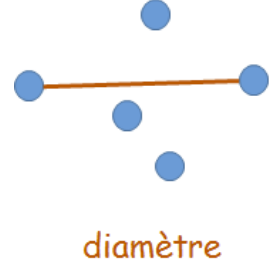
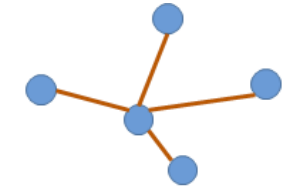
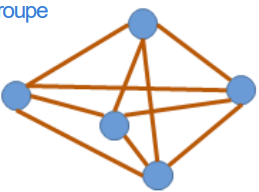
**Jaccard**

Une image contenant texte, horloge, jauge

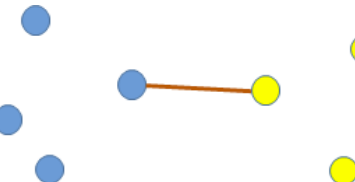
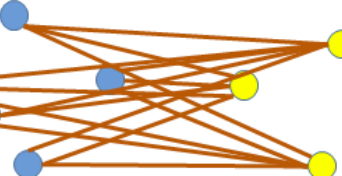
Description générée automatiquement**distance** :

= 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)

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

Une image contenant texte

Description générée automatiquementDistance , Centre

• 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)

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

Approche d’agglomération : 1. Commence avec une partition initiale avec n singletons 2. Fusionne successivement les clusters en fonction d’un critère local 3. Répète jusqu’à ce que toutes les données appartiennent au même cluster

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.

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 ≤ , ils sont voisins

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

Approche : 1. Point au hasard Xi. 2. Regroupe Xi avec ses -voisins (cluster) 3. Si un point est trouvé comme part d’un cluster, son -voisinnage est lui aussi part du même cluster 4. Si ce processus abouti à un cluster d’au moins m points, on le garde. Sinon, ils sont considérés du 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

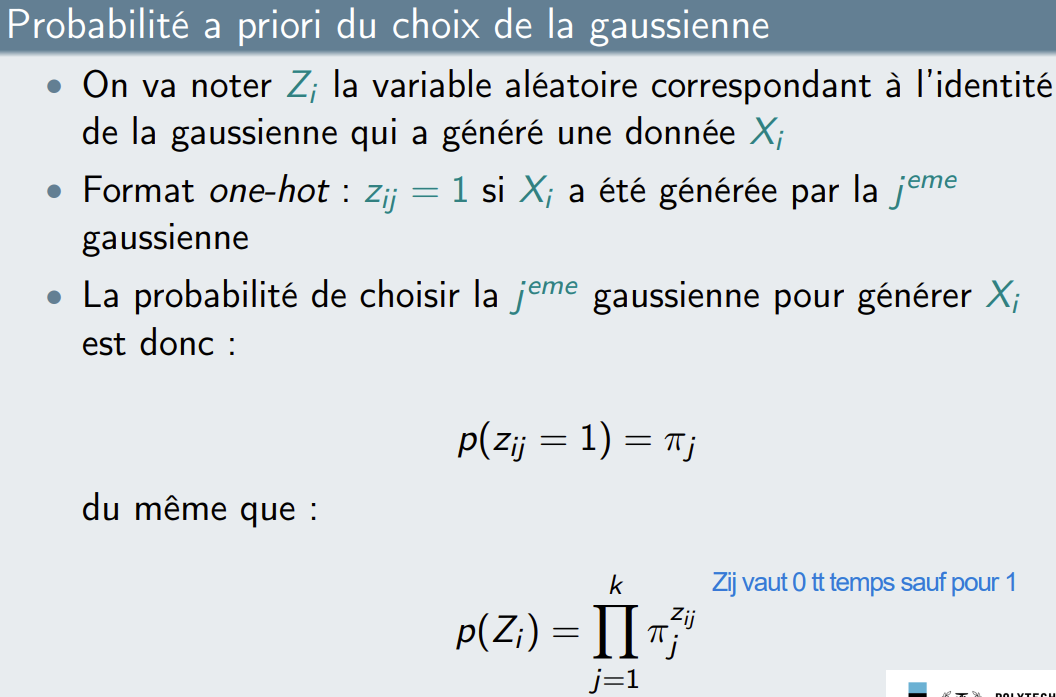
- Qualité clustering dépend bcp params. Mauvais pour jeux de données avec grandes différences densité

**Type 4 : Regroupement par des mélanges**. Mélanges de gaussienne. Produit clusters soft (1 point dans + d’un cluster)

• Permet de savoir les probas d’appartenance des données aux différents clusters

• Suppose que les données ont été générées comme suit :

• Pour i = 1 . . . n, un entier j = 1, . . . , k est choisi selon les probas π1, . . . , πk et Xi est généré selon une loi de proba (Xi|µj, Σj). Autrement dit, les données sont échantillonnées à partir de j = 1, . . . , k différentes distributions gaussiennes, ayant chacune une moyenne µj et une covariance Σj

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement• Nous voulons regrouper nos données en fonction des probabilités d’appartenance à chacune des gaussiennes

Une image contenant texte

Description générée automatiquement• On ne connait pas les gaussiennes (µ, Σ). Il se peut que les données de chaque cluster soient pas générées des gaussiennes. C’est une hypothèse de ce modèle de clustering qu’on peut changer : modèles avec (d’autres) mélanges

Une image contenant texte

Description générée automatiquement

Une image contenant horloge, montre

Description générée automatiquementUne image contenant texte, horloge

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

Une image contenant texte, horloge, montre

Description générée automatiquementsi 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)

Une image contenant texte

Description générée automatiquement**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

**Cours 9 – 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 probabilistes** : • Adaptation directe de l’algo EM (gaussian mixture model). On calcule somme j = 1, . . . , k (voir formule fonction de densité marginale des données). Ceci est un outlier score. Si trop petit -> Outlier trouvé

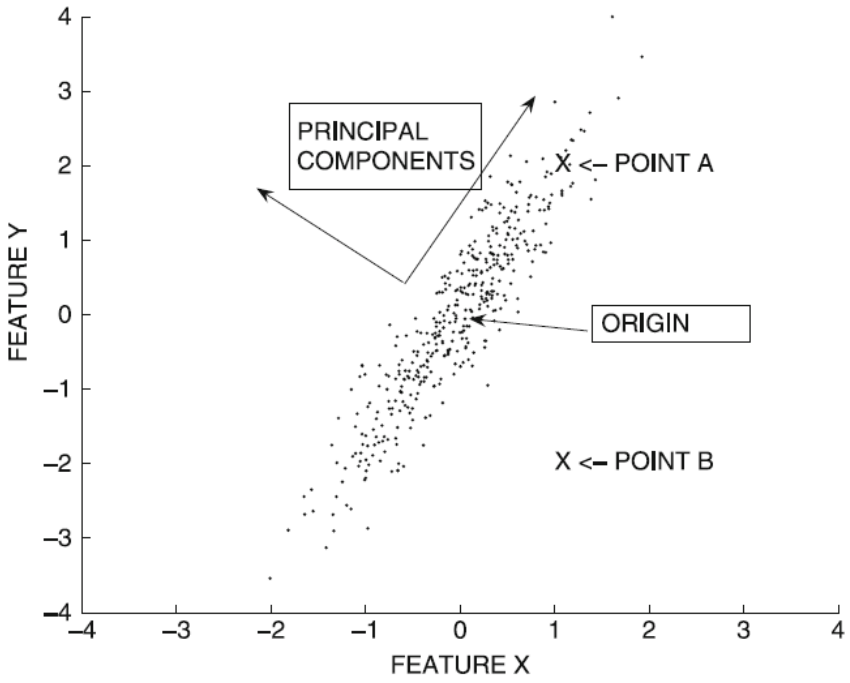
**Modèles par clustering** : • Clustering et détections d’outliers partagent une relation complémentaire bien connue.

• Détection d’outliers en tant que produit secondaire des méthodes de clustering n’est pas une approche appropriée.

• 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.

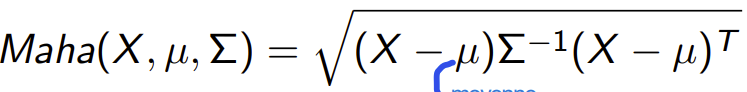
• D’autre part, le segment de O à B est faiblement peuplé.

• 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)

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

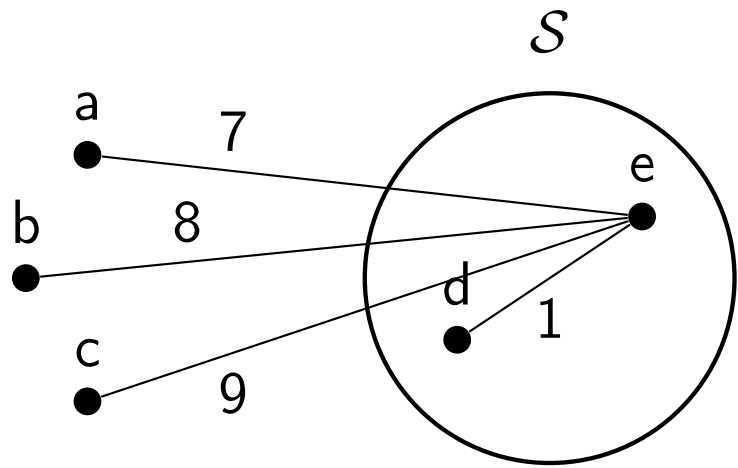
• Calculer toutes les distances par paires entre les points dans S et les points dans D : O(sn).

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

• Le reme outlier score dans l’échantillon est déterminé.

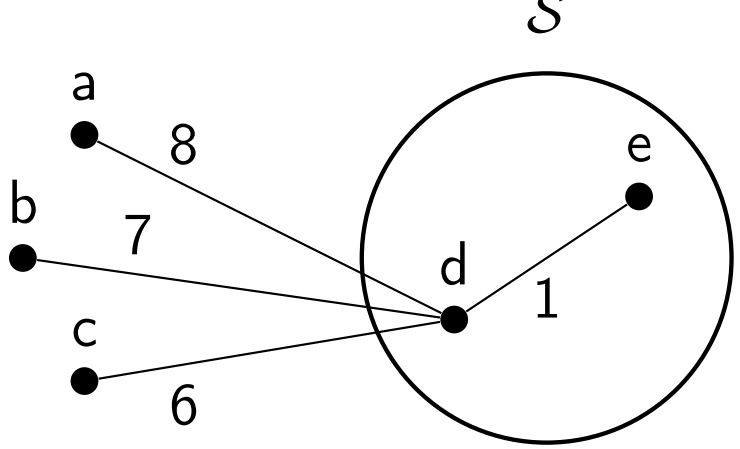
• Il est un lower bound (L) pour le reme outlier score dans l’ensemble total D.• Pour chaque point dans D − S, on connaist seulement un upper bound par rapport à son keme plus proche voisin.

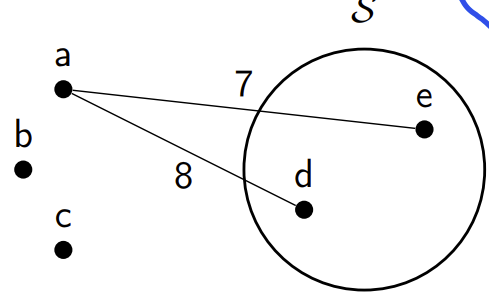
• Si ce upper bound est plus petit que L, alors le point de D − S peut-être exclu comme possible top-r outliers.

Une image contenant table

Description générée automatiquement• 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) = 8.

• 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) = 8 > 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.**Cours 10 – Analyse de réseaux sociaux**

• Réseau social peut être structuré comme un graphe G = (N, A) (N : sommets et A : 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), et ni = |Si| (nb voisins de i)

• Arêtes possibles entre sommets de Si :

• Le coefficient de regroupement du réseau est la valeur moyenne de ηi sur tous les sommets du réseau, avec num = nb liens entre les voisins de i et dénom nb chemins possible entre les voisins ( n! / (n-k)! k! )

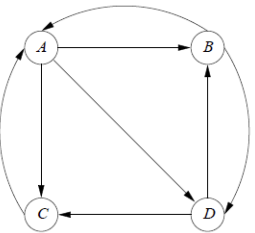
Attachement préférentiel Dans un réseau en croissance, la probabilité qu’un sommet reçoive des nouveaux liens augmente avec son dégrée. Composant connecté géant les nouveaux liens sont plus susceptibles de se joindre aux sommets densément connectés et de haut dégrée dans le réseau.**Mesures de centralité 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 CD(i) 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.• 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

• Centralité de proximité:

• Pour le prestige de proximité, il faut considérer que quelques chemins j i sont inexistants.

• Donc, le premier pas consiste à calculer l’ensemble Influence(i) composé par les sommets qui peuvent atteindre le sommet i • Dist(j, i) est calculée du sommet j au sommet i. Utiliser l’inverse de la distance moyenne ne serait plus juste, les sommets avec moins d’influence doivent être pénalisés.

• 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:

**Centralité d’intermédiarité** : 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. Faire une matrice F avec diagonale et en dessous non remplie et ligne/colonne i vide

• Soit qjk le nombre de plus courts chemins entre les sommets j et k.

• Soit qjk(i) le nombre de ces chemins qui passent par le sommet i.

• Donc, la fraction des chemins fjk(i) qui passe par le sommet i est donné par fjk(i) = qjk(i)/qjk

• fjk(i) : niveau de contrôle que le sommet i a sur les sommets j et k en termes de régulation du flot d’informations

Une image contenant texte

Description générée automatiquement• La centralité d’intermédiarité CI(i) est la valeur moyenne de cette fraction pour toutes les paires de sommets (en excluant le sommet i) :

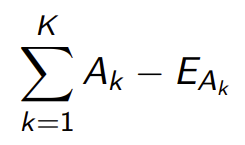
• Contrairement à la centralité de proximité, la centralité d’intermédiarité peut également être définie pour les réseaux déconnectés. Peut être généralisé pour les arêtes aussi.

**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

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.

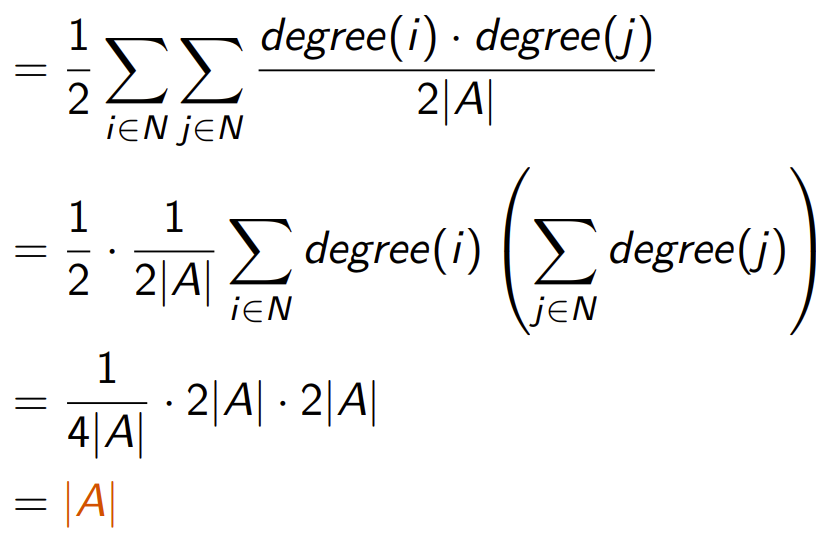
**Modularité** : Critère de clustering pour les réseaux. Étant donné une partition du réseau en K clusters, la modularité Q est directement proportionnelle à :

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. • Nb attendu d’arêtes entre les sommets i et j est égal à :

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



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

Une image contenant texte, horloge, montre

Description générée automatiquementoù 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.

**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 .**Étape 1** : Graphe non-dirigé -> graphe dirigé**. Étape 2** : Remplacer arcs sortants des sommets étiquetés par self-loops

• 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.

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

πk = M · πk−1 processus markovien

à partir de π0 avec (π0)i = 1 dans le cas où le sommet de départ est i, 0 sinon

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

**Cours 11 – 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.

**Maximum commun sous-graphe (MCS)**entre paire de graphes G1 = (N1, E1) et G2 = (N2, E2) est 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

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

Mesures de similarité : , -> NP-difficile

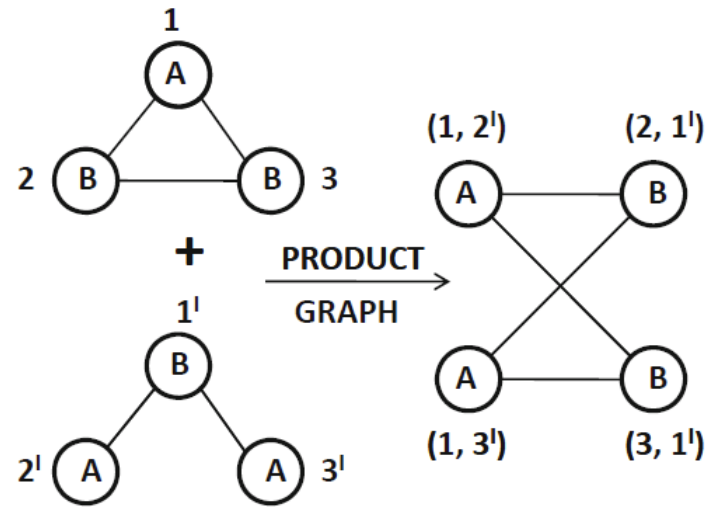
**Distance d’édition** Edit(G1, G2) est égal au coût minimum des opérations d’édition à appliquer au graphe G1 pour le transformer en G2 (node/edge deletion, label substitution, edge insertion) -> NP-difficile

**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)

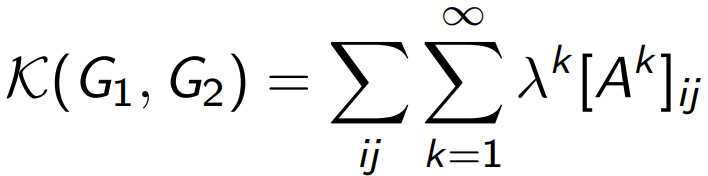
• La fonction φ(·) n’est pas définie directement. Plusieurs façons de définir une similarité de noyau pour les graphes

**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) : VX = {(v1, v2) : v1 ∈ G1 ∧ v2 ∈ G2 ∧ label(v1) = label(v2)}

• 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}

**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 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 .

**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 :

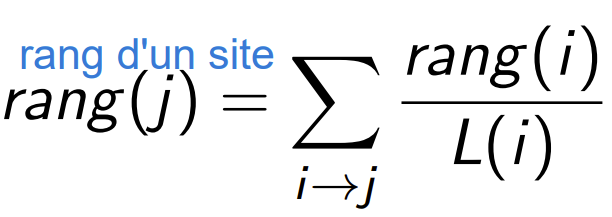
• Morgan Index égal à un vecteur de taille |G| où chaque composante i est égale au nombre de sommets accessibles depuis le sommet i à une distance d’au plus t.• Wiener Index égal à la somme des distances les plus courtes entre toutes les paires de sommets du graphe.

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

**Cours 12 – Fouille de réseaux sociaux** Un seul grand graphe

**PageRank** : 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.

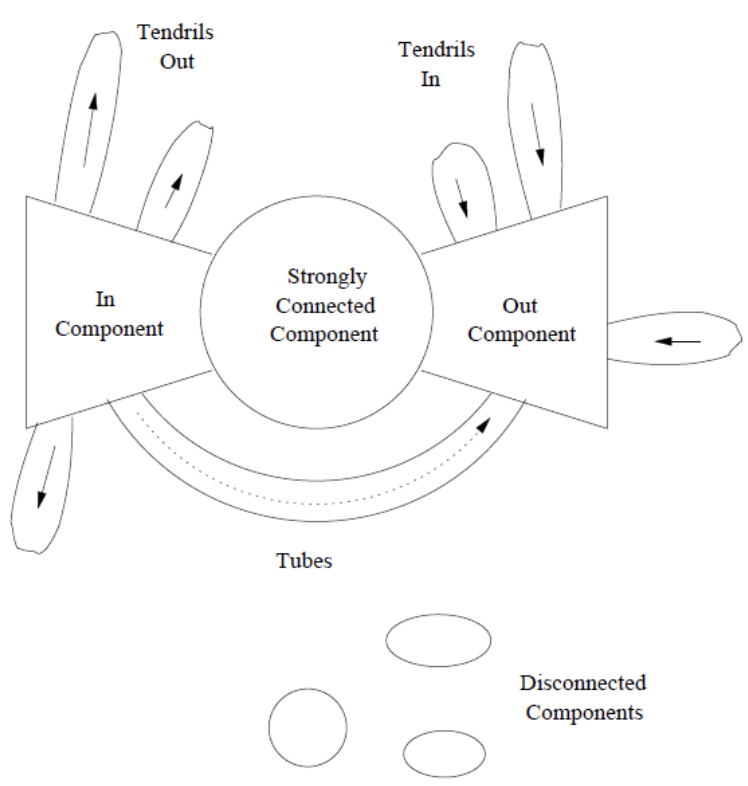
• Mij est la probabilité de que la prochaine page visitée après la page j soit la page i.

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

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

• Le PageRank à la k-eme itération est alors donné par PRk = M · PRk−1

• 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, il saute à une page quelconque du web. On utilise souvent β ∈ [0.8, 0.9]

**Nouveau PageRank** : PRk = βM · PRk−1 + (1 − β)e/n, où e est le vecteur unitaire

Une image contenant texte

Description générée automatiquement

L(i) est le nombre d’arces sortants de i

**Cours 13 – 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

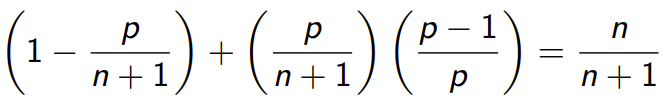
2 Échantillonner une proportion fixe de données dans le flot (disons 1 sur 10)

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)

• 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.

• 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 :

• Conclusion : À l’instant n, les données sont dans P 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

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.

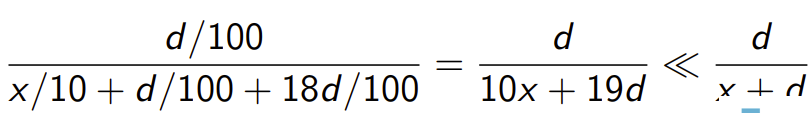
Problème avec approche simple : L’échantillon va contenir x/10 des requêtes uniques.

• Seules d/100 des requêtes doublées seront aussi doublées dans

l’échantillon :

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• 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 (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 et fléchette : code hash h(s) des différentes clés s ∈ S

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

• Quelle est la probabilité qu’une cible reçoive au moins une fléchette ?

• Quelle est la probabilité qu’un bit de B soit 1 ?

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

• Probabilité qu’aucun élément de S marquent un bucket en particulier

• Donc, la probabilité qu’un bucket soit marqué vaut 1 − e−|S|/|B|

• Reprenons notre 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

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)

• 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, donc la fraction du vecteur binaire de bits à 1 est de (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/mln(2)

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

**Comptage de bits** : • Un modèle utile de traitement de flots est celui dont les requêtes sont faites sur une fenêtre de longueur N (les N plus récentes données).• Cas intéressants : N est si grand que les données ne peuvent pas être stockées en mémoire ou sur disque, ou Il y a tellement de flots que leurs fenêtres ne peuvent pas être toutes stockées.• Applications : vente items Amazon (flot :item, bit : transaction), fréquence hastags Twitter (flot : hastag, bit : tweet)

• Étant donné un flot de 0 et 1s… Combien de 1 dans les derniers k bits ? (où k ≤ N) Combien de fois vendu X dans les k dernières ventes ? • Solution évidente (impossible pour k ≈ N) :

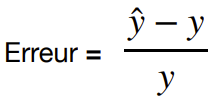
• Stocker les k bits les plus récents - pas possible • Quand un nouveau bit arrive, ignorez le k + 1er bit

Un essai : solution simple. • Q : Combien y a-t-il de 1 dans les derniers k bits ?

• Une solution simple qui ne résout pas vraiment notre problème : hypothèse d’uniformité.

• Maintenir deux compteurs : S : nombre de 1s depuis le début du flot Z : nombre de 0s depuis le début du flot

• Combien y a-t-il de 1s dans les derniers k bits ? Problème si le flot n’est pas uniforme…

• On peut définir plusieurs sous-fenêtres glissantes sur le flot Tailles variables : plus grandes dans le passé, plus petites dans le présent (puissances de 2). Fusionnées au long du temps (max 2 sous-fenêtres de même taille, ne peuvent pas se chevaucher). On garde pour chaque sous-fenêtre le nombre de 1 ainsi que sa taille. ŷ = nb estimé de bits, y= nb réel de bits.

• La pire estimation est illimitée, si la vraie valeur est 0 et qu’on a pas 0.

Méthode DGIM : garantit 3 propriétés intéressantes :

1 Stockage de l’ordre de O(log2 N) bits pour une fenêtre de taille N.

2 Temps de l’ordre de O(log N) pour traiter un nouveau bit.

3 Erreur relative inférieure ou égale à 50%.

Les N derniers bits sont divisés en buckets (sous-fenêtres). Chaque bucket contient :

• Le timestamp le plus récent modulo N (marque le début du bucket, la fin est implicite).

• Le nombre de 1 dans le bucket (appelé taille du bucket).6 contraintes pour DGIM : 1 Le bit à droite (le plus récent) d’un bucket doit être un 1.

2 Tous les 1 doivent être dans un bucket. 3 Chaque bit est au plus dans un bucket.

4 Il y a au plus deux buckets de la même taille. 5 Toutes les tailles sont des puissances de 2.

6 La taille des buckets augmente vers la gauche (vers le passé)

• À chaque fois qu’on a plus que deux sous-fenêtres de même taille, on les fusionne.• On ne connaît pas la distribution des 1 dans le bucket partielle. Si une requête inclut partiellement un bucket, l’estimation DGIM = somme 1 bucket entiers + moitié de 1 du bucket partielle (si seulement 1 bit, on met 1)

• DGIM garantit un seuil maximum (upper bound) d’erreur sur l’estimation de 50% !

Preuve : Supposons que l’estimation nécessite partiellement un bucket

b de taille 2j. Notons la valeur réelle y et l’estimation ŷ. 2 cas : estimation supérieure ou inférieure à la valeur réelle

Une image contenant texte, antenne

Description générée automatiquement1) Estimation **supérieure** valeur réelle : Dans le pire des cas, seul le bit le plus à droite du bucket b est considéré et il n’y a qu’un bucket de chaque taille plus petite que celle de b

2) Estimation **inférieure** valeur réelle : Pire cas, tous les 1 de b sont dans la partie du bucket considéré

• L’estimation du bucket partiel est 1/2 × 2j = 2j−1, alors que la valeur réelle du bucket est 2j • Puisqu’il y a au moins un bucket de chaque taille, la valeur réelle est supérieure à 20 + 21 + · · · + 2j−1 + 2j = 2j+1 − 1• Erreur relative donc inférieure à 50%

**Exercices**

**Une image contenant texte

Description générée automatiquement1. Pourquoi la régression logistique est un classificateur linéaire?**

Séparateur pour un classificateur logistique est donné par (image)

C’est un séparateur linéaire en fonction de X

**2. En quoi consiste un kernel trick et quel est son principal avantage?**

Le kernel trick permet aux Support Vector Machines (SVM) de séparer les données non linéairement séparables en les projetant dans un espace R de dimension plus élevée. L’avantage est que l’on n’a pas besoin de connaître la projection explicitement, mais seulement la similarité entre deux points dans R. Les similarités entre les données sont modélisées à travers des fonctions noyaux (kernel). Cela permet entre autres d’avoir des projections dans des espaces infinis (ex. noyau qui fait l’opération min(x1, x2)).

**3. Soit un jeu de données linéairement séparables. Donnez deux avantages et un inconvénient des machines à support de vecteurs (SVM) linéaires simples par rapport à la régression logistique. Justifiez vos choix en une ligne.**

Avantages: — Les SVMs sont plus robustes, car ils trouvent le séparateur à marge maximale. — Les SVMs ont une variance plus faible, car retirer les individus qui ne sont pas vecteurs de support ne change pas la solution. — Les SVMs sont plus résistent aux données aberrantes que la régression logistique, car les points qui ne sont pas vecteurs de support ne contribuent pas à la solution. Inconvénients : — La classification des SVMs est hard (0 ou 1), alors que la classification de la régression logistique est soft ([0, 1]). — Il n’y a pas de généralisation multiclasse naturelle pour les SVMs contrairement à la régression logistique qui utilise une fonction Softmax. — Ne peut pas être appliqué sur des jeux de données massifs, car la complexité d’apprentissage est O(N2).— La solution trouvée par les SVMs n’est pas aussi simplement interprétable que les coefficients de la régression logistique Attention : — La régression logistique peut séparer parfaitement les deux classes, même en grande dimension. En effet, les données sont linéairement séparables et l’entropie croisée est une fonction convexe, le meilleur séparateur peut donc être trouvé avec la méthode de descente du gradient (minimum global). — Le même ordre de quantité de mémoire O(d) est nécessaire pour les SVMs (vecteurs de support) et pour la régression logistique (coefficients)

**4. Expliquez pourquoi il y a un compromis entre la précision (precision) et le rappel (recall). Comment peut-on augmenter la précision ?** Le compromis vient de la définition du rappel et de la précision. Considérons un modèle qui fait F =FN + FP prédictions incorrectes. La précision et le rappel sont égaux lorsque FP = FN. Lorsque FP augmente (c.-à-d., la précision diminue car le dénominateur augmente) alors F N diminue (c.-à-d., le rappel augmente car le dénominateur diminue). Dans le cas extrême où le rappel vaut 1 (c.-à-d. le modèle prédit uniquement positif), alors la précision est au plus faible. Augmenter la précision : — Augmenter le seuil de détection (threshold) pour que seuls les individus avec une grande valeur de sortie soient classifiés comme appartenant à la classe positive, et ainsi diminuer le nombre de F P. — Les faux positifs sont des exemples négatifs classifiés comme appartenant à la classe positive, il faut donc mettre un poids supérieur à 1 sur la classe négative pour réduire les FP.— Ajouter des individus appartenant à la classe négative ou retirer des individus de la classe positive

**5. 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.

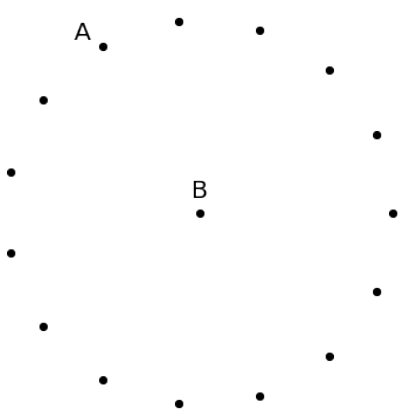
6. **Comment utiliser les k-plus proches voisins pour détecter les données aberrantes ? Quels sont les avantages**

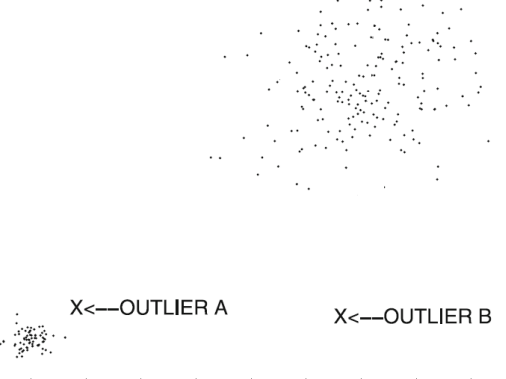
**et les inconvénients de cette approche ?** L’outlier score d’un enregistrement est donné par sa distance avec son k-plus proche voisin. Des variantes considèrent la moyenne des k plus proches voisins. Un enregistrement est considéré outlier si son score est supérieur à un seuil, ou si son score est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

**7. 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).

**8. 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).

**9. 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.

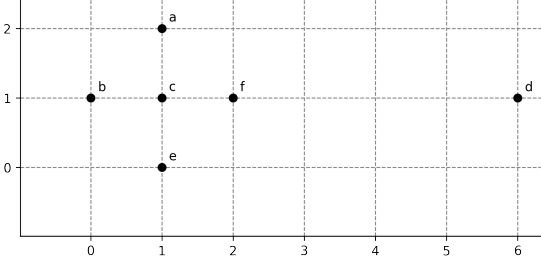
**b) Est-ce que l’outlier indiqué en (a) peut être identifié par son outlier score obtenu en utilisant un modèle de k plus proches voisins basé sur des distances euclidiennes ?** Justifiez. Cette méthode n’est pas appropriée pour détecter l’outlier A parce que les deux clusters présentent de densités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

**10. 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.**11. 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.

**12. 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.

Une image contenant table

Description générée automatiquementLe 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).

On transforme en

car E peut retourner à tous les nœuds ainsi que sur lui-même pour éviter les dead-ends

Ou encore et

Donc, et