Exercice de devoirs n° Z (exercice récapitulatif)

• Cet exercice doit être soumis en ligne dans un fichier ZIP au format ZIP2.numéro_1numéro_assZ où 1numéro et 2number sont les numéros de carte d'identité des étudiants soumettant (par exemple ZIP098765432._012345678_assZ). Le dossier doit contenir : o Un ou plusieurs paquets de code Python 3 comme requis avec toutes les dépendances qui permettent aux méthodes des paquets de s'exécuter (faites attention aux exigences spécifiques car le code doit s'exécuter dans un script de test pré-préparé). o De plus, un document d'accompagnement dactylographié (et non manuscrit et scanné) au format PDF (vous pouvez préparer un document DOC ou DOCX dans Word puis l'enregistrer au format PDF). Le document contiendra des réponses aux questions théoriques ici dans les exigences, comme une explication brève et claire des choix effectués lors de l'écriture du code, des fichiers importants, des classes et des méthodes, etc.

• Les questions de programmation font référence à des données spécifiques. S'il n'existe pas de référence directe à partir de l'ouvrage, ces données sont disponibles sur le site du cours sous forme de fichier ou de référence (télécharger à nouveau les données). Les fichiers ont été mis à jour avant la publication de l'exercice (. • Les réponses de programmation doivent être flexibles afin qu'elles fonctionnent sur toutes les données de votre format de données.

• Le nom et l'identifiant de l'expéditeur doivent être écrits en haut de chacun des fichiers de la solution de l'exercice (le fichier PDF et chacun des fichiers . Le code est sous forme de commentaire.

• Essayez d'écrire du code aussi efficacement que possible dans les limites des contraintes. • Écrivez des noms significatifs pour les classes, les fonctions et les variables. • Il est permis d'ajouter des fichiers py supplémentaires aux fichiers fournis. Ils doivent également être soumis dans un fichier zip. Les fichiers supplémentaires doivent se trouver dans le même répertoire que les fichiers -py attachés à la question.

• Vous devez écrire du code aussi efficacement que possible dans les limites des contraintes. Les essais seront limités dans le temps. • Écrivez des noms significatifs pour les classes, les fonctions et les variables. • S'il y a des notes de documentation dans le code : en anglais uniquement (l'hébreu peut être utilisé dans le document d'accompagnement) et elles doivent être surlignées Points importants ou compliqués dans le code.

• La partition est divisée entre théorique et pratique. Le code et les résultats des exécutions sont considérés comme la partie pratique. Les explications dans le document d'accompagnement de ce que vous avez fait dans la partie pratique comptent pour la note attribuée à la partie théorique.

• Il est permis d'utiliser Internet pour comprendre comment fonctionne tel ou tel algorithme ou comment utiliser Python de manière créative pour faciliter l'exercice. Vous n'êtes pas autorisé à télécharger des packages ou à copier du code qui résout l'exercice à votre place. • La consultation de l'IA est autorisée. Vous ne devez pas copier/coller du code qui vous est fourni par l'IA comme solution complète (ou presque complète) à un problème. La question.

• Ne laissez pas l'IA rédiger le document d'accompagnement à votre place. Si le document d'accompagnement contient des murs de texte d'IA inexpliqué Votre code ne répondant pas aux exigences sera considéré comme une malhonnêteté académique.

• Si vous n'avez pas d'autre choix que d'utiliser une petite partie de code copiée à partir d'une source externe (d'Internet, pas d'autres étudiants), cela doit être explicitement mentionné dans une note dans le code ainsi que dans le document d'accompagnement (il faut préciser clairement quelle est la source, quelle est la taille de la section copiée et si des modifications ont été apportées au code copié). Les étudiants qui mentionnent de tels emprunts de code peuvent voir leur note dans l'exercice diminuée, mais ne seront pas considérés comme ayant agi avec malhonnêteté académique.

• La solution doit être soumise avant la date limite indiquée dans le module (c'est-à-dire sur le site Web du cours) sans faute. Tout retard non approuvé peut entraîner Pour disqualifier l'exercice.

• Quel que soit le nombre de points que vous avez accumulés. Vous ne pouvez pas obtenir plus de 100 dans un exercice.

Sujets d'exercice : Regroupement

Date limite de soumission : 03/06/2025

Exercice final : Algorithmes de clustering sur des données de différents types et tailles. Dans cette tâche, vous devez implémenter des outils de plusieurs types

1. Outils de production de données multidimensionnelles constituées de plusieurs clusters.

2. Un outil de clustering hiérarchique de données relativement petites. 3.

Un outil Means-K pour le clustering de données dans un espace euclidien qui tient dans la mémoire principale. 4. Un outil

BFR pour le clustering de données euclidiennes qui se trouvent sur le disque dur et qui sont composées de clusters distribués

Normal à chaque axe

5. Outil CURE pour le clustering de données euclidiennes qui résident sur le disque dur et dont la distribution de cluster est inconnue

Quelques points importants à propos du projet • Lors

de la mise en œuvre, vous devrez prendre toutes sortes de décisions. Dans le document d'accompagnement qui accompagne votre soumission, vous devrez expliquer les

Ces décisions et pourquoi votre code fait ce que vous voulez. Les dimensions

seront représentées (au moins extérieurement) sous forme de tuples de taille dim • Dans cette tâche, les points dans -

par exemple (2.2,1.3,-4.0) est un point dans l'espace euclidien à 3 dimensions.

• Lorsqu'un algorithme de clustering qui n'enregistre pas les résultats dans un fichier termine son exécution, il enregistre les points en mémoire.

Comme une liste de listes. Par exemple:

[[(3, 4, 7), (4, 5, 6), (3, 4, 6), (5, 6, 7)], [(2, 4, 2), (2, 3, 3), (0, 2, 3), (0, 1, 4), (1, 2, 4), (3, 4, 4)]]

• Les listes de points et de clusters seront enregistrées dans des fichiers au format CSV, qui sont des fichiers texte dans lesquels les champs sont séparés.

Entre virgules. Voir la définition ici. Chaque point occupera une ligne. Si le numéro de cluster est également enregistré dans le fichier, il sera enregistré

comme dernière valeur de chaque ligne.

Autrement dit, sans le numéro de cluster, les lignes ressembleront à ceci :

3,4,5

2,4,2

…

Et avec le numéro de cluster, les lignes ressembleront à ceci :

3,4,5,0

2,4,2,1

…

Partie 1 : (30 points) Environnement de clustering et algorithmes d'utilisation de la mémoire principale.

Dans cette section, vous devez implémenter : 1. Une fonction qui génère des données et les enregistre dans un fichier CSV. 2. Fonctions de clustering qui fonctionnent dans la mémoire principale. 3. Fonctions qui acceptent un fichier CSV avec des données. Un algorithme de clustering qui fonctionne dans la mémoire principale et enregistre les données après l'opération de clustering dans un nouveau fichier CSV.

1) Écrire la fonction

def generate_data(dim, k, n, out_path, points_gen=None, extras={})

La fonction data_generate créera un fichier CSV dans le chemin trouvé dans la variable path_out qui contiendra n points dans des dimensions dim divisées en k clusters. L'objectif n'est pas simplement d'étiqueter k points avec des étiquettes différentes, mais de créer des données de manière à ce qu'elles soient constituées d'un certain nombre de clusters que l'algorithme de clustering peut identifier. La fonction peut accepter une fonction d'assistance gen_point qui créera les points (ou certains d'entre eux, selon votre choix). Vous pouvez décider quels paramètres envoyer à gen_points et ce qu'elle renvoie. Si vous souhaitez définir des arguments supplémentaires pour la fonction, vous pouvez les ajouter au dictionnaire des extras. Expliquez dans le document d'accompagnement comment votre code génère avec succès des données groupées. Expliquez également quels sont les paramètres supplémentaires, le cas échéant. Si vous avez utilisé gen_point, expliquez ce qu'i Et comment ça marche.

Écrire la fonction

def load_points(dans_chemin, dim, n=-1, points=[])

La fonction points_load lit à partir d'un fichier CSV dans le chemin trouvé dans la variable path_in n points dans les dimensions dim. Chaque point sera lu à partir d'une ligne du fichier. S'il y a plus d'une dimension de données dans la ligne points_load, convertissez la première dimension en point et ignorez le reste. S'il y a plus de n lignes dans le fichier points_load, il lira les n premières lignes. Si -n, points_load lira toutes les lignes du fichier. La fonction insérera les points extraits de path_in dans la liste points, qui a la valeur par défaut -1.

2) Écrire la fonction

définition h_clustering(dim, k, points, dist, clusters=[])

La fonction clustering_h effectuera un clustering hiérarchique de haut en bas des points dans les dimensions -dim. Les points sont dans la variable points dans les dimensions -dim. k est le nombre de clusters souhaité. Si None=k, vous devez décider vous-même quand arrêter. Utilisez l'approche apprise en classe pour choisir quand arrêter dans ce cas. dist est la fonction de distance, si None=dist suppose un espace euclidien (vous devrez écrire une fonction qui calcule la distance euclidienne). clusts est une variable de sortie dans laquelle les clusters seront saisis sous forme de liste de listes de points. Expliquez dans votre document d'accompagnement les décisions que vous avez prises et comment cela fonctionne.

3) Écrire la fonction

def k_means(dim, k, n, points, clusters=[])

La fonction means_k effectuera un clustering means-k de points dans des dimensions dim. Les points sont dans la variable points. k est le nombre de clusters souhaité. Si None=k, vous devez décider quelle est la valeur k correcte. Utilisez l'approche apprise en classe pour choisir quand trouver k dans ce cas (cela nécessitera plusieurs lectures avec différentes valeurs k). clusts est une variable de sortie dans laquelle les clusters seront saisis sous forme de liste de listes de points. Expliquez dans votre document d'accompagnement les décisions que vous avez prises et comment cela fonctionne.

4) Écrire la fonction

def save_points(clusters, chemin de sortie, chemin de sortie balisé)

La fonction points_save créera 2 fichiers CSV à partir d'une liste de clusters dans la variable clusters. Le fichier dans le chemin trouvé dans la variable path_out contiendra tous les points des clusters dans un ordre aléatoire. Le fichier dans tagged_path_out contiendra les points dans l'ordre, chaque ligne contenant un champ supplémentaire contenant le numéro de cluster (entre 0 et 1-k, où k est le nombre de clusters).

Partie 2 : (30 points) Algorithmes de clustering d'un grand nombre de points.

1) Trouvez un moyen de générer une grande quantité de points (trop grande pour être conservée en mémoire) et enregistrez-la dans un fichier CSV. Vous pouvez utiliser la fonction data_generate de la partie 1. Expliquez comment votre code (et quelle fonction) exécute la tâche.

2) Écrire la fonction

bfr_cluster(dim, k, n, taille_bloc, chemin_entrée, chemin_sortie)

La fonction cluster_bfr effectuera le clustering BFR. Les points sont dans un fichier CSV situé dans le chemin dans la variable path_in, qui contiendra au moins n points dans au moins des dimensions dim. Chaque ligne représente un point. Si une ligne comporte plus de champs dim, les premiers champs dim de la ligne doivent être considérés comme les dimensions dim du point. k est le nombre de clusters souhaité. Si None=k, vous devez décider quelle est la valeur k correcte. Utilisez l'approche apprise en classe pour choisir quand trouver k dans ce cas (cela nécessitera plusieurs lectures avec différentes valeurs k). size_block est le nombre de points chargés à chaque étape BFR du fichier vers la mémoire principale. n est le nombre de points sur lesquels effectuer le cluster. Sélectionnez les n premières lignes dans path_in. path_out est le fichier CSV dans lequel les points seront enregistrés avec le cluster auquel ils appartiennent. Expliquez dans votre document d'accompagnement les décisions que vous avez prises et comment cela fonctionne.

)3

cure_cluster(dim, k, n, taille_bloc, chemin_entrée, chemin_sortie)

La fonction cluster_cure effectuera le clustering CURE. Les points sont dans un fichier CSV situé dans le chemin dans la variable path_in, qui contiendra au moins n points dans au moins des dimensions dim. Chaque ligne représente un point. Si une ligne comporte plus de champs dim, les premiers champs dim de la ligne doivent être considérés comme les dimensions dim du point. k est le nombre de clusters souhaité. Si None=k, vous devez décider quelle est la valeur k correcte. Utilisez l'approche apprise en classe pour choisir quand trouver k dans ce cas (cela nécessitera plusieurs lectures avec différentes valeurs k). size_block est le nombre de points chargés dans la mémoire principale à chaque étape CURE à partir du fichier (c'est également une limite supérieure du nombre de points chargés en mémoire pendant l'étape d'échantillonnage CURE). n est le nombre de points sur lesquels le clustering est effectué. Sélectionnez les n premières lignes dans path_in. path_out est un fichier CSV dans lequel les points seront enregistrés avec le cluster auquel ils appartiennent. Expliquez dans votre document d'accompagnement les décisions que vous avez prises et comment cela for

Partie 3 : (40 points) Tests et comparaisons

Dans cette section, vous devez exécuter votre code et enregistrer vos résultats dans une feuille de calcul.

Tout d'abord, créez les fichiers

• Créez un petit fichier de données avec 2 dimensions divisées en au moins 5 clusters (environ 1000 lignes au total sont recommandées).

• Créez un fichier de données suffisamment petit pour fonctionner dans la mémoire principale avec des dimensions de 1dim divisées en clusters de . Et aussi 10 < 1 + 1 1k. Les paramètres doivent satisfaire 4 < 3, < 1 divers. , • Créez au moins 2 autres fichiers de ce type avec des tailles différentes. • Vous êtes autorisé et même recommandé de télécharger des données existantes supplémentaires pour tester les algorithmes (du code peut être nécessaire) (Veuillez noter le format.) •

Ne soumettez pas les fichiers de données (sauf le petit). Vous pouvez soumettre des liens vers le lecteur.

Expliquez comment vous avez créé les fichiers et assuré qu'ils étaient divisés en clusters. Expliquez également comment vous avez fait en sorte que la division ne soit pas trop triviale. La soumission doit contenir des scripts Python qui créent les fichiers et des instructions pour les créer.

Exécutez maintenant le clustering hiérarchique et l'algorithme k-means :

Vous devez écrire votre mesure (ou vos mesures) de la qualité du clustering. Ce qui vous fait évaluer le clustering comme meilleur ou pire et ce qui fait que les algorithmes de clustering choisissent si vous ne leur fournissez pas les données à l'avance. Certains de ces éléments auraient déjà dû être écrits dans la partie 1. Il n'est pas nécessaire de tout répéter en détail, mais

Il est important que cela soit écrit quelque part.

Sur le premier fichier, exécutez les algorithmes Tami clustering_h et -means-k.

Montrez comment l'indice correspondant change lorsque les changements :

Voici un exemple d'un tel tableau (il doit refléter exactement ce qui est mesuré pour différentes valeurs) :

Nom du	Algorithme	k=2	k=3	K=4	K=5	K=6	K=7	K=8
fichier	Hiérarchique							
f1.csv f1.csv	k-moyenne							

Ajoutez également un tel tableau pour l'une des sections de fichier d'entrée les plus volumineuses. Pour chacun de vos fichiers, indiquez si vos algorithmes choisissent le bon k.

Vérification de

l'exactitude : doit être connue de vous et enregistrée dans un fichier séparé. Maintenant que vous disposez des fichiers de clustering « corrects » que vous avez créés, vous pouvez vérifier dans quelle mesure vos algorithmes classent les points de vos données dans les clusters « corrects ». Notez que si vous avez créé des clusters proches et pas très faciles à déchiffrer, il est tout à fait naturel que les algorithmes n'atteignent pas toujours une précision de 100 %, et cela ne pose aucun problème. Le tableau pourrait ressembler à ceci :

Nom du	Algorithme	k0	k1	k2	k3	k4	k5
fichier f2.csv	Hiérarchique	99,11 % 97,22 % 97,79 % 99,21			…	…	…
f2.csv	k-moyenne	% 97,12 % 98,98 %			…	…	…

Nom de fichier	Algorithme	k0	k1	k2	k3	k4	k5	k6	k7	k8
f3.csv	hiérarchique 75,22% …			…	…	…	…	…	…	…
f3.csv	k-moyenne	85,32% …		…	…	…	…	…	…	…

La signification des chiffres dans les tableaux ci-dessus est le pourcentage de membres qui ont été assignés au cluster et sont donc censés en faire partie. Notez que votre algorithme de clustering ne choisit pas nécessairement le même ordre de clusters que dans le fichier d'origine (c'est-à-dire que le 1 d'origine peut maintenant être étiqueté comme 3). Une telle inattention peut vous amener à penser à tort que votre algorithme a omis presque entièrement des clusters entiers alors qu'il leur a simplement donné un nom différent. Soyez donc prudent lorsque vous comparez.

Lorsque vous créez des tables, assurez-vous d'avoir un fichier avec un clustering correctement étiqueté lors de la création des données (c'est-à-dire sans algorithme de clustering) et un fichier avec les réponses données par l'algorithme de clustering dont vous souhaitez évaluer la précision. Exécutez maintenant le code qui parcourra les 2 fichiers et comparera les emplacements des points en fonction des deux divisions en .clusters.

Remarque : Étant donné que vous n'êtes pas obligé de soumettre le code qui compare vos résultats au clustering, mais uniquement de signaler les résultats, vous pouvez certainement utiliser l'IA ou des sources externes sans restriction pour créer ce code « correct ».

Tests pour la partie 2 (fichiers volumineux) :

Tout d'abord, créez des fichiers •

Créez 2 fichiers de données d'au moins 10 Go chacun (en fait 4 fichiers si vous enregistrez également ceux marqués)

par cluster). Sauvegardez-les sur votre disque dur. Vous travaillerez dessus.

• Dossier numéro 1 : adapté pour fonctionner avec BFR. • Dossier

numéro 2 : adapté pour fonctionner avec Cure. • Les deux fichiers

auront des dimensions faibles et seront divisés en k clusters. Les paramètres doivent satisfaire 5 < 5, < . et les instructions d'exécution • Ne soumettez pas les

fichiers mais soumettez les scripts Python qui créent les fichiers,

Les scripts.

• Il est très important de créer vos propres fichiers et de ne pas les copier 1 à 1 comme les autres étudiants créent leurs fichiers.

o Les données BFR sont toujours assez similaires mais avec des écarts types différents et provenant de centres différents. Jouer avec dim et -k comme

Même avec la méthode de sélection des écarts types et des centres, il y aura une différence entre les fichiers.

o Vous êtes autorisé à consulter l'IA pour réfléchir à des idées de création de clusters pour l'algorithme Cure. Rappelle-toi toujours

Assurez-vous que les suggestions ont du sens.

Test de précision :

Créez des tableaux comme vous l'avez fait pour les tests de précision de la partie 1 et évaluez le fonctionnement des deux algorithmes sur les deux fichiers (si vous avez créé les fichiers correctement, BFR devrait être moins performant sur les données créées pour Cure).