Université de Sherbrooke

IFT 599 / IFT 799 - Science de données

TP2 : Clustering et détection d'anomalies

Automne 2025

Enseignant

	Courriel	Local	Téléphone
Jordan F. Masakuna	jordan.felicien.masakuna@usherbrooke.ca	D6-0013	+1 873 662-8960

Faculté des Sciences,
Département d'informatique

04 novembre 2025

Sommaire

Dans le cadre de ce travail pratique (TP), deux jeux de données sont mis à la disposition des étudiants et étudiantes. L'objectif est d'exploiter ces jeux de données pour mettre en valeur les concepts de clustering et la détection d'anomalies sur les données complexes de grande dimension abordés dans les Thèmes 4-5 du cours.

Contents

1 Objectifs pédagogiques et compétences développées

L'objectif de ce projet est de conduire une analyse expérimentale et critique sur le clustering et la détection d'anomalie dans le cadre de malédiction de dimensionalité. Il s'agit notamment d'examiner les effets de la malédiction de la dimension sur la détection de structures non convexes, ainsi que d'explorer les capacités des auto-encodeurs et de leurs variantes pour la détection d'anomalies. Ce projet poursuit les objectifs suivants :

- Comprendre la malédiction de la dimensionnalité : expliquer théoriquement et démontrer empiriquement l'impact des dimensions non pertinentes sur le clustering et la détection d'anomalies.
- Maîtriser la détection d'anomalies : mettre en œuvre quelques architectures d'auto-encodeur pour la détection d'anomalies.
- Évaluer et interpréter les résultats : utiliser des métriques d'évaluation et visualiser les résultats en 2D pour interpréter les structures des données.

2 Énoncé

Ce projet s'articule autour de deux thématiques principales. La première concerne le clustering de données à haute dimension. Elle s'appuie sur un premier jeu de données (voir Section 4) permettant d'explorer différentes méthodes de clustering. La seconde thématique porte sur l'identification de points atypiques, en utilisant des techniques non supervisées de détection d'anomalies. Cette partie repose sur un deuxième jeu de données spécifiquement choisi pour mettre en évidence les comportements inhabituels.

Dans le cadre de la segmentation, chaque groupe de travail devra appliquer et comparer les performances de quelques algorithmes de clustering : K-Means, DBSCAN et clustering spectral. Pour chaque méthode de clustering classique, deux approches doivent être mises en œuvre : la première utilise l'ensemble des attributs disponibles, tandis que la seconde applique une réduction du nombre d'attributs à 100 (UMAP et ACP) avant de procéder à la segmentation. L'objectif de ce projet est de comparer les approches classiques de segmentation à une méthode avancée, dans le cadre de l'analyse de données à haute dimension. L'hypothèse sous-jacente est que, face à la malédiction de la dimensionnalité (avec structures non convexes), les approches avancées—mieux adaptées à la complexité et à la structure des données—devraient surpasser les méthodes classiques en termes de qualité de segmentation et de pertinence des clusters obtenus.

Pour la détection d'anomalies, chaque groupe d'étudiant.e.s devrait confronter les résultats obtenus par la méthode Isolation Forest et par un auto-encodeur classique à ceux d'une variante d'auto-encodeur choisie parmi les options suivantes :

- **Denoising autoencoder** : apprend à reconstruire les données originales à partir d'entrées bruitées, ce qui le rend robuste aux perturbations et utile pour la prétraitement ou la détection d'anomalies.
- **Sparse autoencoder** : impose une contrainte de sparsité sur les activations cachées, favorisant des représentations compactes et interprétables.

- **Variational autoencoder (VAE)** : encode une distribution plutôt qu'un point unique, ce qui permet la génération de nouvelles données et l'estimation de l'incertitude.
- **Monte Carlo Dropout autoencoder** : applique le dropout en phase d'inférence pour estimer l'incertitude, ce qui peut améliorer la robustesse du modèle et la détection d'anomalies.

3 Évaluation

Pour évaluation des différents algorithmes, utilisez des métriques d'évaluation internes et externes (pour clustering), des métriques de classification (pour détecion d'anomalies), ainsi que des indicateurs computationnels (pour clustering et détecion d'anomalies).

3.1 Clustering

3.1.1 Métriques internes (sans vérité terrain)

Ces indicateurs évaluent la qualité du cluster sans nécessiter de labels de référence :

- Coefficient de silhouette : mesure la cohérence d'un point avec son propre cluster par rapport aux autres clusters (valeurs entre -1 et 1; plus c'est élevé, mieux c'est).
- Indice de Davies-Bouldin : rapport entre la dispersion intra-cluster et la séparation inter-cluster (plus c'est faible, mieux c'est).
- Indice de Calinski-Harabasz : rapport entre la dispersion inter-cluster et intra-cluster (plus c'est élevé, mieux c'est).

3.1.2 Métriques externes (avec vérité terrain)

Ces indicateurs comparent les clusters obtenus aux labels réels :

- Indice de Rand ajusté (ARI) : mesure la concordance entre les labels réels et prédits, corrigée du hasard (valeurs entre 0 et 1 ; plus c'est élevé, mieux c'est).
- Information mutuelle normalisée (NMI) : mesure basée sur la théorie de l'information pour évaluer la qualité du cluster (valeurs entre 0 et 1 ; plus c'est élevé, mieux c'est).

3.2 Détection d'anomalies

- Accuracy (exactitude) : proportion d'instances correctement assignées à leur classe réelle. Utile lorsque les classes sont équilibrées.
- Recall (rappel) : capacité à identifier correctement les instances d'une classe donnée (sensibilité). Important pour la détection d'anomalies.
- Precision (précision) : proportion d'instances identifiées comme appartenant à une classe qui sont réellement correctes. Utile pour limiter les faux positifs.
- F1-score : moyenne harmonique entre précision et rappel, utile lorsque les classes sont déséquilibrées.

- ROC-AUC : mesure la capacité du modèle à distinguer les classes en évaluant le compromis entre taux de vrais positifs et taux de faux positifs. Plus la courbe est proche du coin supérieur gauche, meilleure est la performance.

Les valeurs de ces métriques sont comprises entre 0 et 1 ; plus c'est élevé, mieux c'est.

3.3 Clustering et détection d'anomalies

Ces indicateurs (métriques computationnelles) évaluent l'efficacité algorithmique :

- Temps d'entraînement : durée d'exécution sur le processeur.
- Utilisation mémoire : consommation maximale de mémoire pendant l'exécution.

3.4 Protocole d'expérimentation

Pour chaque algorithme ou méthode sélectionné(e) pour cette étude, vous devez réaliser un protocole expérimental rigoureux :

- Nombre d'exécutions : effectuez 10 exécutions indépendantes (expérimentations) de chaque algorithme ou méthode sur le même jeu de données et avec le même ensemble de paramètres.
- Initialisation : assurez-vous que chaque exécution utilise une initialisation aléatoire différente (si applicable, par exemple en modifiant la seed du générateur de nombres aléatoires) pour garantir l'indépendance statistique des résultats.

Pour chaque métrique de performance pertinente, affichez les résultats finaux sous la forme suivante : $(\mu \pm \sigma)$ où μ et σ représentent la moyenne arithmétique et l'écart-type des 10 valeurs obtenues pour la métrique respectivement. Elle représente la performance typique de l'algorithme.

4 Jeu de données et programmation

4.1 Jeu de données

4.1.1 Hi-Seq

Ce jeu de données (`https://archive.ics.uci.edu/dataset/401/gene+expression+cancer+rna+seq`) regroupe les niveaux d'expression génique mesurés par la plateforme Illumina HiSeq chez 801 patients atteints de différents types de tumeurs : BRCA (sein), KIRC (rein), COAD (côlon), LUAD (poumon) et PRAD (prostate). Chaque échantillon est représenté par 20531 variables réelles, correspondant à des gènes, et est stocké ligne par ligne. Il y a deux fichiers csv pour ce jeu de données, un (`hiseq_data.csv`) contient les variables indépendantes et l'autre (`hiseq_labels.csv`) contient la variable dépendante—ces deux fichiers sont à concaténer.

4.1.2 ECG

Ce jeu de données (`https://www.kaggle.com/datasets/shayanfazeli/heartbeat`) contient des enregistrements de l'activité électrique du cœur (ECG), mesurée au fil du temps. Il est conçu pour :

- Capturer les motifs typiques des signaux cardiaques normaux, comme les ondes P, QRS et T.
- Identifier les anomalies telles que les arythmies, les battements prématurés ou les irrégularités de fréquence.

Le fichier de données `ecg.npz` contient un ensemble de 4998 enregistrements ECG, chacun représenté par 141 attributs. Les 140 premiers attributs correspondent aux caractéristiques extraites du signal cardiaque, tandis que le 141ème attribut indique la classe associée à chaque enregistrement : une valeur de 0 désigne un signal normal, et une valeur de 1 signale une anomalie. Ce jeu de données est utilisé pour entraîner et évaluer des modèles de détection d'anomalies, permettant de distinguer les signaux cardiaques physiologiquement normaux des cas atypiques ou pathologiques.

Pour lire `ecg.npz`, on utilise le code suivant en Python:

Listing 1: Chargement du fichier ecg.npz

```
import numpy as np
data = np.load('ecg.npz')
df_ecg = data['ecg']
```

Pour la tâche de détection d'anomalies, le jeu de données doit être divisé en trois sous-ensembles distincts. Les données d'entraînement doivent contenir 60% des données normales uniquement, afin de permettre aux modèles non supervisés — notamment les auto-encodeurs — d'apprendre une représentation fidèle du comportement normal. Les données de test doivent regrouper 30% des données normales ainsi que 80% des données anormales, afin d'évaluer la capacité des modèles à détecter les écarts par rapport à la norme. Enfin, les données de validation doivent inclure les 10% restants des données normales et les 20% restants des données anormales. Ce dernier ensemble est utilisé spécifiquement pour déterminer le seuil de décision des auto-encodeurs et de leurs variantes, en calibrant la frontière entre comportements normaux et anomalies à partir des erreurs de reconstruction.

4.2 Programmation

Vous devrez utiliser `Python` comme langage de programmation. Les bibliothèques nécessaires pour ce TP ont été installées et utilisées lors de nos séances pratiques. Il est impératif de citer explicitement vos sources lorsque vous vous inspirez d'analyses réalisées par d'autres auteurs. Toute omission de citation sera considérée comme un acte de plagiat, pouvant entraîner l'attribution d'une note de zéro ainsi que des sanctions disciplinaires. Les

citations peuvent être intégrées directement dans vos programmes sous forme de commentaires, ou regroupées dans une section dédiée de votre rapport du TP2, sous forme de liste de références.

Tous les modèles/algorithmes développés dans le cadre de ce projet doivent être regroupés dans un fichier `models.py`, afin de centraliser les définitions des architectures et des algorithmes utilisés. Le prétraitement des données (nettoyage, normalisation, réduction de dimension, etc.) doit être implémenté dans un fichier distinct nommé `preprocess.py`. Les fonctions utilitaires générales, telles que le chargement des données, la gestion des partitions ou les métriques d'évaluation, doivent être placées dans `utils.py`. L'entraînement des modèles d'autoencodeur doit être réalisé dans un fichier dédié nommé `train.py`.

L'ensemble du code sera orchestré et exécuté à partir d'un `Notebook Jupyter`, qui servira à illustrer les étapes du pipeline, à visualiser les résultats et à commenter les choix méthodologiques.

5 Travail à réaliser

5.1 Préparation des données (15%)

Cette première étape du projet consiste à préparer et explorer les jeux de données afin d'assurer la qualité, la cohérence et la représentativité des informations avant l'application des algorithmes de clustering et détection d'anomalies. Elle inclut, si applicable, le nettoyage des données, l'imputation ou la suppression des valeurs manquantes, la normalisation des variables numériques, ainsi que l'encodage des données catégorielles. Des techniques de réduction de dimension (l'ACP, t-SNE ou UMAP) sont utilisées pour la visualisation. Lors de la visualisation, les instances de chaque classe doivent être représentées par une couleur distincte, afin de faciliter la distinction entre les catégories.

5.2 Clustering (30%)

Dans cette étape, des méthodes de segmentation sont appliquées afin d'identifier les clusters dans les données (5 clusters au total).

Trois approches seront successivement mises en œuvre :

- **K-Means**, pour partitionner les données en segments homogènes.
- **DBSCAN**, qui détecte les clusters denses et isole les points faiblement connectés comme anomalies.
- **Spectral Clustering**, qui exploite les relations de similarité via un graphe de voisinage pour segmenter les données.

Les résultats obtenus seront comparés afin d'évaluer la cohérence des clusters.

5.3 Détection d'anomalies (35%)

Cette section explore la détection d'anomalies à partir de la distribution normale des données, en s'appuyant sur des méthodes d'apprentissage fondées sur l'isolation ou la reconstruction. L'algorithme **Isolation Forest** identifie les observations atypiques en les isolant rapidement dans des arbres de décision, tandis qu'un **auto-encodeur** classique, entraîné sur des données normales, détecte les anomalies par une erreur de reconstruction élevée. Pour affiner l'analyse, des variantes d'auto-encodeurs (denoising, variational, sparse) sont testées afin d'améliorer la modélisation des régularités sous-jacentes—chaque groupe de travail doit choisir une variante d'auto-encodeur. Les performances sont comparées à l'aide de métriques mentionnées ci-haut.

5.4 Présentation des résultats (20%)

Dans votre rapport, vous devez décrire, brièvement, l'objectif et votre démarche pour chaque méthode. Vous devrez fournir des commentaires détaillés sur les paramètres des différents modèles utilisés, ainsi que sur les architectures des auto-encodeurs (i.e., nombre de couches, taille de chaque couche, etc.). Vous devez fournir quelques commentaires sur les résultats de chaque méthode-combinaison pour faciliter la compréhension de vos résultats. Si vous utilisez des ressources Internet, il faut absolument citer les sources aussi. Ne pas citer les sources sera considéré comme une acte de plagiat et pourrait conduire à une note de zéro en plus de s'exposer à des mesures disciplinaires. Il est fortement déconseillé d'utiliser des ressources Internet pour la partie de l'analyse des résultats.

6 Remise du TP

- Le TP doit être fait en équipe de deux à trois personnes.
- Indiquez les noms et Cips (ou matricules) des membres du groupe dans chacun des fichiers que vous soumettez.
- Ne téléchargez pas le jeux de données depuis les liens mentionnés ci-dessus. Utilisez plutôt la version incluse dans le fichier zippé du TP.
- La date de remise du TP est le mardi 25 novembre 2025 23h59'. Afin de garantir l'équité entre toutes et tous, aucun TP ne sera accepté après la date limite fixée.
- Soignez votre rapport, une pénalité de 5% pourrait être appliquée pour un rapport mal rédigé;
- Les fichiers à soumettre sont le rapport (en pdf) et l'ensemble de vos programmes. Ne pas soumettre les données!
- La remise doit être faite par Turnin : http://turnin.dinf.usherbrooke.ca