Projet : Génération de données synthétiques par réduction de dimension et modèles génératifs

MTH2329 – Mathématiques pour l'Intelligence Artificielle 3
Niveau Master - S3

Année universitaire 2025–2026

Contexte et objectifs

Ce projet s'inscrit dans le cadre du module MTH2329 – *Mathématiques pour l'Intelligence Artificielle*. L'objectif est de concevoir et comparer deux approches de génération de données synthétiques à partir d'un jeu de données simulé de courbes gaussiennes.

Vous devrez mettre en œuvre deux pipelines de génération :

1. Une approche classique combinant **ACP (Analyse en Composantes Principales)** et **GMM (Gaussian Mixture Model)**.
2. Une approche basée sur un **autoencodeur (AE)** comme méthode de réduction de dimension, couplée également à un GMM.

Le projet vous amènera à :

- Implémenter et comparer deux méthodes de réduction de dimension linéaire (ACP) et non linéaire (autoencodeur).
- Modéliser la distribution des données en espace latent à l'aide d'un modèle de mélange gaussien.
- Évaluer la qualité des données générées selon plusieurs critères statistiques et visuels.
- Analyser les avantages, inconvénients et domaines d'application de chaque approche.

1 Introduction à la génération de données

Dans de nombreux domaines (santé, finance, vision par ordinateur, etc.), la capacité à générer des données synthétiques réalistes est devenue essentielle :

- **Augmentation de données** : Pour entraîner des modèles de deep learning, surtout lorsque les données réelles sont limitées.
- **Préservation de la vie privée** : Créer des jeux de données synthétiques qui reproduisent les propriétés statistiques des données réelles sans exposer d'informations sensibles.
- **Simulation d'événements rares** : Pour l'entraînement de modèles de détection d'anomalies.
- **Benchmarking et validation** : Tester de nouveaux algorithmes dans des conditions contrôlées.

Concept clé

Défi principal : Les données réelles résident souvent sur une variété (manifold) de faible dimension plongée dans un espace de grande dimension. La modélisation générative vise à apprendre cette structure sous-jacente pour échantillonner de nouveaux points réalistes.

2 Description du projet

2.1 Données de départ

Vous travaillerez avec un jeu de données simulé de courbes gaussiennes, chacune paramétrée par sa moyenne μ et son écart-type σ :

$$f(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Chaque courbe est discrétisée en `nbx` points, formant ainsi un vecteur de grande dimension.

2.2 Pipeline général

Le processus génératif suit les étapes suivantes :

1. **Réduction de dimension** : Passer de l'espace de grande dimension (taille `nbx`) à un espace latent de dimension réduite $d \ll$ nbx.
2. **Modélisation de la densité** : Dans l'espace latent, apprendre la distribution des données à l'aide d'un modèle de mélange gaussien (GMM).
3. **Génération** : Échantillonner de nouveaux points dans l'espace latent à partir du GMM, puis les reconstruire dans l'espace original.

Cadre mathématique

Formalisation du pipeline ACP + GMM :

$$\begin{aligned}
\text{Données centrées : } & X_c = X - \bar{X} \\
\text{ACP : } & Z = X_c V_d \quad \text{(projection sur les } d \text{ premières composantes)} \\
\text{GMM : } & p(z) = \sum_{i=1}^{K} \pi_i \mathcal{N}(z|\mu_i, \Sigma_i) \\
\text{Échantillonnage : } & z_{\text{new}} \sim p(z) \\
\text{Reconstruction : } & x_{\text{new}} = z_{\text{new}} V_d^T + \bar{X}
\end{aligned}$$

Cadre mathématique

Formalisation du pipeline Autoencodeur + GMM :

$$\begin{aligned}
\text{Encodeur} &: z = f_\theta(x) \in \mathbb{R}^d \\
\text{Décodeur} &: \hat{x} = g_\phi(z) \in \mathbb{R}^{\text{nbx}} \\
\text{Perte} &: \mathcal{L} = \|x - \hat{x}\|^2 \\
\text{GMM} &: p(z) = \sum_{i=1}^{K} \pi_i \mathcal{N}(z|\mu_i, \Sigma_i) \\
\text{Échantillonnage} &: z_{\text{new}} \sim p(z) \\
\text{Reconstruction} &: x_{\text{new}} = g_\phi(z_{\text{new}})
\end{aligned}$$

3 Partie 1 : Approche ACP + GMM

Cette partie reprend la méthode classique présentée dans la séance de travail initiale.

Implémentation attendue:

- Génération des courbes gaussiennes.
- Réduction de dimension par ACP (avec `sklearn.decomposition.PCA`).
- Modélisation par GMM (avec `sklearn.mixture.GaussianMixture`).
- Échantillonnage et reconstruction.
- Visualisations : variance expliquée, courbes originales vs générées, clusters dans l'espace latent.

4 Partie 2 : Approche Autoencodeur + GMM

Un autoencodeur est un réseau de neurones composé de deux parties :

- Un **encodeur** qui réduit la dimension des données vers un espace latent.
- Un **décodeur** qui reconstruit les données à partir de l'espace latent.

Vous devrez concevoir un autoencodeur (de préférence avec `TensorFlow/Keras` ou `PyTorch`) adapté à vos courbes gaussiennes.

Une fois l'autoencodeur entraîné :

1. Utilisez l'encodeur pour projeter toutes les données d'entraînement dans l'espace latent Z.
2. Entraînez un GMM sur Z.
3. Échantillonnez de nouveaux z_{new} depuis le GMM.
4. Utilisez le décodeur pour reconstruire les courbes correspondantes.

5 Partie 3 : Comparaison et analyse

Cette partie constitue le cœur du projet. Vous devez comparer systématiquement les deux approches selon les axes suivants :

5.1 Comparaison quantitative

- **Erreur de reconstruction** : MSE entre données originales et reconstruites (sur un ensemble de test).
- **Qualité des générations** :
  - Distribution des paramètres μ et σ estimés sur les courbes générées.
  - Proportions de valeurs négatives (une vraie gaussienne est toujours positive).
  - Distance entre les moyennes et écarts-types des courbes originales et générées.

5.2 Comparaison qualitative

- Visualisation des espaces latents (2D si possible) : les points colorés par les clusters GMM.
- Affichage de courbes générées typiques pour chaque cluster.
- Capacité à générer des courbes variées mais réalistes.

5.3 Expériences complémentaires

1. **Sensibilité à la dimension latente** : Faire varier d (2, 5, 10, 20) et comparer l'impact sur les deux méthodes.
2. **Sensibilité au nombre de clusters GMM** : Tester différentes valeurs de K.
3. **Robustesse au bruit** : Ajouter du bruit gaussien aux données d'entraînement et observer l'effet sur les générations.
4. **Interprétabilité** : Les composantes ACP sont linéaires et souvent interprétables. Les dimensions latentes de l'autoencodeur le sont-elles ?

6 Cahier des charges et rendu

Contenu à fournir

1. **Notebook Jupyter principal (`projet_MTH2329.ipynb`)** contenant :
   - Toutes les étapes d'implémentation, d'expérimentation et de visualisation.
   - Des commentaires explicatifs en français.
   - Une analyse détaillée des résultats.
2. **Scripts Python modulaires** (optionnel mais recommandé) :
   - `data_generation.py` – Génération des courbes gaussiennes.
   - `pca_gmm_pipeline.py` – Pipeline ACP + GMM.
   - `autoencoder_gmm_pipeline.py` – Pipeline Autoencodeur + GMM.
   - `evaluation.py` – Métriques de comparaison.
   - `utils.py` – Fonctions auxiliaires.
3. **Rapport synthétique** (inséré dans le notebook en cellules Markdown) comprenant :
   - Une description de votre démarche.
   - Un comparatif structuré des deux méthodes (avantages/inconvénients).
   - Une discussion sur les choix d'architecture (dimension latente, nombre de clusters, etc.).
   - Des pistes d'amélioration ou d'extension.

Questions de réflexion (à traiter dans le rapport)

1. En quoi l'autoencodeur capture-t-il potentiellement mieux la structure des données que l'ACP dans ce contexte ? Dans quels cas l'ACP pourrait rester préférable ?
2. Comment la dimension latente choisie influence-t-elle la qualité des générations ? Y a-t-il un compromis entre fidélité et diversité ?
3. Quelles métriques vous ont semblé les plus pertinentes pour évaluer la qualité des données générées ? Pourquoi ?
4. Proposez une application réaliste (en traitement du signal, imagerie médicale, etc.) où l'approche autoencodeur + GMM pourrait être utile. Quel gain attendre par rapport à ACP + GMM ?

Modalités de rendu

- **Date limite** : 16 janvier 2026
- **Format** : Une archive `ZIP` nommée `Nom_Prénom_MTH2329_Projet.zip`
- **Contenu** : Le notebook + les scripts + éventuellement un fichier `README.md`

Critères d'évaluation

- Qualité et propreté du code (modularité, commentaires, reproductibilité).
- Exhaustivité des expériences et validité des résultats.
- Profondeur de l'analyse comparative.
- Clarté du rapport et pertinence des interprétations.
- Originalité des visualisations ou des analyses complémentaires.