Université de Strasbourg

Intelligence Artificielle

Licence 3 Informatique

TP : Classification non supervisée d'Iris avec l'algorithme des K-means

L'objectif de ce TP est de mettre en œuvre un algorithme de K-Means afin de partitionner les données du jeu de données des Iris.

Objectifs spécifiques :

- Observer les données, comprendre leur nature et comment les transformer, si besoin, pour les utiliser avec un k-means
- Comprendre le fonctionnement du partitionnement de données avec un K-means de façon à pouvoir mettre en œuvre ce partitionnement dans un programme Python.
- Évaluer les résultats et les mettre en perspectives avec ce que l'on sait des données.

1 Le jeu de données des Iris

... à déjà été présenté dans le TP précédent !

★ *Pourquoi dit-on que l'algorithme des K-means est un algorithme d'apprentissage non supervisé ?*

2 Mise en œuvre de l'algorithme des K-means

L'utilisation de l'algorithme des k-means permet de partitionner des données qui ne sont pas nécessairement étiquetées, dans k partitions distinctes.

★ *Sachant que le jeu de données des Iris contient 3 classes distinctes, combien de partitions allez vous créer dans avec votre programme ?*

Pour information, même si on ne dispose pas de données étiquetées, pour utiliser un K-means il faut de toute façon spécifier une valeur pour k : on qualifie alors ce type de modèles de modèles **semi-supervisés**.

Il est évidement que différentes valeurs de k vont conduire à différents résultats. Sans étiquette, vous ne connaissez pas à priori la valeur à définir (on peut approximer k en utilisant des méthodes dédiées telles que `https://link.springer.com/article/10.1007\%2FBF02294245`).

Construction de partitions Une fois k paramétré, la procédure pour créer les partitions est la suivante :

1. *Initialisation des clusters :* sélectionner k points aléatoirement dans le jeu de données et les utiliser comme centres initiaux des k partitions.
2. *Assigner des instances aux partitions :* pour chaque instance du jeu de données, calculer la distance de cette instance au centre de chaque cluster (distance Euclidienne). Assigner l'instance à la partition la plus proche.
3. *Mise à jour des centres des partitions :* calculer les moyennes de chaque partition. Pour chaque partition, la moyenne devient le centre de la partition.
4. *Répéter* les étapes 2 et 3 tant que les partitions continues à changer (ou bien spécifier un nombre maximum d'itérations, permettant ainsi de pallier des éventuelles oscilations)

★ *Mettez en œuvre cette procédure et visualisez les partitions résultantes avec Matplotlib.*

★ *Utilisez votre algorithme pour différentes valeurs de k et différentes méthodes d'initialisations des partitions. Décrivez les résultats obtenus.*

Visualisation de partitionnements avec K-means selon différentes initialisations : `http://shabal.in/visuals/kmeans/1.html`