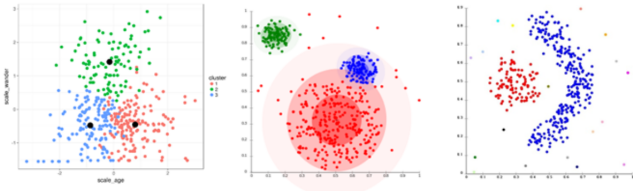
Emmanuel Franck

Année 2019-2020
Mathématiques Appliquées L3

Informatique S6 Projet: Apprentissage non-supervisé: Clustering

1 Introduction

Ce projet se place dans le contexte de l'apprentissage non supervisé et plus précisément de clustering. Contrairement à l'apprentissage supervisé qui construit un modèle à partir d'exemple "d'entrées" x et de "sorties" y. En apprentissage non supervisé, on construit un modèle juste à partir d'exemple de données x. Le **clustering** est un des problèmes de l'apprentissage non supervisé. Il s'agit de dispatcher les données dans différents paquets qui peuvent se voir comme des catégories. On va d'abord construire une classe pour gérer



un nuage de points qui sont nos données. Ensuite on construira une hiérarchie de classe pour gérer les méthodes de clustering.

2 Nuage de point

Objectif (4.0 points) : On va ici construire la classe nuage de point **A partir** du cours numéro 6.
Objectif : Construire la classe **NuagePoint**.
Il s'agit d' un template de classe dépendant d'une classe T qui correspond au type des données (points).
La classe devra contenir les attributs suivants :

- un entier qui correspond au nombre de points,
- un point de type "T" qui contiendra le tableau des points,
- 1 pointeur de fonction :
  - un pointeur nommé "distance" sur les fonctions qui prennent deux "T" et renvoie un double. Cette fonction permettra de calculer la distance entre deux données de type "T".

Elle devra contenir des accesseurs et mutateurs pour l'ensemble des attributs. Pour les pointeurs de fonction, un mutateur modifiera le pointeur (donc la fonction) et l'accesseur évaluera la fonction contenue dans le pointeur pour des paramètres donnés.
Vous devrez écrire le constructeur par défaut, le destructeur, le constructeur par copie et surchargez le "=".

Surchargez l'opérateur "[]" qui renverra le ième point. Surchargez l'opérateur "+" qui va concaténer deux nuages de points. Ajoutez des tests unitaires.

3 Classes de clustering

A partir du cours numéro 9.

Ici on va construire une classe clustering générale puis des classes filles qui correspondront à des méthodes de clustering.

3.1 Classe générale de clustering

Objectif (3.5 points) : Construire la classe **Clustering**.
Il s'agit d' un template de classe dépendant d'une classe T qui décrit ce que doit être une méthode de clustering. La classe devra contenir les attributs suivants :

- un entier qui correspond au nombre de clusters,
- un pointeur vers un objet de type "NuagePoint$< T >$" qui pointera vers un nuage de points ;
- un pointeur de type "entier" qui contiendra un tableau qui a chaque numéro de point stocke son numéro de cluster. Ce tableau sera initialisé avec -10 dans chaque case.

Elle devra contenir des accesseurs et mutateurs pour l'ensemble des attributs. Vous devrez écrire le constructeur par défaut, le destructeur, le constructeur par copie et surchargez le "=".
Surchargez l'opérateur "[]" qui renverra le cluster du ième point. Surchargez l'opérateur de sortie et ajouter des tests unitaires.
On ajoutera une fonction virtuelle pure qui aura vocation à calculer les clusters. Elle ne prendra pas de paramètre.

Dans le futur les clusters seront numérotés de 0 à $n_{clusters} - 1$. et on notera -1 les points qui n'appartiennent à aucun cluster.

3.2 Méthode de K-moyenne

Objectif (4.5 points) : Construire la classe **K-moyenne** qui hérite de la classe clustering.
Il s'agit d'une méthode qui construit les clusters en construisant des centres de clusters et qui ensuite assigne comme cluster à un point celui dans le centre est le plus proche.
L'algorithme est défini ici : `https://fr.wikipedia.org/wiki/K-moyennes`.

La classe devra contenir les attributs suivants :

- un point de type "T" qui contiendra le tableau des centres,
- un entier *niter* qui décrit le nombre d'itérations de la méthode.

Elle devra contenir des accesseurs et mutateurs pour l'ensemble des attributs.
Vous devrez écrire le constructeur par défaut, le destructeur, le constructeur par copie et surchargé le "=". Les constructeurs devront utiliser ceux de la classe mère. Surcharger l'opérateur de sortie et ajouter des tests unitaires. Il faudra construire trois fonctions :

— une qui initialise les centres de clusters en choisissant des points du nuage aléatoirement.
— une qui calcul les nouveaux centres de clusters à partir de la formule de moyenne (voir algorithme)
— la méthode "compute clusters" qui résout l'algorithme en enchaînant des phases de calculs de centres et des phases d'attribution des clusters aux points.

3.3 Méthode DBSCAN

Objectif (4.5 points) : Construire la classe **DBSCAN** qui hérite de la classe clustering.
A partir du cours numéro 12.
Il s'agit d'une méthode qui construit les clusters par une approche de densité. L'algorithme est défini ici : `https://fr.wikipedia.org/wiki/DBSCAN`. Ici on utilisera les vecteurs.

La classe devra contenir les attributs suivants :
— un double "ϵ" qui est un paramètre de la méthode,
— un entier "MinPts" qui est un paramètre de la méthode.
— un pointeur sur des booléen qui servira à savoir si un point a été visité ou pas.

Elle devra contenir des accesseurs et mutateurs pour l'ensemble des attributs.
Vous devrez écrire le constructeur par défaut, le destructeur, le constructeur par copie et surchargé le "=". Les constructeurs devront utiliser ceux de la classe mère.Surcharger l'opérateur de sortie et ajouter des tests unitaires.
Il faudra construire trois fonctions :
— une qui calcule le ϵ voisinage qui prend un point et renvoie un vecteur d'entier (classe "vector ») qui contiendra le numéro des voisins.
— une qui étend le cluster. Elle prend un index de point, un numéro de cluster et une liste de voisin et renvoie une liste de voisin modifiée.
— la fonction "compute clusters" qui code l'algorithme final.

4 Exemples

Objectif (4.0 points) : Coder des exemples et appliquer les deux algorithmes.

On va traiter deux nuages de points et coder deux fonctions qui génèrent ces nuages de points 2D. On va stocker ces points 2D dans des "$valarray < double >$" de dimension deux. Chaque élément du nuage de point sera donc un "$valarray < double >$".

Dans la première :
— on va construire un nuage de 40 points 2D. A chaque fois on va tirer dix points selon deux lois normales (une pour x_1, une pour x_2) de variance 0.05. Afin que les clusters soient bien séparés, les moyennes doivent être éloignées dans le plan 2D.
— Avec cette construction, les 10 premiers points sont dans un cluster, les dix suivants dans le second et ainsi de suite (utile pour débugger).

Appliquez les deux algorithmes et commentez le résultat (les paramètres de DBSCAN, l'initialisation des K-moyenne) etc.

Dans la seconde :
— on va construire un nuage de 40 points 2D.

— les 20 premiers seront générés sur un cercle de rayon 2 avec un peu de bruit (utilisé les coordonnées radiales et on bruite x_1, x_2 avec une loi normale de moyenne zéro).
— les 20 suivants seront générés sur un cercle de rayon 0.5 avec un peu de bruit (utilisé les coordonnées radiales et on bruite x_1, x_2 avec une loi normale de moyenne zéro).

Appliquez les deux algorithmes et commentez le résultat.