K-means

Sergio Peignier

1 Dataset et clustering

Dans ce TP nous allons faire du clustering de données. Définissons ces termes de manière plus précise, un dataset $X = \{x_1, x_2 \dots\}$ est un ensemble d'objets décrits dans $\mathbb{R}^D$ par D attributs (features), c'est à dire les coordonées des objets. Le clustering est une méthode qui a pour but de regrouper des objets d'un dataset de telle sorte à ce que les objets appartenenant à un même groupe partagent un niveau de similarité plus grand entre eux qu'avec des objets d'autres groupes. La notion de similarité n'est pas unique ... on peut la définir de plusieurs façons, mais on utilise souvent une notion de distance.

2 K-means

Vous allez coder un algorithme de clustering très connu et classique: le k-means crée par Stuart Lloyd en 1957. Le but de l'algorithme est de répartir n objets dans k groupes ou clusters, de telle sorte à minimiser la distance (le carré de la distance Euclidienne) entre les points trouvés et les centroides (position moyenne du groupe) des groupes. Chaque groupe sera entièrement caractisé par son centroide, le nombre de clusters à trouver est un paramétre de l'algorithme. Dans cet algorithme chaque objet ne peut appartenir qu'à un seul cluster. Imaginons qu'on a k clusters chacun contenant un ensemble de points du dataset et étant décrit par son centroide, l'algorithme alterne deux étapes:

- Affectation: Chaque objet du dataset est affecté au cluster le plus proche. Les distances entre le centroide du cluster et le point se mesurent en utilisant le carré de la distance Euclidienne. Cettte étape correspond à réaliser un affectation qui minimise la somme des distances Euclidennes au carré entre les objets que le cluster contient et le centroide de celui-ci.

- Mise à jour: Les différents centroides des clusters sont mis à jours. Les coordonnées de chaque centroide sont remplacées par les moyennes des coordonnées des points qui lui sont rattachés.

Après quelques itérations, l'algorithme converge vers un optimum local, c'est à dire que les affectations ne changent plus et que donc les mises à jours ne modifient plus les positions des centroides trouvés. La convergence vers un optimum global n'est pas assurée!

Plusieures méthodes existent pour initialiser les centroides, les méthodes les plus classiques consitent à prendre des points au hasard dans l'espace, ou k objets du dataset.

3 Implémentation

- Télécharger le dataset Iris du site http://archive.ics.uci.edu/ml/
- Charger le dataset dans python (plusieurs options possibles!) si pandas est installé je vous conseille de tester pandas.
- Viualizer le dataset en réalisant un scatter plot ou des projections 2D, vous pouvez choisir la couleur des points en fonction de leur classe.
- Faire une classe K-means et implémenter l'algorithme (je vous laisse réfléchir à la structure :-))
- Executer l'algorithme sur ce dataset et comparer les clusters obtenus et les classes réelles. Vous pouvez faire une matrice de confusion et la visualiser grâce à un heatmap.
- Calculer la somme des distances eucliennes au carré entre les points et les centroides (une fois que l'algorithme a convergé) en fonction du nombre de clusters k qu'on cherche.
- Executez plusieurs fois l'algorithme avec $k = 3$ mais avec des graines différentes, enregistrer à chaque fois la somme des distances eucliennes au carré entre les points et les centroides (une fois que l'algorithme a convergé), faire un histogramme avec les valeures obtenues.

4 Questions

On a vu que l'étape d'affectation minimise la distance (euclidienne au carré) entre les centroides et les objets. Que minimise l'étape de mise à jour? Est-ce que les deux minimisations sont compatibles?

Imaginons qu'on remplace la distance euclidienne au carré par une distance de Manhattan, comment doit-on modifier l'étape de mise à jour?

Jettez un coup d'oeil aux principales techniques existantes (http://scikit-learn.org/stable/modules/clustering.html), essayez d'autres techniques de clustering sur le dataset Iris (il y a encore plein de jeux interessants dans http://archive.ics.uci.edu/ml/).

Je vous passe un lien vers un tutoriel sympa pour faire du clustering hierarchique (mais bon, ce n'est pas le seul, il y a plein de tutos sympas que vous pouvez suivre, soyez curieux) https://joernhees.de/blog/2015/08/26/scipy-hierarchical-clustering-and-dendrogram-tutorial/

Quelle est la complexité computationnelle de cet algorithme?