Algorithme CURE

Guillaume Baud-Berthier Thomas Bellitto Thibault Godin

25 octobre 2013

Dans le cadre du cours *Algorithmes pour la Visualisation de données* nous devions implémenter un algorithme de clustering : CURE (Clustering Using REpresentatives). Comme le nom l'indique, l'idée est de choisir pour chaque groupe de données un jeu de représentants plus significatif que le seul barycentre, tout en gardant une complexité raisonnable. Grâce à cette représentation plus fine du cluster, on peut reconnaître des formes plus complexes que le cercle associé à la distance choisie. Dans CURE le choix des représentants se fait en prenant les points extrêmes du cluster.

Implémentation

Nous avons codé l'algorithme dans le module python de Tulipe. La première étape a donc été de prendre en main ces deux outils avec lesquels nous n'avions que peu d'expérience, et d'utiliser à bon escient les fonctions pré-construites du module Tulipe, comme la récupération des nœuds ou des couleurs. Après avoir lu l'article, nous avons commencé par définir une classe `cluster` puis nous avons suivi les algorithmes donnés, en complétant les détails sur lesquels l'auteur passait rapidement. Nous avons aussi fait le choix, notamment au vu de la taille « raisonnable » des données, d'utiliser une liste triée plutôt qu'un k-d tree associé à un tas. Le code est ainsi plus simple et reste suffisamment rapide.

Pour la partie visualisation, nous colorons les nœuds d'un cluster par une même couleur et actualisons les couleurs des nœuds à chaque étape. On peut ainsi visualiser l'avancement du programme mais le résultat final est assez rarement esthétique : en effet, nous devons attribuer beaucoup de couleurs au début, sans pouvoir prédire lesquelles resteront, et les choix sont donc souvent maladroits. Nous avons donc ajouté une étape de post-traitement qui attribue aux clusters les couleurs recommandées par le site colorbrewer (`http://colorbrewer2.org`), ce qui garantit un résultat lisible, du moment que l'utilisateur demande 12 clusters ou moins.

Au niveau des performances le programme traite le jeu de données (environ 400 points) en une quarantaine de secondes (33.7 pour 5 clusters et 5 représentants).

Analyse des résultats

Pour tester notre programme, nous avons étudié une représentation de la taille des moteurs (ordonnées) en fonction des prix (abscisses). Au premier coup d'œil le nuage de point semble plutôt formé d'un ensemble assez homogène suivant la première bissectrice du plan, une sorte d'ellipse avec quelques points éparpillés sans motif apparent. Le clustering nous aide à raffiner cette analyse.

Quand on lui demande plus de deux clusters, le résultat type est le suivant : un premier groupe selon l'axe (xy), surmonté par un deuxième de taille semblable selon une direction $(1, -1/2)$ environ, puis un ensemble de petits clusters (moins de cinq éléments en général), sans motif discernable.

Le clustering nous aide donc à distinguer essentiellement deux types de voitures : un pour lequel la taille du moteur est directement proportionnelle au prix, et un où au contraire, la taille du moteur diminue à mesure que le prix augmente, puis enfin quelques cas isolés sur lesquels il est difficile de faire une analyse. Pour le second type, on peut suggérer une amélioration de la performance du moteur à taille constante, qui induit un prix plus élevé.

Il pourrait maintenant être intéressant de mener une étude statistique sur les éléments de chaque cluster séparément.