Livret du module MATCHING

Nicolas Bourgeois

1 Prérequis

Il est nécessaire d'avoir quelques bases élémentaires de Python pour pouvoir suivre ce module. Si vous regardez les exercices disponibles sur `ouralou.fr` cela correspond aux trois premiers chapitres (introduction au langage, listes, fonctions).

Le jour du cours, il est nécessaire d'avoir installé Python 3 sur votre ordinateur, ainsi qu'un IDE de votre choix. Les jeux de données et autres ressources utilisées par le cours seront disponibles quelques jours à l'avance à l'adresse `ouralou.fr/Resources/epitech.zip`. Il est conseillé de les télécharger la veille.

2 Plan de cours

Jour 1 : Maximiser le nombre d'allocations

9h-9h15	Présentation du problème de matching
9h15-9h45	Recherche de solutions empiriques sur des exemples simples
9h45-10h	Graphe et structures de données possibles (3)
10h-10h30	Petits exercices avec la structure choisie
Pause	
10h45-11h	Définition de la méthode exhaustive
11h-11h30	Implémentation et crash tests scalés
11h30-11h45	Calculs et notion de complexité
11h45-12h15	Améliorations empiriques
12h15-12h30	Bilan et explications
Pause Déjeuner	
14h-14h15	Le problème du flot max
14h15-14h40	Construction du réseau associé au matching initial
14h40-15h	Algorithme de Ford-Fulkerson
15h-15h45	Implémentation et crash tests scalés
Pause	
16h-16h15	Retour complexité + I/O en Python
16h15-17h	Réalisation d'un programme autonome

Jour 2 : Déterminer des préférences

9h-9h15	Présentation de l'analyse multivariée
9h15-9h45	Importation des données et recherche de représentations
9h45-10h	Construction de distances, principe et exemples
10h-10h30	Mise en pratique sur le jeu de données
Pause	
10h45-11h	Construction de graphes avec seuil simple
11h-11h30	Implémentation sur l'exemple donné
11h30-11h45	Matching vs Clusterisation (cliques)
11h45-12h30	Mise en pratique
Pause Déjeuner	
14h-14h15	La classification hiérarchique ascendante
14h15-14h45	Programmation d'une CHA
14h45-15h	Optimisation : temps vs qualité du résultat
15h-15h45	Test scalés et évalués
Pause	
16h-16h15	Bilan général des notions abordées
16h15-17h	Réalisation d'un prototype global

3 Evaluation

Chaque groupe devra, d'ici le 31 janvier minuit, mettre à ma disposition un répertoire compressé portant le nom de tous les membres du groupe et contenant :

- un jeu de données original sous la forme d'un fichier texte, encodé en utf-8, appelé `data.csv` comportant plusieurs dizaines de champs alphanumériques (le premier étant un id unique) et plusieurs centaines de lignes. Les champs seront séparés par des points-virgules et les lignes de données par de simples retours à la ligne, comme dans les exemples du cours. Idéalement ce jeu de données proviendra d'un exemple réel. A défaut, vous le générerez aléatoirement avec les contraintes suivantes :
  - Certains champs seront qualitatifs, d'autres quantitatifs
  - Il existera des corrélations entre certains champs (par exemple taille, âge et poids dans une population donnée)

  Ce jeu de données sera accompagné d'une question qui justifiera le recours au matching. Par exemple, si ce sont les inscrit-e-s d'un site de rencontres on veut proposer un maximum de couples potentiels ; si c'est des joueurs/-joueuses d'un PvP on veut un maximum de parties par session, etc.
- deux algorithmes `build_graph.py` et `match.py`. Le premier algorithme lit le fichier de données et produit en sortie un graphe (ou une structure de données que vous jugerez appropriée) de compatibilité entre les individus. Il doit donc 1) coder les variables qualitatives 2) normaliser et pondérer

selon ce que vous jugerez pertinent toutes les variables 3) éliminer les variables inutiles 4) agréger les autres de façon à créer une distance d'individu à individu 5) fixer un seuil 6) construire un graphe avec des arêtes là où la distance est inférieure au seuil.

Le second algorithme est plus classique : il résout effectivement le problème de matching sur le graphe généré et propose donc une allocation optimale. Si vous avez construit un problème dans lesquels les associations peuvent contenir plus de deux éléments, il s'agit du coup d'un problème de coloration dans le graphe complémentaire et non de matching, donc np-complet, donc vous vous contenterez d'une heuristique car il n'existe pas d'algo exact. Dans tous les cas il revoie en sortie un fichier `result.csv` constitué d'une liste de paires (ou de groupes) d'ids représentant les appariements déterminés, au format habituel, par exemple :

```
284 ;1945
7327 ;12
8799 ;9554
```

Le score final dépendra de la qualité des données, de la qualité du traitement des données et de la construction des préférences, puis de la distance, puis du graphe, de la rapidité de l'algorithme de matching (ou de la pertience de l'heuristique si vous ne faites pas un matching) et bien sûr comme d'habitude de la qualité générale du code : commentaire, respect des consignes et conventions de nommage. Pas besoin de tests unitaires vu la taille relativement modeste du code en revanche.

4 Besoin d'aide ?

Je serai disponible par internet ou téléphone régulièrement, n'hésitez pas à me demander un rdv par mail (ou me poser directement des questions). Je n'écrirai jamais de code pour vous, mais je répondrai à toute autre question concernant aussi bien l'algorithmique que la syntaxe, les consignes ou encore la faisabilité de telle ou telle approche.

Bon courage.

5 Bibliographie

CORMEN Thomas, LEISERSON Charles, RIVEST Ronald et STEIN Clifford, *Algorithmique*, Dunod, 2010