Castellan Clément,
Le 17 Janvier 2021

Examen IBM

Conduite de projets en science des données et Predictive Analytics

1: IA

Décrire la différence entre approche réaliste et approche utilitariste *dans la démarche scientifique.*

Dans la démarche scientifique, l'approche réaliste pose pour but à la science de créer de la connaissance qui décrit le monde physique ou explore des idées et des concepts. Le but est d'établir des lois. L'approche utilitariste sera plus de l'ordre de l'utile : nous ne cherchons pas à définir des lois ou des connaissances, mais à réaliser une chose qui marche par reproduction ou inspiration de ce que nous observons dans la réalité.

En ce sens, l'IA ne relevant pas d'un domaine de la connaissance mais plutôt d'une méthode pour aborder la connaissance, elle s'inscrit davantage dans une approche utilitariste. Par exemple, s'il faut traiter beaucoup de données ou exécuter des tâches, nous observerons quel mécanisme humain reproduire par biomimétisme pour que la machine puisse réaliser cette tâche. Par exemple, les systèmes experts imitent la logique humaine, les algorithmes génétiques s'inspirent de l'évolutionnisme. Dans cette démarche, il n'y a pas la recherche de lois portant sur l'IA qui en plus réside dans le monde de l'information, l'infosphère et qui n'est pas régit par les mêmes lois que le monde physique.

2: programmation logique/chainage avant

Dans un langage à base de règles simple en chaînage avant (on appelle cela un système de production) on a le programme:

var *input[]=[], result[]=[], i=1, tmp=0;*

when *input.length>0* ***and*** *i >= input.length* ***then*** *result.append(input[tmp]), input.removeAt(tmp), i=1, tmp=0;*
when *input[i] < input[tmp]* ***then*** *tmp=i, i=i+1;*
when *input[i] >= input[tmp]* ***then*** *i=i+1;*

Lorsque l'instruction *input=[2,0,5,4,9];* est exécutée, que contiendront les variables *result* et *input* en retour? Expliquer l'algorithme. Donner sa complexité.

```
inputa = [2,0,5,4,9];
result = [];
i = 1;
tmp = 0;

while len(inputa) != 0:
    if len(inputa) > 0 and i >= len(inputa):
        result.append(inputa[tmp]);
        inputa.pop(tmp);
        i = 1;
        tmp = 0;
    elif inputa[i] < inputa[tmp]:
        tmp = i;
        i = i+1;
    else:
        i = i+1;

    print(inputa);
    print(result);
    print(tmp);
    print(i);
```

```
[2, 0, 5, 4, 9]
[]
1
2
[2, 0, 5, 4, 9]
[]
1
3
[2, 0, 5, 4, 9]
[]
1
4
[2, 0, 5, 4, 9]
[]
1
5
[2, 5, 4, 9]
[0]
0
1
[2, 5, 4, 9]
[0]
0
2
[2, 5, 4, 9]
[0]
0
3
[2, 5, 4, 9]
[0]
0
4
[5, 4, 9]
[0, 2]
0
1
[5, 4, 9]
[0, 2]
1
2
[5, 4, 9]
[0, 2]
1
3
[5, 9]
[0, 2, 4]
0
1
[5, 9]
[0, 2, 4]
0
2
[9]
[0, 2, 4, 5]
0
1
[]
[0, 2, 4, 5, 9]
0
1
```

En chainage avant, le programme possède un ensemble de règles et essaie de deviner toutes les conséquences de celles-ci sur une entrée donnée en exerçant les règles autant qu'elles peuvent être exercées.

Ici les règles sont les suivantes :

- Si la longueur du tableau d'entrée ET si i est supérieur ou égal à la longueur du tableau d'entrée, alors on ajoute au tableau de résultats la valeur du tableau d'entrée dont la coordonnée est la valeur de tmp. Ensuite on retire du tableau d'entrée la valeur dont la coordonnée est tmp. On affecte 1 à i et 0 à tmp.
- Si la valeur du tableau d'entrée à la coordonnée i est strictement inférieure à la valeur du tableau d'entrée à la coordonnée tmp, alors tmp prend la valeur de i puis i est incrémentée de 1.
- Autrement, si la valeur du tableau d'entrée à la coordonnée i est supérieure ou égale à la valeur du tableau d'entrée à la coordonnée tmp, alors i est incrémenté de 1.

En résumé le programme range par ordre croissant dans un tableau « result » les valeurs désordonnées d'un tableau d'entrée « input ».

En sortie nous aurons donc (cf l'implémentation python ci-contre) : input = [ ] et result = [0 , 2 , 4 , 5 , 9].

J'ai compris que la complexité d'un algorithme se détermine dans deux situations, le meilleur des cas et le pire des cas, mais surtout dans le pire des cas. Par contre je n'ai pas compris comment la calculer.

3: Smart City: Trouver sur internet 3 logiciels commerciaux **professionnels** destinés à remplir un rôle similaire à celui du projet SmartDeliveries. En vous basant sur la présentation commerciale, identifiez leurs principales caractéristiques démarquantes (quelles fonctionnalités mettent-ils particulièrement en avant par rapport à la concurrence). Fournir les références utilisées.

En termes de solutions similaires, le cours évoque mapotempo et copilote, nous pouvons également citer géoconcept. Les atouts mis en avant par ces trois concurrents pour se distinguer des autres portent sont les suivants :

- **Mapotempo** : L'expérience utilisateur intuitive, la philosophie open-source et la qualité du support (mises à jour fréquentes, etc…) https://www.mapotempo.com/
- **Géoconcept** : La compatibilité simple via API avec de nombreux appareils, la garantie d'un retour sur investissement, une prise en main simple et assistée. https://fr.geoconcept.com/app-plan-tournees-cloud
- **Copilote** : Propose un service plus général que la simple planification des tournées (gestion stocks, contrôle des chargements, dématérialisation des documents, etc…). https://www.infologic-copilote.fr/erp/logistique/

4: trafic routier

a- Quelles sont les principales variables mesurées par un détecteur de trafic?

Il y a deux valeurs principales mesurées :

- Le nombre de véhicules qui passent sur une route, il s'agit du débit
- Le taux d'occupation, c'est-à-dire le pourcentage de temps pour lequel il y avait un véhicule sur la chaussée. Si le trafic est arrêté, le véhicule ne bouge pas, il est donc sur la chaussée 100% du temps. Cette valeur renseigne sur la vitesse puisque si le taux d'occupation est élevé, la vitesse diminue.

b- Qu'est-ce que le diagramme fondamental d'un détecteur de trafic, pourquoi est-il utile pour mesurer et prévoir la congestion?

Le diagramme fondamental est construit avec le taux d'occupation en abscisse (ou la vitesse), et en ordonnée le nombre de véhicule qui passe. On a constaté que tous les diagrammes fondamentaux ont une courbe en forme de cloche en trois parties :

- Une réponse linéaire à gauche de la cloche, c'est-à-dire un trafic « vert » et fluide car le taux d'occupation est faible. Cela signifie que plus il y a de véhicules qui entrent, plus il y a de véhicules qui sortent (donc le débit, augmente lorsque l'on ajoute des véhicules).
- Un segment « plateau » qui peut faire passer l'indicateur du trafic routier à l'orange. Il s'agit d'un certain seuil où plus on ajoute des véhicules, plus ils vont lentement car ils sont trop près les uns des autres. Ce plateau correspond au débit maximal admissible, si on ajoute des voitures il n'augmentera pas.
- Enfin, quand la courbe en cloche redescend, l'indicateur trafic passe au « rouge ». Cela signifie que le taux d'occupation augmente, les véhicules sont de plus en plus proches les uns des autres mais ne peuvent pas avancer, le débit n'augmente donc pas et l'embouteillage se crée.

Ce diagramme est utile car il permet d'établir trois indicateurs de trafic routier : vert, orange et rouge. Il est aussi possible d'en extraite des courbes comme celle permettant de voir la congestion sur les jours de la semaine (grâce au taux d'occupation). Elle permet par exemple de rendre compte de patterns de congestions (comme le fait qu'en week-end la congestion est plus faible qu'en semaine) et donc cela permet à la ville d'ajuster les vitesses maximales sur un axe. En résumé, le diagramme fondamental permet d'éviter les embouteillages en permettant à la ville d'anticiper via l'établissement de modèles de congestion.

c- quel est le débit typique maximal d'un tronçon à une voie

- En zone urbaine : Environ 10.000 UVP/jours (source Wikipédia)
- Sur voie rapide ou autoroute : Environ 30.000 UVP/jours (source Wikipédia)
- Pourquoi cette différence ? Le débit est donc 3 fois plus important sur autoroutes que sur zone urbaines. Cela dépend d'abord des structures qui ont été pensées et faites pour un débit donné. Cela dépend aussi de l'attention des conducteurs qui détermine l'interdistance des voitures (plus faibles sur autoroute que sur une route de campagne, source Wikipédia). Enfin le nombre d'obstacles sur une route en zone urbaine va limiter son débit maximum (feux, passages à niveaux, etc…) contrairement à une autoroute qui est une ligne droite sans obstacles sur plusieurs kilomètres.

5: temps de parcours

a) Quelles sont les principales variables prédictives du temps de parcours d'un camion de livraison en ville, par ordre d'importance décroissante? (déterminées en cours)

Par ordre d'importance décroissante :

- La distance
- L'identifiant de tournée qui représente des facteurs endogènes
    - Le type de véhicule
    - Le chauffeur qui conduira et donc son expérience
    - La motivation du chauffeur
    - Les circonstances locales qui peuvent survenir
- Prévision du temps de parcours statique
- Congestion
- Angle (coordonnées polaires)

b) Citer 2 facteurs potentiels affectant les temps de parcours et difficiles à mesurer avec les données fournies dans les fichiers fournis en TP.

- Les circonstances locales rares (série de feux rouges, accidents sur les voies)
- Les biais des estimateurs ne sont pas corrigés, les erreurs s'accumulent car la loi des grands nombres ne s'applique pas

6. Prescriptive Analytics

Les affirmations suivantes sont-elles vraies ou fausses:

a- Un problème de décision est dans la classe de complexité NP si et seulement si il n'existe pas d'algorithme polynomial pour le résoudre.

FAUX car $P \subseteq NP$.

b- Dans l'industrie, la majorité des problèmes d'ordonnancement sont résolus grâce à des heuristiques.

VRAI. Aujourd'hui dans l'industrie la résolution des problèmes d'ordonnancement passe encore beaucoup par les heuristiques même si maintenant les industriels tendent de plus en plus vers des méthodes plus génériques, notamment CP pour la planification.

c- Le problème suivant possède exactement trois solutions:
u in {1,3}
v in {1,2}
w in {3,4}
x in {1,5}
y in {4,5}
allDifferent(u,v,w,x,y)

FAUX, il n'y a que deux combinaisons possibles : 12354 et 32415

d- L'algorithme de résolution de CP-Optimizer est un algorithme exact: si un problème d'optimisation est faisable, il garantit de trouver une solution optimale.

VRAI. Ça peut prendre longtemps mais il ressortira avec une solution optimale car il fait une recherche complète.

7. optimisation

a- En cherchant sur internet, décrivez un problème d'optimisation combinatoire non vu dans le cours dont la version de décision est un problème NP-Complet.

Il s'agit du problème du sac à dos. L'idée est que l'on doit remplir un sac à dos avec un ensemble d'objets pesant des poids différents. Le sac à dos supporte un poids maximum. Le but est de remplir le sac à dos en maximisant la valeur totale des poids des objets dans le sac pour en mettre le plus possible, sans dépasser le poids maximum supporté par le sac.

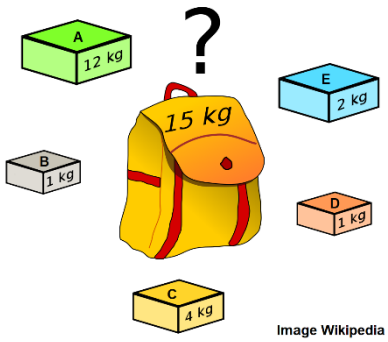

Image Wikipedia

b- Décrivez une petite instance particulière de ce problème d'optimisation (avec des valeurs pour chacune des données).

Par exemple nous pourrions avoir un sac qui supporte 15Kg au maximum avec cinq boites A, B, C, D, E qui pèsent respectivement 12Kg, 1Kg, 4kg, 1kg, 2kg. Le but est de mettre le plus de boites possibles dans le sac.

c- Donnez une solution faisable non-optimale et une solution optimale de cette petite instance.

Solution faisable non optimale : A + B + E
Solution optimale dans l'instance donnée : B + C + D + E

8. programmation par contraintes

Deux principes fondamentaux de la Programmation par Contraintes sont (1) la recherche arborescente et (2) le filtrage du domaine des variables. Décrivez brièvement ces principes, leurs rôles et la façon dont ils sont mis en oeuvre durant la résolution.

La programmation par contraintes c'est proposer une solution qui prend en compte un certain nombre de contraintes : ex un livreur qui doit s'arrêter à des endroits précis. C'est une technique pour résoudre des problèmes d'ordonnancement. Le but est de trouver une combinaison optimale d'objets dans une large collection d'objets finie en fonction de contraintes données.

L'implémentation se fait par la pose du problème et l'énonciation des règles de résolution du problème dans un langage compréhensible pour l'ordinateur, python par exemple.

L'algorithme repose sur deux grands principes :

- Premier principe l'arbre de recherche : Il s'agit d'une représentation en arbre pour retrouver une valeur : si l'algorithme doit trouver une valeur, alors il va essayer toutes les valeurs possibles et pour chacune d'elle essayer toutes les valeurs suivantes possibles. A chaque fois des nœuds sont créés et les nœuds valide sont explorés. Si un nœud ne satisfait pas une règle la branche est supprimée car cette combinaison n'est pas une solution et l'algorithme remonte au nœud précédent.
- Il est possible d'optimiser le tri par arbre. Certaines valeurs sont inutiles à explorer car impossibles selon les règles données, ainsi il est possible de faire un filtrage du domaine des variables pour que l'algorithme n'explore que les variables qui sont consistantes et qui pourraient donc être une solution. Cette méthode repose sur la propagation des contraintes : l'idée est que l'algorithme considère les valeurs possibles puis élimine toutes celles qui sont inconsistantes. Par cette élimination, il va pouvoir enlever davantage de valeurs car le premier retrait de certaines valeurs aura peut-être créé des nouvelles valeurs inconsistantes ou aura permis l'application de règles. Donc il tourne jusqu'à ce qu'il n'y ait plus rien à enlever et cela réduit considérablement le nombre de pistes à explorer.