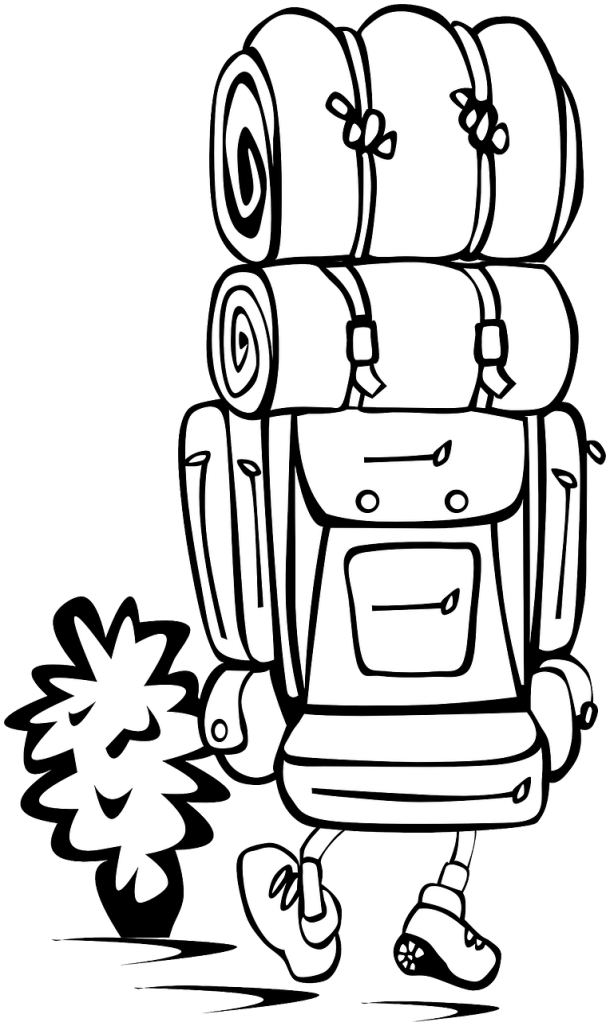
Algorithme glouton et IA



https://www.mathexien.com

Table des matières:

Introduction:

Nous avons déjà vu que lorsqu'un algorithme a une complexité exponentielle (en 2^n), le temps d'exécution devient rapidement trop élevé d'où l'intérêt de trouver d'autres paradigmes de programmation, et notamment des méthodes d'optimisation (des heuristiques)

Nous avons déjà travaillé sur une méthode au programme de première (algoithme des k plus proches voisins) et nous allons dans ce TP travailler sur une deuxième méthode au programme de 1ère: l'algorithme glouton, c'est un algorithme assez naturel qui est utilisé dans la vie de tous les jours.

En Terminale, on continuera à développer des méthodes pour aborder des problèmes complexes (d'un point de vue algorithmique) et dans le supérieur vous aborderez surement d'autres aspects, notamment l'analyse théorique des problèmes NP-complets et l'intelligence artificielle.

Mais, comme beaucoup sont intéressés par le dernier sujet, je vous proposerai à la fin du TP, à titre optionnel, une activité concrète où vous allez développer un réseau de neurones en moins de 3 minutes, que vous pourrez approfondir et intégrer dans vos développements.

Objectifs:

L'objectif de ce TP est:
- d'étudier l'algorithme glouton
- d'initier ceux qui le souhaitent au machine learning

Instructions:

Vous avez chacun un exemplaire de ce document que vous devez compléter par vos réponses aux questions.

1. Découverte

Le principe de l'algorithme glouton (greedy algorithm) est de faire une suite de choix à chaque étape, en maximisant ce qu'on prend à chaque fois selon une règle à définir (d'où le nom: il "bouffe" le maximum à chaque étape), et sans revenir en arrière sur les choix précédents.

Cet algorithme peut trouver la solution optimale dans certaines situations, une solution non optimale (de qualité plus ou moins bonne) dans d'autres situations ou encore ne pas trouver de solution.

Prenant un exemple utilisé tous les jours pour illustrer le principe de cet algorithme, le rendu de monnaie.

Si voulez rendre 9 Dh, il y a plusieurs manières de faire , mais il y en a qu'une optimale, c'est à dire avec le minimum de pièces.

Question 1: Représenter par un arbre (comme en probabilité) les différentes combinaisons des pièces de 1 Dh, 2 Dh, 5 Dh, entourez les feuilles de l'arbre qui permettent d'obtenir 9 Dh et encadrer la solution optimale (on peut s'arrêter quand on dépasse 9 Dh)

Renseignez votre réponse ici

Assez naturellement, on utilise un algorithme glouton, en prenant à chaque fois le maximum : la pièce où le billet ayant la plus grande valeur (inférieure à la monnaie restante)

Par exemple pour rendre 169 Dh, on va:
- rendre 100 Dh, il reste 69 Dh
- rendre 50 Dh, il reste 19 Dh
- rendre 10 Dh, il reste 9 Dh
- rendre 5 Dh, il reste 4 Dh
- rendre 2 Dh, il reste 2 Dh
- rendre 2 Dh, c'est bon, on a tout rendu

Question 2: Donnez l'algorithme en langage naturel (on fera la programmation en Python plus tard) pour rendre la monnaie avec l'algorithme glouton

Renseignez votre réponse ici

Question 3: Trouvez un exemple de système monétaire (vous pouvez enlever des pièces) où l'algorithme glouton ne trouvera pas toujours la solution optimale

Renseignez votre réponse ici

2. Sac à dos

On passe maintenant à une autre situation assez classique où l'on a des objets, chaque objet a un poids et une valeur et l'on veut choisir plusieurs objets pour maximiser la valeur globale tout en respectant une contrainte sur le poids total.

C'est ce qu'on appelle le problème du sac à dos que l'on peut résoudre exactement (vous le ferez en Terminale) ou de manière approchée avec un algorithme glouton: c'est ce qu'on va faire ici.

Prenons un exemple simple pour commencer.

On a trois Objets:
$O_1 : \ poids = 4 \quad valeur = 3$
$O_2 : \ poids = 6 \quad valeur = 5$
$O_3 : \ poids = 3 \quad valeur = 2$

et on veut les mettres dans un sac à dos en maximisant la valeur tout en ne dépassant pas sa capacité maximale (en terme de poids): $poids_{max} = 7$

Pour préparer la généralisation on va utiliser des listes pour:
les poids des objets: $p = [4, 6, 3]$
les valeurs des objets: $v = [3, 5, 2]$
la solution, c'est à dire les objets qu'on va retenir, en mettant à l'index de l'objet 1 si on le prend, et 0 sinon: $s = [?, ?, ?]$
Les objets sont repérés par leur index dans ces listes.

Question 1: Représenter par un arbre (comme en probabilité), toutes les solutions réalisables possibles et identifiez la solution optimale.
NB: vous procéderez comme suit:
1ère étape: choix du 1er objet (soit je le prends $s[0] = 1$ soit je ne le prends pas $s[0] = 0$)
2ème étape: choix du 2ème objet (soit je le prends $s[1] = 1$ soit je ne le prends pas $s[1] = 0$)
3ème étape: choix du 3ème objet (soit je le prends $s[2] = 1$ soit je ne le prends pas $s[2] = 0$)

Renseignez votre réponse ici

Question 2: Déterminez le nombre de feuilles maximales de l'arbre si l'on a n objets. Déduisez en la complexité d'un algorithme qui chercherait la solution exacte parmi toutes les solutions.

Renseignez votre réponse ici

On va maintenant chercher une solution avec un algorithme glouton.

Mais quel critère va t'on essayer de maximiser à chaque étape?
On pourrait prendre la valeur, mais c'est pas très malin (prenez l'exemple d'un objet lourd qui remplit tout le sac alors qu'on a des objets de faible poids qui ont une valeur juste un peu plus faible)
On peut aboutit à la même conclusion en prenant le poids.
Alors, on va prendre un autre critère qui fera une sorte de compromis, le ratio entre la valeur et le poids.

On va donc essayer de maximiser à chaque itération le quotient $q = v/p$

Question 3: Dérouler l'algorithme glouton sur notre exemple en utilisant $q = v/p$ comme critère à maximiser à chaque itération. Comparer à la solution exacte.

Renseignez votre réponse ici

On va maintenant généraliser et construire un programme qui construit la solution d'un problème sac à dos selon l'algorithme glouton.

Les entrées sont:

la liste des poids des n objets: $p = [\, p_0 \,,\, p_1 \,,\, p_2 \,,\, \ldots \,,\, p_{n-1}]$
la liste des valeurs des n objets: $v = [\, v_0 \,,\, v_1 \,,\, v_2 \,,\, \ldots \,,\, v_{n-1}]$
le poids max: p_{max}

La sortie est la solution:
la liste des des n objets pris (1) ou non pris (0): $s = [\, s_0 \,,\, s_1 \,,\, s_2 \,,\, \ldots \,,\, s_{n-1}]$

Question 4: Donnez l'algorithme en langage naturel (on fera la programmation en Python après) pour trouver la solution du problème sac à dos avec l'algorithme glouton.

Renseignez votre réponse ici

Vous pouvez développez le programme en Python , vous êtes libre pour l'implémentation, mais voici tout de même quelques fonction à implémenter obligatoirement:

Génération d'instances: $instanceObjets\,(n : int\,,\, R : int\,) \rightarrow (\, p\,,\, v\,) : tuple$
Cette fonction va générer aléatoirement les données de n objets, c'est à dire les listes p et v
Chaque donnée est un entier de 1 à R généré par : $randint(1,\, R)$

Calcul du poids et de la valeur total d'une instance:
Cette fonction retourne la somme des poids (des valeurs) de tous les objets de l'instance
$poidsInstance(\, I : instance) \rightarrow poidsTotal : int$
$valeurInstance(\, I : instance) \rightarrow valeurTotal : int$

Poids maximal du sac à dos: $poidsMax(\, I : instance) \rightarrow poids_{max} : int$
On prendra la moitié du poids total de l'instance (sauf si on veut tester des cas particuliers)

Solution glouton: $glouton(\, I : instance,\, poids_{max} : int\,) \rightarrow s : list$
La fonction principale qui prend en entrée une instance de donnée et un poids max et retourne la solution glouton à travers la liste des des n objets pris (1) ou non pris (0)

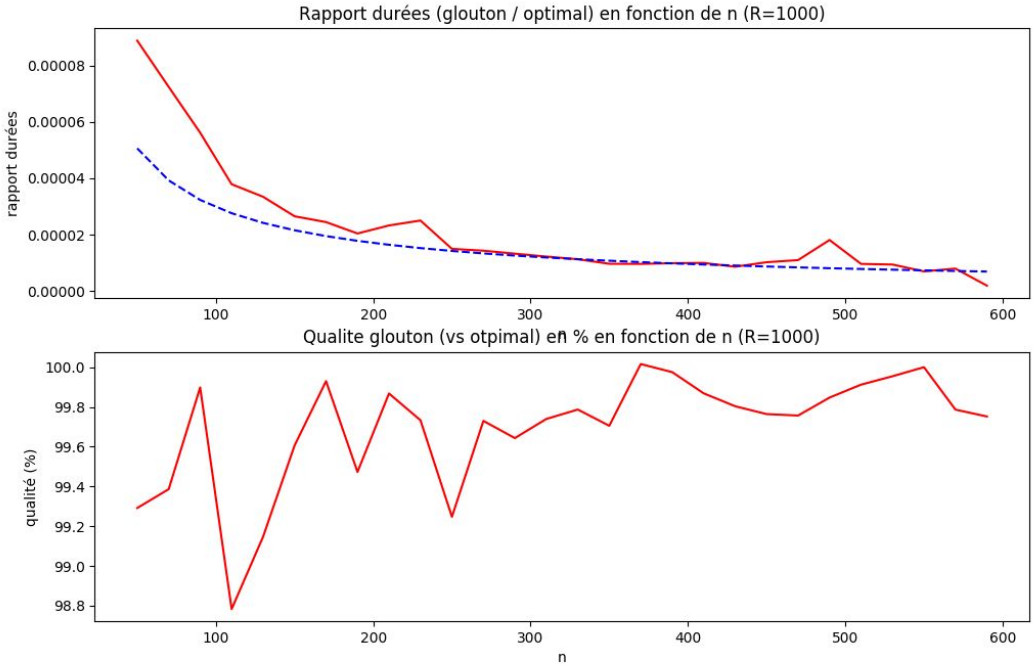
Question 5: Développer un programme en Python pour trouver la solution du problème sac à dos avec l'algorithme glouton.

Renseignez votre réponse ici

joindre votre fichier Python

En option: C'est au programme de Terminale, mais vous pourrez, si vous vous ennuyer pendant les vacances, chercher un algorithme qui donne la solution exacte du problème de sac à dos et analyser ainsi la qualité de l'algorithme glouton et comparer les durées d'exécution des algorithmes.

Voilà ce que je trouve:



On voit clairement que, tant qu'on n'est pas dans une situation de risque fort, l'algo glouton est tout à fait acceptable pour résoudre des problèmes de type sac à dos (qualité supérieur à 99,6% pour $n > 300$)

3. Machine learning (en option)

C’est un sujet vaste et complexe, mais les principes sont assez simples et de manière pratique, des bibliothèques et des banques de données, vous permettent de développer très rapidement un réseau de neurones performant.

Vous pourrez après la première vidéo (7 min):
https://youtu.be/KNAWp2S3w94

et la première activité (3min):
https://colab.research.google.com/github/lmoroney/mlday-tokyo/blob/master/Lab1-Hello-ML-World.ipynb

définir et entraînez votre premier réseau de neurone qui va prévoir les images d’une fonction dont il ne connaît pas la définition.

Pour l’activité pas besoin d’installation, l’environnement lié permet d’exécuter directement le code sur la page web.

La suite de la série vous permettra de définir et d'entraîner un réseau de neurones qui pourra catégoriser une image.

Vous pourrez intégrer cela dans vos projets de développements interactifs: un jeu de chifoumi où le joueur fait le geste par exemple...

Amusez vous bien !!!