Optimisation Financière

Un algorithme pour maximiser le rendement de l'investissement

Pop Alexandru
10/10/2023

Introduction & Aperçu

Introduction

Dans ce rapport, je présenterai un problème d'optimisation financière qui implique la sélection des actions les plus rentables pour des investissements, tout en ayant un budget limité. Ce problème est similaire au problème du sac à dos (Knapsack Problem) et au problème du rendu de monnaie (Coin Change Problem) en informatique.

Points qui seront abordés

Nous explorerons différents algorithmes, leurs avantages et inconvénients.

Le Problème & Problèmes Similaires

Description du problème :

- Le but est de maximiser le profit en sélectionnant les meilleures actions, sans dépasser un budget de 500 euros.

- **Contraintes**:
  - Chaque action peut être achetée une seule fois, et nous ne pouvons pas acheter une fraction d'action.
- **Problèmes similaires**:
  - Ce problème est similaire au problème du sac à dos en informatique, où nous devons maximiser la valeur sans dépasser une certaine capacité.
  - Il est également lié au problème de la monnaie, où nous devons atteindre une somme donnée avec le moins de pièces possible.

Algorithmes Possibles et Leurs Explications

Programmation Dynamique

Utilise les solutions de sous-problèmes pour construire la solution finale. Peut être lent pour de grands ensembles de données et budgets élevés.

Algorithmes Génétiques

Utilise des méthodes d'évolution pour améliorer un ensemble initial de solutions au fil du temps. Le temps d'exécution dépend de la rapidité avec laquelle une bonne solution 'évolue'.

Méthodes Heuristiques

Utilise des 'astuces' ou des 'règles générales' pour trouver une solution assez bonne en un temps court, sans examiner chaque possibilité.

Algorithme de Force Brute

Introduction et complexité simplifiée

L'algorithme de force brute examine toutes les combinaisons possibles pour trouver la meilleure. Cependant, cela prend beaucoup de temps, surtout lorsque le nombre d'actions est élevé.

Avantages et inconvénients

Avantages:

Solution exacte.

Inconvénients:

Très lent pour de grands ensembles de données.

Non Optimal

Pour presque 2000 actions, le temps de calcul serait extrêmement long, le rendant impraticable pour notre cas.

Algorithme Glouton (Greedy)

Introduction et complexité simplifiée

L'algorithme glouton trie les actions en fonction de leur rentabilité et sélectionne les meilleures jusqu'à ce que le budget soit épuisé.

Avantages et inconvénients

Avantages:

Rapide et efficace.

Inconvénients:

Ne garantit pas toujours la solution optimale.

Performances

Force Brute

Nombre de donnes :

20

Temps d'exécution :

```
PS C:\Users\Alex\OneDrive\Desktop\FORMATION\algo> py .\brute_force.py
Temps d'execution: 1.5257675000466406 seconds.
PS C:\Users\Alex\OneDrive\Desktop\FORMATION\algo>
```

Glouton (Greedy)

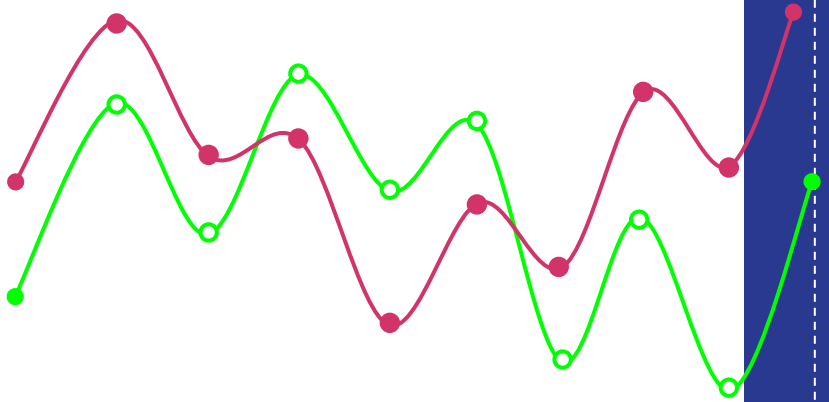
Nombre de donnes :

1500

Temps d'exécution :

```
PS C:\Users\Alex\OneDrive\Desktop\FORMATION\algo> py .\optimi
Temps d'execution: 0.0034202998504042625 seconds.
PS C:\Users\Alex\OneDrive\Desktop\FORMATION\algo>
```

Solution



Nous avons opté pour l'algorithme Greedy en raison de sa rapidité exceptionnelle et de son efficacité dans la résolution de notre problème spécifique d'optimisation financière.

Greedy est non seulement plus rapide mais aussi beaucoup plus simple à mettre en œuvre, ce qui accélère le processus de développement et de déploiement de la solution.