SK.POP.5
6 grudnia 2022

Jan Jędrzejewski
Dominik Łopatecki

Problem plecakowy, PBIL

Projekt Wstępny

Źródła:

1) Prezentacje wykładowe z przedmiotu POP prof. J. Arabasa
2) Artykuł ze strony podanej w literaturze: DOI:10.3390/app11199136
3) https://en.wikipedia.org/wiki/Population-based_incremental_learning
4) https://pl.wikipedia.org/wiki/Problem_plecakowy

Opis projektu

Interpretacja problemu plecakowego

Problem plecakowy, wielokrotnie poruszany na wykładach z optymalizacji, polega na maksymalizacji wartości przedmiotów które wybieramy. Istnieje jedno ograniczenie, wybrane przedmioty muszą mieścić się w plecaku. Zagadnienie nie pozwala na dzielenie przedmiotów na części. Często przedstawiany jako problem złodzieja okradającego sklep. Pragnie on zmieścić przedmioty o jak największej wartości do swojego plecaka.

pi - wartość i-tego przedmiotu
wi - waga i-tego przedmiotu
xi - informacja o tym czy wybieramy przedmiot
W - ograniczenie (maksymalna waga) plecaka

$$max \sum_{i=1}^{n} x_i p_i$$

$$\sum_{i=1}^{n} x_i w_i \leqslant W$$

$$x_i \in \{0,1\}$$

W matematyce problem plecakowy możemy przedstawić w postaci maksymalizacji funkcji zysku. Szukamy optymalnego wektora x o binarnych wartościach [x1, x2, x3, …, xn]. 0 oznacza że nie wybieramu danego przedmiotu, 1 dla zabranych.

Metoda heurystyczna - sortowanie przedmiotów według współczynnika

Do rozwiązania problemu plecakowego można użyć metody heurystycznej. Wyznacznikiem czy dany przedmiot opłaca się zabrać może być stosunek wartości do wagi. Innymi słowami jak duża wartość znajduje się w jednostce wagi. Poniżej przykład.

w - waga przedmiotu,
p - wartość przedmiotu,
W = maksymalna waga

założenia: W=12
Chcemy wybrać przedmioty o maksymalnej wartości

i	1	2	3	4	5	6
p_i	20	5	10	5	6	3
w_i	6	2	5	3	4	3
p_i/w_i	3.33	2.5	2	1.66	1.5	1

W powyższej tabeli umieszczono przedmioty posortowane w kolejności malejącej pod względem atrybutu p/w. Po kolei dokładamy przedmioty do pustego plecaka, w momencie gdy przedmiot się nie zmieści pomijamy go i próbujemy kolejny. Kończymy gdy przejdziemy przez całą listę, bądź gdy wypełnimy plecak plecak. Poniżej kolejne kroki:

1) Do plecaka wkładamy przedmiot 1. Łączna waga: 6; Łączna wartość: 20
2) Do plecaka wkładamy przedmiot 2. Łączna waga: 8; Łączna wartość: 25
3) Do plecaka nie możemy włożyć przedmiotu nr. 3 ponieważ 8 + 5 > 12
4) Do plecaka wkładamy przedmiot 4. Łączna waga: 11; Łączna wartość: 30
5) Do plecaka nie zmieszczą się przedmioty 5 i 6. Kończymy z plecakiem o wartości 30

W plecaku otrzymaliśmy przedmioty 1, 2 i 4. Ich łączna wartość to 30. Nie jest to maksymalna do uzyskania wartość, ale w stosunkowo prosty sposób pozwala zbliżyć się do optymalnego wyniku. Gdybyśmy wyjęli przedmiot 4 i w jego miejsce włożyli przedmiot 5 to waga plecaka byłaby równa ograniczeniu, a wartość plecaka wyniosłaby 31.

Ta metoda nie byłaby efektywna dla dużej liczby przedmiotów i dla stosunkowo niewielkiego plecaka. Duże elementy mogą zapchać plecak i nie będziemy go w stanie wypełnić tak jak w powyższym przykładzie.

Rozwiązanie sztucznej inteligencji

Do rozwiązania problemu plecakowego z pomocą przyjdzie nam uczenie maszynowe i algorytmy przeszukiwania przestrzeni. Wszystkie możliwości spakowania przedmiotów do plecaka potraktujemy jako bogatą dziedzinę zadania. Spośród niej musimy znaleźć punkt optymalny - to znaczy zestawienie przedmiotów które posiadają największą wspólną wartość i mieszczą się w ograniczeniu wagi.

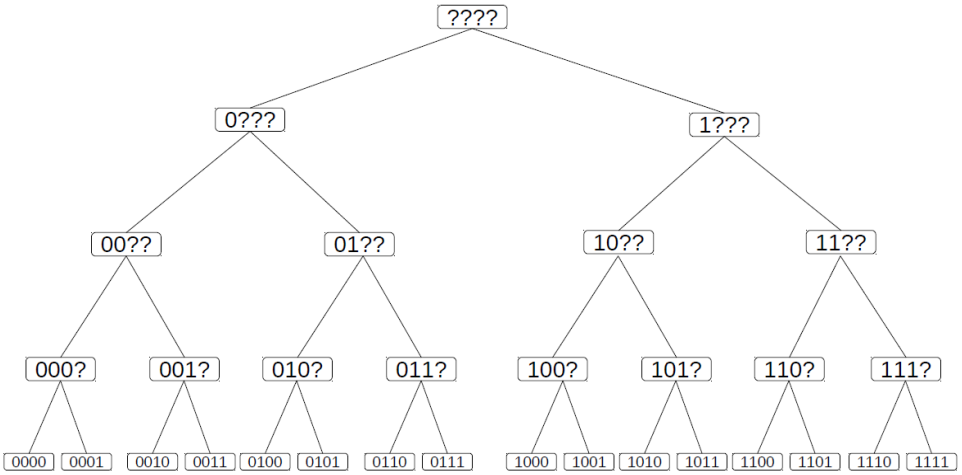
Zestaw przedmiotów będziemy reprezentować poprzez ciągi bitowe $x_i = \{0, 1\}$. Gdzie:

- ➔ 1 oznacza że przedmiot został wybrany do plecaka,
- ➔ 0 przeciwnie - pozostawiamy przedmiot.

Wówczas dla n-przedmiotów mamy 2^n kombinacji różnych sposobów spakowania ich do plecaka. Do przeszukania problemu w którym występuje ponad 30 przedmiotów nie jesteśmy w stanie użyć standardowych metod takich jak porównanie wszystkich możliwości.

Natomiast można do tego zagadnienia wykorzystać algorytmy do przeszukiwania łańcuchów znaków na przykład algorytmy genetyczne. Populacją początkową może być dowolnie wybrany podzbiór przedmiotów. Z każdą iteracją algorytmu poprzez mutację, krzyżowanie oraz selekcję będziemy przybliżać się do optymalnego zestawu przedmiotów.

Również możemy skorzystać z algorytmów przeszukujących struktury drzewiaste np. A*.



Gdy przedstawimy przestrzeń w postaci binarnego drzewa, tak jak powyżej, gdzie korzeniem będzie niezdefiniowany zestaw przedmiotów, a kolejne poziomy będą miały sprecyzowany pierwszy bit oznaczający czy zabieramy przedmiot o danym numerze czy nie. Głębokość takiego drzewa jest równa liczbie przedmiotów n, natomiast liczba liści to 2^n.

Opis wykorzystywanych algorytmów

Algorytm PBIL (Population-Based Incremental Learning)

Algorytm Population-Based Incremental Learning należy do rodziny algorytmów ewolucyjnych, a dokładnie do klasy Estimation of Distribution Algorithm. Algorytmy tej klasy cechują się tym, że mechanizm ewolucyjny realizowany jest przy pomocy modelu probabilistycznego. Dużym uproszczeniem jest fakt, że rozkład prawdopodobieństw każdego z bitów jest niezależny, co pozwala na osobny proces uczenia oraz generowania punktu dla każdej zmiennej.
Rozkład prawdopodobieństwa w PBIL reprezentowany jest jako wektor zawierający prawdopodobieństwo wystąpienie 1 w danym bicie.

$$p^t = [p_1^t, p_2^t, p_3^t, \ldots, p_n^t]$$, gdzie:

t - numer iteracji, n - rozmiar chromosomu

Algorytm PBIL- schemat działania:

1. Tworzymy populację P^t o rozmiarze M, bazując na p^t
2. Wybieramy z P^t N najlepszych rozwiązań tworząc zbiór O^t
3. Następnie aktualizowany jest wektor prawdopodobieństw zgodnie ze wzorem:

$$p^{t+1} = (1 - lr) * p^t + lr * \frac{1}{N} \sum_{x \epsilon O^t} x$$, gdzie:

lr - współczynnik uczenia

4. Mutacja wyliczonego p^{t+1}
5. Jeśli warunek stopu nie jest spełniony przejście do punktu 1.

Jako warunek stopu można zastosować maksymalna liczbę iteracji. Dodatkowo algorytm powinien zakończyć się w sytuacji prawie rozstrzygniętej, czyli takiej gdzie wartość każdego bitu będzie w określonym sąsiedztwie 0 lub 1.

Przykład obliczeń:

1. p^t = [0.5, 0.5, 0.5, 0,5]
2. Zbiór O^t = {[0, 1, 1, 0], [1, 1, 1, 0], [0, 0, 1, 0]}
3. Lr = 0.2
4. p_1^{t+1} = 0.8 * 0.5 + 0.2 * ⅓ = 0.467
5. p_2^{t+1} = 0.8 * 0.5 + 0.2 * ⅔ = 0.533
6. p_3^{t+1} = 0.8 * 0.5 + 0.2 * 1 = 0.6
7. p_4^{t+1} = 0.8 * 0.5 + 0.2 * 0 = 0.4

Uzyskany rezultat jest zgodny z przewidywaniami, prawdopodobieństwo wystąpienia jedynki wzrosło w bitach, dla których większość osobników ze zbioru O^t posiada ten bit ustawiony na 1. Natomiast analogicznie zmalało, gdy większość osobników na tym bicie ma wartość 0.

IP (Integer Programing)

Jako algorytm porównawczy zastosujemy metodę programowania dynamicznego, która pozwoli na uzyskanie optimum globalnego.

Zasada działania algorytmu

Tworzymy tablicę **A** o wymiarach **n+1** $\times$ **W+1**, gdzie **n** - liczba przedmiotów, **W** - pojemność plecaka. Komórka **(i,j)** w naszej tablicy będzie przechowywało największą możliwą wartość, przy założeniu, że pojemność plecaka jest nie większa niż **j** oraz bierzemy pod uwagę pierwsze **i** przedmiotów. Dodatkowo tablica **c[n]** zawiera informacje o wartościach przedmiotów, a tablica **w[n]** o wagach przedmiotów.
Komórki o indeksach **(0,j)** oraz **(i, 0)** dla każdego **i** oraz **j** wypełniamy 0. Następnie schemat działania algorytmu wygląda w następujący sposób:

- $A(i,j) = A(i-1,\ j)$, jeśli wiemy, że **i-ty** przedmiot nie zmieści się do plecaka o pojemności **j**
- $A(i,j) = max(A(i-1, j),\ A(i-1,\ j-w[i]) + c[i])$, jeśli **i-ty** przedmiot zmieści się do plecaka o pojemności **j**

Rozwiązanie znajduje się w A**(n, W)** .

Przykład:

- c = [3, 4, 6]
- w = [2, 3, 4]
- W = 6
-

i/j	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	3	3	3	3	3
2	0	0	3	4	4	7	7
3	0	0	3	4	6	7	9

- Otrzymany rezultat to 9
  - Możemy to łatwo zweryfikować:
    - Bierzemy tylko przedmiot nr 1, wartość plecaka będzie równa 3
    - Bierzemy tylko przedmiot nr 2, wartość plecaka będzie równa 4
    - Bierzemy tylko przedmiot nr 3, wartość plecaka będzie równa 6
    - Bierzemy tylko przedmioty nr 1 i 2, wartość plecaka będzie równa 7
    - Bierzemy tylko przedmioty nr 1 i 3, **wartość plecaka będzie równa 9**
    - Bierzemy tylko przedmioty nr 2 i 3, za duża suma wag
    - Bierzemy wszystkie przedmioty, za duża suma wag
- **Przykładowe wyliczenia:**
  - Dla komórki (3,6) korzystamy ze wzoru:
    $A(i,j) = max(A(i-1,j),\ A(i-1,\ j-w[i]) + c[i])$,
    który zwraca wartość $A(i-1,\ j-w[i]) + c[i]$
  - Dla komórki (2, 2) korzystamy ze wzoru: $A(i,j) = A(i-1,\ j)$

- Dla komórki (3, 5) korzystamy ze wzoru:
  $A(i,j) = max(A(i-1,j), A(i-1, j-w[i]) + c[i])$,
  który zwraca wartość $A(i-1,j)$

Plan eksperymentów

Ilość przedmiotów

Dla opisanych algorytmów jak i dla metody heurystycznej opisanej powyżej przeprowadzimy eksperymenty porównujące ich działanie pod względem liczby rozpatrywanych przedmiotów. Obiektem naszego zainteresowania będzie końcowa wartość plecaka uzyskana przez każdą z tych metod, jak i czas wykonania. Możliwe że metody dające wyższy wynik będą znacznie wolniejsze i w rzeczywistości mało praktyczne w porównaniu do szybszych metod.

Przeprowadzimy eksperymenty dla zbiorów o następujących licznościach przedmiotów:
n = [10, 30, 40, 100, 500]

Maksymalna waga plecaka

Dla powyższych liczb elementów przeprowadzimy badania dla różnych rozmiarów plecaka - maksymalnej wagi przedmiotów w plecaku. Plecak o niewielkim rozmiarze, w porównaniu do wagi pojedynczego przedmiotu, łatwiej jest zapełnić ponieważ nie ma wielu możliwości spakowania różnych zestawów przedmiotów. Algorytmy bardziej zaawansowane powinny lepiej sobie radzić przy dużych rozmiarach plecaka.

Na potrzeby naszych eksperymentów przyjmiemy dwa możliwe rozmiary plecaka:
1. W = 2*v gdzie v-maksymalna waga jednego przedmiotu
2. W = połowie sumy wag wszystkich przedmiotów

Parametry algorytmu PBIL

Dla algorytmu można ustawić poniższe parametry:
- M - rozmiar generowanej populacji
- N - liczba wyselekcjonowanych najlepszych chromosomów
- lr - learning rate (pl. wskaźnik uczenia się)

Od parametru M zależy liczba możliwych wygenerowanych przypadków w każdym kroku algorytmu. Są one generowane na podstawie prawdopodobieństwa wystąpienia. Z nich zostaje wybranych N najlepsych, które modyfikują kolejną populację i są poddawane mutacji.

Parametr learning rate odpowiada za przyrost uczenia się, im jest mniejszy tym bardziej “konserwatywny” - mniej podatny na zmiany w jednym kroku algorytmu. Im większa wartość lr tym większy zmiany w populacji zachodzą na skutek aktualizowania populacji.

Metody porównania algorytmów

Dla każdej z założonych liczb elementów przeprowadzimy po cztery eksperymenty, dla każdego algorytmu na tych samych zbiorach danych. Czas wykonania oraz łączną wartość wybranych elementów uśrednimy.

Zaprezentujemy rezultaty na wykresach zależności:

- czasu wykonania od liczby elementów,
- sumy wartości przedmiotów od liczby elementów,
- sumy wartości przedmiotów od czasu wykonania dla konkretnej liczby elementów

Dodatkowo przeprowadzamy eksperymenty na zbiorze danych skorelowanych i porównamy z wynikami dla danych nieskorelowanych.

Zbiory danych

Powyżej opisane eksperymenty planujemy przeprowadzić na zbiorach danych wygenerowanych w sposób przedstawiony w poniższej tabeli. Dla wyróżnionych trzech stopni korelacji danych:

	Poziom korelacji	Waga	Wartość
1	Nieskorelowane	U[1, v]	U[1, v]
2	Częściowo skorelowane	U[1, v]	waga + U[-v/2, v/2]
3	Skorelowane	U[1, v]	waga + v/2

v - maksymalna wartość wagi, dla naszych badań przyjmiemy wartość v = 20
U[a, b] - dane losowane rozkładem ciągłym z przedziału od a do b