Uczenie maszynowe

22.01.2025

Podstawy uczenia ze wzmocnieniem

Uczenie przez wzmacnianie (ang. *RL - Reinforcement Learning*) jest trzecim z głównych nurtów uczenia maszynowego. Jego zadaniem jest interakcja ze środowiskiem na podstawie zbieranych informacji. W uczeniu ze wzmocnieniem wyróżnia się trzy główne elementy:

- **Środowisko** - zadanie lub symulacja, z którym algorytm wchodzi w interakcje. Celem uczenia jest maksymalizacja nagrody zwracanej przez środowisko, czyli nauczenie agenta osiągania w nim najwyższego wyniku (np wygrania największej ilości gier);
- **Agent** - wchodzi w interakcję ze środowiskiem, ucząc się najkorzystniejszego oddziaływania z nim. Za zachowanie agenta odpowiada tzw. **polityka**, czyli funkcja zwracająca akcję (najczęściej jest to sieć neuronowa);
- **Bufor** - magazyn danych przechowujący informacje zebrane przez agenta w trakcie uczenia, które następnie są wykorzystywane do jego wytrenowania.

W ogólnym przypadku agent może nie mieć pełnej informacji o swoim środowisku, jak również precyzyjnego (a nwet żadnego) opisu swoich działań i ich skutków. Inaczej mówiąc, agent zostaje umieszczony w środowisku, którego nie zna, i musi się nauczyć skutecznie w nim działać, aby maksymalizować pewne kryterium, udostępniane mu w formie wzmocnień.

Procesy decyzyjne Markowa

Zakłada się , że probabilistyczny model skutków akcji agenta jako zagadnienie podstawowe jest dyskretnym procesem Markowa (MDP), jednak agent nie zna jego parametrów.

W procesach decyzyjnych Markowa przejścia między stanami zależą od stanu bieżącego i wektora akcji, który jest stosowany do danego systemu. Formalnie MDP jest określony przez:

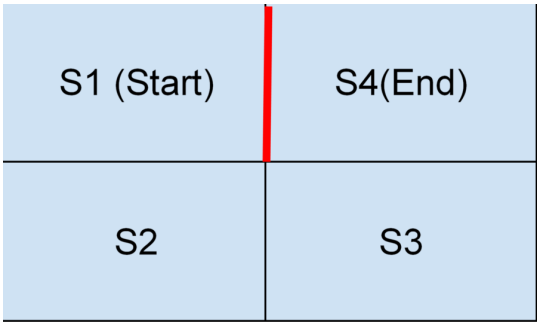
- zbiór możliwych stanów S
- zbiór możliwych akcji A
- nagrody/kary R
- polityka π^*
- wartość, v

Zadanie polega na zbadaniu możliwych stanów s poprzez podejmowanie działań A i wymyśleniu optymalnej polityki π^*, która maksymalizuje wartości v na podstawie nagród i kar R.

MDPtoolbox

Pakiet `MDPToolbox` wykorzystuje procesy Markowa do nauki wzmacniania.

1. Aby zdefiniować elementy uczenia RL, należy przypisać etykiety do każdego ze stanów w macierzy nawigacyjnej. Na początek rozważmy macierz 2x2:



`S1` jest stanem początkowym, `S4` - końcowym. Nie można przejść bezpośrednio z `S1` do `S4` z powodu ściany. Zatem z `S1` można tylko przejść do `S2` lub pozostać w `S1`.

Stąd macierz `w dół` będzie miała niezerowe prawdopodobieństwa tylko dla `S1` i `S2` w pierwszym wierszu. Podobnie możemy zdefiniować prawdopodobieństwa dla każdej akcji w każdym stanie.

2. Teraz możemy zdefiniować zbiór akcji: `up`, `down`, `left`, `right` dla macierzy stanów 2x2. Uwaga: jest to macierz prawdopodobieństw, gdzie w każdym wierszu ich suma musi wynosić 1:

```
# Up Action
up=matrix(c( 1, 0, 0, 0,
             0.7, 0.2, 0.1, 0,
             0, 0.1, 0.2, 0.7,
             0, 0, 0, 1),
          nrow=4,ncol=4,byrow=TRUE)

# Down Action
down=matrix(c(0.3, 0.7, 0, 0,
              0, 0.9, 0.1, 0,
              0, 0.1, 0.9, 0,
              0, 0, 0.7, 0.3),
            nrow=4,ncol=4,byrow=TRUE)

# Left Action
left=matrix(c( 0.9, 0.1, 0, 0,
               0.1, 0.9, 0, 0,
               0, 0.7, 0.2, 0.1,
               0, 0, 0.1, 0.9),
            nrow=4,ncol=4,byrow=TRUE)

# Right Action
right=matrix(c( 0.9, 0.1, 0, 0,
                0.1, 0.2, 0.7, 0,
                0, 0, 0.9, 0.1,
                0, 0, 0.1, 0.9),
```

```
             nrow=4,ncol=4,byrow=TRUE)

# Combined Actions matrix
Actions=list(up=up, down=down, left=left, right=right)
```

3. Zdefiniowanie kar i nagród

Jedyną karą jest mała kara za każdy dodatkowy krok. Określmy ją jako -1.

Nagroda jest otrzymywana po osiągnięciu stanu `S4`. Ustawmy wagę na poziomie +10. W ten sposób możemy utworzyć macierz R:

```
Rewards=matrix(c( -1, -1, -1, -1,
                  -1, -1, -1, -1,
                  -1, -1, -1, -1,
                  10, 10, 10, 10),
               nrow=4,ncol=4,byrow=TRUE)
```

4. Następnie algorytm musi znaleźć optymalną politykę i jej wartość. W tym celu posłużymy się funkcją `mdp_policy_iteration()`, która wymaga ustawienia akcji, nagród oraz rabatu jako danych wejściowych do obliczenia wyników. Rabat jest używany do zmniejszenia wartości bieżącej nagrody lub kary w miarę wykonywania każdego z kroków.

```
library(MDPtoolbox)
```

```
## Loading required package: Matrix
```

```
## Loading required package: linprog
```

```
## Loading required package: lpSolve
```

```
solver=mdp_policy_iteration(P=Actions, R=Rewards, discount = 0.1)
```

Wynik daje nam politykę, wartość na każdym kroku oraz dodatkowo liczbę iteracji i czas potrzebny na wykonanie. Jak wiemy, polityka powinna określać właściwą ścieżkę do osiągnięcia stanu końcowego S4. Używamy funkcji polityki, aby poznać macierze używane do definiowania polityki, a następnie nazwy z listy działań.

```
solver$policy
```

```
## [1] 2 4 1 1
```

```
names(Actions)[solver$policy]
```

```
## [1] "down"  "right" "up"    "up"
```

Wartości są zawarte w `v` i pokazują nagrodę na każdym kroku.

```
solver$V
```

```
## [1] -1.106604 -1.048661 -0.237458 11.111111
```

Iteracje i czas mogą być użyte do śledzenia liczby iteracji oraz czasu, aby monitorować złożoność.

```
solver$iter
```

```
## [1] 2
```

```
solver$time
```

```
## Time difference of 0.03915 secs
```

5. Spróbuj wykonać powyższe kroki dla macierzy 3x3.