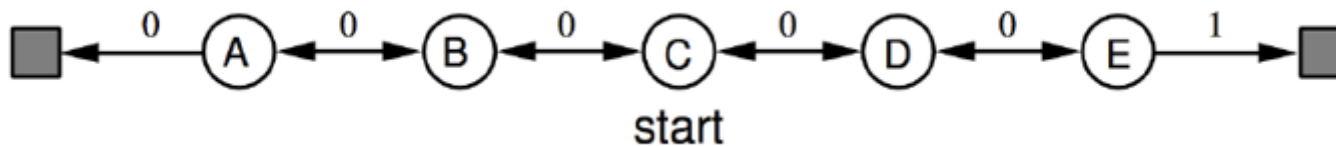


▼ Opis problemu

Poniżej przedstawiona jest implementacja algorytmu **TD(0)** do problemu **random walk**.

Diagram:



Przypomnijmy podstawowe fakty:

- **Kółkami** oznaczone są **stany nieterminalne**, **kwadratami** stany **terminalne**.
- Wszystkie **epizody rozpoczynają** się w środkowym stanie **C**.
- W dowolnym stanie nieterminalnym **prawdopodobieństwa ruchu w lewo i ruchu w prawo** są równe i wynoszą **0.5** (to jest polityka **pi**).
- Nad strzałkami widoczne są **wartości nagród**. Tylko przejście ze stanu **E** do **stanu terminalnego po prawej stronie** skutkuje nagrodą **R=1**. Poza tym wszystkie nagrody wynoszą **0**.

▼ Algorytm

Implementujemy algorytm:

Tabular TD(0) for estimating v_π

Input: the policy π to be evaluated

Algorithm parameter: step size $\alpha \in (0, 1]$

Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(\text{terminal}) = 0$

Loop for each episode:

 Initialize S

 Loop for each step of episode:

$A \leftarrow$ action given by π for S

 Take action A , observe R, S'

$V(S) \leftarrow V(S) + \alpha[R + \gamma V(S') - V(S)]$

$S \leftarrow S'$

 until S is terminal

▼ Implementacja

Zacznijmy od importu potrzebnej biblioteki:

```
import numpy as np
```

Stany oznaczamy cyframi począwszy od lewej strony:

- Stan terminalny z lewej strony - 0
- Stan A - 1

- Stan B - **2**
- Stan C - **3**
- Stan D - **4**
- Stan E - **5**
- Stan terminalny z *prawej strony* - **6**

Wartości początkowe **V** dla wszystkich stanów:

```
V = np.zeros(7)
print(V)
```

```
[0. 0. 0. 0. 0. 0. 0.]
```

```
epochs = 1000 #liczb epok czyli to ile epizodów uwzględnimy w wyliczeniu V (im więcej tym lepiej)
alpha = 0.1
gamma = 1.0 #bez zdyskontowania
#V = np.zeros(env.env.nS)
```

```
for i in range(epochs):
    state = 3 #stan początkowy w każdym epizodzie
    while True:
        action = np.random.randint(2)
        if action==0:
            next_state = state - 1
        else:
            next_state = state + 1
        if next_state == 6:
            R=1
        else:
            R=0
        V[state] = V[state] + alpha * (R + gamma*V[next_state] - V[state])
        state = next_state
```

```
if state == 0 or state == 6:  
    break
```

#UWAGA: do losowania akcji (w prawo lub w lewo) można wykorzystać metodę `np.random.randint(2)`, która zwraca liczbę 0 lub 1.

Wypiszmy wyliczone wartości **V**:

```
print(V)  
  
[0.          0.17640368 0.34788477 0.48064029 0.7367174  0.83793971  
 0.          ]
```

Wartości **teoretyczne** podane w wykładzie (**slajd 19**):

```
print([0,1/6,2/6,3/6,4/6,5/6,0])  
  
[0, 0.16666666666666666, 0.3333333333333333, 0.5, 0.6666666666666666, 0.8333333333333334, 0]
```

▼ Polecenie

Przetestuj działanie powyższego algorytmu dla **3 wybranych par wartości parametrów alpha i gamma**. Podaj wyliczone wartości **V**. Skomentuj uzyskane rezultaty.

DO UZUPEŁNIENIA

✓ 0 s ukończono o 12:49

