Zaawansowane Metody Inteligencji Obliczeniowej

Lab 6: Temporal-Difference Learning

Michał Kempka Marek Wydmuch

8 kwietnia 2021

Fundusze Europejskie Polska Cyfrowa
Rzeczpospolita Polska
Unia Europejska Europejski Fundusz Rozwoju Regionalnego

"Akademia Innowacyjnych Zastosowań Technologii Cyfrowych (AI Tech)",
projekt finansowany ze środków Programu Operacyjnego Polska Cyfrowa POPC.03.02.00-00-0001/20

1 Wprowadzenie

Metody Monte-Carlo, omówione na poprzednich ćwiczeniach pozwalają nam na nauczenie się wartości polityki i optymalnej polityki dla danego środowiska bez znajomości modelu. Metody te bazują na stopniowym polepszaniu estymacji v_π na podstawie zysków zdobytych w czasie całego epizodu. Takie podejście może być bardzo wolne w przypadku gdy epizody są długie (lub wcale nie ma epizodów!). By zaadresować ten problem zaproponowano tzw. **Temporal-Difference Learning (TDL)**, który łączy zalety metod Monte-Carlo (estymacje wyciągane na podstawie próbkowania ze środowiska) i programowania dynamicznego (rekurencyjne polepszanie estymacji na podstawie dotychczasowych estymacji dla następników - ang. bootstrapping). TDL w przeciwieństwie do MC, nie potrzebuje całego epizodu by polepszyć swoje estymacje, lecz może wykonywać uaktualnienia w każdym kroku na podstawie błędu TD (ang. Temporal-Difference Error (TDE)) oznaczany również jako δ_t:

$$\delta_t = \overbrace{V(S_t)}^{\substack{\text{estymacja} \\ \text{dla obecnego stanu}}} - (R_{t+1} + \gamma \overbrace{V(S_{t+1})}^{\substack{\text{estymacja} \\ \text{dla następnika}}}) \tag{1}$$

Jak widać z Równania 1 TDE odnosi się do różnicy między estymacją dla obecnego stanu $v(s_t)$, a estymacją dla stanu następnego połączoną z właśnie otrzymaną nagrodą. TDE w różnych wariantach i formach będzie spotykać w wielu algorytmach uczenia ze wzmocnieniem.

2 TD(0)

W każdym kroku otrzymujemy krotkę S, S', R i załóżmy, że chcielibyśmy wyestymować v_π dla każdego stanu i zadanej polityki poprzez rozwiązanie klasycznego problemu regresji. Chcemy oczywiście by TDE był jak najbliżej zera więc sformułujemy problem jako minimalizację uśrednionego (po stanie i czasie) błędu kwadratowego:

$$\forall_{s \in \mathcal{S}} \min \mathbb{E}\left[\frac{1}{2} TDE_t(S)^2\right] = \min \mathbb{E}\left[\frac{1}{2}\Big(V(S) - (R + \gamma V(S')\Big)^2\right] \tag{2}$$

2.1 Gradient Descent

By rozwiązać problem z równania 2 możemy użyć klasycznej metody Online Gradient Descent (OGD). Wyprowadź regułę aktualizacji aktualnej estymacji $V(S)$ dla pojedynczej krotki S, S', R w ogólnej formie oraz zakładając, że $V(S)$ jest stablicowane. We wzorze użyj współczynnika dyskontowego γ i wielkości kroku α (ang. step-size/learning rate).

Rozwiązanie

$$
\begin{aligned}
V(S) \leftarrow V(S) - \alpha \nabla \left(\frac{1}{2} TDE(S, S', R)^2 \right) \\
= V(S) - \alpha \nabla \left(\frac{1}{2} \Big(V(S) - (R + \gamma V(S')) \Big)^2 \right) \\
= V(S) - \alpha \Big(V(S) - (R + \gamma V(S')) \Big)
\end{aligned}
$$

Tak oto wyprowadziliśmy algorytm TD(0) (one-step (nazwany tak zdaje się z historycznych powodów). Należy jednak zauważyć, że ogólna formulacja pozwala nam na użycie nie tylko tabelarycznej wersji $V(S)$, lecz dowolnej funkcji z parametrami $V(S, \theta)$, gdzie gradient będzie liczony względem tych parametrów. Dodatkowo, zamiast standardowego OGD/SGD możemy użyć dowolnych rozszerzeń i usprawnień np. Adam, RMSProp - wystarczy zdefiniować funkcję straty (ang. loss function).

Algorytm 1: Ewaluacja polityki za pomocą algorytmu TD(0)

Inicjalizacja:
Dowolnie zainicjalizowane $V(s)$ dla wszystkich $s \in \mathcal{S}$
$V(s) \leftarrow 0$ jeśli s jest stanem terminalnym.
Dana dowolna polityka π.

repeat
  if S *nieustawiony lub terminalny* **then**
    rozpocznij nowy epizod i zainicjalizuj
    $S \leftarrow S_0$
  Wykonaj akcję $\pi(S)$, zaobserwuj nagrodę R i następnik S'.
  $V(S) \leftarrow V(S) - \alpha(V(S) - (R + \gamma v(S'))$
  $S \leftarrow S'$
until *warunek stopu*;

2.2 Monte Carlo Error, a TD Error

Wyraź błąd Monte Carlo $V(S_t) - G_t$ przy pomocy błędu TD δ_t, zakładając, że estymaty V są stałe.

Rozwiązanie

$$
\begin{aligned}
V(S_t) - G_t &= V(S_t) - (R_{t+1} + \gamma G_{t+1} + \gamma V(S_{t+1}) - \gamma V(S_{t+1})) \\
&= V(S_t) - (R_{t+1} + \gamma V(S_{t+1})) - \gamma G_{t+1} + \gamma V(S_{t+1}) \\
&= \delta_t + \gamma (V(S_{t+1}) - G_{t+1}) \\
&= \delta_t + \gamma \delta_{t+1} + \gamma^2 (V(S_{t+2}) - G_{t+2}) \\
&= \delta_t + \gamma \delta_{t+1} + \cdots + \gamma^{T-t} (V(S_T) - G_T) \\
&= \delta_t + \gamma \delta_{t+1} + \cdots + \gamma^{T-t} (0 - 0) \\
&= \sum_{i=t}^{T-1} \gamma^{i-t} \delta_i
\end{aligned}
$$

Pytanie: Algorytm TD(0) uczy się wartości stanów dla zadanej polityki. Jak zmodyfikować go by umiał polepszyć politykę, lub znaleźć politykę optymalną?

Odpowiedź: Pierwszym problemem jest fakt, że nie mamy modelu przejść - prawdopodobieństw i nagród, które są potrzebne do wyznaczenia, która akcja jest najlepsza.

Mając model możemy decydować, którą akcję wybrać - na przykład zachłannie, tę która maksymalizuje przewidywane zyski (według optymalnego równania Bellmana). Niestety natrafiamy tu na pewien problem - nasza estymacja $V(S)$ silnie zależy od polityki więc może się okazać, że nigdy nie natrafimy na optymalną ścieżkę bo nasza polityka 'będzie miała pecha'. W tym momencie, zamiast $V(S)$ algorytm może uczyć się $Q(s, a)$. Tak oto otrzymujemy algorytm zwany q-learningiem.

3 SARSA

Algorytm 2: Pseudokod dla algorytmu SARSA

Inicjalizacja:
Dowolnie zainicjalizowane $Q(s,a)$ dla wszystkich $s \in \mathcal{S}, a \in \mathcal{A}$
$Q(s,\cdot) \leftarrow 0$ jeśli s jest stanem terminalnym
Dana dowolna polityka π (np. ϵ-greedy) zgodna z początkowym Q

repeat
  if S *nieustawiony lub terminalny* **then**
    Rozpocznij nowy epizod
    $S \leftarrow S_0$
    $A \leftarrow \pi(S)$
  Wykonaj akcję A, zaobserwuj nagrodę R i następnik S'
  $A' \leftarrow \pi(S')$
  $Q(S,A) \leftarrow Q(S,A) - \alpha(Q(S,A) - (R + \gamma Q(S',A'))$
  Zaktualizuj π zgodnie z Q
  $S \leftarrow S'$
  $A \leftarrow A'$
until *WarunekStopu*;

Wykorzystajmy teraz TD(0) do wyznaczenia polityki tak jak to robiliśmy przy analizie Monte Carlo. Jednym z najprostszych z tego typów algorytmów jest SARSA (S_t,A_t,R_{t+1},S_{t+1},A_{t+1}) zaprezentowana w Algorytmie 2.

Pytanie: Czy Algorytm 2 SARSA jest on-policy czy off-policy)?

Odpowiedź: On-policy, estymacja wartości Q jest aktualizowana używając wartości Q od stanu i akcji wybranej przez aktualnie używaną politykę, tym samym wartości Q dla wszystkich par stan,akcja są estymowane z założeniem, że działanie zgodnie z aktualną polityką będzie kontynuowane.

Pytanie: Czy Algorytm 2 SARSA wymaga modelu świata czy jest model-free?

Odpowiedź: Jest model-free, nie wymaga modelu świata.

4 Q-learning

Algorytm 3: Pseudokod dla algorytmu Q-Learning

Inicjalizacja:
Dowolnie zainicjalizowane $Q(s,a)$ dla wszystkich $s \in \mathcal{S}, a \in \mathcal{A}$
$Q(s,\cdot) \leftarrow 0$ jeśli s jest stanem terminalnym
Dana dowolna polityka π zgodna z początkowym Q

repeat
  if S *nieustawiony lub terminalny* **then**
    Rozpocznij nowy epizod
    $S \leftarrow S_0$
  $A \leftarrow \pi(S)$
  Wykonaj akcję A, zaobserwuj nagrodę R i następnik S'
  $Q(S,A) \leftarrow Q(S,A) - \alpha(Q(S,A) - (R + \gamma \max_{a \in \mathcal{A}} Q(S',a))$
  Zaktualizuj π zgodnie z Q
  $S \leftarrow S'$
until *warunek stopu*;

Jednym z najbardziej znanych i wpływowych algorytmów TD jest Q-learning zaprezentowany w Algorytmie 3.

Pytanie: Q-learning klasyfikuje się jako algorytm off-policy. Co to oznacza?

Odpowiedź: Q-learning aktualizuje wartości Q używając Q od następnego stanu i zachłannej akcji, tym samym estymuje on całkowitą dyskontowaną przyszłą nagrodę dla każdej pary stan, akcja zakładają zachłanną politykę, mimo, że algorytm nie podąża za zachłanną polityką.

Pytanie: Załóżmy, że polityka π jest całkowicie zachłanna. Czy w takim wypadku Q-learning będzie odpowiadać algorytmowi SARSA (będzie wybierać te same akcje i dokonywać tych samych aktualizacji wartości Q)?

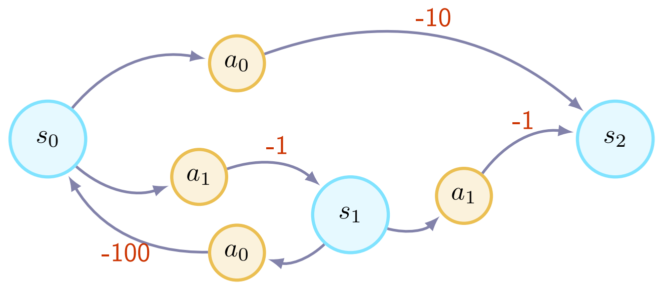
Odpowiedź: Zwróćmy uwagę, że wybór kolejnej akcji w metodzie SARSA obywa się przed aktualizacja wartości Q, a w Q-learning po. Może to czasem prowadzić do sytuacji, że nastąpi zmiana zachłannej akcji dla polityki i wtedy SARSA i Q-learning wykonają inną zachłanną akcję.

Pytanie: Q-learning nie potrzebuje znać modelu świata (ang. model-free). Czemu nie jest tak w przypadku analogicznego algorytmu uczącego się optymalnej polityki na podstawie $V(S)$

Odpowiedź: Mając tylko wartości $V(S)$ nie jesteśmy w stanie wyznaczyć polityki, gdyż brakuje nam prawdopodobieństw do wyliczenia wartości akcji w każdym stanie a tym podstawy do wybrania najlepszych akcji.

4.1 SARSA vs Q-learning

Zasymuluj działanie algorytmu first visit Monte Carlo, SARSA oraz Q-learning z polityką ϵ-greedy gdzie $\epsilon = 0.5$ dla środowiska przedstawionego na Rysunku 1. Początkowo $Q(s, a) = 0$ dla wszystkich $s \in \mathcal{S}, a \in \mathcal{A}$, $\gamma = 0.5$ i $\alpha = 0.5$.



Rysunek 1: Diagram przedstawiający proste MDP z trzema stanami. Nagrody i następniki są deterministyczne, stan początkowy to s_0 a terminalny s_2.

Rozwiązanie

* - oznacza, że akcja była zachłanna. W wypadku remisów brana jest pierwsza akcja w kolejności.

4.1.1 SARSA

i	S	A	R	S'	A'	$Q(s_0, a_0)$	$Q(s_0, a_1)$	$Q(s_1, a_0)$	$Q(s_1, a_1)$	$Q(s_2, *)$
0	-	-	-	-	-	0.00	0.00	0.00	0.00	0.00
1	s_0	a_0^*	-10	s_2	a_0^*	-5.00	0.00	0.00	0.00	0.00
2	s_0	a_1^*	-1	s_1	a_1	-5.00	-0.50	0.00	0.00	0.00
3	s_1	a_1	-1	s_2	a_0^*	-5.00	-0.50	0.00	-0.50	0.00
4	s_0	a_1^*	-1	s_1	a_0^*	-5.00	-0.75	0.00	-0.50	0.00
5	s_1	a_0^*	-100	s_0	a_1^*	-5.00	-0.75	-50.19	-0.50	0.00
6	s_0	a_1^*	-1	s_1	a_0	-5.00	-13.42	-50.19	-0.50	0.00
7	s_1	a_0	-100	s_0	a_0^*	-5.00	-13.42	-76.34	-0.50	0.00
8	s_0	a_0^*	-10	s_2	a_0^*	-7.50	-13.42	-76.34	-0.50	0.00
9	s_0	a_0^*	-10	s_2	a_0^*	-8.75	-13.42	-76.34	-0.50	0.00
10	s_0	a_0^*	-10	s_2	a_0^*	-9.38	-13.42	-76.34	-0.50	0.00

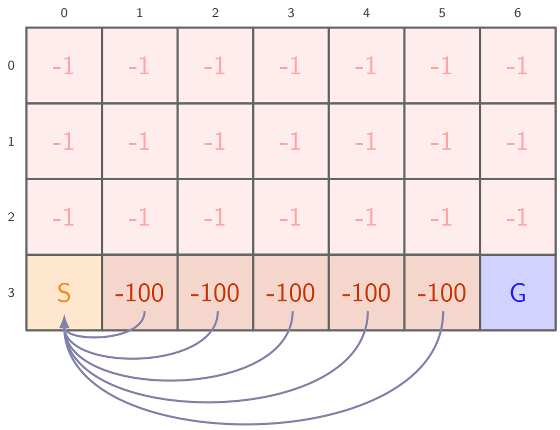
4.1.2 Q-learning

i	S	A	R	S'	$Q(s_0,a_0)$	$Q(s_0,a_1)$	$Q(s_1,a_0)$	$Q(s_1,a_1)$	$Q(s_2,*)$
0	-	-	-	-	0.00	0.00	0.00	0.00	0.00
1	s_0	a_1	-1	s_1	0.00	-0.50	0.00	0.00	0.00
2	s_1	a_0^*	-100	s_0	0.00	-0.50	-50.00	0.00	0.00
3	s_0	a_0^*	-10	s_2	-5.00	-0.50	-50.00	0.00	0.00
4	s_0	a_0	-10	s_2	-7.50	-0.50	-50.00	0.00	0.00
5	s_0	a_1^*	-1	s_1	-7.50	-0.75	-50.00	0.00	0.00
6	s_1	a_1^*	-1	s_2	-7.50	-0.75	-50.00	-0.50	0.00
7	s_0	a_1^*	-1	s_1	-7.50	-1.00	-50.00	-0.50	0.00
8	s_1	a_1^*	-1	s_2	-7.50	-1.00	-50.00	-0.75	0.00
9	s_0	a_1^*	-1	s_1	-7.50	-1.19	-50.00	-0.75	0.00
10	s_1	a_1^*	-1	s_2	-7.50	-1.19	-50.00	-0.88	0.00

4.2 Chodzenie po klifie

Rozważmy proste środowisko chodzenia po klifie przedstawione na Rysunku 2. Rozpoczynamy w stanie S, środowisko kończy się po dotarciu do stanu G. Możliwe akcje to ruch w górę, dół, lewo, prawo o ile istnieje stan we wskazanym kierunku. Kara -1 występuje na wszystkich przejściach poza stanem G i stanem na dole, który dają karę -100 i cofa do stanu S (co symbolizuje spadnięcie z klifu).

Rozważmy algorytm SARSA oraz Q-learning z polityką ϵ-greedy gdzie $\epsilon > 0.1$. Jaką politykę wyznaczy algorytm SARASa, a jaką Q-learning dla tego środowiska? Co należałby zrobić aby oba algorytmy wyznaczyły optymalną politykę?



Rysunek 2: Środowisko chodzenia po klifie

5 TD(λ)

Istnieje też sposób na połączenie programowania dynamicznego i metod MC w bardziej gładki sposób. Metody MC uaktualniają estymaty na podstawie nagród zebranych w czasie całego epizodu, a TD(0) używa tylko najbliższej nagrody i obecnej estymacji dla stanu następnego (bootstrapping). Możemy jednak przechodzić płynnie miedzy tymi dwoma rozwiązaniami poprzez sumowanie nagród z przyszłych n akcji i bootstrapowaniu na podstawie stanu n kroków późniejszego (ostatnie n epizodów użyje dokładnego zwrotu jak w MC). n wyznacza nam zatem pewien horyzont czasowy dla aktualizacji programowania dynamicznego.

$$G_t^{t+n}(S_t) = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^n V(S_{t+n}) \tag{3}$$

Dla $n = 1$ otrzymamy TD(0) zaś $n = T$ da nam metody Monte Carlo (gdzie T to ostatni stan - metody MC w opisanej formie nie będą działać na nieskończonych epizodach).

Pytanie: Co jeśli spróbujemy zastosować podobne rozszerzenie do Q-learningu? **Odpowiedź:** Okaże się, że w momencie gdy będziemy wykonywać akcje eksploracyjne, przestanie on być off-policy, a zacznie działać jak bardziej jak SARSA. Pojawił się za to algorytm $Q(\lambda)$, który przeprowadza bootstrap w momencie wykonania pierwszej akcji eksploracyjnej.

W ogólności, można wykorzystać dowolnie ważoną średnią estymacji z różnym horyzontem czasowym. Użycie $\lambda^{n-1}; \lambda \in [0, 1]$ jako wag da nam algorytm TD(λ):

$$L_t = (1 - \lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} G_t^{t+n}(S_{t+n}) + \lambda^{T-t-1} G_t \tag{4}$$

Jak widać użycie $\lambda = 0$ daje nam TD(0) zaś $\lambda = 1$ daje MC.

Literatura

[1] Russell, S. and Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. Prentice Hall, third edition.

[2] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. The MIT Press, second edition.

Fundusze Europejskie
Polska Cyfrowa
Rzeczpospolita Polska
Unia Europejska
Europejski Fundusz Rozwoju Regionalnego

"Akademia Innowacyjnych Zastosowań Technologii Cyfrowych (AI Tech)",
projekt finansowany ze środków Programu Operacyjnego Polska Cyfrowa POPC.03.02.00-00-0001/20