Uczenie Maszynowe — Laboratorium 01

Radosław Łazarz

2 listopada 2022

1 Co warto wiedzieć?

W ramach tego laboratorium naszym zadaniem będzie dokończenie implementacji i przetestowanie na prostym problemie poniższego algorytmu (n-krokowego sterowania SARSA w wariancie poza-polityką).

Off-policy n-step Sarsa for estimating $Q \approx q_*$ or q_π

Input: an arbitrary behavior policy b such that $b(a|s) > 0$, for all $s \in \mathcal{S}, a \in \mathcal{A}$
Initialize $Q(s,a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}$
Initialize π to be greedy with respect to Q, or as a fixed given policy
Algorithm parameters: step size $\alpha \in (0,1]$, a positive integer n
All store and access operations (for S_t, A_t, and R_t) can take their index mod $n+1$

Loop for each episode:
  Initialize and store $S_0 \neq$ terminal
  Select and store an action $A_0 \sim b(\cdot|S_0)$
  $T \leftarrow \infty$
  Loop for $t = 0, 1, 2, \ldots$:
  | If $t < T$, then:
  | Take action A_t
  | Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
  | If S_{t+1} is terminal, then:
  | $T \leftarrow t+1$
  | else:
  | Select and store an action $A_{t+1} \sim b(\cdot|S_{t+1})$
  | $\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)
  | If $\tau \geq 0$:
  | $\rho \leftarrow \prod_{i=\tau+1}^{\min(\tau+n, T-1)} \frac{\pi(A_i|S_i)}{b(A_i|S_i)}$ $(\rho_{\tau+1:\tau+n})$
  | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n, T)} \gamma^{i-\tau-1} R_i$
  | If $\tau + n < T$, then: $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$ $(G_{\tau:\tau+n})$
  | $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha\rho\left[G - Q(S_\tau, A_\tau)\right]$
  | If π is being learned, then ensure that $\pi(\cdot|S_\tau)$ is greedy wrt Q
  Until $\tau = T - 1$

2 Jaki problem mamy przed sobą?

Problem do rozwiązania to znalezienie sposobu na możliwie najszybszy przejazd po zadanym zakręcie.

- Zakręt specyfikowany jest jako wejściowa mapa, na której:
  - pola białe oznaczają miejsca dopuszczone do ruchu (asfalt);
  - pola zielone oznaczają linie startu;
  - pola czerwone oznaczają dopuszczalne pozycje końcowe.

- Każdy epizod zaczyna się w losowym polu startowym.
- Każda akcja skutkuje karą -1 (minęła jedna jednostka cennego czasu!).
- Wyjątkiem jest sytuacja, gdy na skutek akcji samochód znajdzie się w czerwonej strefie końcowej — wtedy kara wynosi 0 (nie ma jej).
- Stan samochodu charakteryzują cztery zmienne — jego położenie w osiach X i Y oraz prędkość w tych samych kierunkach.
- Każda akcja polega na zmianie jednej lub obu składowych prędkość o 1, 0 lub -1.
- Wynikowa prędkość nie może wynieść 0 w obu kierunkach (za wyjątkiem pozycji startowej).
- Prędkość w osi X musi zawierać się w zakresie $[0, 3]$, zaś w osi Y w zakresie $[-3, 3]$.

- Za każdym razem istnieje niewielkie prawdopodobieństwo, że układ sterowania zawiedzie i wybrana akcja nie odniesie skutku (nie zmieni się prędkość).
- Gdy samochód wyjedzie poza dozwolony obszar jego prędkość jest zerowana, a następnie zostaje on ponownie przeniesiony na losową pozycję startową.

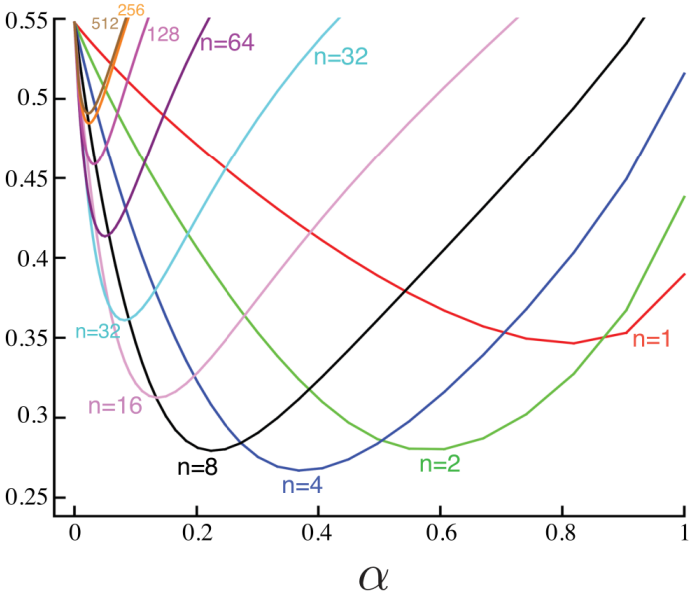
3 Co jest już gotowe?

Załącznikiem do tej instrukcji jest paczka o następującej strukturze:

- `problem.py` — zawiera klasy specyfikujące rozważany problem;
- `solution.py` — zawiera klasę implementującą rozwiązanie (**z lukami do uzupełnienia!**);
- `utils.py` — kilka pomocniczych funkcji do wizualizowania wyników;
- `corners` — katalog zawierający specyfikacje zakrętów o rosnącym stopniu trudności (od `b` do `d`);
- `plots` — katalog na wynikowe wizualizacje;
- `requirements.txt` — specyfikacja wymaganych bibliotek.

4 Co należy zrobić?

- Uruchomić aplikację dla testowego losowego algorytmu i sprawdzić, czy wszystko działa.
- Uzupełnić luki w kodzie właściwego rozwiązania (są oznaczone jako `TODO`).
- Sprawdzić, czy algorytm uczy się przejazdu przez prosty zakręt testowy `corner_b`.
- Jeżeli algorytm z sukcesem uczy się przejazdu przez `corner_b`, to zwizualizować kilka przykładowych tras dla znalezionej optymalnej polityki.
  - Co trzeba w tym celu zmienić? Pamiętajmy o tym, że w trakcie uczenia się do wyboru akcji używana jest nieoptymalna polityka eksplorująca.
- Jeżeli odnieśliśmy sukces, to sprawdźmy działanie dla trudniejszego wariantu `corner_c`.
  - W tym przypadku zbadajmy wpływ parametru α i liczby kroków n na wyniki, robiąc studium parametryczne podobne do tego poniżej.



- Na koniec dla optymalnego zestawu parametrów odpalmy najtrudniejszy przypadek `corner_d` i oceńmy wyniki (tu uczenie może już trwać dość długo!).
- Otrzymane wyniki przedstawiamy w zwartym i czytelnym raporcie, wrzucamy wraz z kodem rozwiązania w odpowiednie miejsce na platformie UPEL.