Inverse reinforcement learning – definicja i metody rozwiązywania

Jan Karwowski

Zakład Sztucznej Inteligencji i Metod Obliczeniowych
Wydział Matematyki i Nauk Informacyjnych PW

19 X 2022

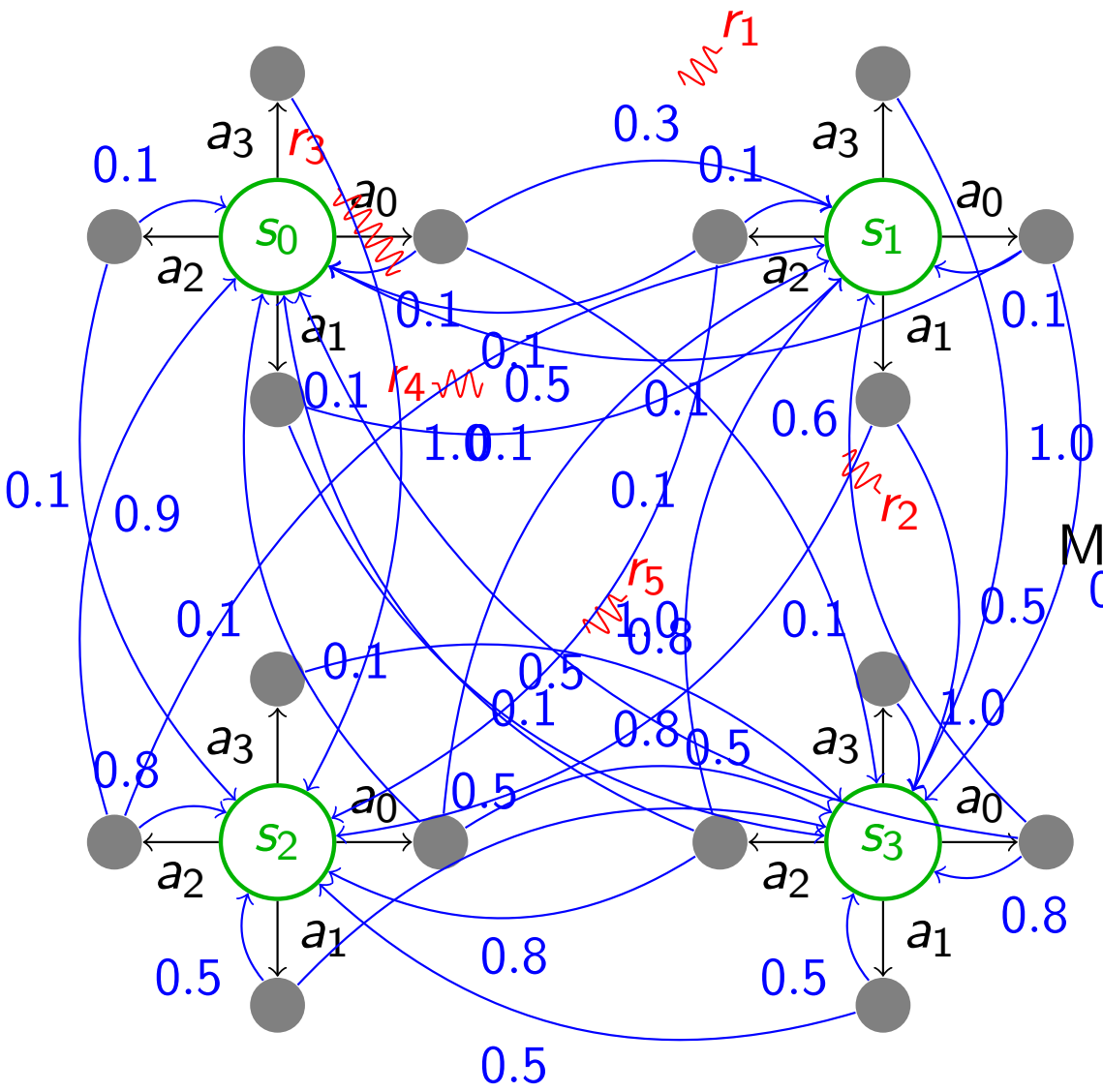
Informacje wstępne — Proces Markova

MDP

$$MDP = (S, A, T, \gamma, D, R)$$

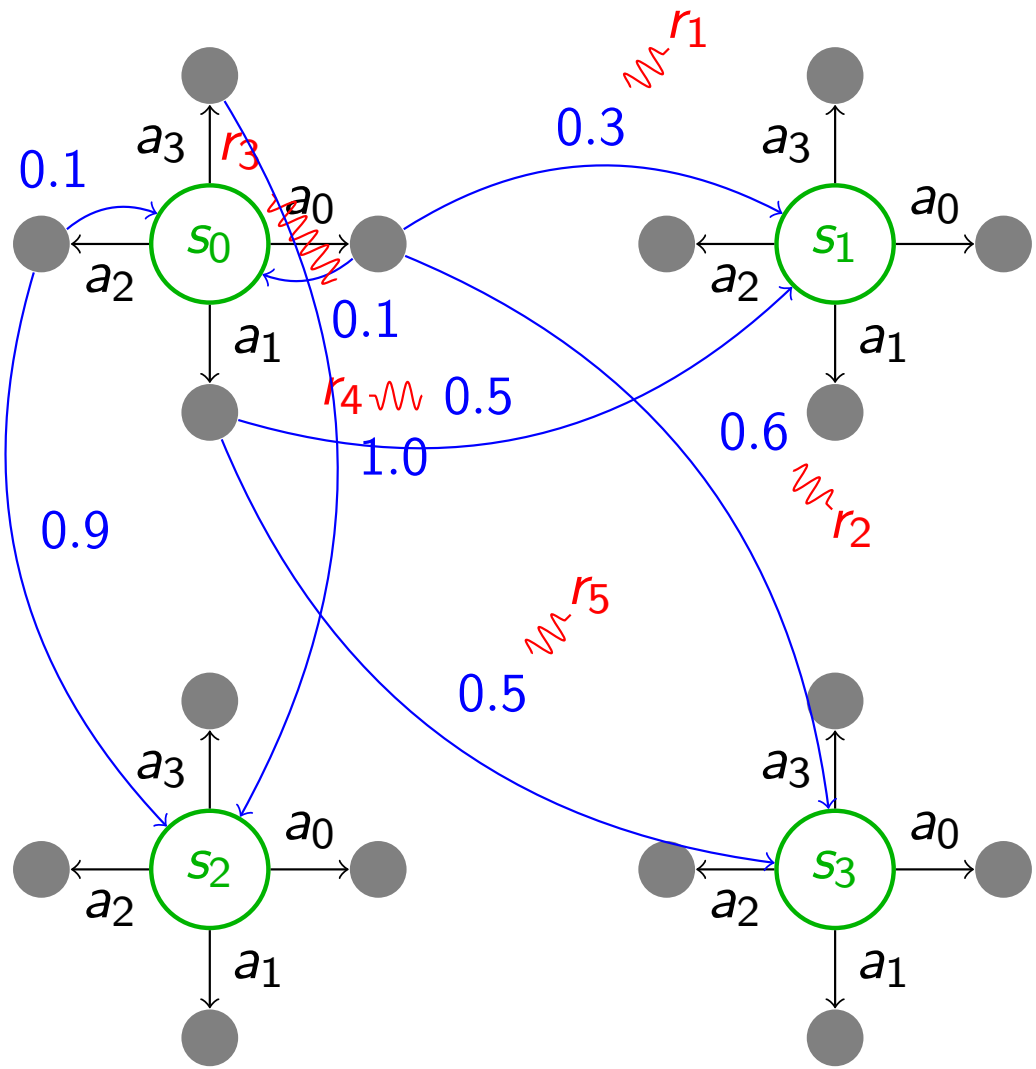
- S — zbiór stanów
- A — zbiór akcji
- $T = \{P_{sa}\}$ — zbiór wektorów prawdopodobieństw przejść do stanu po wybraniu akcji a w stanie s
- γ — współczynnik dyskontowania
- D — rozkład prawdopodobieństwa stanu początkowego
- $R : S \to \mathbb{R}$, odpowiednio regularna. Przyjmijmy, że ograniczona. Wypłata uzyskiwana, za każdym razem, gdy zostanie osiągnięty stan S.

MDP — Rysunek



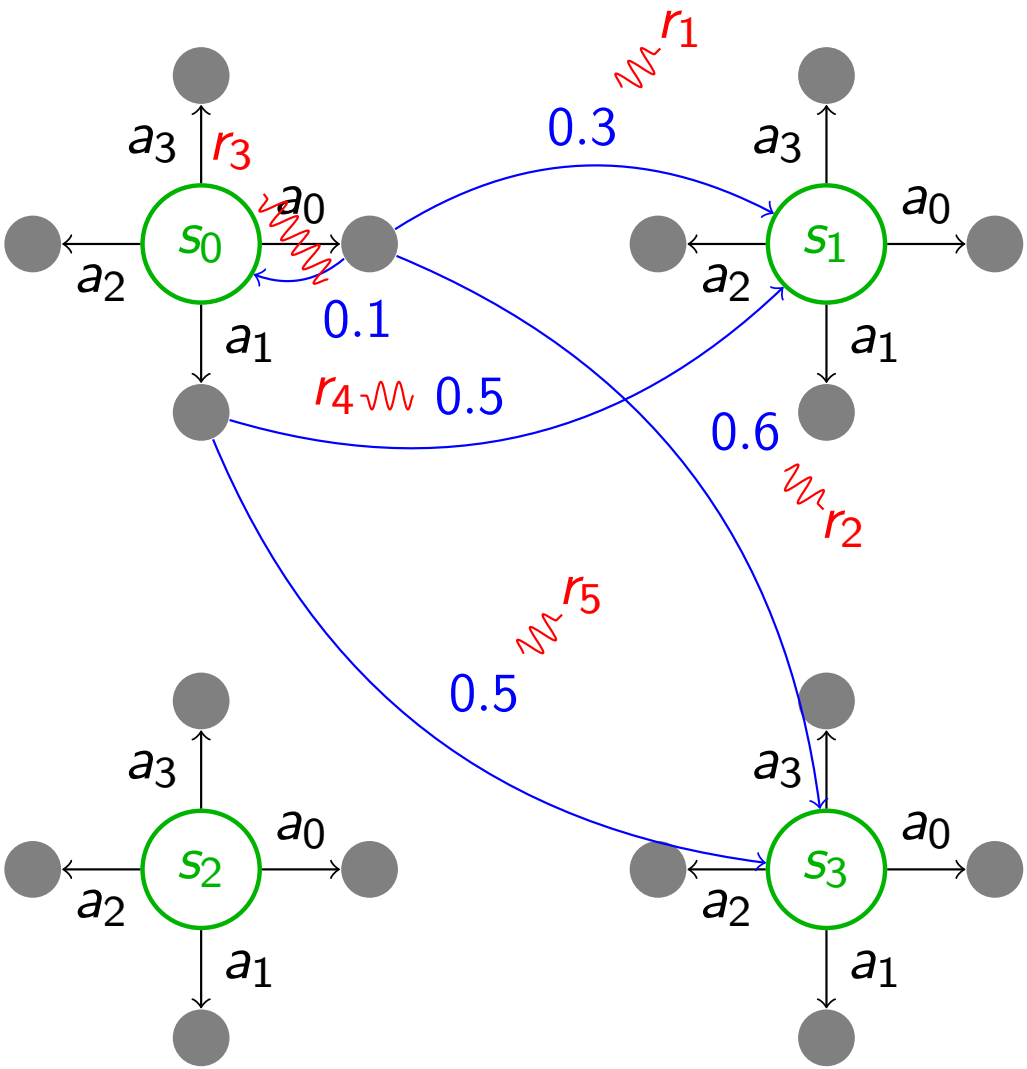
MDP=(S , A , T , γ , D , R)

MDP — Rysunek



MDP=(S , A , T , γ , D , R)

MDP — Rysunek



$\mathrm{MDP}=(\,\mathrm{S}\,,\,\mathrm{A}\,,\,\mathrm{T}\,,\,\gamma\,,\,\mathrm{D}\,,\,\mathrm{R}\,)$

Polityka

- Zwykle oznaczenie π
- Opis sposobu wyboru akcji w każdym ze stanów
- W najprostszym przypadku przypisanie każdemu ze stanów dokładnie jednego ruchu do zagrania
- Bardziej zaawansowany przypadek – polityka to rozkład prawdopodobieństwa

Wartość polityki – dyskontowana suma wypłat

- t — numer kroku, r_t — wartość wypłaty po t-tym kroku
- $\mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_r\right]$, gdy wybierane są ruchy według polityki π

Reinforcement learning

- Agent musi nauczyć się polityki decyzyjnej maksymalizującej wartość
- Agent uczy się na podstawie doświadczeń zebranych w poprzednich iteracjach polityki
- Wejście: $MDP = (S, A, T, \gamma, D, R)$
- Wyjście: polityka π^*

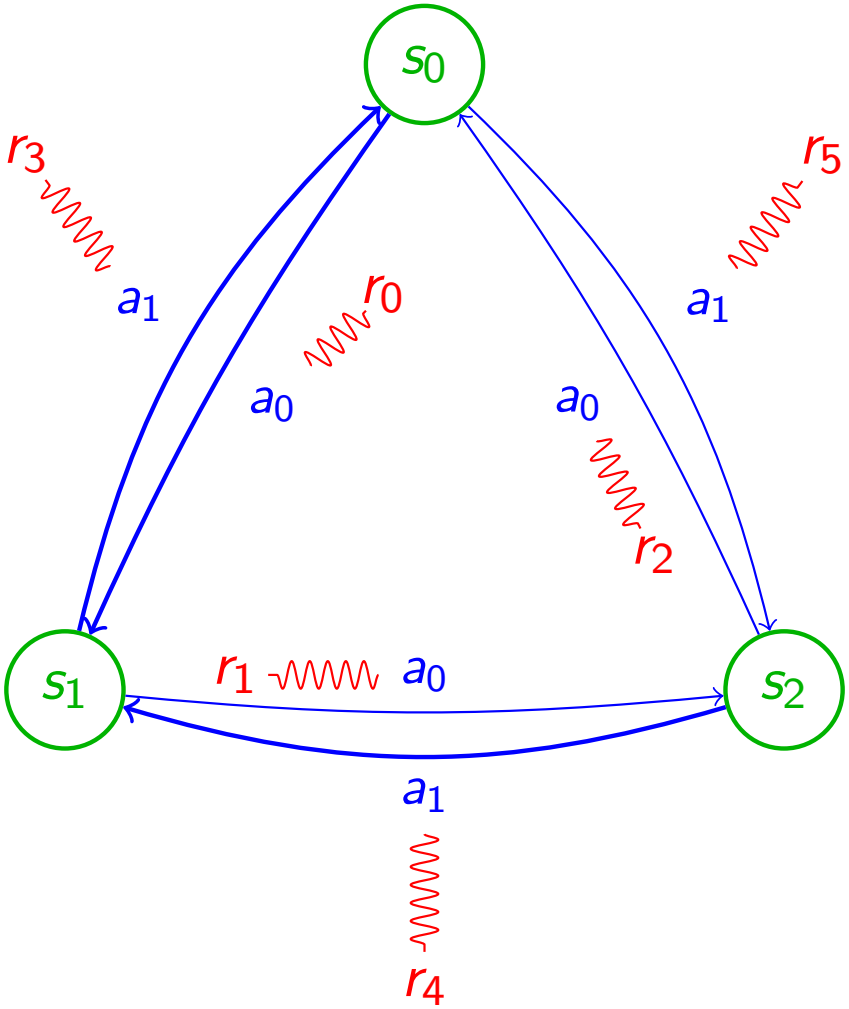
Inverse reinforcement learning

- System musi nauczyć się funkcji wypłat, tak, aby zadana polityka maksymalizowała wartość.
- Wejście: MDP=(S , A , T , γ , D , ~~R~~) oraz π^*
- Wyjście: funkcja R

Andrew Y. Ng and Stuart Russell. "Algorithms for Inverse Reinforcement Learning". In: *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*. Ed. by Pat Langley. Morgan Kaufmann, 2000, pp. 663–670. ISBN: 1-55860-707-2

Niejednoznaczność rozwiązania

Istnieje wiele funkcji R, dla których dana polityka jest optymalna.

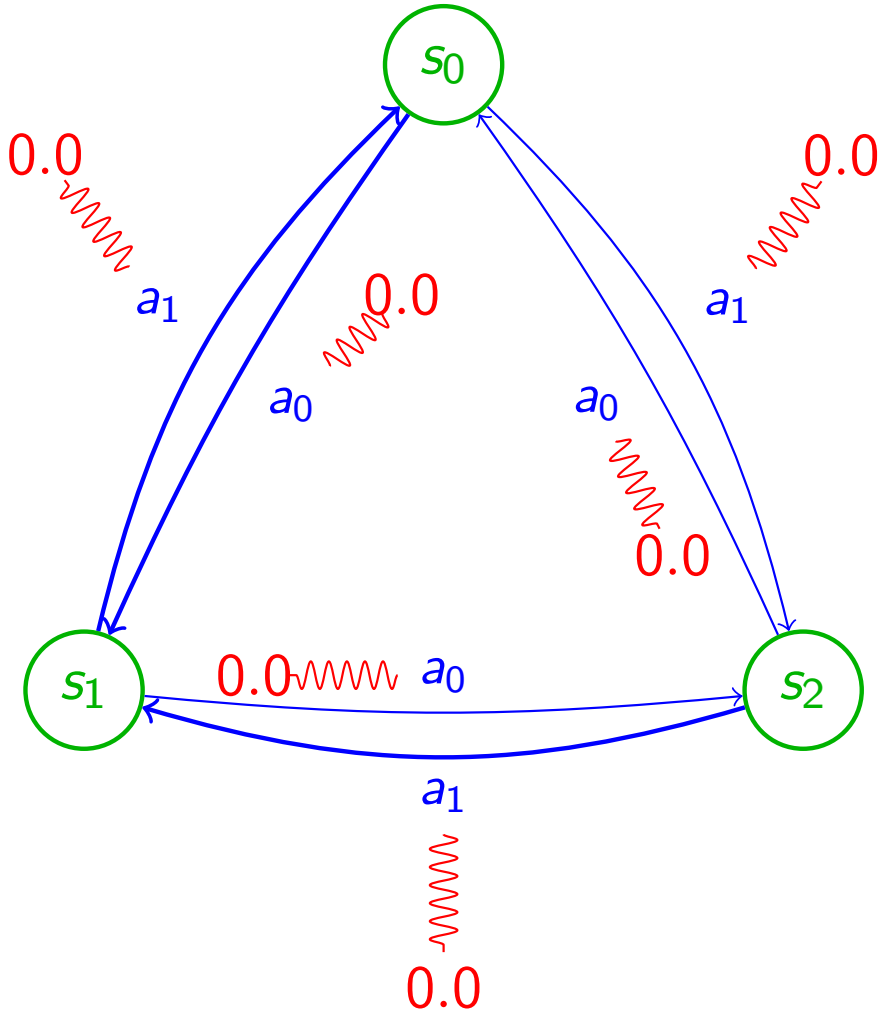


π^*

- $s_0 : a_0$
- $s_1 : a_1$
- $s_2 : a_1$

Niejednoznaczność rozwiązania

Istnieje wiele funkcji R, dla których dana polityka jest optymalna.

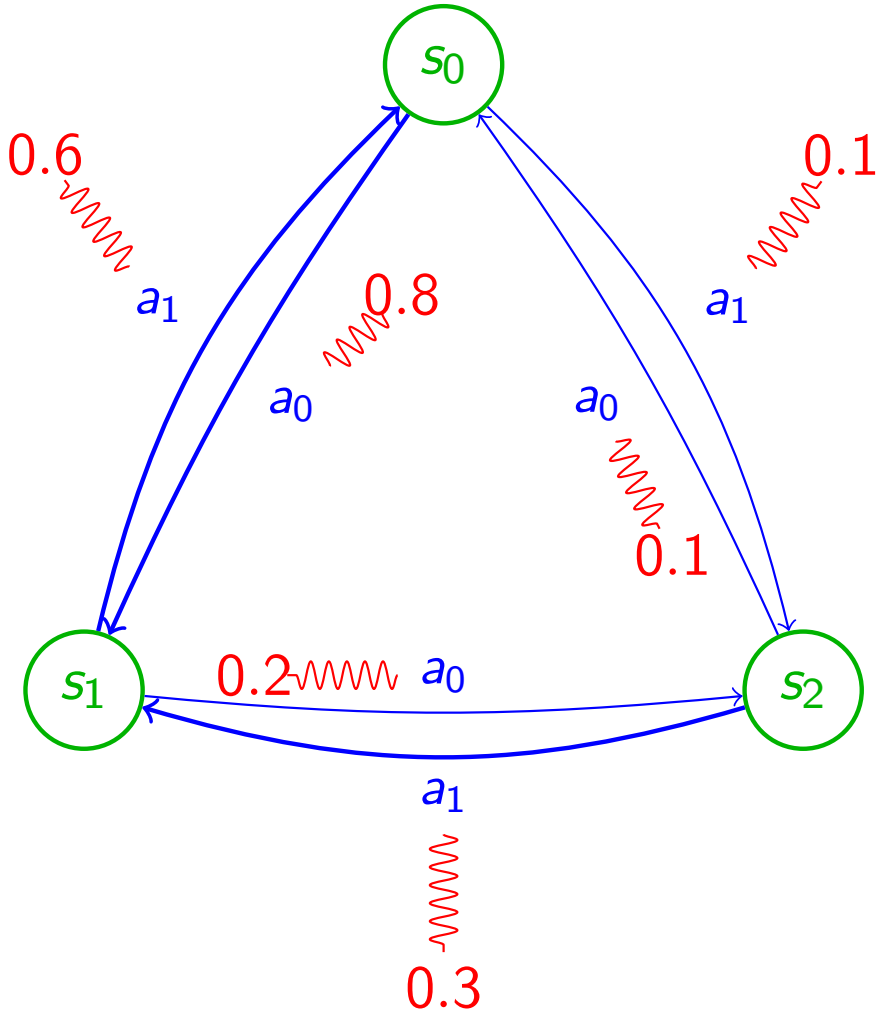


π^*

- $s_0 : a_0$
- $s_1 : a_1$
- $s_2 : a_1$

Niejednoznaczność rozwiązania

Istnieje wiele funkcji R, dla których dana polityka jest optymalna.

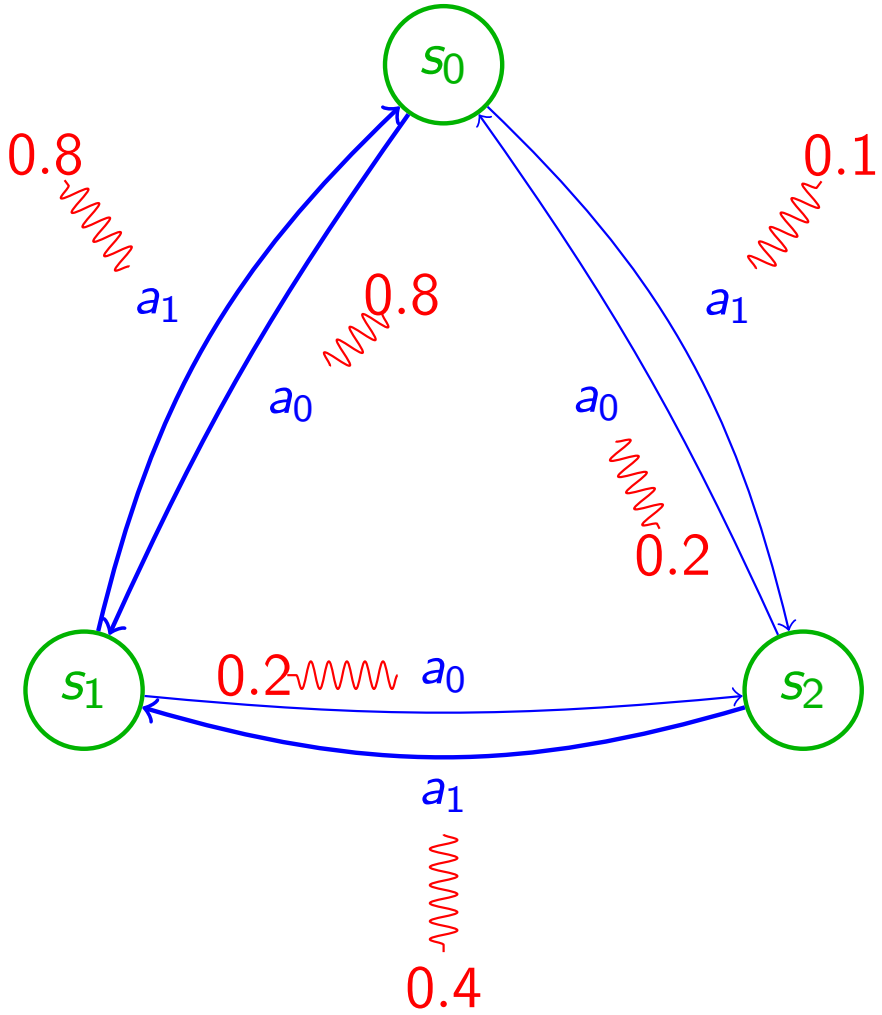


π^*

- $s_0 : a_0$
- $s_1 : a_1$
- $s_2 : a_1$

Niejednoznaczność rozwiązania

Istnieje wiele funkcji R, dla których dana polityka jest optymalna.



π^*

- $s_0 : a_0$
- $s_1 : a_1$
- $s_2 : a_1$

Postulaty do „dobrego” rozwiązania

- Możliwie duża różnica pomiędzy wartością dla polityki π^* a każdą z pozostałych polityk
- Niezbyt duże wartości funkcji wypłat

Skończona przestrzeń stanów I

Andrew Y. Ng and Stuart Russell. "Algorithms for Inverse Reinforcement Learning". In: *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*. Ed. by Pat Langley. Morgan Kaufmann, 2000, pp. 663–670. ISBN: 1-55860-707-2

Dodatkowe założenie: polityka π^* jest znana w jawnej postaci, poprzez przypisanie dokładnie jednej akcji do każdego stanu.

Skończona przestrzeń stanów

- Można stablicować funkcję wypłat
- Polityka to przypisanie akcji do każdego ze skończonej liczby stanów

Skończona przestrzeń stanów II

Zabieg wstępny

- W każdym stanie z osobna, zamienić numery akcji tak, żeby akcja wybrana w π^* miała oznaczenie a_1.
- Bez straty ogólności, upraszcza notację w opisie implementacji.

Szkic rozwiązania

- Program liniowy
- Optymalizuje wektor wartości funkcji wypłat
- Uwzględnia wszystkie zboczenia z polityki π^* w dokładnie jednym stanie
- Zapewnia, żeby w żadnym z tych przypadków wartość sumy dla polityki ze zmianą nie była wyższa niż wartość dla π^*

Skończona przestrzeń stanów III

Program liniowy

$$
\begin{aligned}
\text{maximize} \quad & \textstyle\sum_{i=1}^{N} \min_{a \in \{a_2, \dots, a_k\}} \{(\boldsymbol{P}_{a_1}(i) - \boldsymbol{P}_a(i)) \\
& (\boldsymbol{I} - \gamma \boldsymbol{P}_{a_1})^{-1} \boldsymbol{R}\} - \lambda ||\boldsymbol{R}||_1 \\
\text{s.t.} \quad & (\boldsymbol{P}_{a_1} - \boldsymbol{P}_a)(\boldsymbol{I} - \gamma \boldsymbol{P}_{a_1})^{-1} \boldsymbol{R} \succeq 0 \\
& \qquad \forall a \in A \setminus a_1 \\
& |\boldsymbol{R}_i| \leq R_{\max}, \quad i = 1, \dots, N
\end{aligned}
$$

- R wektor wypłat, jedna wypłata dla jednego stanu
- P_a macierz przejścia gdy została zagrana akcja a. W wierszach stan poprzedzający, w kolumnach stan następny.
- $P_a(i)$ — i-ty wiersz macierzy

Źródło: [2]

Nieskończona przestrzeń stanów I

Andrew Y. Ng and Stuart Russell. "Algorithms for Inverse Reinforcement Learning". In: *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*. Ed. by Pat Langley. Morgan Kaufmann, 2000, pp. 663–670. ISBN: 1-55860-707-2

Założenia

- $S = \mathbb{R}^n$
- Funkcja wypłaty jest kombinacją liniową pewnych funkcji bazowych: $R(s) = \alpha_1\phi_1(s) + \alpha_2\phi_2(s) + \cdots + \alpha_d\phi_d(s)$
  - Problem: takie sformułowanie nie obejmuje wszystkich możliwych funkcji wypłat. To oznacza, że może istnieć polityka π^*, dla której nie ma takich α_i, żeby ta polityka była optymalną.
- Pozostaje w mocy założenie o przenumerowaniu akcji tak, żeby w każdym stanie akcja a_1 była akcją wybraną przez π^*

Nieskończona przestrzeń stanów II

Spostrzeżenie

- Niech $V^{\pi}(s)$ – wartość oczekiwana sumy wypłat w stanie s dla polityki π
- Przy funkcji wypłat jak we wcześniejszych założeniach $V^{\pi} = \alpha_1 V_1^{\pi} + \alpha_2 V_1^{\pi} + \cdots \alpha_d V_d^{\pi}$

Dopuszczalne rozwiązanie

Dobrać takie współczynniki α, żeby:

$$\mathrm{E}_{s' \sim P_{sa_1}} [V^{\pi}(s')] \geq \mathrm{E}_{s' \sim P_{sa}} [V^{\pi}(s')] \qquad (10)$$

Dla każdego stanu s'

$\mathrm{S} = \mathbb{R}^n$

Może się zdarzyć, że nie ma rozwiązania.

Nieskończona przestrzeń stanów III

Funkcja kary

$$p(x) = \begin{cases} x, & \text{gdy} x \geqslant 0 \\ 2x, & \text{w p. p.} \end{cases}$$

Rzeczywiste rozwiązanie

- $S_0 \subset S$ — skończony zbiór spróbkowanych stanów
- $$\begin{aligned} &\text{maximize} \sum_{s \in S_0} \min_{a \in \{a_2, \ldots, a_k\}} \{ \\ &\qquad p(\mathrm{E}_{s' \sim P_{sa_1}}[V^{\pi}(s')] - \mathrm{E}_{s' \sim P_{sa}}[V^{\pi}(s')]) \} \\ &\quad \text{s.t.} \ |\alpha_i| \leq 1, \ \ i = 1, \ldots, d \end{aligned}$$
- Brak ograniczeń — może się zdarzyć, ze π^* nie będzie optymalną polityką

Polityka optymalna nie jest jawnie dana I

Andrew Y. Ng and Stuart Russell. "Algorithms for Inverse Reinforcement Learning". In: *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*. Ed. by Pat Langley. Morgan Kaufmann, 2000, pp. 663–670. ISBN: 1-55860-707-2

Założenia

W porównaniu z poprzednimi:

- π^* nie jest zadana wprost, ale jest możliwości przeprowadzania symulacji działania
- Nie są dane macierze przejścia w MDP, ale jest możliwość symulacji dla dowolnej polityki

Polityka optymalna nie jest jawnie dana II

Działanie

- Wybierz losową politykę π_1
- $k =\leftarrow 1$
- Oblicz $\hat{V}_i^{\pi_k}(s_0)$ dla każdej funkcji składowej wypłaty poprzez symulacje Monte Carlo
- Rozwiąż program

$$\begin{array}{ll} \text{maximize} & \sum_{i=1}^{k} p\left(\hat{V}^{\pi^*}(s_0) - \hat{V}^{\pi_i}(s_0)\right) \\ \text{s.t.} & |\alpha_i| \leq 1, \quad i = 1, \ldots, d \end{array}$$

- $k \leftarrow k + 1$
- Dla tak wyznaczonych współczynników α znajdź strategię π_k, która jest optymalną polityką przy bieżącej funkcji wypłaty.

Eksperymenty I

Andrew Y. Ng and Stuart Russell. “Algorithms for Inverse Reinforcement Learning”. In: *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*. Ed. by Pat Langley. Morgan Kaufmann, 2000, pp. 663–670. ISBN: 1-55860-707-2

Eksperymenty II

Eksperyment 1

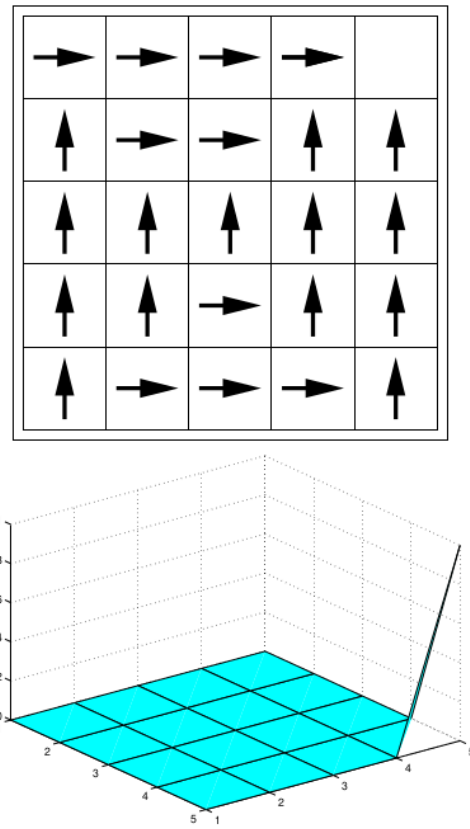


Figure 1. Top: 5x5 grid world with optimal policy. *Bottom*: True reward function.

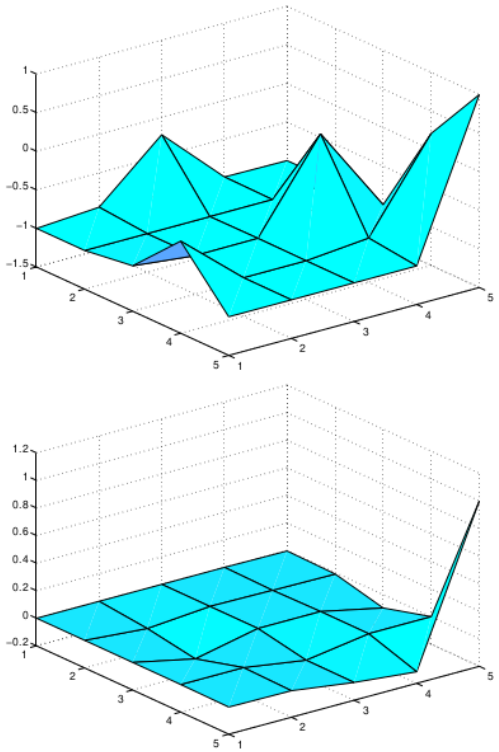


Figure 2. Inverse RL on the 5×5 grid. *Top*: $\lambda = 0$. *Bottom*: $\lambda = 1.05$.

30% szansy na ruch w losowym kierunku.

Eksperymenty III

Eksperyment 2

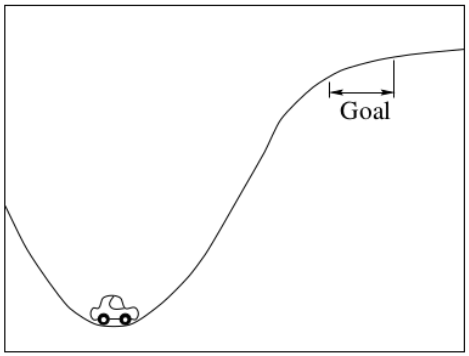


Figure 3. Cartoon of the mountain–car problem (not shown to scale).

- Próbkowanie stanów: siatka 120×120
- Wypłata: 26 funkcji gaussowskich parametryzowanych tylko x, o maksimach równo rozłożonych na osi

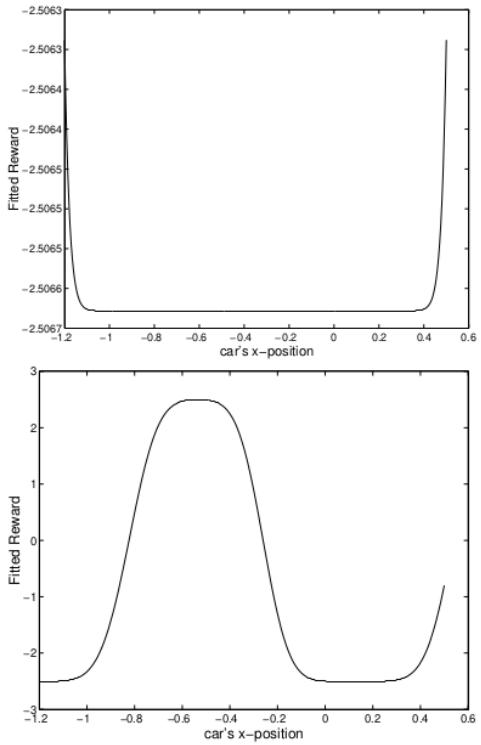


Figure 4. Typical solutions found by IRL for the mountain–car. *Top*: Original problem (note scale on y axis). *Bottom*: Problem of parking at bottom of hill.

- Parkowanie w zakresie $[-0.72, 0.32]$

Eksperymenty IV

Eksperyment 3

- Ciągła wersja gry z chodzeniem po siatce
- Ruch: 0.2 w wybranym kierunku + zaburzenie z kostki $[-0.1, 0.1]^2$
- Wypłaty: 1 w kostce $[0.8, 1]^2$
- $\gamma = 0.9$
- Wypłata: 15×15 funkcji gaussowskich równo rozłożonych na polu gry.
- Szacowanie Monte Carlo 5000 uśrednionych przebiegów z limitem 30 kroków
- Próbkowanie stanów 50×50

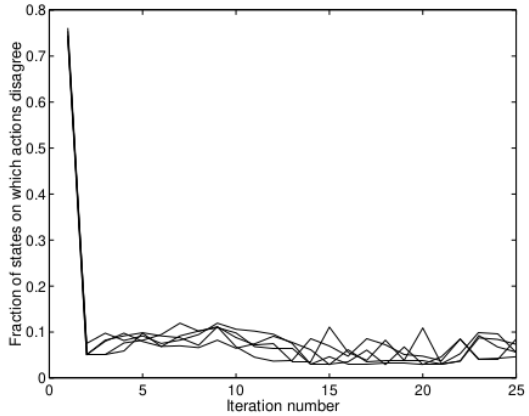


Figure 5. Results on the continuous grid world, for 5 runs. *Top*: Fraction of states on which the fitted reward's optimal policy disagrees with the true optimal policy, plotted against iteration number. *Bottom*: The value of the fitted reward's optimal policy. (Estimates are from 50000 Monte Carlo trials of length 50 each; negligible errorbars).

Uzyskanie polityki podobnej do zadanej I

Pieter Abbeel and Andrew Y. Ng. "Apprenticeship learning via inverse reinforcement learning". In: *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*. Ed. by Carla E. Brodley. Vol. 69. ACM International Conference Proceeding Series. ACM, 2004. DOI: 10.1145/1015330.1015430. URL: https://doi.org/10.1145/1015330.1015430

Problem

Znaleźć politykę, która, przy nieznanej funkcji wypłaty, osiągnie wartość zbliżona do zadanej polityki (niekoniecznie optymalnej).

Uzyskanie polityki podobnej do zadanej II

Założenia

Podobne jak poprzednio.

- Dany jest MDP bez funkcji wypłaty
- Zakładamy, że funkcja wypłaty jest postaci $R(s) = \sum_i^k w_i \phi_i(s)$
- Dana jest polityka eksperta π_E

Uzyskanie polityki podobnej do zadanej III

Algorytm

1. Randomly pick some policy $\pi^{(0)}$, compute (or approximate via Monte Carlo) $\mu^{(0)} = \mu(\pi^{(0)})$, and set $i = 1$.
2. Compute $t^{(i)} = \max_{w:\|w\|_2 \leq 1} \min_{j \in \{0..(i-1)\}} w^T(\mu_E - \mu^{(j)})$, and let $w^{(i)}$ be the value of w that attains this maximum.
3. If $t^{(i)} \leq \epsilon$, then terminate.
4. Using the RL algorithm, compute the optimal policy $\pi^{(i)}$ for the MDP using rewards $R = (w^{(i)})^T \phi$.
5. Compute (or estimate) $\mu^{(i)} = \mu(\pi^{(i)})$.
6. Set $i = i + 1$, and go back to step 2.

Oznaczenia:

- $\mu(\pi)$ — [estymowany] wektor wartości oczekiwanych sumy wypłat przy polityce π dla każdej składowej funkcji R.

Uzyskanie polityki podobnej do zadanej IV

Eksperyment 1

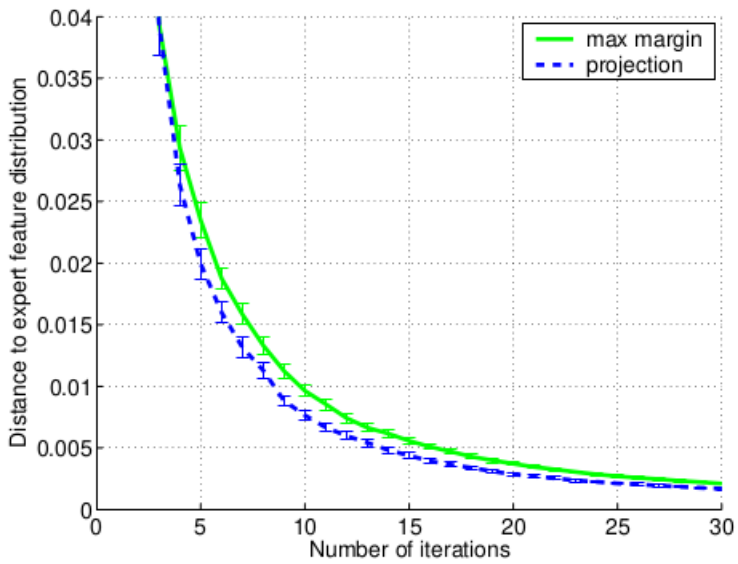


Figure 3. A comparison of the convergence speeds of the max-margin and projection versions of the algorithm on a 128x128 grid. Euclidean distance to the expert's feature expectations is plotted as a function of the number of iterations. We rescaled the feature expectations by $(1-\gamma)$ such that they are in $[0,1]^k$. The plot shows averages over 40 runs, with 1 s.e. errorbars.

- Chodzenie po siatce
- 128×128 węzłów
- Wypłaty: 64 funkcje bazowe – podział przestrzeni na grupy 16×16 – przynależność do grupy.
- $\gamma - 0.99$

Uzyskanie polityki podobnej do zadanej V

Eksperyment 2

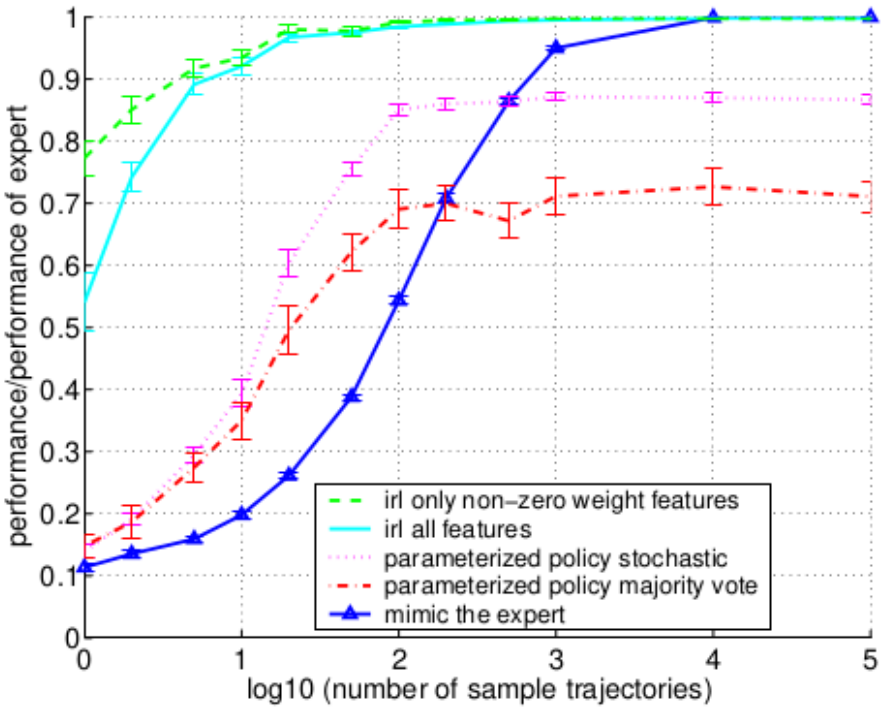


Figure 4. Plot of performance vs. number of sampled trajectories from the expert. (Shown in color, where available.) Averages over 20 instances are plotted, with 1 s.e. errorbars. Note the base-10 logarithm scale on the x-axis.

Uzyskanie polityki podobnej do zadanej VI

Eksperyment 3

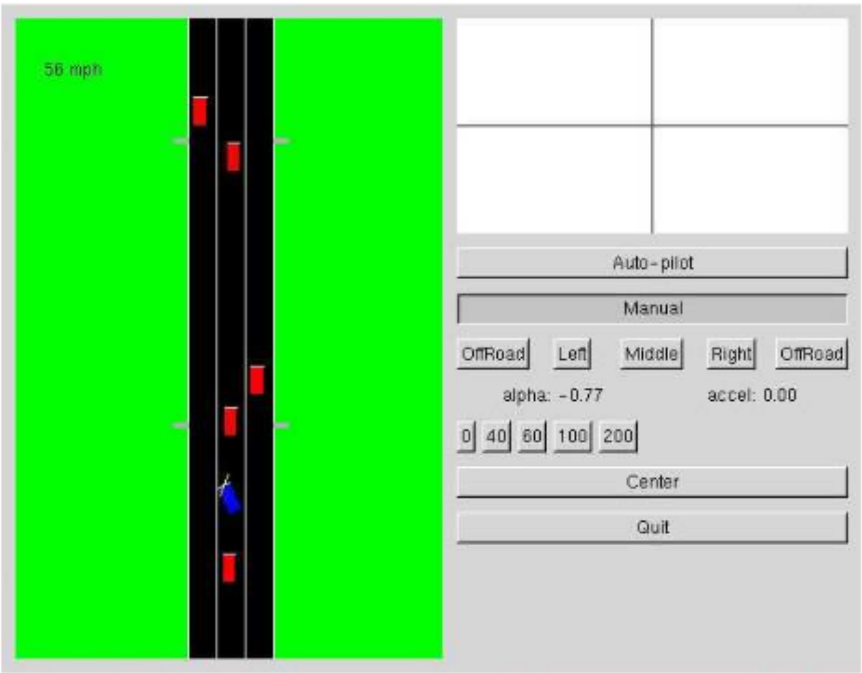


Figure 5. Screenshot of driving simulator.

in every instance. We considered five styles:

1. Nice: The highest priority is to avoid collisions with other cars; we also prefer the right lane over the middle lane over the left lane, over driving off-road.
2. Nasty: Hit as many other cars as possible.
3. Right lane nice: Drive in the right lane, but go off-road to avoid hitting cars in the right lane.
4. Right lane nasty: Drive off-road on the right, but get back onto the road to hit cars in the right lane.
5. Middle lane: Drive in the middle lane, ignoring all other cars (thus crashing into all other cars in the middle lane).

Uzyskanie polityki podobnej do zadanej VII

Eksperyment 3 – wyniki

Table 1. Feature expectations of teacher $\hat{\mu}_E$ and of selected/learned policy $\mu(\tilde{\pi})$ (as estimated by Monte Carlo). and weights w corresponding to the reward function that had been used to generate the policy shown. (Note for compactness, only 6 of the more interesting features, out of a total of 15 features, are shown here.)

		Collision	Offroad Left	LeftLane	MiddleLane	RightLane	Offroad Right
1	$\hat{\mu}_E$	0.0000	0.0000	0.1325	0.2033	0.5983	0.0658
	$\mu(\tilde{\pi})$	0.0001	0.0004	0.0904	0.2287	0.6041	0.0764
	$\tilde{w}$	-0.0767	-0.0439	0.0077	0.0078	0.0318	-0.0035
2	$\hat{\mu}_E$	0.1167	0.0000	0.0633	0.4667	0.4700	0.0000
	$\mu(\tilde{\pi})$	0.1332	0.0000	0.1045	0.3196	0.5759	0.0000
	$\tilde{w}$	0.2340	-0.1098	0.0092	0.0487	0.0576	-0.0056
3	$\hat{\mu}_E$	0.0000	0.0000	0.0000	0.0033	0.7058	0.2908
	$\mu(\tilde{\pi})$	0.0000	0.0000	0.0000	0.0000	0.7447	0.2554
	$\tilde{w}$	-0.1056	-0.0051	-0.0573	-0.0386	0.0929	0.0081
4	$\hat{\mu}_E$	0.0600	0.0000	0.0000	0.0033	0.2908	0.7058
	$\mu(\tilde{\pi})$	0.0569	0.0000	0.0000	0.0000	0.2666	0.7334
	$\tilde{w}$	0.1079	-0.0001	-0.0487	-0.0666	0.0590	0.0564
5	$\hat{\mu}_E$	0.0600	0.0000	0.0000	1.0000	0.0000	0.0000
	$\mu(\tilde{\pi})$	0.0542	0.0000	0.0000	1.0000	0.0000	0.0000
	$\tilde{w}$	0.0094	-0.0108	-0.2765	0.8126	-0.5099	-0.0154

Wideo: `http://www.cs.stanford.edu/~pabbeel/irl/`

Literatura I

[1] Pieter Abbeel and Andrew Y. Ng. "Apprenticeship learning via inverse reinforcement learning". In: *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*. Ed. by Carla E. Brodley. Vol. 69. ACM International Conference Proceeding Series. ACM, 2004. DOI: 10.1145/1015330.1015430. URL: https://doi.org/10.1145/1015330.1015430.

[2] Andrew Y. Ng and Stuart Russell. "Algorithms for Inverse Reinforcement Learning". In: *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*. Ed. by Pat Langley. Morgan Kaufmann, 2000, pp. 663–670. ISBN: 1-55860-707-2.