Sztuczna inteligencja. TD learning, gry bez tur i logika

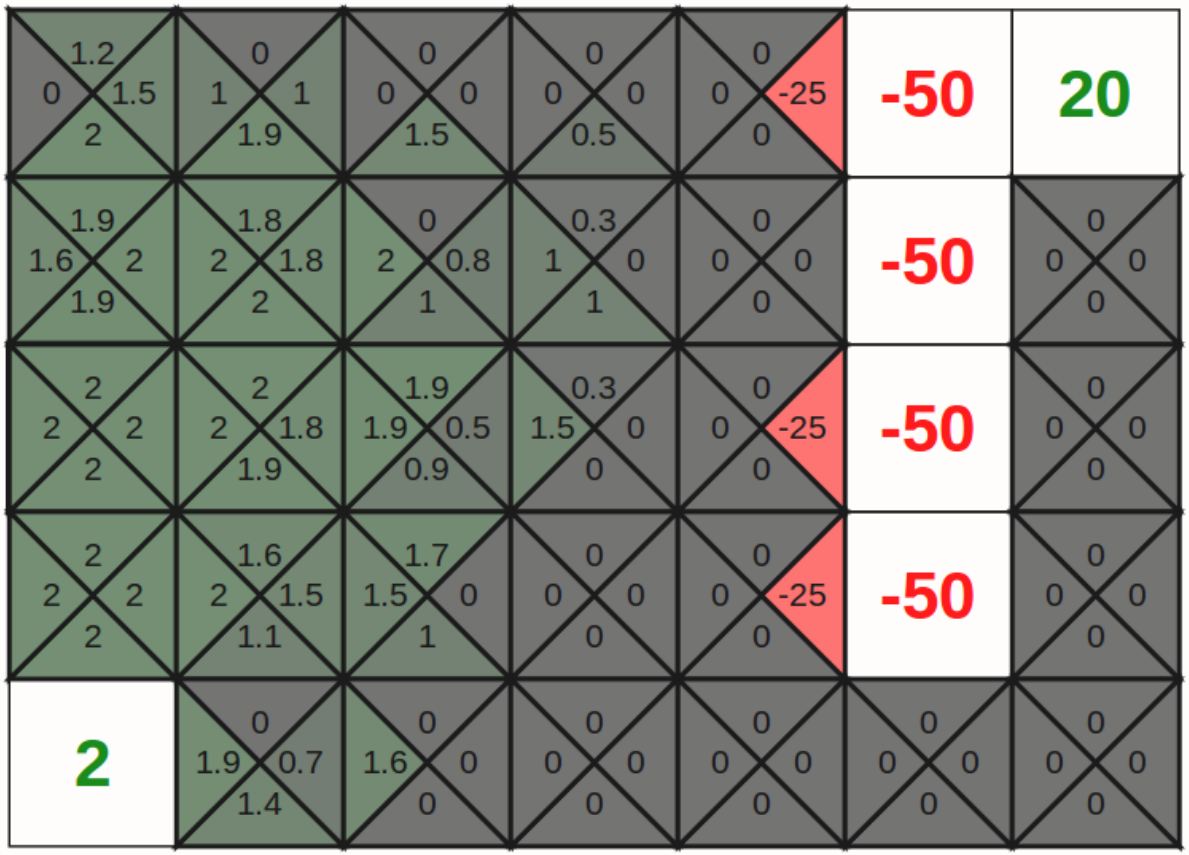
Paweł Rychlikowski

Instytut Informatyki UWr

4 czerwca 2018

Większy wulkan

Problem: large state spaces, hard to explore



Average utility: 0.44

CS221 / Spring 2018 / Sadigh

Regresja liniowa w uczeniu ze wzmocnieniem

- Definiujemy
$$\hat{Q}_{\text{opt}}(s, a; \mathbf{w}) = \mathbf{w} \cdot \phi(s, a)$$
- Przykładowe cechy:
  - Czy akcja równa się R? (0/1)
  - Czy akcja równa się U? (0/1)
  - Czy x = 5?

TD-learning

- Temporal Difference Learning – metoda uczenia wartości V (używana na przykład w grach)
- Idea:
  - Generuj dane z rozgrywek
  - Naucz się wag funkcji heurystycznej analizując te dane

Uwaga

W najprostszym przypadku (czyli liniowym) mamy:

$$V(s; w) = w \cdot \phi(s)$$

Generowanie danych

Definicja

Polityka odruchów (reflex policy) – to strategia agenta, w której podejmuje decyzje analizując przybliżoną funkcję oceniającą konsekwencje działań (w grach: stany po ruchu)

- Do generowania danych możemy wykorzystać politykę odruchową (czyli naszą aktualną funkcję oceniającą)
- **Problem**: tak wygenerujemy tylko jedną rozgrywkę (albo bardzo niezróżnicowaną populację rozgrywek)

Generowanie danych (2)

Konieczne jest wprowadzenie losowości:

a) Polityka ε-zachłanna (pamiętajmy o zmianie znaczenia V dla Mina i Maxa)

b) Losowanie zgodne z prawdopodobieństwem „softmaxowym", czyli:

$$P(s,a) = \frac{e^{V(\text{succ}(s,a))}}{\sum_{a' \in \text{Actions(s)}} e^{V(\text{succ}(s,a'))}}$$

c) Dla gier z z rzucaniem kostkami (z elementem losowym) można wybierać zawsze optymalne ruchy (sama gra zapewnia czynnik eksploracyjny)

Reguła TD-learningu

- **Predykcja**: $V(s; w)$
- **Cel**: $r + \gamma V(s'; w)$ (s' to kolejny stan w rozgrywce)

Streszczenie reguły

W przypadku większości gier ($\gamma = 1$, nagroda na końcu), sprowadza się to do:

a) Staraj się, by podczas dobrych gier wartość planszy po ruchu zbytnio się nie zmieniała (dla minmaxa i optymalnej strategii powinna być ona stała)

b) Zwiększaj wartośc sytuacji bliskich zwycięstwa (wykorzystanie r pod koniec).

Ogólne zasady

- Funkcja celu:

$$\frac{1}{2}(\text{prediction}(w) - \text{target})^2$$

- Gradient:

$$(\text{prediction}(w) - \text{target})\nabla_w(\text{prediction}(w))$$

- Reguła uaktualniania:

$$w \leftarrow w - \eta((\text{prediction}(w) - \text{target})\nabla_w(\text{prediction}(w)))$$

Algorytm TD-learning

Algorytm

Dla każdego s, a, r, s' wykonuj:

$$w \leftarrow w - \eta(V(s; w) - (r + \gamma V(s', w)))\nabla_w V(s; w)$$

Dla funkcji liniowej:

$$V(s, w) = w \cdot \phi(s)$$

mamy

$$\nabla_w V(s, w) = \phi(s)$$

Porównanie TD-learning i Q-learning

Algorithm: TD learning

On each (s, a, r, s'):

$$\mathbf{w} \leftarrow \mathbf{w} - \eta[\underbrace{\hat{V}_\pi(s; \mathbf{w})}_{\text{prediction}} - \underbrace{(r + \gamma \hat{V}_\pi(s'; \mathbf{w}))}_{\text{target}}]\nabla_{\mathbf{w}} \hat{V}_\pi(s; \mathbf{w})$$

Algorithm: Q-learning

On each (s, a, r, s'):

$$\mathbf{w} \leftarrow \mathbf{w} - \eta[\underbrace{\hat{Q}_{\text{opt}}(s, a; \mathbf{w})}_{\text{prediction}} - \underbrace{(r + \gamma \max_{a' \in \text{Actions}(s)} \hat{Q}_{\text{opt}}(s', a'; \mathbf{w}))}_{\text{target}}]\nabla_{\mathbf{w}} \hat{Q}_{\text{opt}}(s, a; \mathbf{w})$$

Inne uwagi o uczeniu w grach

- Można łączyć TD learning z uczeniem polityki. AlphaGo Zero tak właśnie robiło.
- Można stosować metody poprawiania polityki/oceny:
  a) Bazujące na alpha-beta search (tak uczyć funkcję oceniającą, żeby miała „inteligencję" taką, jak poprzednia wersja)
  b) MCTS policy improvement (żeby nowa polityka udawała jak najlepiej starą wspomagającą się symulacjami MCTS)

Uwaga

Takie metody były używane w różnych słynnych programach:

1. Warcaby (Samuel, 1965)
2. Tryktrak, czyli Backgammon (Tesauro, ok. 1990)
3. AlphaGoZero (DeepMing, 2017)

Gry z jedną turą

- Powiemy sobie trochę o grach z jedną turą
- Ale takich, w których gracze podejmują swoje decyzje jednocześnie

Rozważamy gry z sumą zerową.

Gra w zgadywanie (Morra 2)

- Mamy dwóch graczy:
  A) Zgadywacz
  B) Zmyłek

  którzy na sygnał pokazują 1 lub 2 palce.
- Jeżeli Zgadywacz nie zgadnie (pokazał coś innego niż Zmyłek), daje Zmyłkowi 3 dolary.
- Jeżeli Zgadywacz zgadnie, to dostaje od Zmyłka:
  - jak pokazali 1 palec, to 2 dolary
  - jak pokazali 2 palce, to 4 dolary

Macierz wypłat

Definicja

Taką grę zadajemy za pomocą macierzy wypłat, w której $V_{a,b}$ jest wynikiem gry z punktu widzenia pierwszego gracza.

Nasza gra:

Zg/Zm	1 palec	2 palec
1 palec	2	-3
2 palce	-3	4

Strategie

- Czysta strategia: zawsze a
- Mieszana strategia: rozkład prawdopobieństwa na akcjach

Proste fakty

- Jak Zmyłek będzie grał cały czas to samo, to Zgadywacz wygra każdą turę (i odwrotnie)
- Muszą zatem stosować strategie mieszane, ale jakie?

Wartość gry

Definicja

Wartość gry dla dwóch strategii graczy jest równa:

$$V(\pi_A, \pi_B) = \sum_{a,b} \pi_A(a)\pi_B(b)V(a,b)$$

Przykładowo: Zgadywacz zawsze zgaduje 1, Zmyłek wybiera akcję losowo z prawdopodobieństwem 0.5.

Wynik: $-\frac{1}{2}$ (tak samo często zyskuje 2 jak traci 3 dolary)

Strategia mieszana vs czysta

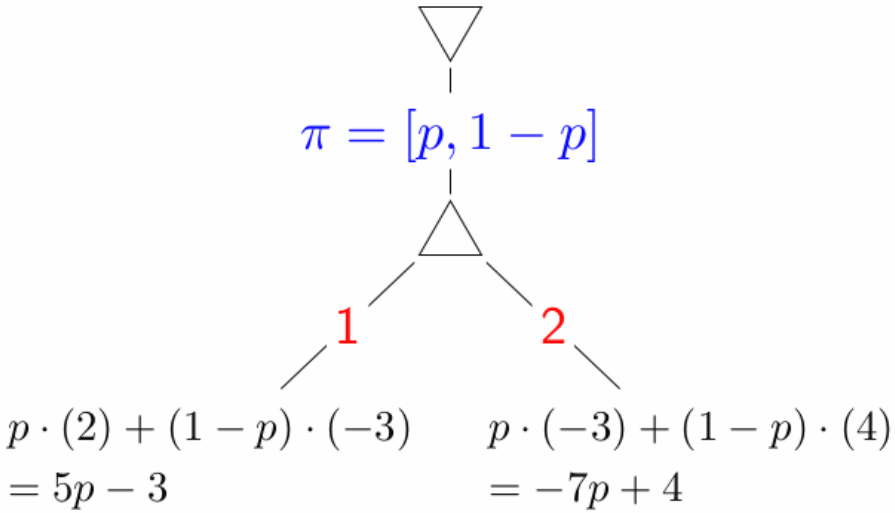
Uwaga

Jeżeli gracz A zapowie, że będzie grał strategią mieszaną (i ją poda), wówczas gracz B może grać strategią czystą (i osiągnie optymalny wynik).

Dlaczego?

Znalezienie optymalnej strategii

Zaczyna gracz B – Zmyłek.
Wybiera strategię mieszaną z parametrem p



Wartość takiej gry to

$$\min_{p \in [0,1]} (\max(5p - 3, -7p + 4))$$

Zauważmy, dla jakich p wygrywa lewe, dla jakich prawe i co z tego wynika.

Znalezienie optymalnej strategii (2)

- W powyższej grze, Zmyłek osiągnie najlepszy wynik, gdy przyjmie $p = \frac{7}{12}$, wynik ten to $-\frac{1}{12}$
- Ok, on zaczynał, miał trudniej – a gdyby zaczynał Zgadywacz? I podał swoją strategię mieszaną?

Wynik gry

Wynik jest dokładnie taki sam, czyli $\mathbf{-\frac{1}{12}}$!

Twierdzenie von Neumana

Twierdzenie, von Neuman, 1928

Dla każdej jednoczesnej gry dwuosobowej o sumie zerowej ze skończoną liczbą akcji mamy:

$$\max_{\pi_A} \min_{\pi_B} V(\pi_A, \pi_B) = \min_{\pi_B} \max_{\pi_A} V(\pi_A, \pi_B)$$

dla dowolnych mieszanych polityk π_A, π_B.

- Można ujawnić swoją politykę optymalną!
- **Dowód**: pomijamy, programowanie liniowe, przedmiot J.B.
- Algorytm: programowanie liniowe

Gry wieloturowe

- Można o grze wieloturowej myśleć jako o grze jednoturowej
- Gracze na sygnał kładą przed sobą opis strategii (program)

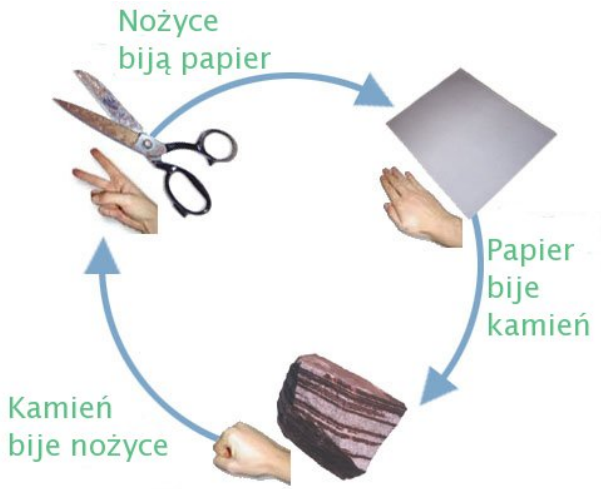
Uwaga

Optymalną strategią jest MinMax (ExpectMinMax w grach losowych). Ale wiedząc o strategii gracza różnej od optymalnej możemy oczywiście ugrać więcej.

Eksperyment myślowy

Jak zmienia się strategia kółka i krzyżyka, gdy wiemy, że oponent gra losowo? (i zaczyna).

Papier, nożyce, kamień



Źródło: Wikipedia

- Optymalna strategia: oczywiście losowa, ale ...
- który człowiek (nie dysponując kostką do gry), przegrawszy 3 razy z rzędu jako papier pokaże papier?

Wyobraźmy sobie turniej, w którym gra N ludzi i K programów. Mecz to wiele tur. Jak napisać taki program?

Co pomijamy

- Gry o sumie niezerowej, w których dochodzi możliwość kooperacji.
- Punkt równawagi Nasha (jest zawsze para strategii, że żaden gracz nie chce jej zmienić, wiedząc, że ten drugi nie zmienia).
- Agent musi zdecydować, czy ma być miły dla innego agenta (i budować reputację przy wielu rozgrywkach, słynny dylemat więźnia).

Paradygmaty modelowania świata

1. Bazujące na **stanach**: przeszukiwanie, MDP, gry
2. Bazujące na **zmiennych**: CSP, sieci Bayesowskie (jeszcze przed nami)
3. Bazujące na **logice**: logika zdaniowa, logiki modalne, logika 1-go rzędu

Zajmiemy się teraz logiką. Zaczniemy od **logiki zdaniowej**.

Sposób definiowania logiki (ogólnie)

Musimy podać 3 składniki:

1. **Składnię** – jak pisac formuły
2. **Semantykę** – co znaczą formuły, kiedy są prawdziwe
3. **Reguły wnioskowania** – jak z prawdziwych formuł wnioskować inne, również prawdziwe

Logiki mają różną siłę wyrazu i dają procedury o różnej złożoności (musimy **balansować** pomiędzy siłą wyrazu a obliczeniową trudnością logiki)

Logika zdaniowa

(dobrze ją pamiętamy z Logiki dla informatyków)

- zmienne zdaniowe (przyjmują wartości 0/1)
- spójniki: $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$, $\neg$

Przykłady

- pada $\wedge \neg$ mam-parasol $\rightarrow$ jestem-mokry $\vee$ mam-kurtkę
- $\neg\ (p \vee q) \leftrightarrow (\neg\ p \wedge \neg\ q)$

Model (logika zdaniowa)

- Modelem w logice zdaniowej jest przypisanie zmiennym wartości logicznych.
  - Ogólnie o modelu myślimy jako o naszej wizji świata.
- Interpretacją formuły przy zadanym modelu jest zdefiniowana rekurencyjnie wartość formuły:
  - $I(a, w) = w(a)$, jeżeli a jest zmienną
  - $I(f_1 \vee f_2, w) = I(f_1, w) \vee I(f_2, w)$
- Składnia (syntax) vs semantyka

Modele i formuły

Definicja

Formuła f jest spełnialna, czyli ma model, jeżeli istnieje takie w, ze $I(f, w) = 1$.

Uwaga

Takich przypisań jest skończenie wiele, stąd mamy prosty (wykładniczy) algorytm sprawdzania, czy formuła ma model (jest spełnialna)

Uwaga 2

Formuła = zwięzły zapis zbioru modeli.

Baza wiedzy

- Formuły to zdania opisujące świat.
- Naturalne jest myślenie o zbiorze takich formuł (do którego możemy dodawać nowe fakty).
- Taki zbiór często nazywamy **bazą wiedzy**.

Koniunkcyjna postać normalna

Definicje

1. literał - zmienna albo $\neg$ zmienna
2. klauzula - $l_1 \vee \cdots \vee l_n$ (gdzie l_i to literał)
3. formuła w **CNF** - $c_1 \wedge \cdots \wedge c_n$, gdzie c_i jest klauzulą

Dlaczego CNF jest fajna?

- Każdą formułę można przekształcić do CNF (czasem płacąc wykładniczym wzrostem jej długości, ćwiczenia)
- Koniunkcja klauzul = zbiór klauzul = baza wiedzy = zbiór więzów
- Jak mamy zbiór formuł (bazę wiedzy, niekoniecznie w CNF) to wykładniczość dotyczy pojedynczej formuły, a nie całej bazy.

Fakt

Dla każdej formuły F logiki zdaniowej istnieje **niezbyt duża** formuła F' w CNF spełnialna wtedy i tylko wtedy, gdy F jest spełnialna (łatwy dowód na ćwiczeniach).

CSP a spełnialność

- Sprawdzanie spełnialności formuły boolowskiej jest zadaniem rozwiązywania więzów
- Nie dziwi zatem, że podstawowe algorytmy (stosowane w praktyce) są dość podobne (backtracking + propagacja).

Uwaga

Współczesne SAT-solvery radzą sobie z milionami klauzul i setkami tysięcy zmiennych

A NP-zupełność?

- Oczywiście problem CNF-SAT (spełnialności formuły w CNF) jest NP-zupełny.
- Nie spodziewamy się istnienia algorytmu wielomianowego (znane algorytmy mają pesymistyczny czas wykładniczy).

Pytanie

Dlaczego SAT-Solvery działają dobrze?

Pytanie jest trudne, i tak do końca nie ma odpowiedzi. Jedyne, co można powiedzieć, że widocznie znaczna część w praktyce spotykanych formuł jest w jakimś sensie **łatwa**.

Algorytm DPLL

- Algrytm Davisa–Putnama–Logemanna–Lovelanda (DPLL) jest algorytmem znajdującym spełniające podstawienie dla formuły CNF.
- Jest **zupełny** – tzn. zawsze kończy się z prawidłowym wynikiem, może działać długo

Uwaga

Stanowi bazę współczesnych SAT-Solverów (modulo jedna modyfikacja, o której powiemy za chwilę).

Algorytm DPLL (2)

Definicje

a) **Klauzula jednostkowa** (unit clause) – klauzula zwierająca 1 literał

b) **Czysty literał** – literał, który występuja tylko jako pozytywny, lub tylko jako negatywny (czyli z jedną polaryzacją).

Jakie wnioskowanie można przeprowadzić korzystając z tych pojęć:

a) unit propagation – klazule jednostkowe można spełnić na 1 sposób (spełniając literał), wstawiając wartość logiczną do innych klauzul możemy zrobić nowe klauzule jednostkowe.

b) „Opłaca się" przypisywać czystym literałom wartość true (bo?).