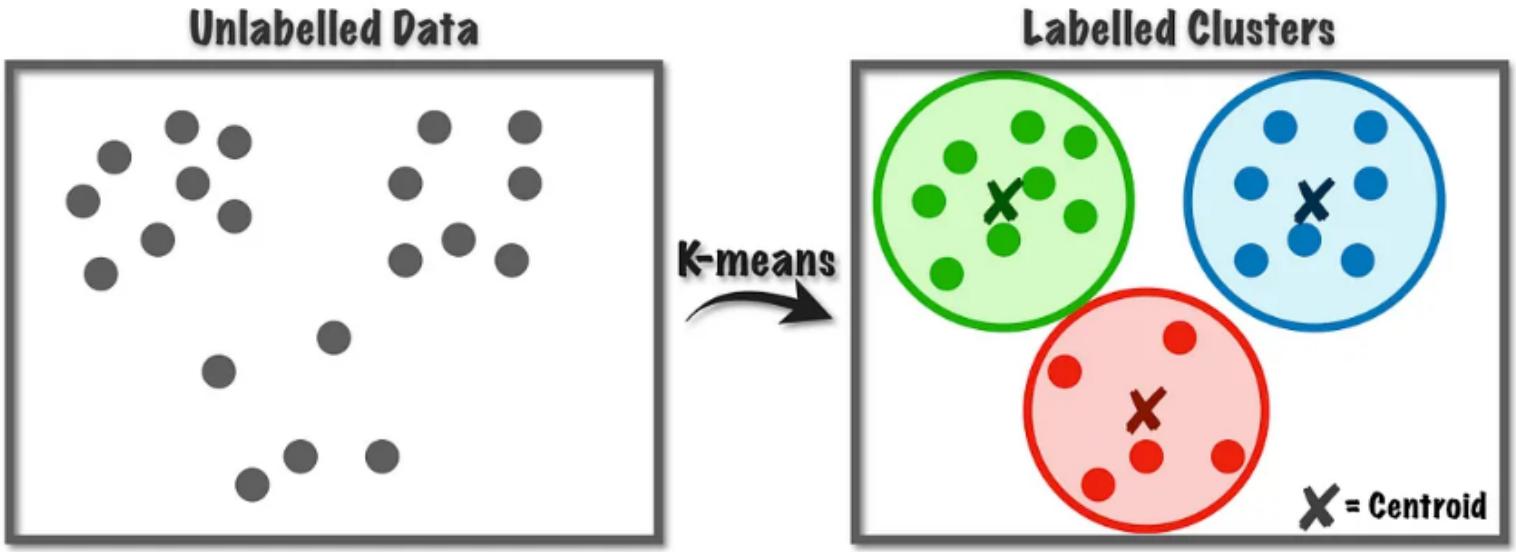
Uczenie. Logika

Paweł Rychlikowski

Instytut Informatyki UWr

12 czerwca 2023

K-średnich. Przypomnienie



K-średnich. Kilka uwag końcowych

- Możemy powtarzać losowanie punktów kilka razy i wybrać najmniejszy błąd
- Możemy wykonać (w celu przyspieszenia) algorytm dla podpopulacji punktów (i potem tylko 1-2 etapy dla wszystkich punktów)
- Zauważmy, że im większe K, tym (średnio) mniejszy błąd grupowania

Pytanie

Jak wykorzystać ostatnie spostrzeżenie do wyboru wartości K (wskazówka: jak wygląda wykres wartości błędu w zależności od K)

Błąd w zależności od liczby klastrów

- 1 klaster – **maksymalny błąd**
- N klastrów – **minimalny błąd** (każdy swoim reprezentantem)
- K jest mniejsze od naturalnej liczby skupień – pewne klastry są połączeniem wielu, duży błąd

Uwaga

Gdy zwiększamy K, do pewnego momentu błąd maleje znacznie, od któregoś – krzywa się wypłaszcza
ten moment to rzeczywista liczba skupień

Wykrywanie nieprawidłowości

Potencjalnie bardzo użyteczne zadanie: można na przykład analizować pomiary różnych parametrów jakiegoś skomplikowanego systemu (skrzydło samolotu pasażerskiego) i zauważać, że coś dziwnego się dzieje

- **Pytanie:** Jak wykorzystać algorytm K-means do wykrywania anomalii?
- Charakterystyka anomalii:
  - Daleko od centrum
  - Najbliżsi sąsiedzi należą do różnych klastrów

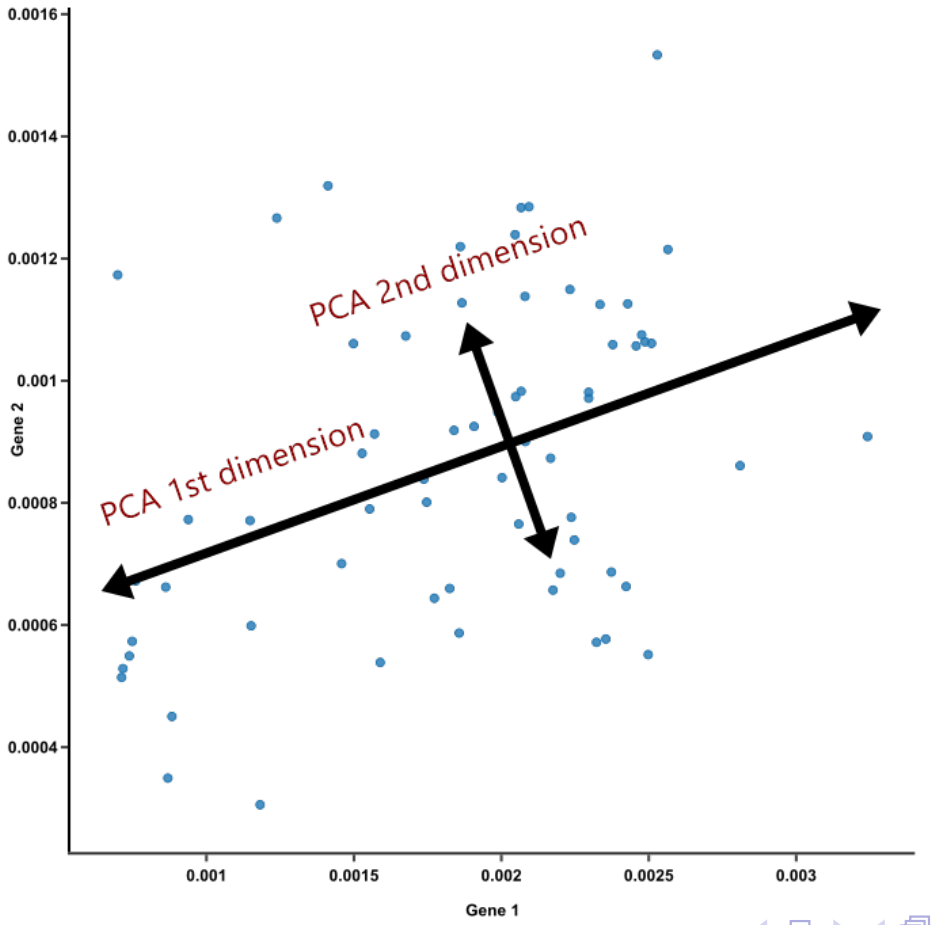
Wykonanie algorytmu K-średnich i analiza poszczególnych punktów daje możliwość zidentyfikowania „dziwnych" elementów (potencjalnych nieprawidłowości).

Redukcja wymiarów

- **Cel**: „zagęszczenie danych” (umożliwiające, być może, lepsze działanie innych algorytmów)
- **Dodatkowa korzyść**: jak zredukujemy liczbę wymiarów do 2, to możemy zbiór danych ładnie narysować (i być może zobaczyć jakieś prawidłowości)
- Redukcja wymiarów oznacza usunięcie informacji, ale być może usuniemy nieistotne informacje, czyli szum.
- Przykładową metodą (omawianą na algebrze) jest PCA, czyli analiza głównych składowych

Principal Component Analysis

- Identyfikujemy osie, które odpowiadają za największą zmienność danych
- Obracamy przestrzeń i pozostawiamy tylko najważniejsze wymiary.



Algorytmy wizualizacji

- Mamy punkty w przestrzeni wielowymiarowej.
- Chcemy przypisać im punkty na płaszczyźnie (2D)
- Jak? (wskazówka: potrafimy liczyć odległość w przestrzeni wielowymiarowej)

Ogólna zasada

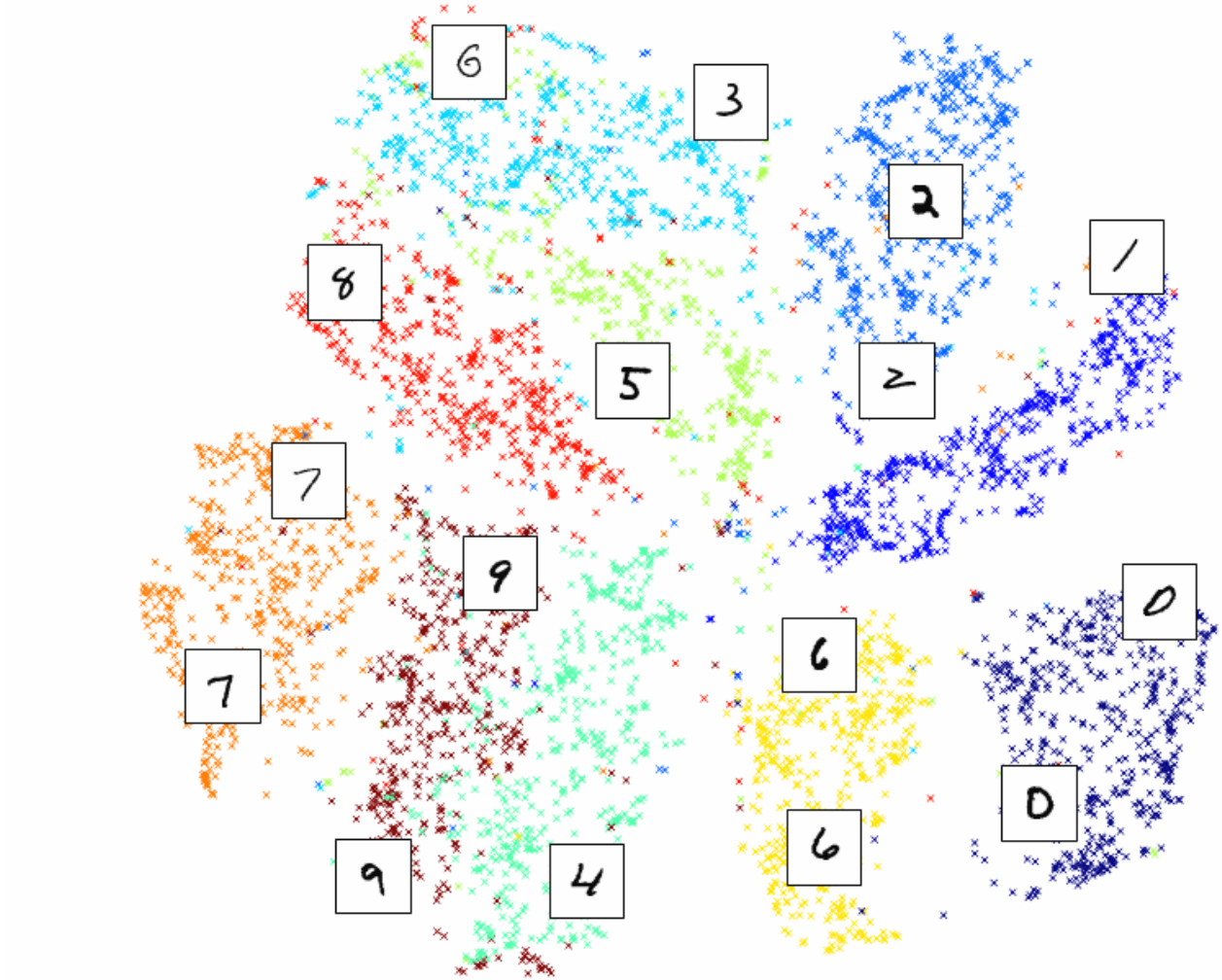
Staramy się, by odległości w 2D odpowiadały tym w oryginalnej przestrzeni (np. 500D)

Niektóre algorytmy można interpretować jako tworzenie układu punktów połączonych sprężynkami, punkty „podobne" się przyciągają, odległe – odpychają, szukamy równowagi tego układu dynamicznego.

Wizualizacja obrazów cyfr (MNIST)

Algorytm t-SNE (który można intepretować „sprężynkowo”):

MNIST dataset – Two-dimensional embedding of 70,000 handwritten digits with t-SNE



Uczenie funkcji oceniającej (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)

Uczenie funkcji oceniającej jako zadanie klasyfikacji/regresji

- **Klasyfikacja**: kto wygra (czy V jest dodatnie?)
- **Regresja**: wartość V

Idea

- Dla każdej wygenerowanej partii dokładnie wiemy, kto ją wygrał (powiedzmy, że MAX).
- Zatem wszystkie stany powinny wskazywać na przewagę MAXa (im późniejsze, tym bardziej)

Temporal Difference Learning

Reguła TD-learningu

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

Uwaga

W przypadku standardowych gier lepiej wypłacać nagrodę po każdym ruchu (bo wiemy kto wygrał, zobacz tablica).

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'):

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

Regresja liniowa w uczeniu ze wzmocnieniem

- Definiujemy
$$\hat{Q}_{\text{opt}}(s, a; \mathbf{w}) = \mathbf{w} \cdot \phi(s, a)$$
- Weźmy grę w Dżunglę z losowym przeciwnikiem (zauważmy, że to jest MDP)
- Przykładowe cechy (propozycje?):
  - Czy jest bicie (0/1)?
  - Czy jest bicie słonia, szczura, kota (dla każdej bierki cecha)?
  - Czy ruch zbliża do jamy przeciwnika? (0/1)
  - Czy ruch jest skokiem zbliżającym do jamy?
  - Czy najbliższa jamie bierka się zbliżyła?
  - Czy podchodzimy pod bicie?
  - Czy bijemy bierkę w pułapce
  - Czy ruch jest do przodu (w lewo, w prawo, w dół)?
  - Czy zajmujemy któreś z wskazanych pól (związanych z atakiem bądź obroną)

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)

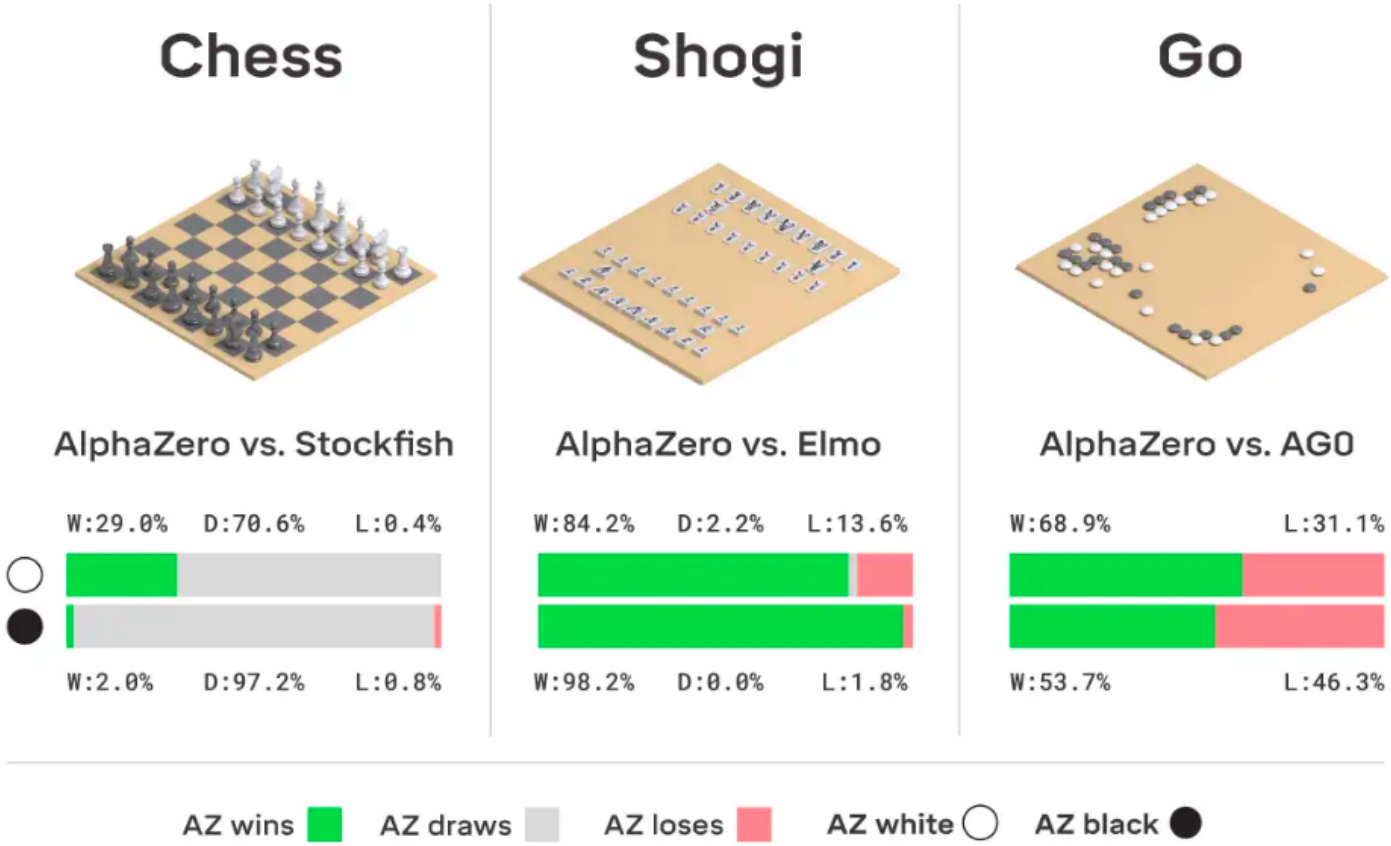
AlphaZero

- AlphaGo wygrało z Lee Sedolem (drugi gracz na świecie).

AlphaZero

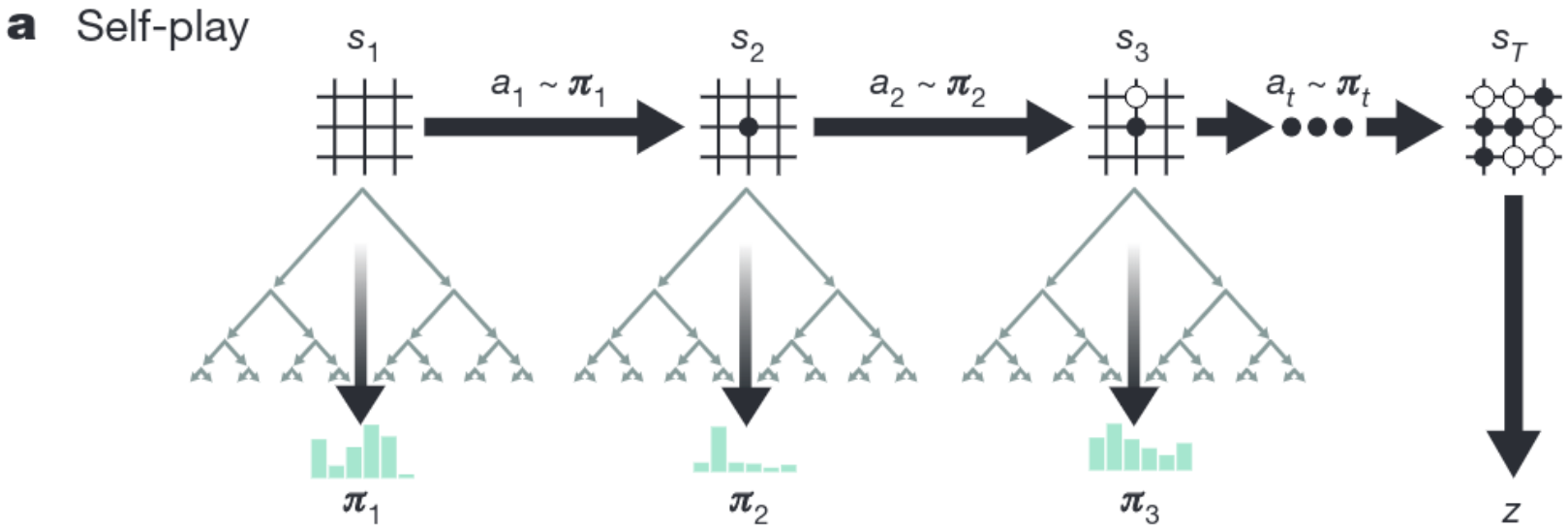
- AlphaGo wygrało z Lee Sedolem (drugi gracz na świecie). Zawierało elementy uczenia z historycznych partii – uczenie z nadzorem, jaki ruch zrobił dobry gracz, który w sytuacji X wygrał partię.
- AlphaZero – uczył się tylko grając sam ze sobą!
- Bardzo dobre (lub znakomite) wyniki w grach:
  - Go (mistrz Wszechświata)
  - Szachy (ale nie poziom mistrza świata)
  - Shoggi (jako przykład innej gry, która też działa)

AlphaZero



https://deepmind.com/blog/article/alphazero-shedding-new-light-grand-games-chess-shogi-and-go

AlphaZero. Rozgrywka



Drobna uwaga

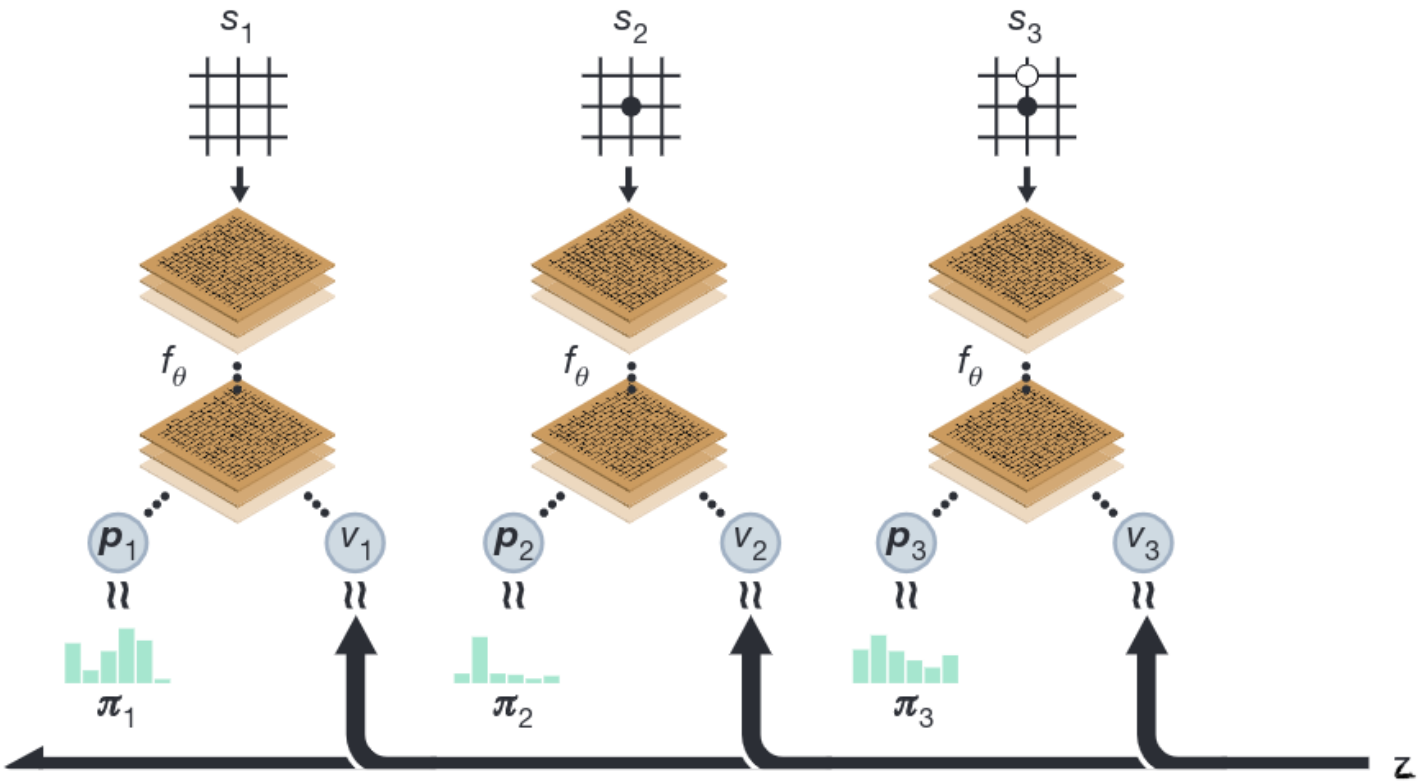
Trochę zmodyfikowany MCTS, który zamiast symulacji używa aktualnej funkcji heurystycznej.

AlphaZero. Analiza rozgrywki

Cele uczenia

1. Funkcja wybierająca ruch
2. Funkcja oceniająca planszę

b Neural network training



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**.

Logika zdaniowa

- 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
  - $I(f_1 \vee f_2, w) = I(f_1, w) \vee I(f_2, w)$
  - (...) – inne, podobne reguły
- 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)

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
- Jak mamy zbiór formuł (bazę wiedzy, niekoniecznie w CNF) to wykładniczość dotyczy pojedynczej formuły, a nie całej bazy.

CSP a spełnialność

- Sprawdzanie spełnialności formuły boolowskiej jest zadaniem rozwiązywania więzów (baza wiedzy = zbiór 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**.