Deep Learning Szeminárium
7. előadás: Megerősítéses Tanulás

Zombori Zsolt

MTA, Rényi Alfréd Matematikai Kutatóintézet

Forrás

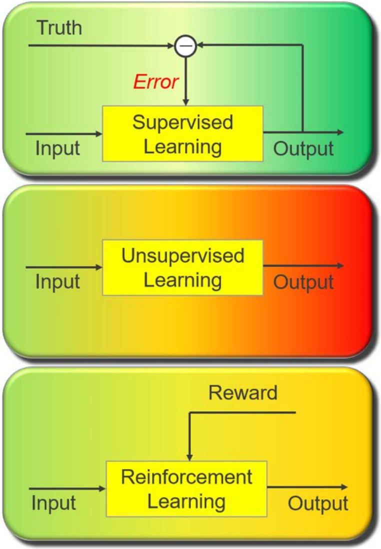
Stanford online:

`https://www.youtube.com/watch?v=lvoHnicueoE`

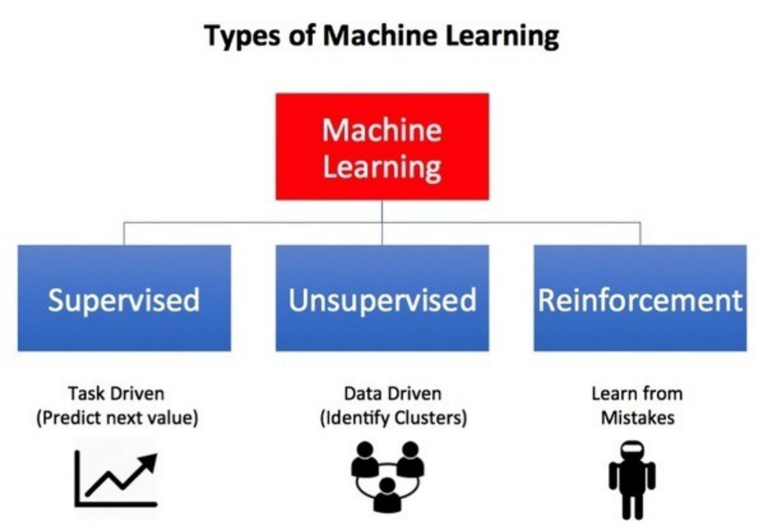
Megerősítéses tanulás - Sakk

- **Cél**: Győzelem
- **Állapot**: Tábla állása
- **Cselekvés**: Lépés egy bábuval
- **Jutalom**: 1 a győztes meccs végén

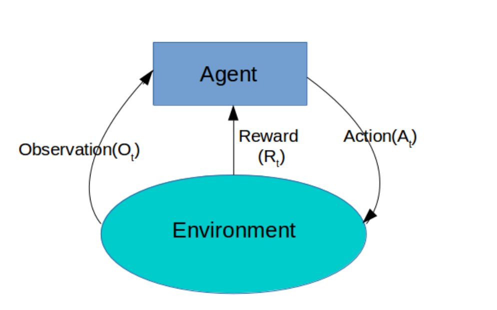
Különféle tanulási feladatok



Különféle tanulási feladatok

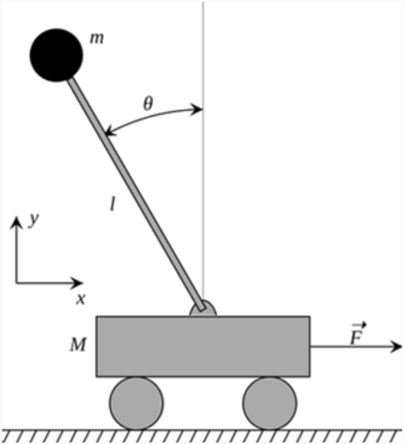


Megerősítéses tanulás



- **Cél** (Objective)
- **Állapotok** (State)
- **Cselekvések** (Action)
- **Jutalom** (Reward)

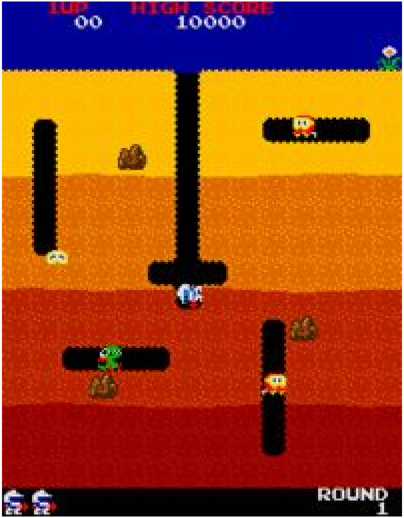
Megerősítéses tanulás - Cart Pole



- **Cél**: Rúd függőlegesen álljon
- **Állapot**: Sebesség, szögsebesség, pozíció
- **Cselekvés**: Viszintes erő a kocsira
- **Jutalom**: Minden pillanatban a függőlegestől való eltérés szöge (Θ)

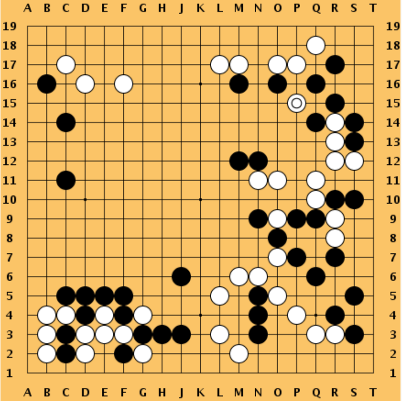
Megerősítéses tanulás - Robot mozgás

- **Cél**: Gyors haladás
- **Állapot**: Csuklók helyzete
- **Cselekvés**: Nyomaték a csuklómotorokra
- **Jutalom**: Minden időpontban a céltól mért távolság reciproka



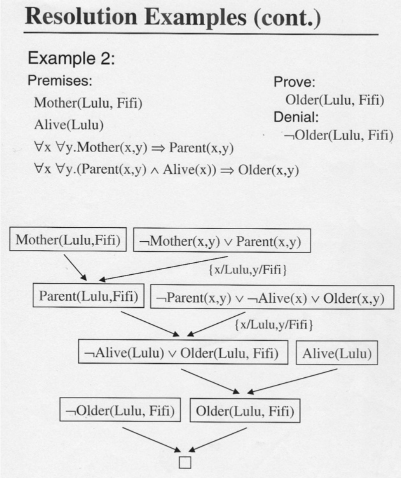
- **Cél**: Minél több pont összegyűjtése
- **Állapot**: Nyers pixeles kép
- **Cselekvés**: Bábu lépegetése
- **Jutalom**: Játék során kapott pontok

Megerősítéses tanulás - Go



- **Cél**: Győzelem
- **Állapot**: Tábla állása
- **Cselekvés**: Kő lerakása
- **Jutalom**: 1 a győztes meccs végén

Megerősítéses tanulás - Tételbizonyítás



- **Cél**: Tétel bebizonyítása
- **Állapot**: Axiómák és levezetett lemmák
- **Cselekvés**: Következtetési lépés
- **Jutalom**: 1 ha sikerült bizonyítást találni

Markov Decision Process (MDP)

A megerősítéses tanulás matematikai formalizmusa
Egy dinamikus rendszer működését írja le

- $\mathcal{S}$: állapotok halmaza
- $\mathcal{A}$: akciók halmaza
- $\mathcal{R}$: jutalom eloszlás: $(s \in \mathcal{S}, a \in \mathcal{A}) \rightarrow$ reward
- $\mathcal{P}$: állapotátmeneti eloszlás: $(s \in \mathcal{S}, a \in \mathcal{A}) \rightarrow$ next state
- γ: diszkont ráta

Markov Decision Process (MDP)

- Kezdőállapot mintavételezése: $s_0 \sim P(s_0)$
- $t = 0$ időpillanattól kezdve iterálunk, amíg kész nem vagyunk:
  - Az ágens választ egy a_t akciót
  - A környezetből mintavételezünk jutalmat $r_t \sim R(s_t, a_t)$
  - A környezetből mintavételezünk új állapotot: $s_{t+1} \sim P(s_t, a_t)$
  - Az ágens megkapja a jutalmat (r_t) és a következő állapotot (s_{t+1})
- Az ágensnek szüksége van egy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ függvényre, mely meghatározza, milyen akciót válasszon. Ezt hívjuk **policy** függvénynek.
- Cél: optimális policy megtalálása, mely maximalizálja a teljes jutalmat: $\sum_{t \geq 0} \gamma^t r_t$

Markov Decision Process (MDP)

A folyamat általában sztochasztikus, ezért az összes jutalom várható értékét maximalizáljuk

$$\pi^* = \arg\max_{\pi} \mathbb{E}[\sum_{t \geq 0} \gamma^t r_t | \pi]$$

Egy adott policy-t követve (s_i, a_i, r_i) hármasok sorozatát kapjuk, ezt hívjuk **trajektóriának**.

Egy állapot értéke

- **Value** függvény: $V^{\pi}(s) = \mathbb{E}[\sum_{t \geq 0} \gamma^t r_t | s_0 = s, \pi]$
- **Q-Value** függvény: $Q^{\pi}(s,a) = \mathbb{E}[\sum_{t \geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi]$

$Q^*(s,a) = \max_{\pi} \mathbb{E}[\sum_{t \geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi]$

$Q^*(s,a)$ kielégíti a **Bellman egyenlőséget**:

$Q^*(s,a) = \mathbb{E}_{s' \sim P(s,a), r \sim R(s,a)}[r + \gamma \max_{a'} Q^*(s', a') | s, a]$

Value Iteration algoritmus

Optimális policy-t iteratív frissítéssel közelítjük a Bellman egyenlet alapján:

$$Q_{i+1}(s,a) = \mathbb{E}_{s' \sim P(s,a), r \sim R(s,a)}[r + \gamma \max_{a'} Q_i(s',a') | s,a]$$

Initialize $V(s)$ to arbitrary values
Repeat
  For all $s \in S$
    For all $a \in \mathcal{A}$
      $Q(s,a) \leftarrow E[r|s,a] + \gamma \sum_{s' \in S} P(s'|s,a) V(s')$
    $V(s) \leftarrow \max_a Q(s,a)$
Until $V(s)$ converge

Value Iteration - nehézségek

- Q_i konvergál Q^*-hoz
- Nagyon lassan konvergál
- Rengeteg mintára van szükség, be kell járnia a teljes álapotteret
- Nem skálázható
- Ötlet: használjunk függvényapproximációt!

Deep q-learning algoritmus

- Használjunk mély neurális hálót approximációhoz!
- $Q(s, a; \Theta) \approx Q^*(s, a)$
- A tanítás során a Bellman hibát minimalizáljuk
- $y_i = \mathbb{E}_{s' \sim P(s,a), r \sim R(s,a)}[r + \gamma \max_{a'} Q(s', a'; \Theta_i) | s, a]$
- $L_i(\Theta_i) = \mathbb{E}_{s,a}[(y_i - Q(s, a; \Theta_i))^2]$

Deep q-learning pszeudokód

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Deep q-learning - nehézségek

- $Q^*(s, a)$ rendkívül komplikált lehet
- Az állapot tér rendkívül magas dimenziós lehet
- Sok esetben a policy ennél sokkal egyszerűbb
- Ötlet: Tanuljuk közvetlenül a policy-t!

Policy Gradiens Tanulás

- Veszünk egy paraméterezett policy osztályt
- $\Pi = \{\pi_\Theta, \Theta \in R^m\}$
- Egy policy jósága: $J(\Theta) = \mathbb{E}[\sum_{t \geq 0} \gamma^t r_t | \pi_\Theta]$
- Keressük az optimális Θ^* paramétert
- Gradiens emelkedéssel

REINFORCE algoritmus

Egy népszerű policy gradiens módszer.

$J(\Theta) = \mathbb{E}_{\tau \sim p(\tau,\Theta)}[r(\tau)] = \int_\tau r(\tau)p(\tau,\Theta)d\tau$ (τ egy trajektória)

$\nabla_\Theta J(\Theta) = \int_\tau r(\tau)\nabla_\Theta p(\tau,\Theta)d\tau$

Ez nem számolható/közelíthető hatékonyan. Helyette:

$\nabla_\Theta p(\tau,\Theta) = p(\tau,\Theta)\frac{\nabla_\Theta p(\tau,\Theta)}{p(\tau,\Theta)} = p(\tau,\Theta)\nabla_\Theta \log p(\tau,\Theta)$

$\nabla_\Theta J(\Theta) = \int_\tau (r(\tau)\nabla_\Theta \log p(\tau,\Theta))p(\tau,\Theta)d\tau$

$\nabla_\Theta J(\Theta) = \mathbb{E}_{\tau \sim p(\tau,\Theta)}[r(\tau)\nabla_\Theta \log p(\tau,\Theta))]$

Tudjuk közelíteni Monte Carlo mintavételezéssel.

REINFORCE algoritmus

Az átmeneti valószínűségeket nem feltétlenül ismerjük.

$p(\tau, \Theta) = \prod_{t \geq 0} p(s_{t+1}|s_t, a_t)\pi_\Theta(a_t|s_t)$

$\log p(\tau, \Theta) = \sum_{t \geq 0}(\log p(s_{t+1}|s_t, a_t) + \log \pi_\Theta(a_t|s_t))$

$\nabla_\Theta \log p(\tau, \Theta) = \sum_{t \geq 0} \nabla_\Theta \log \pi_\Theta(a_t|s_t)$

A gradiens kiszámításához nem kell tudjuk az átmeneti valószínűségeket. Egyetlen trajektória alapján a gradiens közelítése:

$$\nabla_\Theta J(\Theta) \approx \sum_{t \geq 0} r(\tau) \nabla_\Theta \log \pi_\Theta(a_t|s_t)$$

REINFORCE - nehézségek

- Nagy variancia
- Credit Assignment Problem: egy trajektória értéke mely akciókhoz köthető?
- Rengeteg adatot igényel a konvergenciához
- Varianca redukció egy aktívan kutatott kérdés

REINFORCE - variancia redukció

- Egy akcióhoz csak a trakjektória későbbi részéből származó jutalom tartozzon:
  $\nabla_\Theta J(\Theta) \approx \sum_{t \geq 0} (\sum_{t` \geq t} r_{t'}) \nabla_\Theta \log \pi_\Theta(a_t|s_t)$
- Diszkont ráta: elhanyagoljuk a hosszútávú hatásokat:
  $\nabla_\Theta J(\Theta) \approx \sum_{t \geq 0} (\sum_{t` \geq t} \gamma^{t'-t} r_{t'}) \nabla_\Theta \log \pi_\Theta(a_t|s_t)$
- Baseline modell $b(s)$: Egy trajektória értéke önmagában nem mindig értelmes. Kell hozzá egy viszonyítási alap.
  - Legegyszerűbb: múltbeli trajektóriák értékének mozgó átlaga.
  - Jobb: Adott s állapotból induló trajektóriák várható értéke.

  $\nabla_\Theta J(\Theta) \approx \sum_{t \geq 0} (\sum_{t` \geq t} \gamma^{t'-t} r_{t'} - b(s_t)) \nabla_\Theta \log \pi_\Theta(a_t|s_t)$

REINFORCE pszeudokód

Algorithm 1 "Vanilla" policy gradient algorithm

Initialize policy parameter θ, baseline b

for iteration$=1, 2, \dots$ **do**

  Collect a set of trajectories by executing the current policy

  At each timestep in each trajectory, compute

   the *return* $R_t = \sum_{t'=t}^{T-1} \gamma^{t'-t} r_{t'}$, and

   the *advantage estimate* $\hat{A}_t = R_t - b(s_t)$.

  Re-fit the baseline, by minimizing $\|b(s_t) - R_t\|^2$,

   summed over all trajectories and timesteps.

  Update the policy, using a policy gradient estimate $\hat{g}$,

   which is a sum of terms $\nabla_\theta \log \pi(a_t \mid s_t, \theta)\hat{A}_t$

end for

Actor - Critic algoritmus

- Policy gradiens tanulás és Q tanulás egyesítése
- Critic: $V^\pi(s)$: s állapotból induló trajektóriák várható értéke
- Actor: $Q^\pi(s, a)$: s állapotból az a akció várható értéke
- Advantage: $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$
- Célfüggvény: $\min_V \max_Q A^\pi(s, a)$
- Actor tanulás: policy gradiens módszer, viszont kisebb a variancia
- Critic tanulás: Value Iteration, viszont elég az Actor által generált mintákra szakosodnia