Hlboké učenie posilňovaním (hodnotové funkcie)

(Strojové učenie II)

M. Mach

Katedra kybernetiky a umelej inteligencie, FEI, TUKE

marec 2023

DNN ako aproximátor hodnotových funkcií

- Estimácia hodnôt
  - príznaky - mapovanie stavov na príznaky
    - v prechádzajúcom mapovanie bolo pevne dané, ale môže byť tiež parametrizované
    - pre získanie užitočných príznakov prispôsobených riešenej úlohe
  - hodnotové funkcie - mapovanie príznakov na hodnoty
    - tak ako v prechádzajúcom mapovanie parametrizované
- Aproximácia

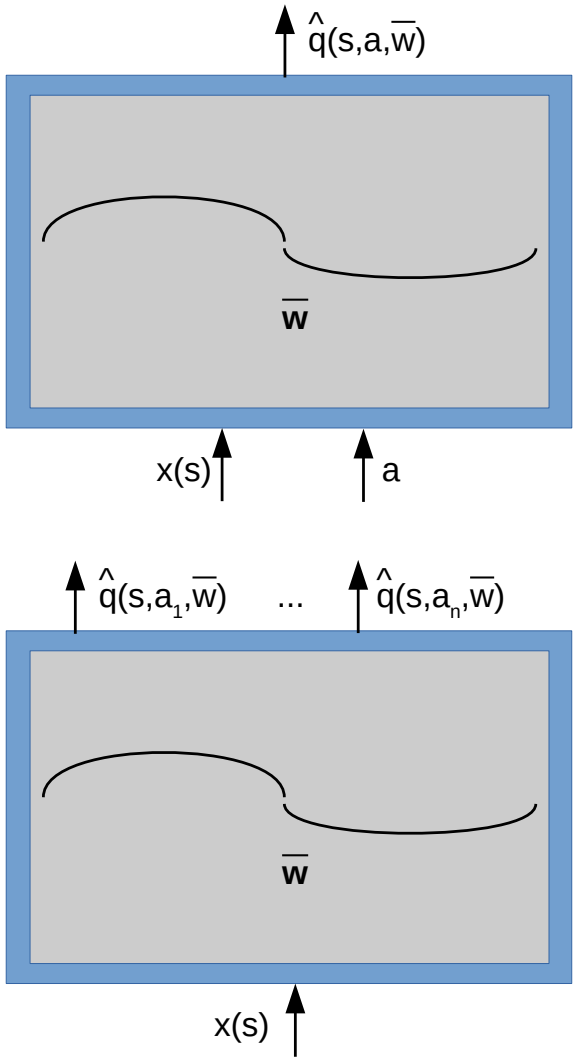
$$v_\pi(s) \approx \hat{v}(x(s), \bar{w}) \quad | \quad \hat{v}(x(s, \bar{w}_1), \bar{w}_2)$$
$$q_\pi(s, a) \approx \hat{q}(x(s, a), \bar{w}) \quad | \quad \hat{q}(x(s, a, \bar{w}_1), \bar{w}_2)$$

- Použitie ako nelineárny aproximátor

Aproximácia hodnôt

- Cieľ: nájsť $\bar{w}$ minimalizujúce
$$L(\bar{w}) = E_{s\sim\mu_\pi}[(v_\pi(s) - \hat{v}(s,\bar{w}))^2]$$
kde $\mu_\pi(s)$ je stacionárna distribúcia návštevnosti stavov (indukovaná politikou π a dynamikou $P^a_{ss'}$)
- riešenie pomocou GD
$$\Delta\bar{w} = -\tfrac{1}{2}\alpha\nabla L(\bar{w}) = \alpha E_{s\sim\mu_\pi}[(v_\pi(s) - \hat{v}(s,\bar{w}))\nabla\hat{v}(s,\bar{w})]$$
- riešenie pomocou SGD
$$\Delta\bar{w} = \alpha(U_t - \hat{v}(s,\bar{w}))\nabla\hat{v}(s,\bar{w})$$
kde U_t je odhad hodnoty kumulatívnej odmeny
  - DP: $U_t = E_\pi[R_{t+1} + \gamma\hat{v}(S_{t+1},\bar{w}_t)|S_t = s]$
  - MC: $U_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$
  - TD: $U_t = R_{t+1} + \gamma\hat{v}(S_{t+1},\bar{w}_t)$

Štruktúra aproximátora



- Typ siete
  - MLP - mapovanie hodnotovej funkcie
  - CNN - mapovanie príznakov a hodnotovej funkcie
- Príklad MLP

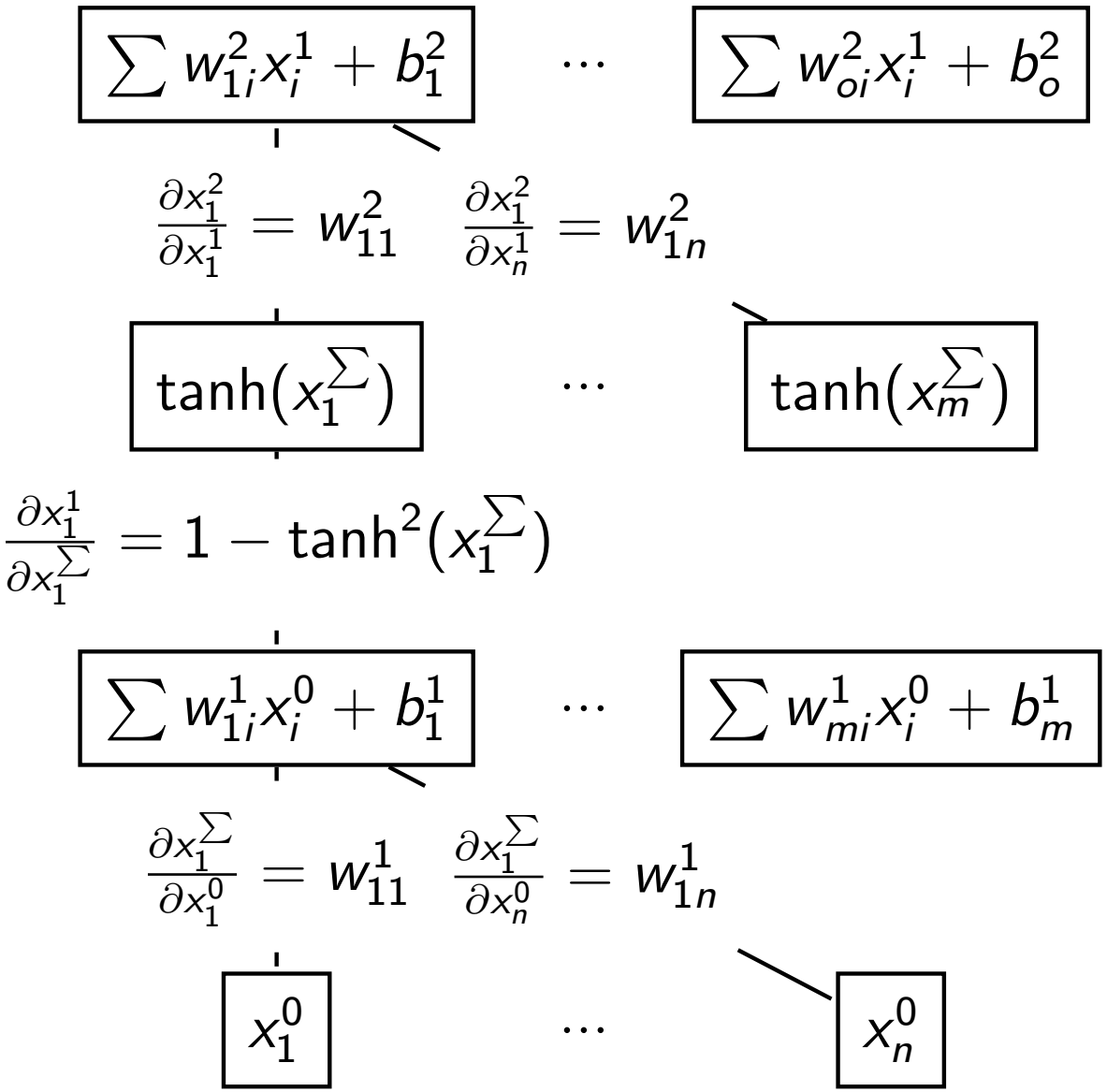
  $\hat{q}(s, a, \bar{w}) =$

  $W_2 * tanh(W_1 * x(s) + b_1) + b_2$

  kde učené parametre $\bar{w}$ sú

  $\bar{w} = \{W_1, b_1, W_2, b_2\}$
- Ako určiť gradient?

Počítanie gradientu $\nabla \hat{q}(s, a, \bar{w})$



- Určovanie gradientu
  - vytvorenie výpočtového graf
  - gradienty medzi uzlami
  - akumulácia gradientu pozdĺž ciest v grafe
- Automatické derivovanie

Hlboký Q-Learning

- Aktualizácia parametrov
  $$\Delta \bar{w} = \alpha(U_t - \hat{q}(S_t, A_t, \bar{w}))\nabla \hat{q}(S_t, A_t, \bar{w})$$
  kde
  $$U_t = R_{t+1} + \gamma \max_a \hat{q}(S_{t+1}, a, \bar{w}_t)$$
- Pseudo-loss
  $$L(\bar{w}) = \frac{1}{2}\left(R_{t+1} + \gamma \max_a \hat{q}(S_{t+1}, a, \bar{w}_t) - \hat{q}(S_t, A_t, \bar{w})\right)^2$$
  s ignorovaním závislosti $\hat{q}(S_{t+1}, a, \bar{w}_t)$ na $\bar{w}$
- Triáda nestability (aproximácia, bootstrapping, off-policy) plne v akcii

(Ne)kompatibilita DL a RL

- Hlboké učenie predpokladá
  - veľké množstvo trénovacích dát
  - metóda SGD vyžaduje vzorkovanie (príklady) i.i.d
    - výber príkladu do vzorky nezávisí od predošlých príkladov
    - príklady do vzorky sú vyberané z rovnakej distribúcie pravdepodobnosti (žiadna fluktuácia distribúcie)
  - použitie dávkového učenia poskytuje lepšie výsledky ako použitie samostatných príkladov
- Učenie posilňovaním
  - TD robí aktualizáciu váhovej funkcie po každom kroku (po každom príklade)
  - zmena exploračnej politiky má za následok zmenu distribúcie dát
  - zriedkavá a (niekedy značne) vzdialená odmena
  - za sebou nasledujúce vzorky sú silne korelované (za sebou nasledujúce kroky v získavanej sekvencii)

DQN (Deep Q-Network)

- V kontexte hrania Atari 2600 games
- Aproximátor = CNN
- Spätné prehrávanie skúsenosti
  - udržiava pamäť (limitovanej veľkosti) získaných skúsenosti
    - po zaplnení pamäti vypúšťa najstaršiu skúsenosť
  - z pamäti vyberá skúseností
    - výber viacerých skúseností - podpora dávkovej aktualizácie aproximátora
    - náhodný výber - redukcia korelácie medzi po sebe nasledujúcimi aktualizáciami
  - prechod z inkrementálneho prístupu na dávkový prístup
  - lepšie využitie dát - skúsenosť môže byť použitá viacnásobne
- Voľba off-policy - aktuálne parametre aproximátora môžu byť iné ako v čase generovania skúsenosti

Zvýšenie stability DQN

- Napriek prechodu z inkrementálneho učena na dávkové (spätné prehrávanie) - rezervy v stabilite
- Neskôr pridané
  - použitie dvoch sietí - hlavnej a cieľovej
    - hlavná sieť sa učí (stále sa mení)
    - cieľová sieť ostáva (po nejakú) dobu rovnaká - používa sa na určovanie odhadu
    - iba raz za nejaký čas sa cieľová sieť aktualizuje
  - obmedzenie δ_t
    - derivácia $\mid \delta \mid$ je iba $+-1$, zatiaľ čo derivácia δ^2 je 2δ (neobmedzená)
    - obmedzenie TD chyby δ spôsobom $\max(\min(\delta^2, 1), -1)$ spôsobí obmedzenie derivácie δ^2
    - korešponduje s tým, ak $L(\bar{w})$ mimo intervalu $< -1, 1 >$ je absolútna hodnota a v intervale kvadratická

Algoritmus DQN

Initialize replay memory D to capacity N
Initialize $\hat{q}(s, a, \bar{w})$ with random weights
Initialize target $\hat{q}(s, a, \bar{w}_T)$ with weights $\bar{w}_T = \bar{w}$

for episode $= 1, M$ do
  Initialize sequence with S_1
  for $t = 1, T$ do
    Choose A_t using policy derived from $\hat{q}(S_t, a, \bar{w})$ (e.g. ϵ-greedy)
    Take action A_t and observe R_{t+1} and S_{t+1}
    Store transition $(S_t, A_t, R_{t+1}, S_{t+1})$ in D
    Sample random minibatch of transitions $(S_j, A_j, R_{j+1}, S_{j+1})$ from D
    for each transition in minibatch

$$y_j = \begin{cases} R_{t+1} & \text{for terminal } S_{t+1} \\ R_{t+1} + \gamma \max_a \hat{q}(S_{t+1}, a, \bar{w}_T) & \text{for non-terminal } S_{t+1} \end{cases}$$

$$target_j = (y_j - \hat{q}(S_t, A_t, \bar{w}))^2$$

    end for
    Perform a gradient descent step on the selected minibatch
    Every C steps reset $\bar{w}_T = \bar{w}$
  end for
end for

Dvojité DQN

- Mäkká divergencia - v úvodnej etape hodnoty divergujú avšak po nejakej dobe sa zotavia
- Pre potlačenie tohto typu divergencie sa používa dvojité učenie (redukuje nadhodnocovaný odhad)
- Použitie dvoch sietí (ktoré v DQN už aj tak sú)

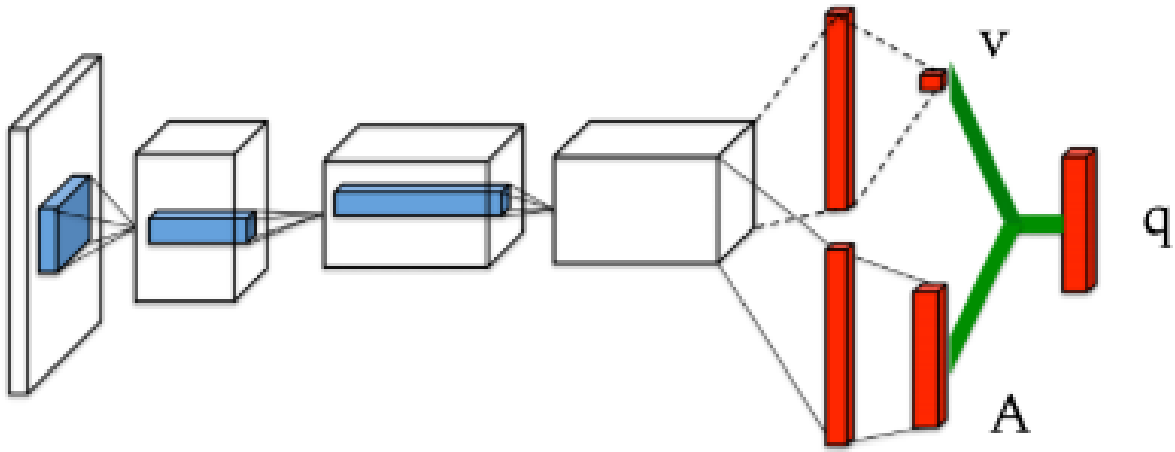
$$L(\bar{w}) = \frac{1}{2}\left(R_{t+1} + \gamma \hat{q}_T(S_{t+1}, arg\,\max_a \hat{q}(S_{t+1}, a, \bar{w}), \bar{w}_T) - \hat{q}(S_t, A_t, \bar{w})\right)^2$$

  - pracovná sieť
    - mení parametre $\bar{w}$ tak, aby sa hodnoty $\hat{q}$ približovali odhadnutému cieľu
  - cieľová sieť
    - používa sa pre odhad cieľa (ohodnotenie akcie) na základe bootstrappingu: $\max_a \hat{q}_T(s, a, \bar{w}_T)$
    - udržiava svoje parametre $\bar{w}_T$ fixované po dlhú dobu
    - mení hodnoty parametrov periodicky iba raz za čas (vymení svoje parametre za parametre pracovnej siete)

Prednostné prehrávanie skúsenosti

- DQN vzorkuje pamäť uniformne - skúsenosti sú prehrávané s tou frekvenciou, s akou boli získavané
  - žiadny ohľad na dôležitosť jednotlivých skúseností
- Väčšia priorita tým skúsenostiam, od ktorých sa očakáva väčší vplyv na učenie siete
  - priorita i-tej skúsenosti je $\mid \delta_i \mid$ kde δ_i je TD-chyba v čase vkladania skúsenosti do pamäte
  - priorita je aktualizovaná pri opätovnom prehrávaní
- Problémy
  - Strata rôznorodosti $\rightarrow$ stochastická prioritizácia
    - pravdepodobnosť i-tej skúsenosti $P(i) = \frac{|\delta_i|^\alpha}{\sum |\delta_j|^\alpha}$
  - Vzorkovací bias $\rightarrow$ vzorkovanie podľa dôležitosti
    - váha i-tej skúsenosti $w_i = \left(\frac{1}{N}\frac{1}{P(i)}\right)^\beta$
    - normalizácia váženej i-tej skúsenosti $\frac{w_i \delta_i}{\max_j w_j}$

Dueling DQN



©*Ziyu Wang et al:Dueling Network Architectures for Deep RL*

- Dve vetvy - oddelený odhad pre $\hat{v}(s)$ a zisk (zvýšenie/zníženie) pre jednotlivé akcie

$$\hat{q}(s, a, \bar{w}, \bar{\alpha}, \bar{\beta}) = \hat{v}(s, \bar{w}, \bar{\beta}) + (A(s, a, \bar{w}, \bar{\alpha}) - \max_{a'} A(s, a', \bar{w}, \bar{\alpha}))$$

- Náhrada maxima priemerom
  - lineárna implementácia
  - zlepšenie stability (pomalšia zmena ako pri max)
  - strata originálnej sémantiky pre $\hat{v}$ $(v(s) \neq \max_a q(s, a))$

Viackrokový odhad

- Odhad cieľa dopredu o n krokov

  $G_{t:t+n} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n max_a \hat{q}(S_{t+n}, a, \bar{w})$
- Väčší dôraz na dáta a menší na odhad (menej greedy)
- Odmena je sčasti on-policy a sčasti bootstrap
  - menšie spoliehanie sa na bootstrapping
  - redukuje pravdepodobnosť divergencie
- Aktualizácia aproximátora

  $$\Delta \bar{w} = \alpha(G_{t:t+n} - \hat{q}(S_t, A_t, \bar{w}))\nabla \hat{q}(S_t, A_t, \bar{w})$$

Šumové siete

- Explorační ϵ-greedy politika je problematická
  - keď je potrebné vykonať väčší počet akcií na to, aby bolo možné získať prvú odmenu
- Prvá lineárna vrstva siete je kombináciou dvoch vetiev - deterministickej a stochastickej
- Vrstva je definovaná

$$y = (b + Wx) + (b_{noisy} \odot \epsilon^b + (W_{noisy} \odot \epsilon^w)x)$$

kde $\odot$ je súčin po elementoch a ϵ^b a ϵ^w sú náhodné premenné ($\epsilon^{b|w} \sim N(0, 1)$)

- Časom sa sieť naučí ignorovať šumovú vetvu, ale rôznou rýchlosťou v rôznych častiach priestoru stavov
  - stavovo podmienená explorácia

Algoritmus Rainbow

- Vylepšenie základného DQN o vybranú sadu rozšírení
- Rainbow = DQN
  - + dvojité DQN
  - + prednostné prehrávanie skúseností
  - + dueling DQN
  - + viackrokový odhad
  - + distribučné učenie posilňovaním
  - + šumové siete
- Nevýhody zlúčenia mnohých techník
  - výpočtovo náročný - pomalý
  - príliš veľa parametrov treba vhodne nastaviť

Spojité akcie

- TD chyba

  $\delta_t = R_{t+1} + \gamma \max_a \hat{q}(S_{t+1}, a, \bar{w})$
- Výber najlepšej akcie je problematický (akcií je nekonečne veľa)
- Možné riešenia
  - diskretizácia priestoru akcií
  - optimalizácia
  - $\max_a Q(s, a) \approx max\{Q(s, a_1), Q(s, a_2), \ldots, Q(s, a_n)\}$ kde $(a_1, a_2, \ldots, a_n)$ sú generované podľa nejakej distribúcie (napr. uniformnej)
  - zmena prístupu - actor-critic algoritmy (najpopulárnejšie)
  - obmedzenie hodnotovej funkcie na triedu ľahko optimizovateľných funkcií

NAF (Normalized Advantage Functions)

- Hlavná myšlienka je zmeniť reprezentáciu $\hat{q}(s, a, \bar{w})$ tak, aby sa ľahko dalo učiť $argmax_a$
- Vychádza z princípu Dueling DQN

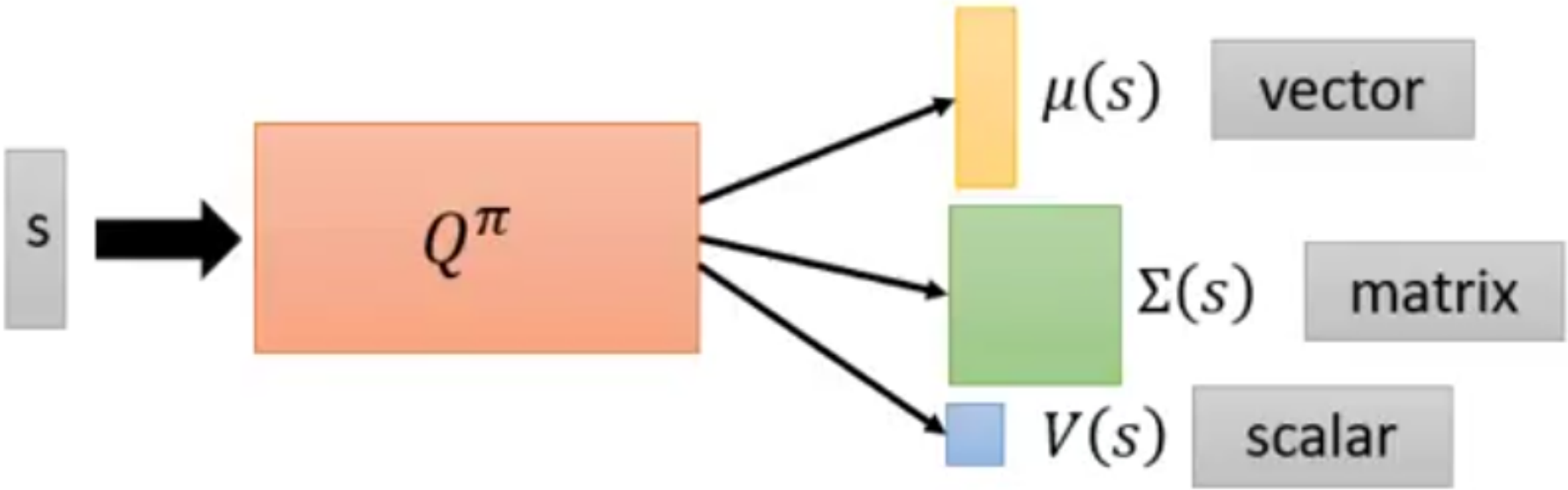
$$\hat{q}(s, a, \bar{w}^Q) = \hat{v}(s, \bar{w}^V) + A(s, a, \bar{w}^A)$$

- Sieť má dve časti - odhad hodnoty stavu a odhad zisku jednotlivých akcií
- Zisková funkcia

$$A(s, a, \bar{w}^A) = -\frac{1}{2}(a - \mu(s, \bar{w}^\mu))^T \sum(s, \bar{w}^P)(a - \mu(s, \bar{w}^\mu))$$

CIT Centrum inteligentných technológií

NAF - štruktúra siete



©*Hung-yi Lee: DRL Lecture 5: Q-learning (Continuous Action)*

- $\sum(s)$ je parametrizovaná $\sum(s) = L(s)L(s)^T$
  - $L(s)$ je dolná trojuholníková matica
    - na výstupe siete nie je priamo $\sum$ ale iba prvky z dolného trojuholníka matice L
  - $\sum(s)$ je pozitívne definitná matica $(\bar{x}^T L \bar{x} > 0)$
  - $A(s, a) \leq 0$

Činnosť algoritmu NAF

$$\hat{q}(s, a, \bar{w}^Q) = \hat{v}(s, \bar{w}^V) + A(s, a, \bar{w}^A)$$
$$A(s, a, \bar{w}^A) = -\frac{1}{2}(a - \mu(s, \bar{w}^\mu))^T \sum(s, \bar{w}^P)(a - \mu(s, \bar{w}^\mu))$$

- Maximá sa dajú určiť analyticky
  $$argmax_a \hat{q}(s, a, \bar{w}^Q) = \mu(s, \bar{w}^\mu)$$
  $$max_a \hat{q}(s, a, \bar{w}^Q) = \hat{v}(s, \bar{w}^V)$$
- TD chyba sa dá vyjadriť bez explicitného *max*
  $$\delta_t = R_{t+1} + \gamma \hat{v}(S_{t+1}, \bar{w}^{V'}) - \hat{q}(S_t, A_t, \bar{w}^A)$$
- GD minimalizuje na základe dávky vybranej z pamäte náhodným výberom
  $$\frac{1}{N} \sum_{i=1}^{N} (\delta_i)^2$$
- Váhy cieľovej siete sú aktualizované podľa hlavnej
  $$\bar{w}^{Q'} = \tau \bar{w}^Q + (1 - \tau) \bar{w}^{Q'}$$

Algoritmus NAF

Initialize normalized network randomly $\hat{q}(s, a, \bar{w}^Q)$
Initialize target network $\hat{q}'$ with weights $\bar{w}^{Q'} \leftarrow \bar{w}^Q$
Initialize replay buffer $D \leftarrow \emptyset$

for episode $= 1, M$ do
  Initialize a random process N for action exploration
  Receive initial state $S_1 \sim p(S_1)$
  for $t = 1, T$ do
    Select action $A_t = \mu(S_t, \bar{w}^\mu) + N_t$
    Take action A_t and observe R_{t+1} and S_{t+1}
    Store transition $(S_t, A_t, R_{t+1}, S_{t+1})$ in D
    for *iteration* $= 1, I$ do
      Sample a random minibatch of m transitions from D
      Set $y_i = R_{t+1} + \gamma \hat{v}'(S_{t+1}, \bar{w}^{V'})$
      Update $\bar{w}^Q$ by minimizing loss $L = \frac{1}{N} \sum_i (y_i - \hat{q}(S_t, A_t, \bar{w}^Q))^2$
      Update the target network $\bar{w}^{Q'} \leftarrow \tau \bar{w}^Q + (1 - \tau) \bar{w}^{Q'}$
    end for
  end for
end for

Čiastočná pozorovateľnosť

- Agentove pozorovanie mu nemusí poskytnúť všetku informáciu, potrebnú pre správne rozhodnutie $O_t \neq S_t^e$
- Formálne sa jedná o POMDP: (S, A, P, R, O, Ω)
  - O - množina možných pozorovaní ($O_t \in O$)
  - Ω - distribúcia pravdepodobnosti, s ktorou je generované pozorovanie z aktuálneho stavu ($O_t \sim \Omega(S_t)$)
- Agent musí svoje chápanie stavu S_t^a konštruovať z histórie pozorovaní (použije n posledných pozorovaní)
  - spojenie pozorovaní do jedného výsledného stavu $S_t = (O_t, O_{t-1}, \ldots, O_{t-n+1})$ - podstatné zväčšenie priestoru stavov
  - podpora časových radov cez rekurentné učenie

DRQN (Deep Recurrent Q-Network)

- Používa rekurentnú vrstvu (LSTM) na zapamätanie predošlých pozorovaní
- Štruktúra siete - tri druhy vrstiev
  - konvolučné vrstvy (predspracovanie pozorovaní a extrakcia príznakov)
  - rekurentná vrstva (pamätanie časovej zmeny)
  - plne prepojené vrstvy (predikcia Q hodnôt)
- Snaha čo najmenej zmeniť DQN
  - náhrada prvej plne prepojenej vrstvy LSTM vrstvou rovnakej veľkosti

CIT Centrum inteligentných technológií

DRQN - prehrávanie skúseností

- Náhodný výber jednotlivých krokov nepostačuje, pre učenie LSTM je potrebná sekvencia za sebou idúcich skúseností
- Dva možné typy aktualizácií
  - sekvenčné aktualizácie
    - epizódy sú vyberané náhodne
    - prehrávajú sa celé epizódy (začína na začiatku)
    - skrytý stav LSTM sa používa počas celého prehrávania
    - lepšie trénovanie LSTM
  - náhodné aktualizácie
    - epizódy vyberané náhodne, ale prehrá sa iba časť z nich
    - prehrávanie začína z náhodne vybraného stavu
    - dĺžka prehrávanej časti epizódy je obmedzená
    - skrytý stav LSTM sa nuluje pred každou epizódou (LSTM sa učí ťažšie dlhšie časové závislosti)
    - pokryje väčšiu časť stavového priestoru