Learn What Not to Learn: Action Elimination with Deep Reinforcement Learning

Дмитрий Пыркин

18 октября 2018 г.

DQN remind

Target

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

DQN remind

Target

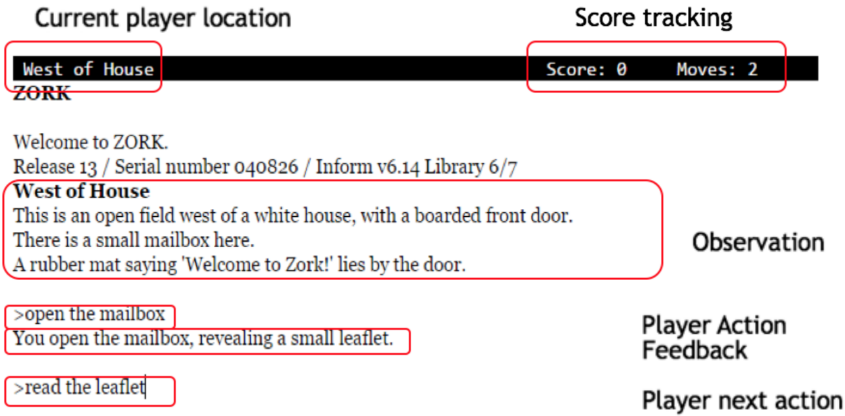
$$Q^*(s, a) = \mathbb{E}_{s'}\left[R(s, a, s') + \gamma \max_{a'} Q^*(s', a')|s, a\right]$$

Loss

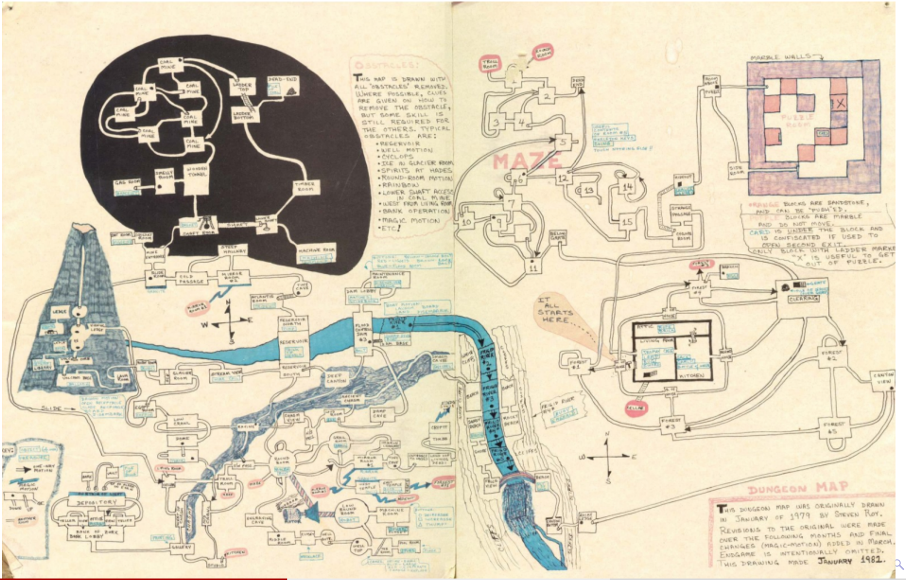
$$L(\theta) = \left(\underbrace{R(s, a, s') + \gamma \max_{a'} Q(s', a'; \theta^-)}_{\text{target}} - Q(s, a; \theta)\right)^2$$

Problem statement

1. Large action space
2. Many irrelevant actions


Current player location
Score tracking
West of House
Score: 0 Moves: 2
ZORK
Welcome to ZORK.
Release 13 / Serial number 040826 / Inform v6.14 Library 6/7
West of House
This is an open field west of a white house, with a boarded front door.
There is a small mailbox here.
A rubber mat saying 'Welcome to Zork!' lies by the door.
Observation
>open the mailbox
You open the mailbox, revealing a small leaflet.
Player Action Feedback
>read the leaflet
Player next action

The world of Zork



Sample complexity

Definition

Sample complexity of MDP is a minimum number of samples per state-action pair such that:

$$\mathbb{P}(|V(s) - V^*(s)| \geq \varepsilon) < \delta$$

Sample complexity

Definition

Sample complexity of MDP is a minimum number of samples per state-action pair such that:

$$\mathbb{P}(|V(s) - V^*(s)| \geq \varepsilon) < \delta$$

Complexity for MDP [2]

$$\widetilde{O}\left(\varepsilon^{-2}(1-\gamma)^{-3}\log\frac{1}{\delta}\right)$$

How to deal with it?

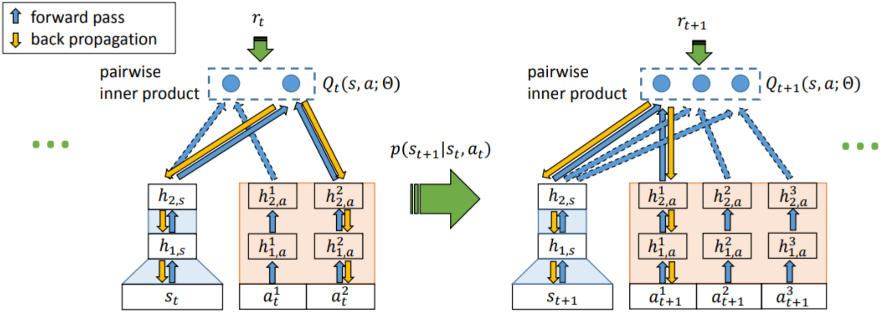
1. Hierarchical DQN.
2. State-action pair representation.
3. Factorizing the action space into binary subspace.
4. Embed discrete actions into continuous space.

DQN with an unbounded action space [3]

1. Learn representations for the states and actions with two different DNNs.

DQN with an unbounded action space [3]

1. Learn representations for the states and actions with two different DNNs.
2. Models the Q values as an inner product between representation vectors.



forward pass
back propagation
r_t
pairwise inner product
$Q_t(s, a; \Theta)$
$h_{2,s}$
$h_{1,s}$
s_t
$h^1_{2,a}$
$h^2_{2,a}$
$h^1_{1,a}$
$h^2_{1,a}$
a^1_t
a^2_t
$p(s_{t+1}|s_t, a_t)$
r_{t+1}
pairwise inner product
$Q_{t+1}(s, a; \Theta)$
$h_{2,s}$
$h_{1,s}$
s_{t+1}
$h^1_{2,a}$
$h^2_{2,a}$
$h^3_{2,a}$
$h^1_{1,a}$
$h^2_{1,a}$
$h^3_{1,a}$
a^1_{t+1}
a^2_{t+1}
a^3_{t+1}

Elimination signal

- After executing an action, the agent also observes a binary *elimination signal* $e(s, a)$.
- $e(s, a) = 1$ if action a may be eliminated in state s.

Contextual bandits

State representation

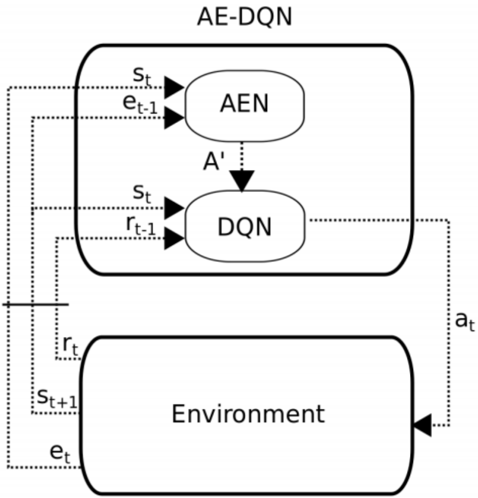
Let $x(s_t) \in \mathbb{R}^d$ be the feature representation of state s_t.

Linearity assumption

$$e_t(s_t, a) = {\theta_a^*}^T x(s_t) + \mathcal{N}(0, \sigma^2)$$

Linear regression task

$$||X_{t,a}\theta_{t,a} - E_{t,a}||_2^2 + \lambda||\theta_{t,a}||_2^2$$


AE-DQN
s_t
e_{t-1}
AEN
A'
s_t
r_{t-1}
DQN
a_t
r_t
s_{t+1}
Environment
e_t

Concurrent Learning

Action elimination network (AEN)

Find a minimal valid action space.

Concurrent Learning

Action elimination network (AEN)

Find a minimal valid action space.

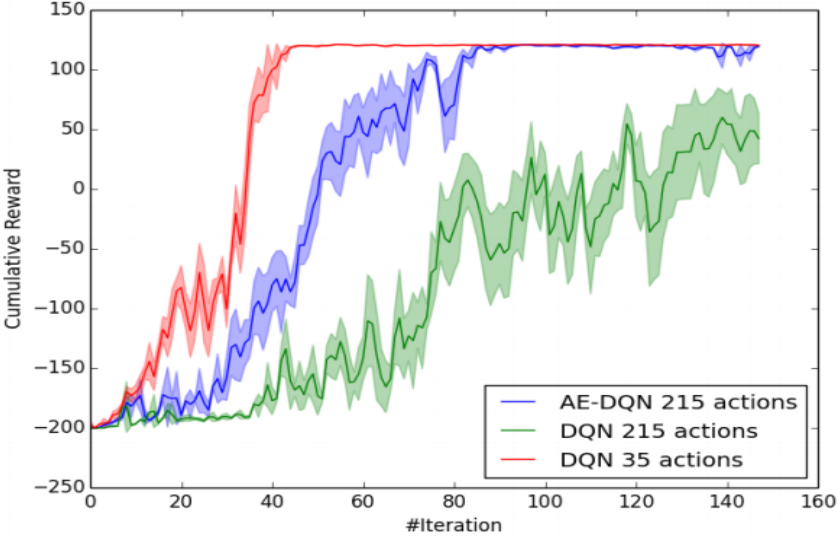
Deep Q-learning network (DQN)

Find an optimal policy.

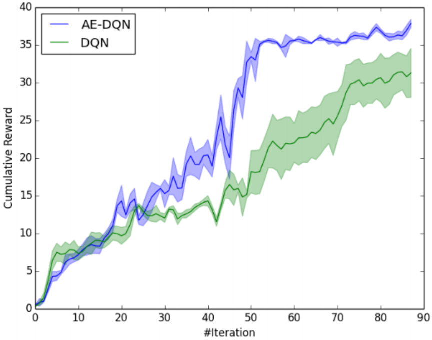
Algorithm

1. Get relevant actions $\mathcal{A}'$ from AEN.
2. Make action ε greedy from $\mathcal{A}'$.
3. Store $(s_t, a_t, s_{t+1}, r_t, e_t)$.
4. Update networks.

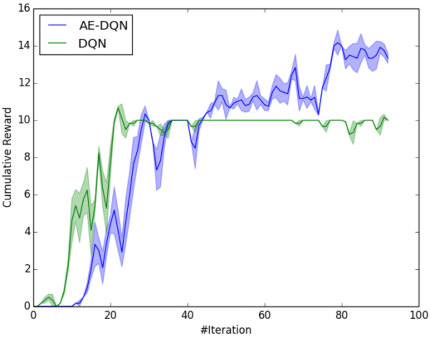
Troll Quest results



Zork results



Minimal action set (131)



"Open Zork"action set (1227)

References

[1] Tom Zahavy, Matan Haroush, Nadav Merlis, Daniel J. Mankowitz, Shie Mannor *Learn What Not to Learn: Action Elimination with Deep Reinforcement Learning*

[2] Lattimore, T., and Hutter, M. 2012. *Pac bounds for discounted mdps.*

[3] He, J.; Chen, J.; He, X.; Gao, J.; Li, L.; Deng, L.; and Ostendorf, M. 2015. *Deep reinforcement learning with an unbounded action space.*

Algorithm 1 deep Q-learning with action elimination

Input: $\epsilon, \beta, \ell, \lambda, C, L, N$
Initialize AEN and DQN with random weights ω, θ respectively, and set target networks Q^-, E^- with a copy of θ, ω
Define $\phi(s) \leftarrow$ LastLayerActivations$(E(s))$
Initialize Replay Memory D to capacity N
for t = 1,2,. . . , **do**
    a_t = ACT$(s_t, Q, E^-, V^{-1}, \epsilon, \ell, \beta)$
    Execute action a_t and observe $\{r_t, e_t, s_{t+1}\}$
    Store transition $\{s_t, a_t, r_t, e_t, s_{t+1}\}$ in D
    Sample transitions

$$\{s_j, a_j, r_j, e_j, s_{j+1}\}_{j=1}^m \in D$$

    $y_j =$ Targets $\left(s_{j+1}, r_j, \gamma, Q^-, E^-, V^{-1}, \beta, \ell\right)$
    $\theta = \theta - \nabla_\theta \sum_j \left(y_j - Q(s_j, a_j; \theta)\right)^2$
    $\omega = \omega - \nabla_\omega \sum_j \left(e_j - \mathrm{E}(s_j, a_j; \omega)\right)^2$
    If $(t \bmod C) = 0 : Q^- \leftarrow Q$
    If $(t \bmod L) = 0 :$

$$E^-, V^{-1} \leftarrow \text{AENUpdate}(E, \lambda, D)$$

end for

function ACT$(s, Q, E, V^{-1}, \epsilon, \beta, \ell)$
    $A' \leftarrow \{a : E(s)_a - \sqrt{\beta\phi(s)^T V_a^{-1}\phi(s)} < \ell\}$
    With probability ϵ, return Uniform(A')
    Otherwise, return $\arg\max_{a \in A'} Q(s, a)$
end function
function TARGETS$(s, r, \gamma, Q, E, V^{-1}, \beta, \ell)$
    if s is a terminal state **then** return r **end if**
    $A' \leftarrow \{a : E(s)_a - \sqrt{\beta\phi(s)^T V_a^{-1}\phi(s)} < \ell\}$
    return $(r + \gamma \max_{a \in A'} Q(s, a))$
end function
function AENUPDATE(E^-, λ, D)
    for $a \in A$ **do**
        $V_a^{-1} = \left(\sum_{j:a_j=a} \phi(s_j)\phi(s_j)^T + \lambda I\right)^{-1}$
        $b_a = \sum_{j:a_j=a} \phi(s_j)^T e_j$
        Set LastLayer$(E_a^-) \leftarrow V_a^{-1} b_a$
    end for
    return E^-, V^{-1}
end function