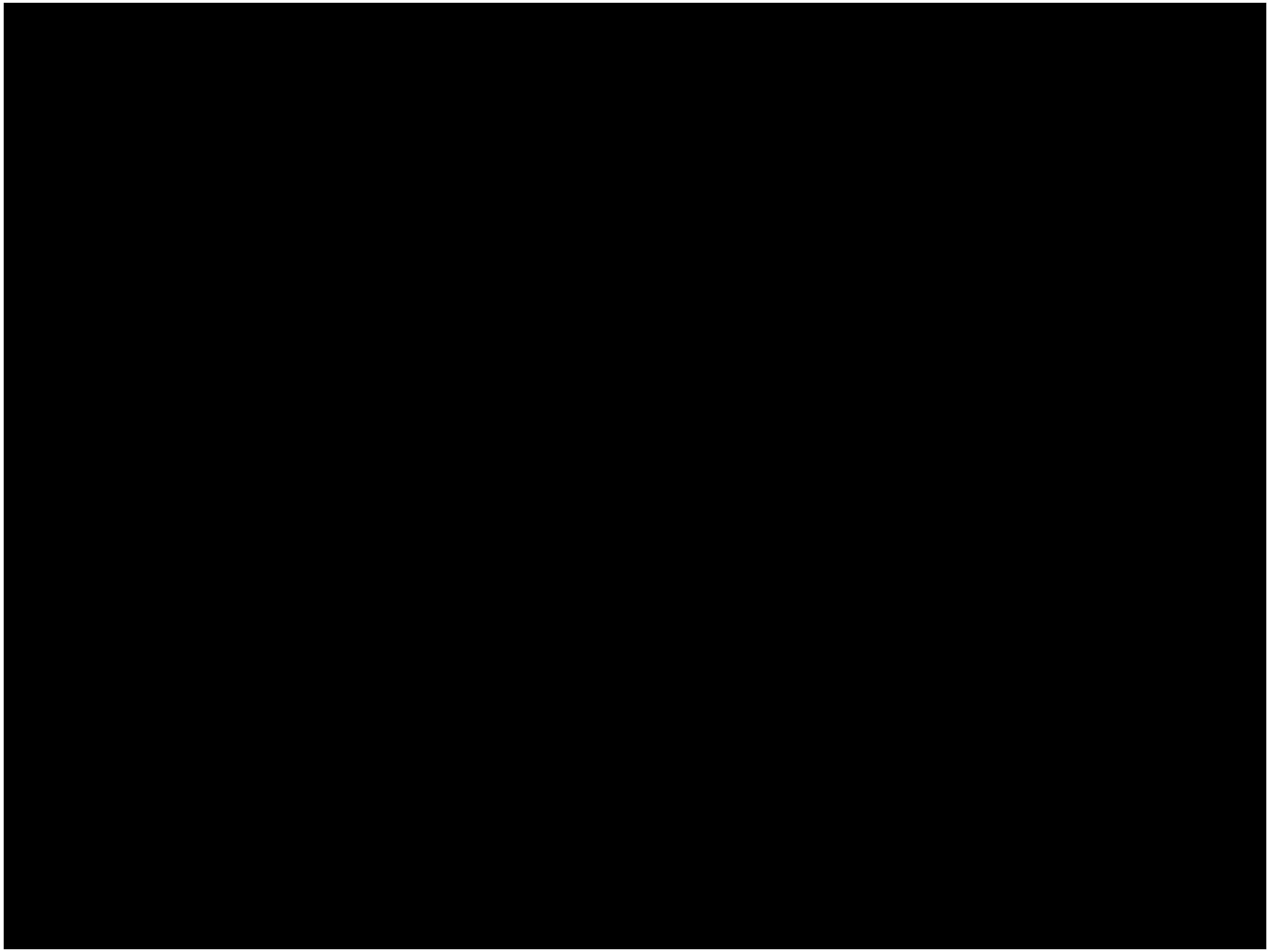
REINFORCEMENT LEARNING

Larry Page: Where’s Google going next?

DeepMind's DQN playing Breakout



Contents

- Introduction to Reinforcement Learning
- Deep Q-Learning

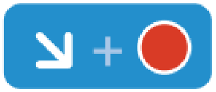
INTRODUCTION TO REINFORCEMENT LEARNING

Contents

- Reinforcement Learning
- Markov Decision Process
- Value function
- Bellman equation
- Action value function (Q-function)

Supervised Learning

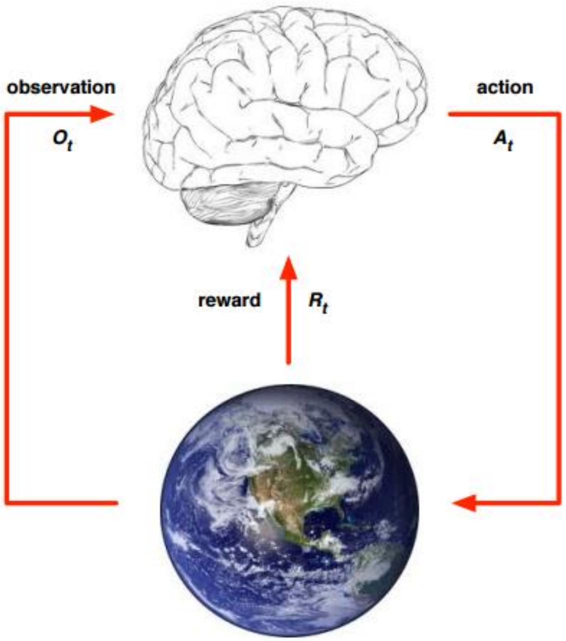
- Training samples: $\{(x_i, y_i)\}$
- Training goal:
  - To find a function $y_i \simeq f(x_i)$
- Atari game example:

$(x_i, y_i) = ($  ,  $)$

Game state   Joystick control

Reinforcement Learning

- Reinforcement learning is an area of machine learning concerned with how software **agents** ought to take actions in an environment so as to maximize some notion of cumulative reward.



Atari Example

Reinforcement Learning

- Learning from interaction
- Goal-oriented learning
- Learning about, from, and while interacting with an external environment
- Learning what to do—how to map situations to actions—so as to maximize a numerical reward signal

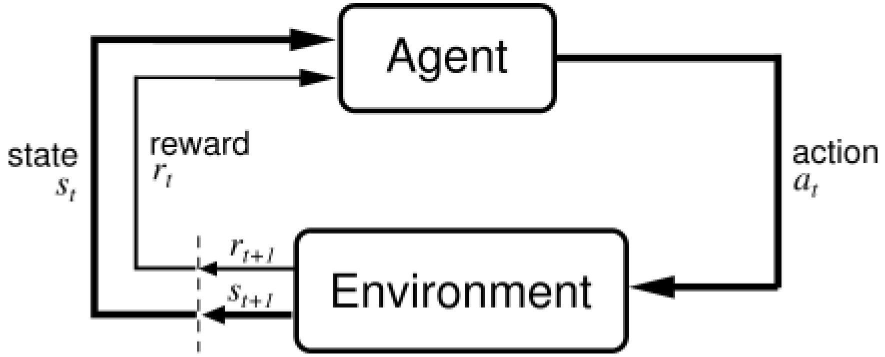
Key Features of RL

- Learner is not told which actions to take
- Trial-and-Error search
- Possibility of delayed reward (sacrifice short-term gains for gre ater long-term gains)
- The need to explore and exploit
- Considers the whole problem of a goal-directed agent interactin g with an uncertain environment

THEORY

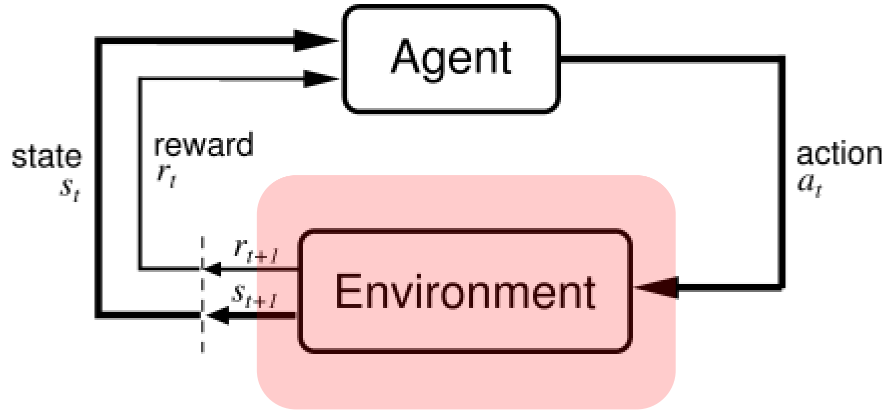
Reinforcement Learning Setting

- S – set of states
- A – set of actions
- $R : S \times A \to R$ – reward for given state and action.



Reinforcement Learning Setting

- The agent-environment interaction in reinforcement learning



$$P\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t, s_{t-1}, a_{t-1}, \ldots, s_0, a_0\}$$

Reward

- Discount rate: $\gamma \in [0,1)$ – It is the discount factor, which represents the difference in importance between future rewards and present rewards.

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

Markov Decision Process

- Markov Decision Process
  - **a reinforcement learning task** that satisfies the **Markov Property**

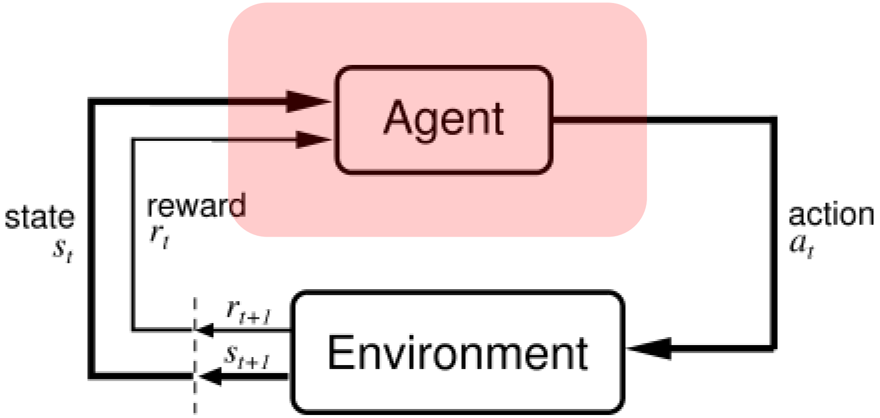
$$P\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t, s_{t-1}, a_{t-1}, \dots, s_0, a_0\}$$
$$= P\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t\}$$

$$P_{ss'}^{a} = P\{s_{t+1} = s' \mid s_t = s, a_t = a\}$$

$$R_{ss'}^{a} = E\{r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s'\}$$

Policy

- Policy π
  - A policy π is a mapping from each state, $s \in S$, and action $a \in A(s)$, to the probability $\pi(s, a)$ of tacking action a when in state s.



Value Functions

- State-value function for policy π.

$$V^{\pi}(s) = E\{R_t | s_t = s\} = E\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\}$$

- Action-value function for policy π.

$$Q^{\pi}(s,a) = E\{R_t | s_t = s, a_t = a\} = E\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\}$$

Optimal Value Functions

- $V^*(s) = \max_\pi V^\pi(s)$

- $Q^*(s,a) = \max_\pi Q^\pi(s,a)$
  $= E\{r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a\}$

Bellman Equation

- Bellman Equation for V^{π}
  - $V^{\pi}(s) = \sum_a \pi(a,s) \sum_{s'} P_{ss'}^{a} \left[R_{ss'}^{a} + \gamma V^{\pi}(s')\right]$
- Bellman Equation for $V^*(s)$
  - $V^*(s) = \max_a \sum_{s'} P_{ss'}^{a} \left[R_{ss'}^{a} + \gamma V^*(s')\right]$
- Bellman equation for $Q^*(s,a)$
  - $Q^*(s,a) = \sum_{s'} P_{ss'}^{a} \left[R_{ss'}^{a} + \gamma \max_{a'} Q^*(s',a')\right]$

Bellman Equation (Fixed policy)

- Bellman Equation for V^{π}: $a = \pi(s)$
  - $V^{\pi}(s) = \sum_{s'} P_{ss'}^{a} R_{ss'}^{a} + \gamma \sum_{s'} P_{ss'}^{a} V^{\pi}(s') = \boldsymbol{R(s,a)} + \gamma \sum_{s'} P_{ss'}^{a} V^{\pi}(s')$

- Bellman Equation for $V^{*}(s)$: $a = \pi(s)$
  - $V^{*}(s) = \max_{a} \sum_{s'} P_{ss'}^{a} \left[R_{ss'}^{a} + \gamma V^{*}(s')\right] = \boldsymbol{R(s,a)} + \gamma \max_{a} \sum_{s'} P_{ss'}^{a} V^{*}(s')$

- Bellman equation for $Q^{*}(s,a)$
  - $Q^{*}(s,a) = \sum_{s'} P_{ss'}^{a} \left[R_{ss'}^{a} + \gamma \max_{a'} Q^{*}(s',a')\right]$

LEARNING METHOD

A COMPLETE MODEL OF THE ENVIRONMENT'S DYNAMICS IS GIVEN

Policy Evaluation

Policy Evaluation: for a given policy π, compute the state-value f unction V^{π}

- Recall:
  - $V^{\pi}(s) = \sum_a \pi(a,s) \sum_{s'} P_{ss'}^{a} \left[R_{ss'}^{a} + \gamma V^{\pi}(s') \right]$
    - A system of $|S|$ simultaneous linear equations

Policy Evaluation

Input π, the policy to be evaluated

Initialize $V(s) = 0$, for all $s \in \mathcal{S}^+$

Repeat

$\quad \Delta \leftarrow 0$

$\quad$ For each $s \in \mathcal{S}$:

$\quad\quad v \leftarrow V(s)$

$\quad\quad V(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s') \right]$

$\quad\quad \Delta \leftarrow \max(\Delta, |v - V(s)|)$

until $\Delta < \theta$ (a small positive number)

Output $V \approx V^\pi$

Policy Improvement

- Suppose we have computed V for a deterministic policy π.
- For a given state s, would it be better to do an action $a \neq \pi(s)$?
- The value of doing a in state s is:

  - $$\begin{aligned} Q^{\pi}(s,a) &= \sum_{s'} P_{ss'}^{a} \left[R_{ss'}^{a} + \gamma V^{\pi} \sum_{a'} \pi(s',a') Q^{\pi}(s',a') \right] \\ &= \sum_{s'} P_{ss'}^{a} \left[R_{ss'}^{a} + \gamma V^{\pi}(s') \right] \end{aligned}$$

- It is better to take an action a for state s if and only if
  - $Q^{\pi}(s,a) > V^{\pi}(s)$

Policy Iteration

1. Initialization

   $V(s) \in \Re$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation

   Repeat

   $\quad \Delta \leftarrow 0$

   $\quad$ For each $s \in \mathcal{S}$:

   $\quad\quad v \leftarrow V(s)$

   $\quad\quad V(s) \leftarrow \sum_{s'} \mathcal{P}_{ss'}^{\pi(s)} \left[\mathcal{R}_{ss'}^{\pi(s)} + \gamma V(s') \right]$

   $\quad\quad \Delta \leftarrow \max(\Delta, |v - V(s)|)$

   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement

   policy-stable $\leftarrow$ true

   For each $s \in \mathcal{S}$:

   $\quad b \leftarrow \pi(s)$

   $\quad \pi(s) \leftarrow \arg\max_a \sum_{s'} \mathcal{P}_{ss'}^{a} \left[\mathcal{R}_{ss'}^{a} + \gamma V(s') \right]$

   $\quad$ If $b \neq \pi(s)$, then *policy-stable* $\leftarrow$ false

   If *policy-stable*, then stop; else go to 2

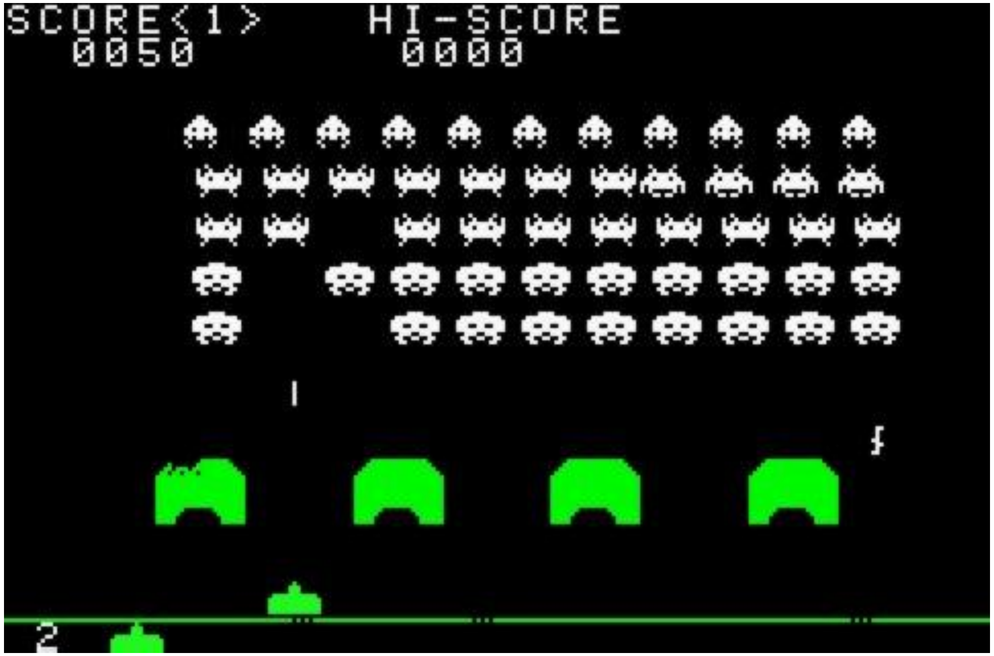
Other methods

- ….
- ….

DEEP–Q LEARNING

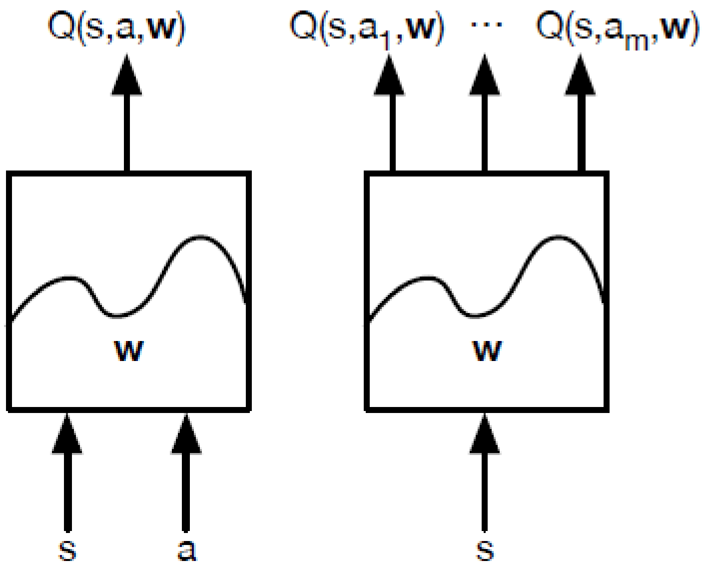
Introduction

- We want to perform human-level control by using deep reinforcement learning.
- Reinforcement learning apply to Atari 2600 platforms presentative classic game.



Q-Networks

- Represent action value function by Q-network with weights w
  - $Q(s, a; w) \simeq Q^*(s, a)$



Q–Learning

- Optimal Q-values should obey Bellman equation
  - Bellman equation for $Q^*(s,a)$
    - $Q^*(s,a) = \sum_{s'} P_{ss'}^{a} \left[R_{ss'}^{a} + \gamma \max_{a'} Q^*(s',a')\right]$
    - $Q^*(s,a;w) = \sum_{s'} P_{ss'}^{a} \left[R_{ss'}^{a} + \gamma \max_{a'} Q^*(s',a';w)\right] = r + \gamma \max_{a'} Q^*(s',a';w)$

- Treat right hand side $r + \gamma \max_{a'} Q^*(s',a';w)$ as a target
- Minimize MSE loss by stochastic gradient descent

$$I = \left(r + \gamma \max_{a} Q(s', a', \mathbf{w}) - Q(s, a, \mathbf{w})\right)^2$$

- Converges to Q^* using table lookup representation
- But diverges using neural networks due to:
  - Correlations between samples
  - Non-stationary targets

Deep Q-Networks

To remove correlations, build data-set from agent's own experience

s_1, a_1, r_2, s_2
s_2, a_2, r_3, s_3
s_3, a_3, r_4, s_4
$\ldots$
$s_t, a_t, r_{t+1}, s_{t+1}$

$s_t, a_t, r_{t+1}, s_{t+1} \rightarrow$

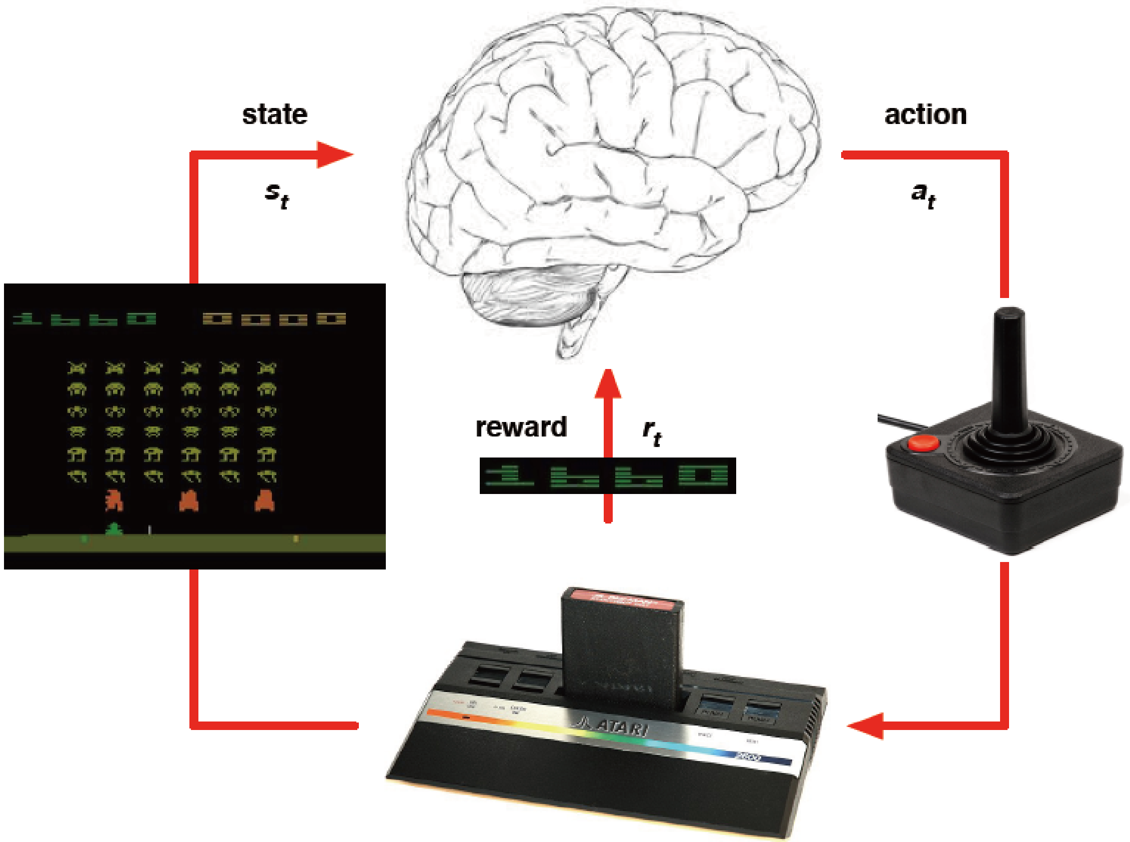
$\rightarrow \quad s, a, r, s'$

Sample experiences from data-set and apply update

$$I = \left(r + \gamma \max_{a'} Q(s', a', \mathbf{w}^{-}) - Q(s, a, \mathbf{w}) \right)^2$$

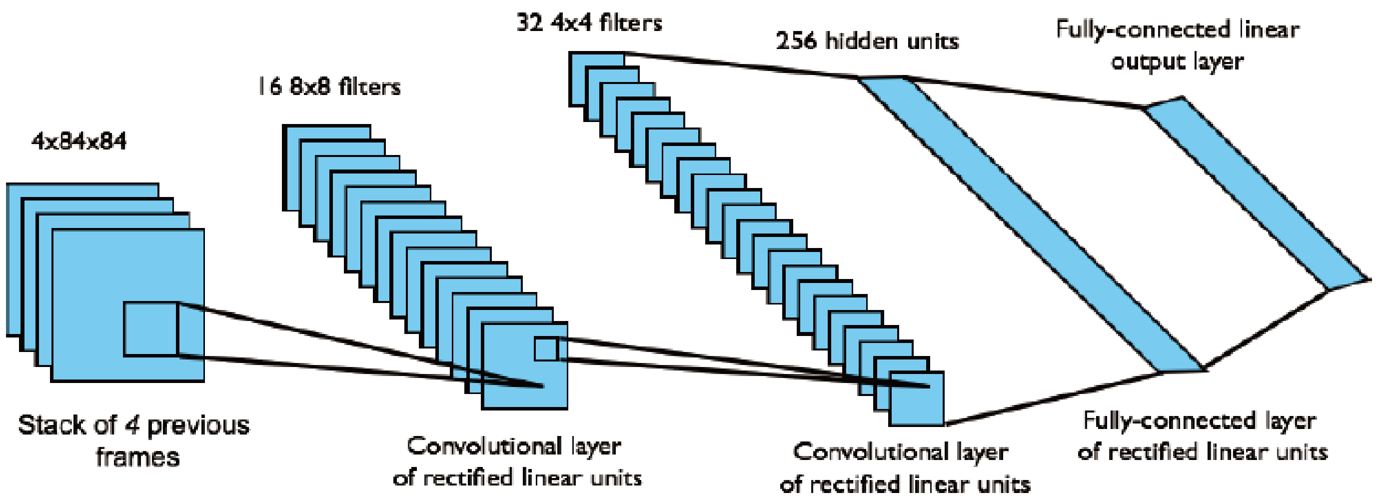
To deal with non-stationarity, target parameters $\mathbf{w}^{-}$ are held fixed

Deep Reinforcement Learning in Atari



DQN in Atari

- End-to-end learning of values $Q(s, a)$ from pixels s
- Input state s is stack of raw pixels from last 4 frames
- Output is $Q(s, a)$ for 18 joystick/button positions
- Reward is change in score for that step



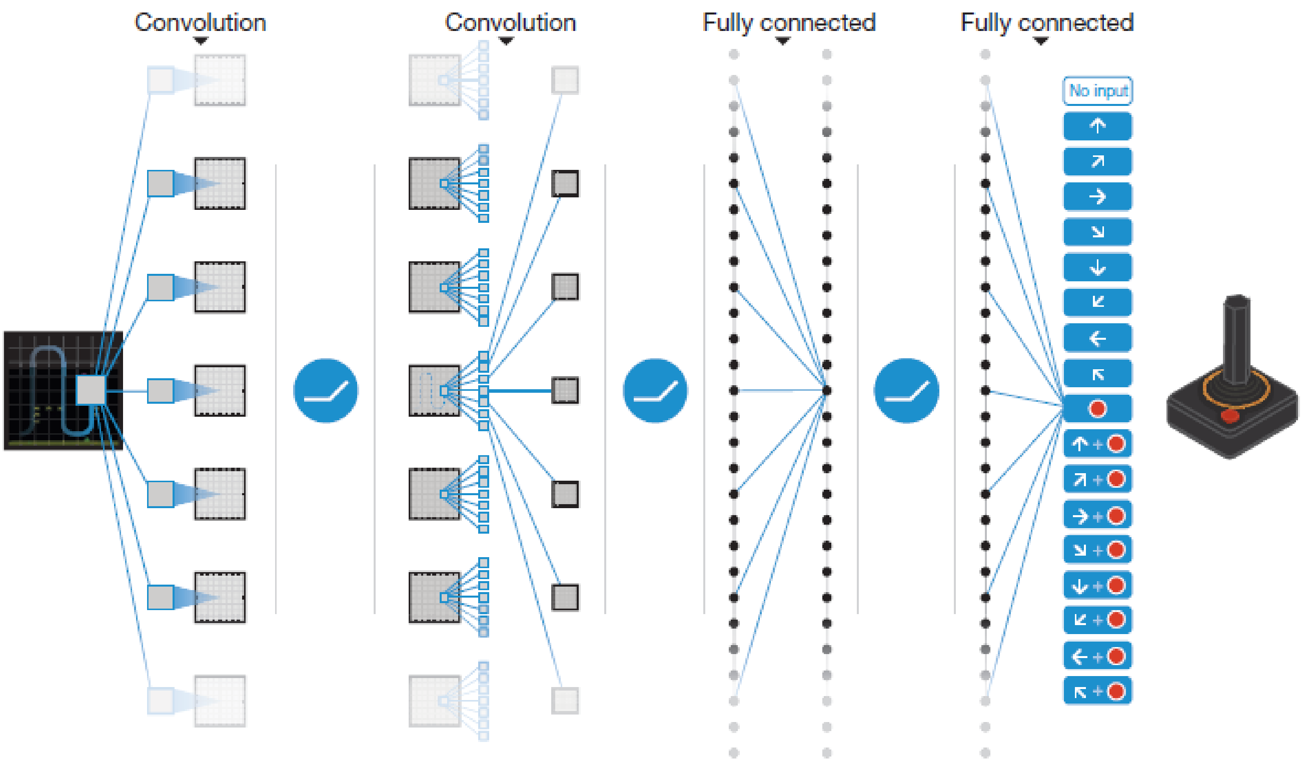
Network architecture and hyperparameters fixed across all games

Algorithm

Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$
For episode $= 1, M$ **do**
    Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
    For $t = 1$,T **do**
        With probability ε select a random action a_t
        otherwise select $a_t = \operatorname{argmax}_a Q(\phi(s_t), a; \theta)$
        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 D
        Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D
        Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^{-}\right) & \text{otherwise} \end{cases}$
        Perform a gradient descent step on $\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$ with respect to the network parameters θ
        Every C steps reset $\hat{Q} = Q$
    End For
End For

ϕ means stacking recent 4 images.

Architecture



From Pixels to Actions: Human-level control through Deep Reinforcement Learning

Posted: Wednesday, February 25, 2015

g+1 622 Tweet 232 Like 407

Posted by Dharshan Kumaran and Demis Hassabis, Google DeepMind, London