Generalization and Function Approximation

COMP4211

THE DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

Generalization and Function Approximation

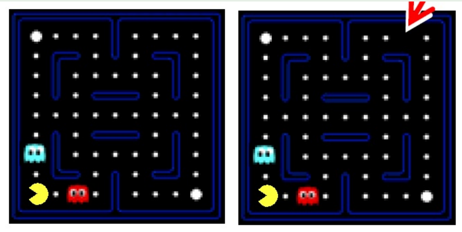
Example (9×9 Go)

- $|S| = 10^{38}$ and $|A| = 81$
- too many states to visit them all in training
- too many states to hold the Q-tables in memory

what should we do?

Generalize from Experience

Example



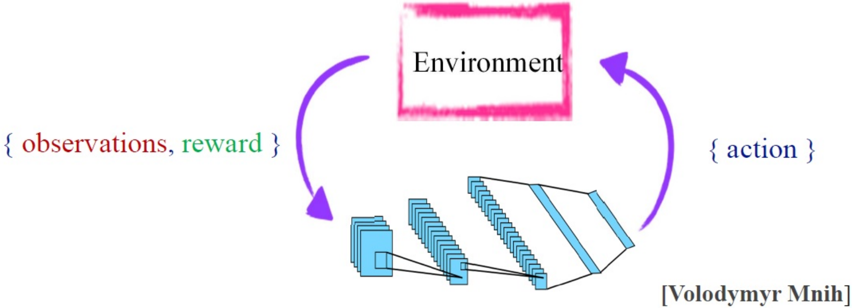
good or bad?

- learn about a few states from experience
- generalize that experience to new, similar states

Replace $\hat{Q}$ table with a function approximator

Example (function approximator= neural net)

deep reinforcement learning



Feature-Based Representations

describe a state using a vector of features

- features are functions from states to real numbers that capture important properties of the state

Example

- distance to closest ghost
- distance to closest dot
- number of ghosts
- $1/(\text{dist to dot})^2$

- can also describe a Q-state (s, a) with features
  - e.g., action moves closer to food

Function Approximation

linear feature functions

$$Q(s, a) = w_1 f_1(s, a) + w_2 f_2(s, a) + \cdots + w_n f_n(s, a)$$

- experience is summed up in only a few numbers

how to update w_i's?

Q-learning

- $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[\text{difference}]$
- difference $= r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)$

Function Approximation...

$$\begin{aligned} \text{error}(w) &= \frac{1}{2}\left(y - \sum_k w_k f_k(x)\right)^2 \\ \frac{\partial \text{error}(\text{w})}{\partial w_i} &= -\left(y - \sum_k w_k f_k(x)\right) f_i(x) \\ w_i &\leftarrow w_i + \alpha \left(y - \sum_k w_k f_k(x)\right) f_i(x) \end{aligned}$$

In Q-learning

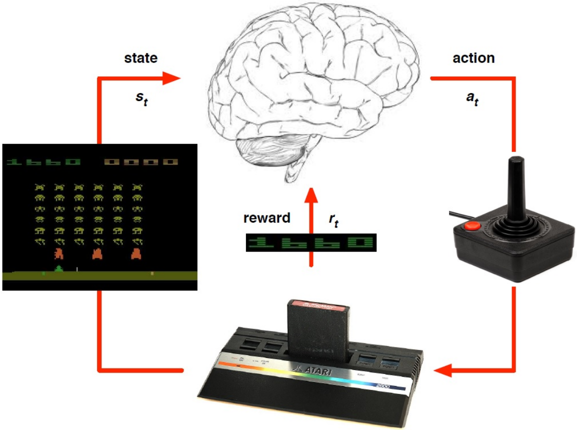
- target: $r_t + \gamma \max_a Q(s_{t+1}, a)$
- prediction: $Q(s_t, a_t)$
- $w_i \leftarrow w_i + \alpha[\text{difference}] f_i(s_t, a_t)$

Q-learning with Linear Approximators

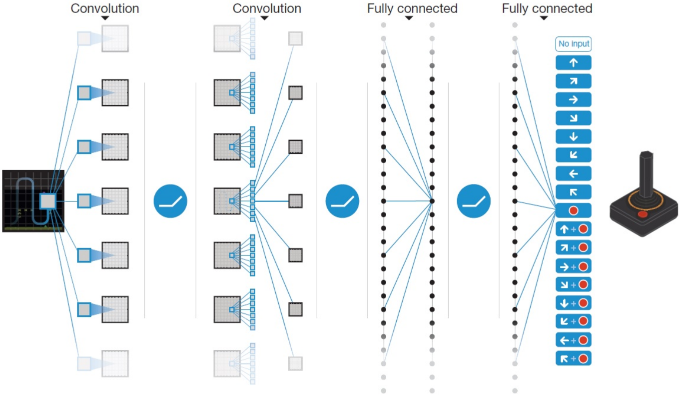
begin
  initialize parameter values;
  repeat
    select an action a and execute it;
    receive immediate reward r;
    observe the new state s';
    difference $= r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)$;
    for $i = 1$ **to** n **do**
      $w_i \leftarrow w_i + \alpha[\text{difference}] f_i(s_t, a_t)$;
    end
    $s \leftarrow s'$;
  until;
end

Deep Q-Network (DQN)

- used on Atari games

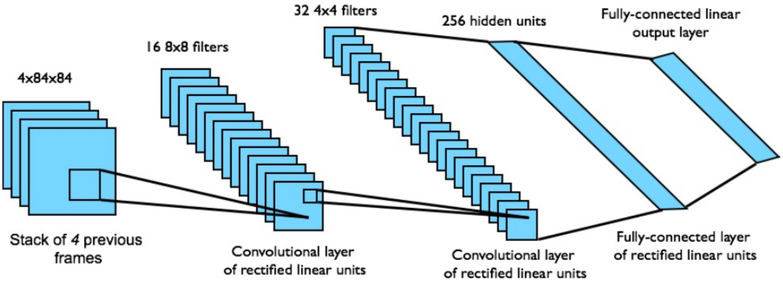


Deep Q-Network in Atari...



- compute Q-values for all possible actions in a given state with only a single forward pass through the network

Deep Q-Network in Atari...



- learning of values $Q(s, a)$ from pixels
- input state s: stack of raw pixels from last 4 frames
- output: $Q(s, a)$ for 18 joystick/button positions
- reward: change in score for that step

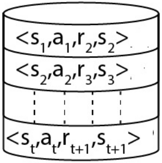
Problem

at each time-step, agent experience $e_t = (s_t, a_t, r_t, s_{t+1})$

- each experience is only used once
- experiences are highly correlated

Experience Replay

- store e_t's in a data set $D_t = \{e_1, \ldots, e_t\}$, pooled over many episodes into a replay memory



- apply Q-learning updates to samples of experience, (s, a, r, s') drawn at random from the pool of stored samples

advantages

- each step of experience is potentially used in many weight updates $\rightarrow$ greater data efficiency
- randomizing the samples breaks sample correlations
- empirically, avoid oscillations or divergence in the parameters

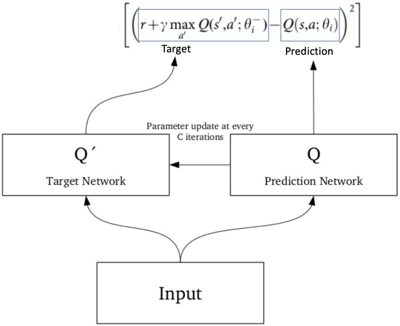
Another Problem

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t(r_{t+1} + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t))$$

- Q is used in both evaluating (s, a) and also in action selection
- possibly leading to oscillations or divergence of the policy

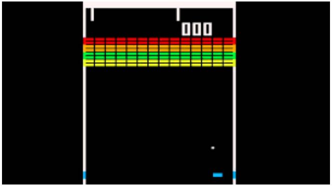
use a separate network for generating the targets in the Q-learning update

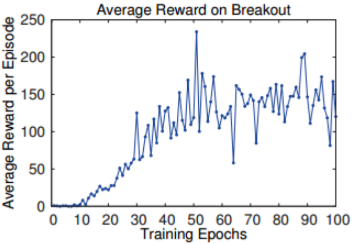
$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t(r_{t+1} + \gamma \max_a \hat{Q}_t(s_{t+1}, a) - Q_t(s_t, a_t))$$

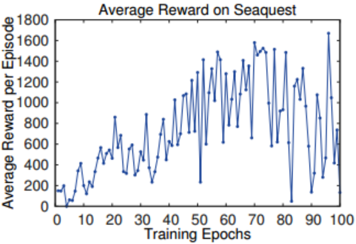


- use $\hat{Q}$ for generating the Q-learning targets
- every C updates we clone the network Q to obtain a target network $\hat{Q}$

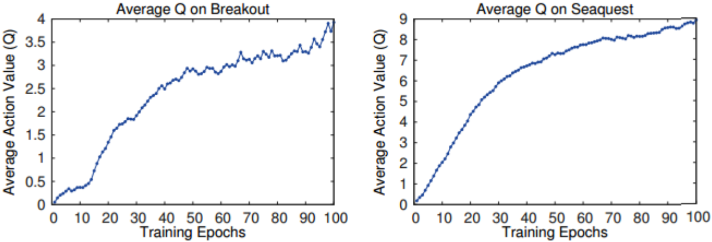
Results: Atari Games



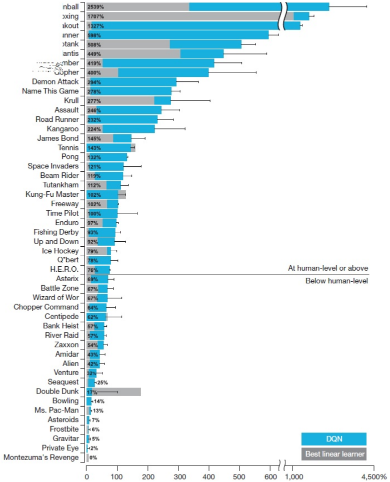




Results: Atari Games...



Performance



- performance is normalized with respect to a professional human games tester (100% level) and random play (0% level)