Lecture 7: DQN

Reinforcement Learning with TensorFlow&OpenAI Gym
Sung Kim <hunkim+ml@gmail.com>
Q-function Approximation: Q-Nets

(1) state, $s$

(2) quality (reward)
for all actions
(eg, $[0.5, 0.1, 0.0, 0.8]$)
LEFT: 0.5,
RIGHT 0.1
UP: 0.0,
DOWN: 0.8)
Q-Nets are unstable
Convergence

\( \hat{Q} \) denote learner’s current approximation to \( Q \).

\[
\min_{\theta} \sum_{t=0}^{T} \left[ \hat{Q}(s_t, a_t | \theta) - (r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a' | \theta)) \right]^2
\]

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

Tutorial: Deep Reinforcement Learning, David Silver, Google DeepMind
Reinforcement + Neural Net

There are some research papers on the topic:

- Reinforcement Learning Using Neural Networks, with Applications to Motor Control
- Reinforcement Learning Neural Network To The Problem Of Autonomous Mobile Robot Obstacle Avoidance

And some code:

- Code examples for neural network reinforcement learning.

Those are just some of the top google search results on the topic. The first couple of papers look like they're pretty good, although I haven't read them personally. I think you'll find even more information on neural networks with reinforcement learning if you do a quick search on Google Scholar.

But diverges using neural networks due to:
- Correlations between samples
- Non-stationary targets

DQN paper
www.nature.com/articles/nature14236

DQN source code:
sites.google.com/a/deepmind.com/dqn/

Tutorial: Deep Reinforcement Learning, David Silver, Google DeepMind
Two big issues

- But diverges using neural networks due to:
  - Correlations between samples
  - Non-stationary targets
1. Correlations between samples

**Algorithm 1** Deep Q-learning

1. Initialize action-value function $Q$ with random weights.
2. 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 $\epsilon$ 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 $\phi_{t+1} = \phi(s_{t+1})$.
   - 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.
3. End for.
4. End for.

Playing Atari with Deep Reinforcement Learning - University of Toronto by V Mnih et al.
1. Correlations between samples

Algorithm 1 Deep Q-learning

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 $\epsilon$ 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 $\phi_{t+1} = \phi(s_{t+1})$
    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
1. Correlations between samples
Prerequisite: http://hunkim.github.io/ml/ or https://www.inflearn.com/course/기본적인-머신러닝-딥러닝-강좌/
1. Correlations between samples
2. Non-stationary targets

\[
\min_{\theta} \sum_{t=0}^{T} \left[ \hat{Q}(s_t, a_t | \theta) - (r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a' | \theta)) \right]^2
\]
2. Non-stationary targets

\[
\min_{\theta} \sum_{t=0}^{T} \left[ \hat{Q}(s_t, a_t | \theta) - (r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a' | \theta)) \right]^2
\]

\[
\hat{Y} = \hat{Q}(s_t, a_t | \theta)
\]

\[
Y = r_t + \gamma \max_{a'} \hat{Q}_{\theta}(s_{t+1}, a' | \theta)
\]
DQN’s three solutions

1. Go deep

2. Capture and replay
   • Correlations between samples

3. Separate networks: create a target network
   • Non-stationary targets

Tutorial: Deep Reinforcement Learning, David Silver, Google DeepMind
Human-level control through deep reinforcement learning, Nature
http://www.nature.com/nature/journal/v518/n7540/full/nature14236.html
Solution 1: go deep

- 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

ICML 2016 Tutorial: Deep Reinforcement Learning, David Silver, Google DeepMind
Problem 2: correlations between samples
Solution 2: experience replay

Deep Q-Networks (DQN): Experience Replay

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

\[
\begin{align*}
{s_1, a_1, r_2, s_2} \\
{s_2, a_2, r_3, s_3} \\
{s_3, a_3, r_4, s_4} \\
\vdots \\
{s_t, a_t, r_{t+1}, s_{t+1}}
\end{align*}
\]

Sample experiences from data-set and apply update

\[
\min_{\theta} \sum_{t=0}^{T} [\hat{Q}(s_t, a_t|\theta) - (r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a'|\theta))]^2
\]

To deal with non-stationarity, target parameters are held fixed.

Capture & Replay

ICML 2016 Tutorial: Deep Reinforcement Learning, David Silver, Google DeepMind
Solution 2: experience replay

Algorithm 1 Deep Q-learning with Experience Replay

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 $\epsilon$ 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 $s_{t+1} = s_t, a_{t+1} = a_t$, and preprocessed $\phi_{t+1} = \phi(x_{t+1})$
    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 $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for
Problem 2: correlations between samples

$s_1, a_1, r_2, s_2$

$s_2, a_2, r_3, s_3$

$s_3, a_3, r_4, s_4$

$\ldots$

$s_t, a_t, r_{t+1}, s_{t+1}$
Problem 3: non-stationary targets

\[
\min_{\theta} \sum_{t=0}^{T} \left[ \hat{Q}(s_t, a_t | \theta) - (r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a' | \theta)) \right]^2
\]

\[
\hat{Y} = \hat{Q}(s_t, a_t | \theta)
\]

\[
Y = r_t + \gamma \max_{a'} \hat{Q}_\theta(s_{t+1}, a' | \theta)
\]
Human-level control through deep reinforcement learning, Nature
http://www.nature.com/nature/journal/v518/n7540/full/nature14236.html
Solution 3: separate target network

\[
\min_{\theta} \sum_{t=0}^{T} \left[ \hat{Q}(s_t, a_t | \theta) - (r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a' | \bar{\theta})) \right]^2
\]
Solution 3: copy network

Sample random minibatch of transitions \( (\phi_j, a_j, r_j, \phi_{j+1}) \) from \( D \)

Set \( y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases} \)

Perform a gradient descent step on \( (y_j - Q(\phi_j, a_j; \theta))^2 \) with respect to the network parameters \( \theta \)

Every \( C \) steps reset \( Q \)

\[ \text{(1) } s \quad \rightarrow \quad \text{input layer} \quad \rightarrow \quad \text{hidden layer 1} \quad \rightarrow \quad \text{hidden layer 2} \quad \rightarrow \quad \text{output layer} \quad \rightarrow \quad \text{(2) } Ws \]

\[ \text{(1) } s \quad \rightarrow \quad \text{input layer} \quad \rightarrow \quad \text{hidden layer 1} \quad \rightarrow \quad \text{hidden layer 2} \quad \rightarrow \quad \text{output layer} \quad \rightarrow \quad \text{(2) } Y \text{ (target)} \]
Algorithm 1: deep Q-learning with experience replay.

Initialize replay memory \( D \) to capacity \( N \)
Initialize action-value function \( Q \) with random weights \( \theta \)
Initialize target action-value function \( \hat{Q} \) with weights \( \theta(0) = \theta \)

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 \( \varepsilon \) select a random action \( a_t \)
otherwise select \( a_t = \text{argmax}_a Q(\phi(s_t), a; \theta) \)
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 preprocessed \( \phi_{t+1} = \phi(s_{t+1}) \)
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 \( y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_a \hat{Q}(\phi_{j+1}, a; \theta) & \text{otherwise} \end{cases} \)
Perform a gradient descent step on \( \left(y_j - Q(\phi_j, a_j; \theta)\right)^2 \) with respect to the network parameters \( \theta \)
Every \( C \) steps reset \( \hat{Q} = Q \)

End For

End For

Human-level control through deep reinforcement learning, Nature
http://www.nature.com/nature/journal/v518/n7540/full/nature14236.html
Next
Lab: DQN