Lecture 6: Q-Network

Reinforcement Learning with TensorFlow & OpenAI Gym
Sung Kim <hunkim+ml@gmail.com>
Q-Table (16x4)

(1) state, s
(2) action, a
(3) quality (reward) for the given action (eg, LEFT: 0.5, RIGHT 0.1 UP: 0.0, DOWN: 0.8)

Q (s, a)
Q-learning Test
created by Jae Hyun Lee (jaehyunlee25@gmail.com)

<table>
<thead>
<tr>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPISODE: 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OPTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIZE: 4</td>
</tr>
<tr>
<td>Hole: reward -1</td>
</tr>
</tbody>
</table>

```
START

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

```

http://computingkoreanlab.com/app/jAI/jQLearning/
Q-Table (16x4)

(1) state, s
(2) action, a
(3) quality (reward) for the given action (eg, LEFT: 0.5, RIGHT 0.1, UP: 0.0, DOWN: 0.8)
Q-Table (?)

100x100 maze

80x80 pixel + 2 color (black/white)
pow(2, 80*80)?

Sung's MacBook Pro: qlearning hunkim$ python
Python 2.7.10 (default, Jul 30 2016, 19:40:32)
[Clang 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.34)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> pow(2, 80*80),
390815922664323873317461428361483673112676810704634122725066747207685355684301383817204958869117433070781824
345011184483028127299512473215765131666243694265195661755211450607650816799767580471720769300054414257889999968
2218145590731711215852375861916259684264866953944533853878020623210968986095655234800673206169578993893064621
394479007844552265475090325302609329069451170857396116842051110072780702996021975530468334392852283568123658
054474934577299768778927200578050584569281027959847841942880157513920578725804851336909371096131251468207589
95253399174861917220440109925365542544333467577974014533174115766832347511737830030067538031141178372089229
1860208840935427421876878495172143644788379083826461955232814452670024106634029782444314857725946984834066258
955213768118705885537084067810415837310293134293532224816692313350648857707611678657107208778727572115167144
96844554782437658117922884282324307657985008269944091087969017270165493533858541923739452891021946640039871
13901469451748274843820362034911702772858708285498619302796396590110309838995711294755319519006994199572
87157799733920151855469490347042463313419056713812651377581876770518591303000260334804048038748524234393838
534388638439846726058760432652194890060981900676541083211709433923554061356424944564563951032375578180928992
07647679659306704845779425650490109789823249286354015437424049924706852952567127100571306646256947045781135
74409288140528268714804050826437685344138382175580528795674046854193370709194541655491392625479753506217540
329184028243756578451089005278842826096937883527288450754271846667367090471859996737723112934060270451409659
56603443929450215599355255438319887909035361713548867020994349284913996584689674033162649588710526967617555789
71650189168353148260794391708438199220888781289029685829505315773902138539907176041428571960169709472040532
81297451190566948103174751340033033351723361119313337995407378348343950340587998781783253376L
Q-Table (? x ?)

(1) state, s
(2) action, a

Q (s, a)

(3) quality (reward) for the given action (eg, LEFT: 0.5)
Q-function Approximation

(1) state, s
(2) action, a
(3) quality (reward) for the given action (eg, LEFT: 0.5)
Q-function Approximation

(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-function Approximation

\[ s \rightarrow \text{State} \rightarrow \text{Network} \rightarrow \text{Q-value} \]

\[ a \rightarrow \text{Action} \rightarrow \text{Q-value} \]

\[ s \rightarrow \text{State} \rightarrow \text{Network} \rightarrow \text{Q-value 1} \rightarrow \text{Q-value 2} \rightarrow \text{Q-value 3} \]

\[ \text{all actions} \]
Q-Network training (linear regression)

\[ H(x) = Wx \]

\[ \text{cost}(W) = \frac{1}{m} \sum_{i=1}^{m} (Wx^{(i)} - y^{(i)})^2 \]
Q-Network training (linear regression)

\[ H(x) = Wx \]

\[ cost(W) = \frac{1}{m} \sum_{i=1}^{m} (Wx^{(i)} - y^{(i)})^2 \]
Q-Network training (linear regression)

\[ Q(s) = r + \gamma \max Q(s') \]

\[ cost(W) = (Ws - y)^2 \]
Q-Network training (math notations)

- Approximate $Q^*$ function using $\theta$
  $$\hat{Q}(s, a | \theta) \sim Q^*(s, a)$$

$$Q(s)$$

Q-Network training (math notations)

- Approximate $Q^*$ function using $\theta$
  \[ \hat{Q}(s, a|\theta) \sim Q^*(s, a) \]

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

\[ Q(s) \]

\[ W_s \]

(1) \( s \) → input layer → hidden layer 1 → hidden layer 2 → output layer → (2) \( W_s \)

\[ Q(s) \]
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 $a_t = \text{max}_a Q^*(\phi(s_t), a; \theta)$
        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 $y_j = \begin{cases} 
        r_j & \text{for terminal } \phi_{j+1} \\
        r_j + \gamma \text{max}_a Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1}
        \end{cases}$
        Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
    end for
end for
Y label and loss function

Set \( y_j = \begin{cases} \frac{r_j}{r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta)} & \text{for terminal } \phi_{j+1} \\ \frac{r_j}{r_j + \max_{a'} Q(\phi_{j+1}, a'; \theta)} & \text{for non-terminal } \phi_{j+1} \end{cases} \)

Perform a gradient descent step on \( (y_j - Q(\phi_j, a_j; \theta))^2 \) according to equation 3.
Deterministic or Stochastic?

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:

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a')]$$
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
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: Deep, Replay, Separated networks
Next
Lab: Q-network