Lecture 5: Windy Frozen Lake
Nondeterministic world!

Reinforcement Learning with TensorFlow & OpenAI Gym
Sung Kim <hunkim+ml@gmail.com>
Windy Frozen Lake
Deterministic VS Stochastic (nondeterministic)

- **In deterministic models** the output of the model is fully determined by the parameter values and the initial conditions.
- **Stochastic models** possess some inherent randomness.
  - The same set of parameter values and initial conditions will lead to an ensemble of different outputs.
Deterministic

```
# Register FrozenLake with is_slippery False
register(
    id='FrozenLake-v3',
    entry_point='gym.envs.toy_text:FrozenLakeEnv',
    kwargs={
        'map_name': '4x4',
        'is_slippery': False
    }
)

env = gym.make('FrozenLake-v3')
```
Stochastic (non-deterministic)

```python
env = gym.make('FrozenLake-v0')
```
Stochastic (non-deterministic) worlds

- Unfortunately, our Q-learning (for deterministic worlds) does not work anymore
- Why not?

\[ \hat{Q}(s, a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a') \]
Our previous Q-learning does not work

```python
env = gym.make('FrozenLake-v0')
```

Score over time: 0.0165
Why does not work in stochastic (non-deterministic) worlds?
Stochastic (non-deterministic) world

- Solution?
  - Listen to $Q(s')$ (just a little bit)
  - Update $Q(s)$ little bit (learning rate)

- Like our life mentors
  - Don’t just listen and follow one mentor
  - Need to listen from many mentors
Your Career Needs Many Mentors, Not Just One

20.1.2017 15.00
Stochastic (non-deterministic) world
Learning incrementally

\[ Q(s, a) \leftarrow r + \gamma \max_{a'} Q(s', a') \]

- Learning rate, \( \alpha \)
  - \( \alpha = 0.1 \)

\[ Q(s, a) \leftarrow (1 - \alpha) Q(s, a) + \alpha \, [r + \gamma \max_{a'} Q(s', a')] \]
Learning with learning rate

\[ Q(s, a) \leftarrow r + \gamma \max_{a'} Q(s', a') \]

\[ Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a')] \]
Learning with learning rate

\[ Q(s, a) \leftarrow r + \gamma \max_{a'} Q(s', a') \]
Q-learning algorithm

For each $s, a$ initialize table entry $\hat{Q}(s, a) \leftarrow 0$

Observe current state $s$

Do forever:

• Select an action $a$ and execute it

• Receive immediate reward $r$

• Observe the new state $s'$

• Update the table entry for $\hat{Q}(s, a)$ as follows:

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a')]$$

• $s \leftarrow s'$

Convergence

\[ \hat{Q} \text{ denote learner’s current approximation to } Q. \]

\[ \hat{Q}(s, a) \leftarrow (1 - \alpha)\hat{Q}(s, a) + \alpha [r + \gamma \max_{a'} \hat{Q}(s', a')] \]

Can still prove convergence of \( \hat{Q} \) to \( Q \) [Watkins and Dayan, 1992]

Machine Learning, Tom Mitchell, 1997
Next

Lab: Stochastic worlds