Empirical Comparison of Sarsa and Q-Learning on FrozenLake

BY PAUL PEREIRA

Review : Sarsa

Sarsa: An on-policy TD control algorithm

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (e.g., ϵ-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ϵ-greedy)
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

Review Q-Learning

Q-learning: An off-policy TD control algorithm

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
    Initialize S
    Repeat (for each step of episode):
        Choose A from S using policy derived from Q (e.g., ϵ-greedy)
        Take action A, observe R, S'
        $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
        $S \leftarrow S'$
    until S is terminal

FrozenLake

- Square grid of size 16
- 4 actions possible : up, down, right, left
- Start in the upper left corner, want to reach bottom right corner
- Ice is slippery : The outcome of picking an action is uncertain
- Holes in the ice : Game Over
- Reward of 1 for reaching the goal, 0 otherwise

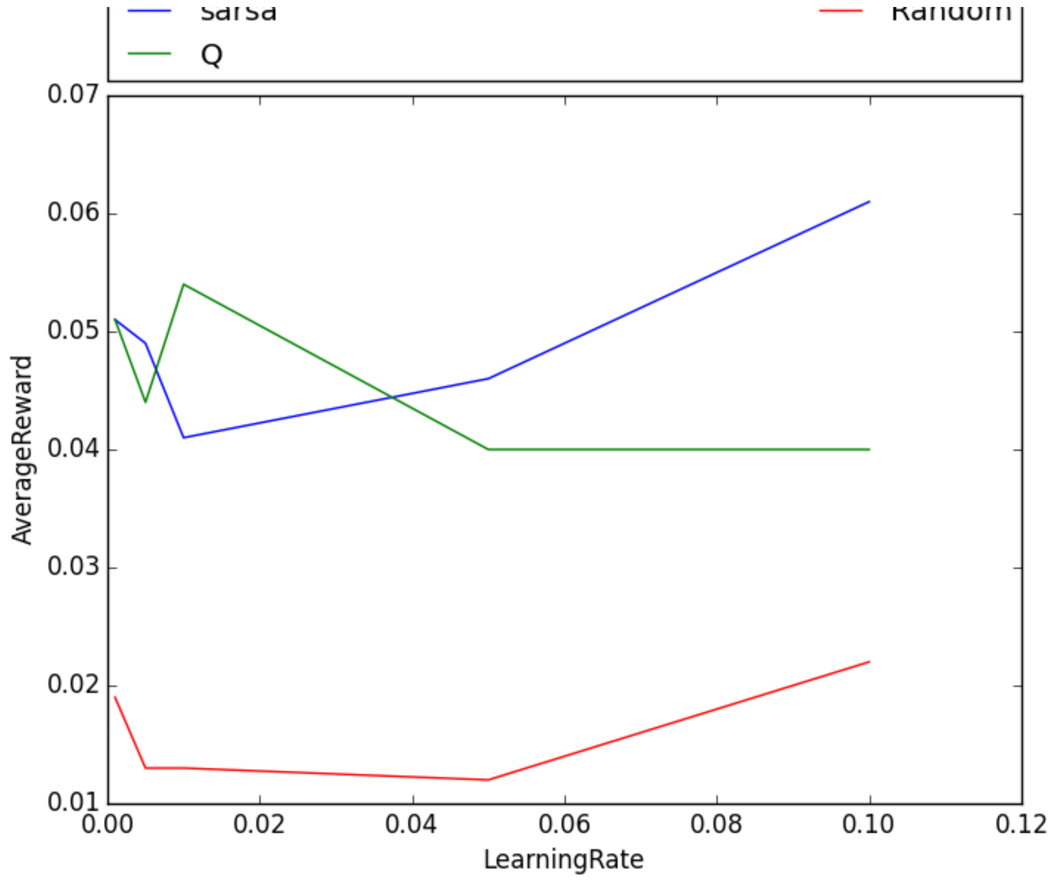
```
SFFF
FHFH
FFFH
HFFG
```

Methodology

- Learn Q function using both methods over 1000 episodes with various learning rates
- Pause the learning
- Run 100 episodes to estimate average reward per game player
- Compare with random baseline

Results

- Both Sarsa and Q-Learning allow some improvement over the baseline random player



Sarsa state-action value function

- Sarsa is capable of learning which actions leads to falling in a hole (for example state 5)
- However, the results obtained get worse as we get closer to the goal (for example state 14)

```
[  7.09663794e-06   1.91761051e-04   1.84392159e-05   1.04313709e-05]
[  1.77332940e-06   8.23371627e-05   1.36010720e-06   2.43161167e-06]
[  1.26770673e-06   2.09654542e-04   1.64499827e-08   1.70839123e-06]
[  5.51914103e-07   2.70443815e-05   0.00000000e+00   1.94526855e-08]
[  1.96536740e-05   4.28459734e-04   1.16428179e-06   8.54491367e-06]
[ 0.  0.  0.  0.]
[  1.76053546e-06   1.86468335e-06   1.42573555e-03   0.00000000e+00]
[ 0.  0.  0.  0.]
[  5.09081573e-06   1.30042216e-03   4.93733886e-06   3.72934825e-05]
[  2.84483955e-06   4.94883517e-03   3.61092855e-05   2.97925225e-06]
[ 0.00146897  0.01631333  0.          0.        ]
[ 0.  0.  0.  0.]
[ 0.  0.  0.  0.]
[ 0.          0.          0.          0.01215345]
[  7.21876917e-05   2.97010000e-02   9.98329948e-03   1.01217178e-01]
[ 0.  0.  0.  0.]
```

```
SFFF
FHFH
FFFH
HFFG
```

Q-Learning state-action value function

- Q-Learning state-action value function seems much better (for example state 14)
- Also capable of learning where the holes are and in which direction to move (for example state 7)

```
[ 0.01132315  0.02702013  0.00945407  0.01052944]
[ 0.00411806  0.01623654  0.00243397  0.00419786]
[  4.16501055e-03   2.06185465e-02   1.71705654e-03   3.94124206e-06]
[ 0.00045919  0.00921068  0.          0.        ]
[ 0.00423758  0.03278595  0.00678821  0.00480472]
[ 0.  0.  0.  0.]
[ 0.          0.03709488  0.          0.        ]
[ 0.  0.  0.  0.]
[ 0.0033923   0.06838827  0.01802792  0.00975713]
[ 0.01286358  0.15302528  0.00053918  0.01265633]
[ 0.01180913  0.16715664  0.01801976  0.00143159]
[ 0.  0.  0.  0.]
[ 0.  0.  0.  0.]
[ 0.00068427  0.02253025  0.01283698  0.20271962]
[ 0.          0.          0.          0.46341489]
[ 0.  0.  0.  0.]
```

```
SFFF
FHFH
FFFH
HFFG
```

Reference

[1] Richard S. Sutton and Andrew G. Barto, "Reinforcement learning: An introduction", Second Edition, MIT Press

[2] https://gym.openai.com/envs/FrozenLake-v0