RL for Sports: Cricket

Navin Mordani, M.Sc. student, Computer Science, McGill University
MohammadReza Davari, M.Sc. student, Computer Science, Concordia University
Julyan Keller-Baruch, M.Sc. student, Human Genetics, McGill University

Cricket: A Gentleman's Game

- 2 Teams
- One team bats first and achieves a score, the other tries to beat that score.
- In a batting turn team gets fixed number of opportunities to hit the ball.
- Member can hit the ball with different aggression levels
- More the aggression more the points on success and more the risk of failure.
- Failure results in elimination of the member.
- All members eliminated means turn over.
- Score is the sum of points at each hit.

Environments

<table>
<tr><th></th><th>Team (Agent) 1</th><th>Team (Agent) 2</th></tr>
<tr><td>Action Space</td><td colspan="2"><table><tr><td><u>Action</u></td><td>1</td><td>2</td><td>4</td><td>6</td></tr><tr><td><u>P(Success)</u></td><td>0.95</td><td>0.88</td><td>0.8</td><td>0.6</td></tr></table></td></tr>
<tr><td>State Variables</td><td>● Balls hit (time steps elapsed)
● Members eliminated</td><td>● Balls hit (time steps elapsed)
● Members eliminated
● Points to the target</td></tr>
<tr><td>Reward</td><td>Points earned at the hit</td><td>● Non-zero at terminal states
● +1 if target achieved else -1</td></tr>
</table>

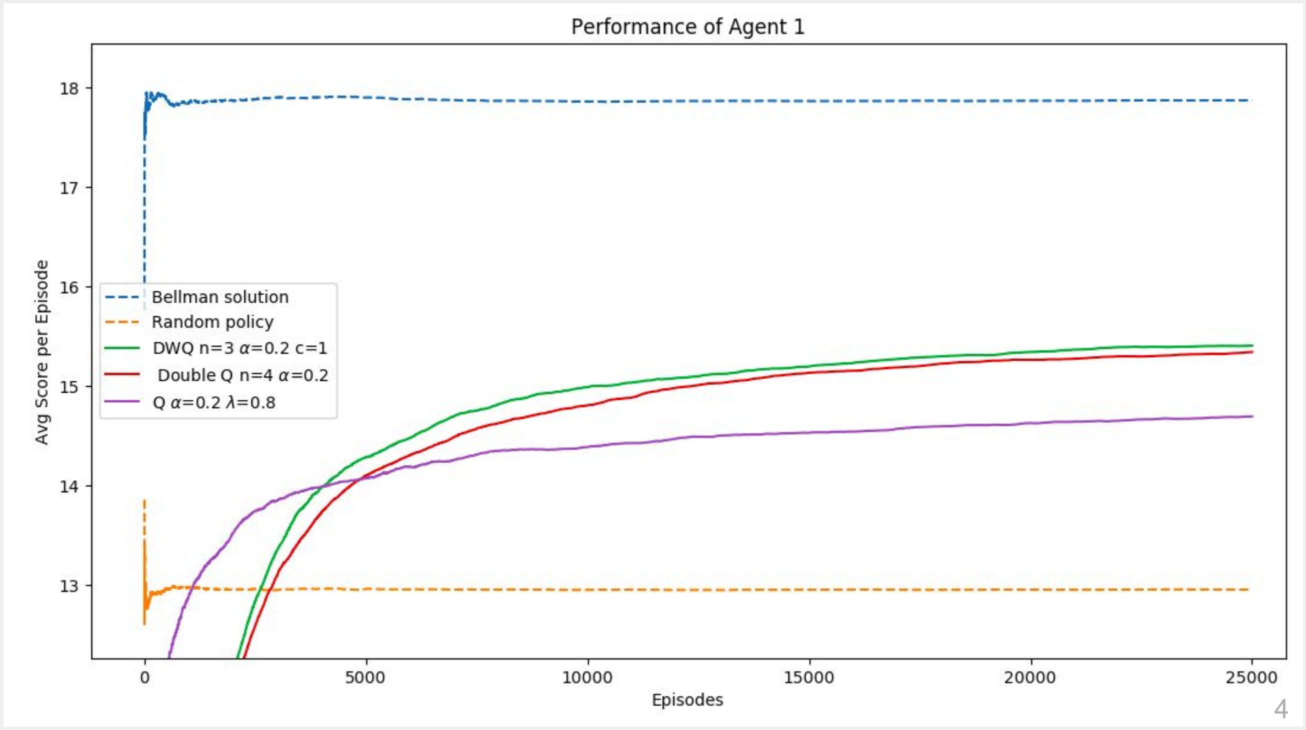
We built agents to play the game with 6 hits and 2 members in each team.

Learning Team 1 Strategy

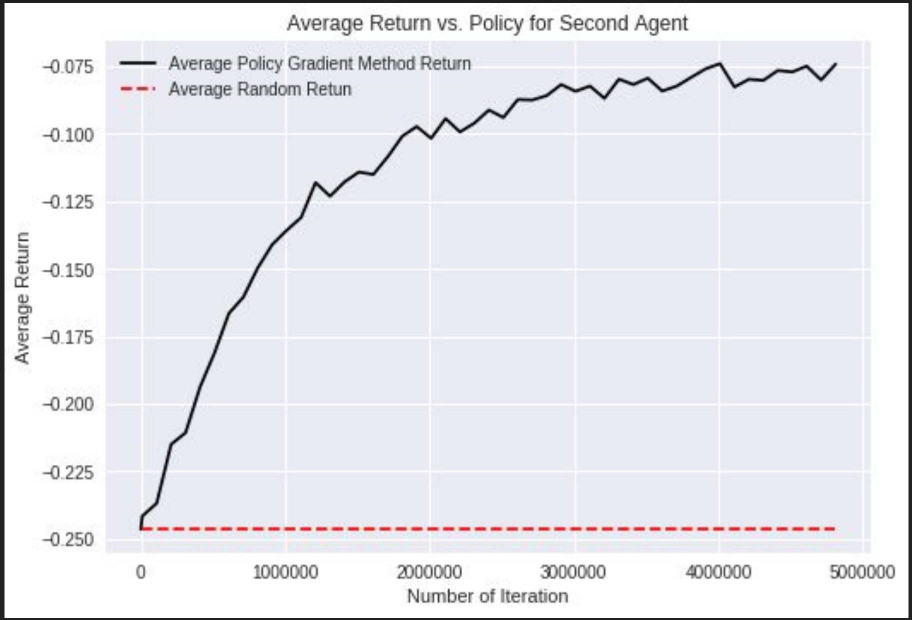
Double weighted Q-learning update.

$$a^* \leftarrow \arg\max_a Q^U(s',a)$$
$$a_L \leftarrow \arg\min_a Q^U(s',a)$$
$$\beta^U \leftarrow \frac{|Q^V(s',a^*)-Q^V(s',a_L)|}{c+|Q^V(s',a^*)-Q^V(s',a_L)|}$$
$$\delta \leftarrow r+\gamma[\beta^U Q^U(s',a^*)+(1-\beta^U)Q^V(s',a^*)]-Q^U(s,a)$$
$$Q^U(s,a) \leftarrow Q^U(s,a)+\alpha^U(s,a)\delta$$

Credits: Z. Zhang, Z. Pan, and M. J. Kochenderfer, “Weighted Double Q-learning,” *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 2017



Learning Team 2 Strategy



Performance of Agent 2

Policy	Avg. Return	%Winning
Random	-0.25	~12%
Learned	-0.075	~46.25%

Conclusion and Future Work

- Significantly higher performance than random implies learning
- Due to the stochastic nature of the problem learning needs to be carried out for large number of episodes and parameter tuning is tricky
- Finish experiment using kernel-based value function approximation.
  - Euclidean distance kernel
  - Other easily implementable kernels, such as RBF
  - Domain specific kernel.
    - Maybe we can use our knowledge of the game to define state-similarity in smart way.
- Would like to explore larger state spaces
  - To get a better sense of the limitations of each of our methods.