CAP6671 Intelligent Systems

Lecture 13:
Multi-agent Reinforcement Learning

Instructor: Dr. Gita Sukthankar
Email: gitars@eecs.ucf.edu
Schedule: T & Th 9:00-10:15am
Location: HEC 302
Office Hours (in HEC 232):
T & Th 10:30am-12
Presentations on Related Work

- 30 minute presentation on one paper that you think is relevant to your project
- Presentation should include:
  - Strengths/weaknesses of paper
  - Detailed presentation of how the system described in the paper works
  - Discussion of how the paper relates to your project
- Email me about when you want to present your paper
Strengths/Problems of Paper
Strengths/Problems of Paper

- **Pros:**
  - Works in a non-Markovian domain
  - Handles multi-agent teams
  - Large state space
  - Has been generalized to the network routing domain

- **Cons:**
  - Much of the work is done by the feature selection mechanism
  - No convergence guarantee
  - Relies on intermediate reinforcement
Characteristics

- Opaque-transition
  - No simple function which governs the transition from one state to another

- Chained agents
  - No single agent’s actions can achieve the goal

- Team-partitioned
  - Each teammate learns and uses a separate part of the table based on its position in the team
What is the agent trying to learn?
Action Representation

Actions = possible kicks towards all the corners of the field

Agent is trying which of 8 possible kicks it should make when it has possession of the ball
How many Q-values are there?

- Number of states: $22^{10^9}$
- # Q-values is normally equal to SxA
- Q-table used in this system only has 88 values; each player only learns 8 Q-values per game
- How is this done?
  - Answer: by the use of a single feature (kick success)
Algorithm

- State generalization
  - Move from raw representation of state space through the outputs (f: S->V)

- Value function learning
  - Determine whether each possible action is likely to succeed (represented by Q-values)

- Action selection
  - Update Q-table based on action chosen and reward received
Value Function Learning

- Insight: use high level action-dependent features to describe the state space
- Only need to consider the features relevant to the action being performed

<table>
<thead>
<tr>
<th>$c(s, a_0)$</th>
<th>$c(s, a_1)$</th>
<th>$Q(v, a_0)$</th>
<th>$Q(v, a_1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_0$</td>
<td>$u_0$</td>
<td>$q_{0,0}$</td>
<td>$q_{1,0}$</td>
</tr>
<tr>
<td>$u_0$</td>
<td>$u_1$</td>
<td>$q_{0,0}$</td>
<td>$q_{1,1}$</td>
</tr>
<tr>
<td>$u_1$</td>
<td>$u_0$</td>
<td>$q_{0,1}$</td>
<td>$q_{1,0}$</td>
</tr>
<tr>
<td>$u_1$</td>
<td>$u_1$</td>
<td>$q_{0,1}$</td>
<td>$q_{1,1}$</td>
</tr>
<tr>
<td>$u_1$</td>
<td>$u_2$</td>
<td>$q_{0,2}$</td>
<td>$q_{1,2}$</td>
</tr>
<tr>
<td>$u_2$</td>
<td>$u_0$</td>
<td>$q_{0,2}$</td>
<td>$q_{1,0}$</td>
</tr>
<tr>
<td>$u_2$</td>
<td>$u_1$</td>
<td>$q_{0,2}$</td>
<td>$q_{1,1}$</td>
</tr>
<tr>
<td>$u_2$</td>
<td>$u_1$</td>
<td>$q_{0,2}$</td>
<td>$q_{1,1}$</td>
</tr>
</tbody>
</table>

$\Rightarrow$

<table>
<thead>
<tr>
<th>$c(s, a_1)$</th>
<th>$Q(v, a_0)$</th>
<th>$Q(v, a_1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_0$</td>
<td>$q_{0,0}$</td>
<td>$q_{1,0}$</td>
</tr>
<tr>
<td>$u_1$</td>
<td>$q_{0,1}$</td>
<td>$q_{1,1}$</td>
</tr>
<tr>
<td>$u_2$</td>
<td>$q_{0,2}$</td>
<td>$q_{1,2}$</td>
</tr>
</tbody>
</table>

- Normally features are independent of actions
Learning Rule

- Modified version of the Q-learning which only takes into account immediate reward and not future transitions

\[
Q(v, a) = Q(v, a) + \alpha (r - Q(v, a))
\]

- Standard Q-learning

\[
Q(s, a) := Q(s, a) + \alpha (r + \gamma \max_{a'} Q(s', a') - Q(s, a))
\]
Action Selection

- Features themselves are discriminative of whether the action should be used
- Hence never consider actions when the feature lacks a certain value (filter based on \(W\))
- If only successful kicks are considered then effectively there is only one state (SuccessfulKick) and 8 actions
- Number of Q-values learned by each agent is 8!
- During the game the actions can be selected using any standard exploration policy (e.g., Boltzmann, greedy)
Layered Approach

- Divide and conquer the learning task
- Low level modules make a big difference to success
- Predicting kick success
  - Use 200 continuous value attributes describing teammate/opponent positions
  - C4.5 decision tree algorithm
- Intercepting ball
  - Trained a neural network specifically for the interception behavior
• Use intermediate rewards rather than just rewarding at goals
• Base reward functions are scaled by time events occur
• Must be observed by agent doing the learning
Results

- Evaluated vs. random team and switching team
- Random team just makes random passes
- Switching team uses hand-coded policy in which the agents all stay towards one half of field
- Interesting notes:
  - 11 players got an average of 1490 action reinforcement pairs per game
  - Each action is only tried on average 186 times over 160 games
Results

Cumulative Goals vs. Game Number

Learning
Random

Goals vs. Game Number

Learning team
Switching team

|U|=2, e=DT
|W|=1
|U|=2, |W|=2, e=rand
|U|=2, |W|=1, e=heur

|U|=1
|W|=1
(b.1) (b.2) (b.3) (b.4) (b.5)
Conclusion

- Divide and conquer your problem
- Layering outputs of multiple classifiers is a very effective approach
- RL will be more effective if state space is small