## Grid Game

<table>
<thead>
<tr>
<th></th>
<th>R</th>
<th></th>
<th>P1</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>M</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>M</td>
<td></td>
<td>M</td>
</tr>
<tr>
<td>M</td>
<td>M</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>M</td>
<td></td>
<td>P3</td>
</tr>
</tbody>
</table>

*Artificial Intelligence: Foundations of Computational Agents*
Prize

- Prize could be in one of the corners or no prize
- \( r(P) = +10 \)
- When the prize is taken it disappears until a new prize is respawned with certain probability.
### Damage

- At each timestep a monster can appear in any of the M cells.
- If a monster appears in the agent’s cell then the agent gets damaged.
- If the agent has already been damaged then it receives $r(D, M) = -10$.
- The agent can get repaired by visiting the R cell.

<table>
<thead>
<tr>
<th>P0</th>
<th>R</th>
<th>P1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
<td></td>
</tr>
<tr>
<td></td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>P2</td>
<td></td>
<td>P3</td>
</tr>
</tbody>
</table>
State

- Fully observable environment
- Represent the state as \((X, Y, P, D)\)
  - \(X\): X position of the agent
  - \(Y\): Y position of the agent
  - \(P\): position of the prize (\(P=4\) - no prize)
  - \(D\): 1 if the agent is damaged, otherwise 0
Linear Function Approximation

- State-action value function:

\[ Q_w(s, a) = w_0 + w_1 F_1(s, a) + \ldots + w_n F_n(s, a) \]
Linear Function Approximation

- State-action value function:

\[ Q_w(s, a) = w_0 + w_1 F_1(s, a) + ... + w_n F_n(s, a) \]

- What features can we choose?
## Possible Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_1(s, a)$</td>
<td>1 - if action $a$ would most likely take the agent from state $s$ into a location where a monster could appear; 0 - otherwise</td>
</tr>
<tr>
<td>$F_2(s, a)$</td>
<td>1 - if action $a$ would most likely take the agent into a wall; 0 - otherwise</td>
</tr>
<tr>
<td>$F_3(s, a)$</td>
<td>1 - if action $a$ would most likely take the agent toward a prize; 0 - otherwise</td>
</tr>
<tr>
<td>$F_4(s, a)$</td>
<td>1 - if the agent is damaged and action $a$ would most likely take it to the repair station; 0 - otherwise</td>
</tr>
<tr>
<td>$F_5(s, a)$</td>
<td>1 - if the agent is damaged and action $a$ would most likely take it to a monster; 0 - otherwise</td>
</tr>
</tbody>
</table>
### Possible Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_6(s, a)$</td>
<td>1 - if the agent is damaged in state $s$; 0 - otherwise</td>
</tr>
<tr>
<td>$F_7(s, a)$</td>
<td>1 - if the agent is <em>not</em> damaged in state $s$; 0 - otherwise</td>
</tr>
<tr>
<td>$F_8(s, a)$</td>
<td>1 - if the agent is damaged and there is a prize in the direction of action $a$; 0 - otherwise</td>
</tr>
<tr>
<td>$F_9(s, a)$</td>
<td>1 - if the agent is <em>not</em> damaged and there is a prize in the direction of action $a$; 0 - otherwise</td>
</tr>
</tbody>
</table>
### Possible Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_{10}(s, a)$</td>
<td>distance from left wall if prize at location P0</td>
</tr>
<tr>
<td>$F_{11}(s, a)$</td>
<td>distance from right wall if prize at location P0</td>
</tr>
<tr>
<td>$F_{12-29}(s, a)$</td>
<td>Similar to $F_{10}$ and $F_{11}$ for different wall and prize combinations</td>
</tr>
</tbody>
</table>
Training with SARSA

- Let $\delta = r + \gamma Q(s',a') - Q(s,a)$ then update the weights with $w_i \leftarrow w_i + \eta \delta F_i(s,a)$

$$Q(s, a) = 2.0 - 1.0 \times F_1(s, a) - 0.4 \times F_2(s, a) - 1.3 \times F_3(s, a)$$
- $- 0.5 \times F_4(s, a) - 1.2 \times F_5(s, a) - 1.6 \times F_6(s, a)$
- $+ 3.5 \times F_7(s, a) + 0.6 \times F_8(s, a) + 0.6 \times F_9(s, a)$
- $- 0.0 \times F_{10}(s, a) + 1.0 \times F_{11}(s, a) + \ldots$
Assignment 2: Programming Task

Svetlin Penkov
School of Informatics
Mar 17, 2017
Setup

- Repository: [https://github.com/ipab-rad/rl-cw2](https://github.com/ipab-rad/rl-cw2)
- The README file provides lots of information
- On a DICE machine:

  ```
  > git clone [https://github.com/ipab-rad/rl-cw2](https://github.com/ipab-rad/rl-cw2)
  ```
Sensing

- The same agent interface for playing Enduro

- The sensing capabilities of the agent are enhanced
  - Road
  - Cars
  - Speed
  - Grid
Road Grid

- Plain 2-dimensional array containing \([x, y]\) points in pixel coordinates.
- **Note:** The image which is displayed is a scaled version of the actual game frame.
Road Grid

road[0]  

road[11]
Road Grid

[r[0] for r in road]  [r[10] for r in road]
Cars

- Dictionary containing the size and location of each car in the game frame
- \((x, y)\) top left pixel coordinate
- \((w, h)\) size in pixels

```json
{
    'self': (x, y, w, h),
    'others': [(x1, y1, w1, h1), (x2, y2, w2, h2), (x3, y3, w3, h3)]
}
```
Speed

- Speed relative to the opponents in the range $[-50, 50]$
- The speed is set to -50 when the agent collides
- If the agent is moving as fast as possible its speed is 50
Grid
Questions?