Comparison of Machine Learning Techniques in Pursuit and Evasion Games

Jacky Baltes and Yong-Joo Park
Centre for Imaging Technology and Robotics
University of Auckland, New Zealand

Motivation

- Simple games: Achievement of static goals. Avoidance of static obstacles
- Next step: Achievement of dynamic goals. Avoidance of dynamic obstacles

Pursuit and Evasion Games are simple enough to analyze them. Yet, complex enough to provide more meaningful results.

Optimal solution for some pursuit and evasion games are known.

Homicidal Chauffeur Game

Game played between two parties on a finite playing field (Empty Parking Lot).
- Pursuer is trying to catch the evader. Pursuer has higher velocity, but larger turn radius.
- Evader is trying to escape from the pursuer. Slower, but more maneuverable
- Modeled as non-holonomic unicycles

Isaacs developed optimal strategy for infinite playing field.
Playing surface is broken up into:
- UP: usable part where pursuer wins
- NUP: non–usable part where evader wins
- BUP: boundary of the non–usable/usable part

Optimal Algorithm

\[ \phi_p = \begin{cases} \text{sgn}(s) & \text{if } s \neq 0 \\ \text{random} & \text{else} \end{cases} \]

\[ \phi_E = \cos^{-1} \sqrt{1 - v_P^2} \]

Results

<table>
<thead>
<tr>
<th>Pursuit</th>
<th>Random</th>
<th>Optimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>62(4.49)</td>
<td>0(0)</td>
</tr>
<tr>
<td>Optimal</td>
<td>71(4.25)</td>
<td>25(3.22)</td>
</tr>
<tr>
<td>Genetic Algorithm</td>
<td>62(3.86)</td>
<td>8(2.19)</td>
</tr>
<tr>
<td>k–NN Learner</td>
<td>67(4.28)</td>
<td>4(0.63)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Evasion</th>
<th>Random</th>
<th>Optimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>62(4.20)</td>
<td>0(0.89)</td>
</tr>
<tr>
<td>Sum</td>
<td>324</td>
<td>37</td>
</tr>
</tbody>
</table>

Genetic Algorithm

- Used the Samuel System (Grefenstette)
- Maximum of 40 rules
- Each rule consists of:
  - Conditional part: range, bearing, heading
  - Action part: one of 9 steering angles (full right to full left)
- Example Rule:
  \[ if \ 1 \leq \text{range} < 250 \&\& \ 5 < \text{bearing} < 5 \&\& \ 90 < \text{heading} < 120 \ then \ \phi = -1 \ (\text{full left}) \]

- Rule selection based on rule strength
- Genetic operators: Rule mutation, Rule crossover, Rule merging
- Plan crossover
- k–NN Algorithm
  - Cases are represented as range, bearing, and heading.
  - Similarity metric is based on average sum of these features.
- Reinforcement Learner
  - Q Learner with standard update rule
  - Reward Evader: 10 * t if Evader caught at time t
  - Reward Pursuer: 10 (T–t) if Pursuer catches Evader at time t

Optimal Control

\[ \phi_p = \begin{cases} \text{sgn}(s) & \text{if } s \neq 0 \\ \text{random} & \text{else} \end{cases} \]

\[ \phi_E = \cos^{-1} \sqrt{1 - v_P^2} \]

Genetic Algorithm

- Used the Samuel System (Grefenstette)
- Maximum of 40 rules
- Each rule consists of:
  - Conditional part: range, bearing, heading
  - Action part: one of 9 steering angles (full right to full left)
- Example Rule:
  \[ if \ 1 \leq \text{range} < 250 \&\& \ 5 < \text{bearing} < 5 \&\& \ 90 < \text{heading} < 120 \ then \ \phi = -1 \ (\text{full left}) \]

- Rule selection based on rule strength
- Genetic operators: Rule mutation, Rule crossover, Rule merging
- Plan crossover
- k–NN Algorithm
  - Cases are represented as range, bearing, and heading.
  - Similarity metric is based on average sum of these features.
- Reinforcement Learner
  - Q Learner with standard update rule
  - Reward Evader: 10 * t if Evader caught at time t
  - Reward Pursuer: 10 (T–t) if Pursuer catches Evader at time t