Deep Q-Learning applied to air tactical deconfliction

CS231N – Spring 2022

Marta Palomar Toledano
mpalomar@stanford.edu
Introduction and Problem Statement

**Purpose**
Create a tool for Conflict Detection and Resolution (CDR) between flying vehicles based on their trajectories using AI.

**Problem statement**
One aircraft (ownship) is following its nominal trajectory towards a target and one (or more) intruders appear presenting a potential collision conflict. **Goal:** apply a series of avoidance maneuvers until all conflicts are solved and retake nominal trajectory to the target.

- Relevant due to growth in the volume of air operations due to improvement in autonomous technologies (Unmanned Air Vehicles).
Methodology

Main methods employed: CNN + DQN architecture to create an agent.

- Input data (points from a trajectory) converted to image format. Idea from literature:
  - By using images, the model can easily adapt to a large number of intruders without altering state space.
  - Take advantage of CNNs (data augmentation, etc)
- Input data are synthetically generated due to lack of real ATC data from air conflicts.
- OpenAI Gym custom environment + Tensorflow backend.
Learning workflow

- **Take step**: given an action from the available action space, the aircraft moves accordingly and a certain reward is received.

$$r(a) = \begin{cases} 
0 & d_{\text{intruder}} > d_{\text{critical}} \\
\frac{d_{\text{critical}} - d_{\text{intruder}}}{d_{\text{critical}}} & d_{\text{intruder}} \leq d_{\text{critical}}
\end{cases}$$
Learning workflow

- **Store in agent memory**: the implemented DQN agent stores the outputs from the `.step()` function, which will be employed for model updates.
- The new state is stacked with the previous three frames:

```
<table>
<thead>
<tr>
<th>t-3</th>
<th>t-2</th>
<th>t-1</th>
<th>State t</th>
</tr>
</thead>
</table>
```

*New state
Reward
Done True/False

- New state
- Get action
- Add experience
- If NOT terminal state
Learning workflow

- **Get action**: given the new state, the CNN model is called in order to predict the next action.
- The action is chosen based on an $\epsilon$-greedy policy.

$$a(t) = \begin{cases} \max_{a} Q_t(a) & \text{probability } 1-\epsilon \\ \text{any}(a) & \text{probability } \epsilon \end{cases}$$

Q-value for each possible action
Learning workflow

- **Add experience**: the `model.fit()` function is called in order to train the model with the experience stored from the last state.
- Summary of model hyperparameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>Learning rate</td>
<td>0.025</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>Discount factor</td>
<td>0.95</td>
</tr>
<tr>
<td>$\epsilon_0$</td>
<td>Initial $\epsilon$</td>
<td>0.9</td>
</tr>
<tr>
<td>$\epsilon_{\text{min}}$</td>
<td>Minimum $\epsilon$</td>
<td>0.05</td>
</tr>
<tr>
<td>$\epsilon_{\text{decay}}$</td>
<td>Decy value for $\epsilon$</td>
<td>0.0009</td>
</tr>
<tr>
<td>$\beta_1$</td>
<td>Adam’s 1st moment decay</td>
<td>0.9</td>
</tr>
<tr>
<td>$\beta_2$</td>
<td>Adam’s 2nd moment decay</td>
<td>0.999</td>
</tr>
</tbody>
</table>

- The Q-values associated to each action and state after each step are updated.
Two performance metrics were found to be relevant to the CDR problem:

- The Success Rate (%) is the percentage of simulations/episodes in which the ownship reaches the target.
- The Conflict Resolution Rate (%) is the percentage of conflict situations (separation distance lower than a threshold) that the model solves successfully.

Important to get 100% CRR in order to enforce safe operations.
Evaluation: Comparison with other methods (DQN without CNNs)

- Assess the benefits of using image representation + CNNs for this kind of problems → creation of an identical RL model using only state information as arrays. Removal of the convolutional layers.
- Similar performance on success rate (regardless the number of intruders the SR converges to 100% for both models).
- Evaluation of predicted best trajectories:

![Diagram showing comparison between DQN with and without CNNs for predicted trajectories.](attachment:image.png)
Deep Q-Learning applied to air tactical deconfliction

CS231N – Spring 2022

Marta Palomar Toledano
mpalomar@stanford.edu