Reinforcement Learning for Missile Evasion and Guidance Using Deep Neural network(DQN)

Anil Katwal

December 12, 2024

Introduction

- The problem involves evading missile threats using interceptors guided by Reinforcement Learning
- Approach: **Deep Q-Learning (DQN)** for controlling the interceptor's actions and distract from hit to the target.
- Objective: **Maximize the chance of interception** while avoiding the missile's target.

Environment and State Representation

State Space Representation:

$$\text{State} = \begin{bmatrix} x_m & y_m & x_i & y_i \end{bmatrix}$$

- x_m, y_m: Missile's position.
- x_i, y_i: Interceptor's position.

Action Space:

1. No adjustment: Maintain trajectory.
2. Turn left: Adjust heading -10°.
3. Turn right: Adjust heading $+10^\circ$.
4. Positive reward for hitting the target.
5. Negative reward for interceptor collision.
6. Dense reward based on distances between missile and target/interceptor.

Dynamics Update Equations

Missile Dynamics:

$$x_m(t+1) = x_m(t) + v_m \cos(\theta_m)$$

$$y_m(t+1) = y_m(t) + v_m \sin(\theta_m)$$

Interceptor Dynamics:

$$\theta_i = \arctan\left(\frac{y_m - y_i}{x_m - x_i}\right)$$

$$x_i(t+1) = x_i(t) + v_i \cos(\theta_i)$$

$$y_i(t+1) = y_i(t) + v_i \sin(\theta_i)$$

Parameters:

- v_m, v_i: Missile and interceptor speeds.
- θ_m, θ_i: Heading angles.

Missile Dynamics:

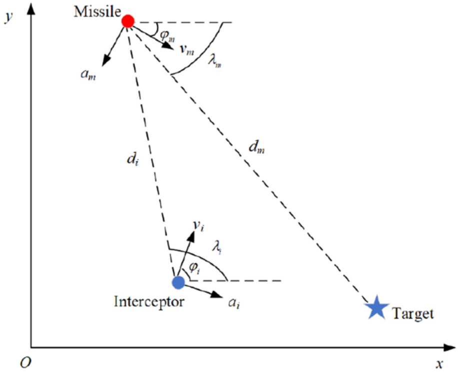


Figure: Actual trajectory and parameter of missile among these parameter only 4 state space is consider.

Q-Learning: Approach and Challenges

Q-Learning Update Rule:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right]$$

- α: Learning rate.
- γ: Discount factor.
- r: Reward for the current action.

Reward System:

- Positive reward: Achieving the target.
- Negative reward: Being intercepted or straying off-course.

Challenges:

- Infeasible for high-dimensional or continuous state spaces.

Transition to Deep Q-Networks (DQN)

DQN Approximation:

$$Q(s, a; \theta) \approx Q(s, a)$$

- θ: Neural network weights.

Loss Function:

$$L(\theta) = \mathbb{E}_{(s,a,r,s')\sim D}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta') - Q(s, a; \theta)\right)^2\right]$$

Enhancements:

- **Experience Replay:** Stores transitions (s, a, r, s') for training.
- **Target Network:** Stabilizes training by maintaining a separate network.

State Update Process

State Update Steps:

1. Observe the current state:

$$s = \begin{bmatrix} x_m & y_m & x_i & y_i \end{bmatrix}$$

2. Predict Q-values for all actions:

$$Q(s, a; \theta)$$

3. Choose the action with the highest Q-value:

$$a = \arg\max_a Q(s, a; \theta)$$

4. Execute the action, observe s', and update Q-value.

DQN Agent Architecture

- The agent uses a **Q-network** (a deep neural network).
- **State Input:** 4D vector representing missile and interceptor positions.
- **Action Output:** Q-values for 3 actions.
- **Learning:** The agent learns through **Experience Replay** and **Target Networks**.

Q-Network Structure

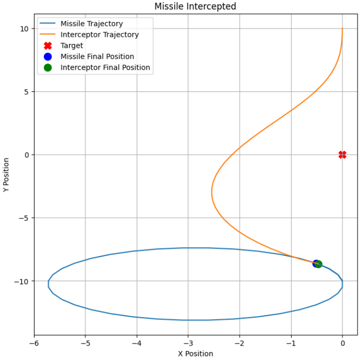
- Input layer: 4D state vector
- Hidden layers: Dense layers with ReLU activation
- Output layer: 3 Q-values (one for each action)

Training Process

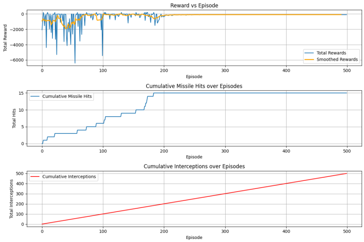
- **Exploration vs Exploitation:** Epsilon-greedy strategy.
- **Replay Buffer:** Stores experiences (state, action, reward, next state, done).
- **Target Network Update:** Periodically update target Q-network weights.
- **Loss Function:** Mean Squared Error between predicted Q-values and target Q-values.

Simulation Results

- **Episodes:** 500
  - Average reward per episode.
  - Number of successful intercept and number of hits on target.



(a) Successfully intercept

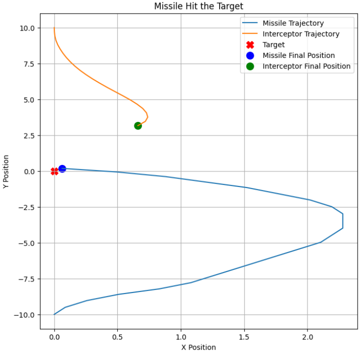


(b) Reward per Episode

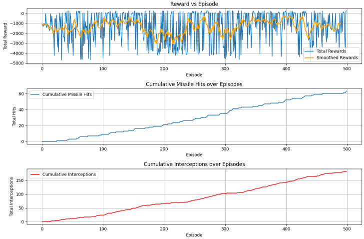
Figure: Interceptor and missile velocity are the same but less number of epochs.

Simulation Results (Continued)

- **Episodes:** 500
  - Average reward per episode.
  - Number of successful interceptions and hits on target.



(a) Successfully hit on target

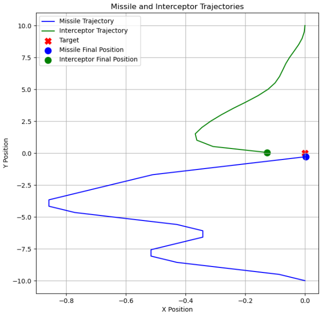


(b) Low Rewards

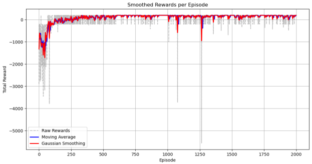
Figure: Interceptor and missile velocity are the same, but interception is highly penalized.

Simulation Results(continued)

- **Episodes:** 2000
  - Average reward per episode.
  - successfully hiting on target.



(a) Successfully hit on target

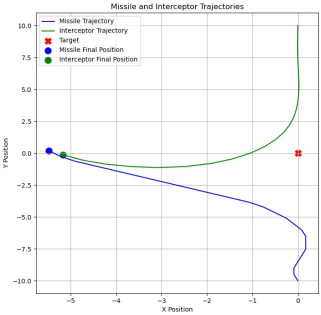


(b) Smoothed Rewards

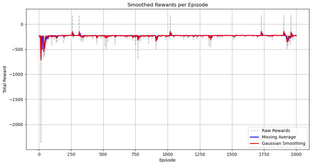
Figure: Missile and interceptor velocity are the same, with increased episodes.

Simulation Results (Continued)

- **Episodes:** 2000
  - Average reward per episode.
  - Number of successful interceptions.



(a) avoid to hit target

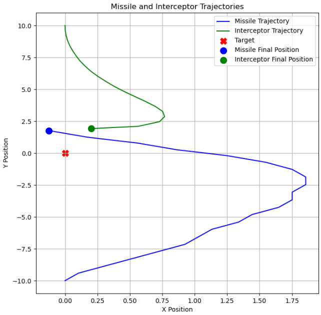


(b) Smoothed Rewards

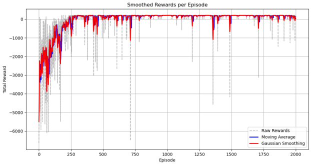
Figure: Interceptor velocity is higher, but it is highly penalized.

Simulation Results (Continued)

- **Episodes:** 2000
  - Average reward per episode.
  - avoid to Hits on target without interception.



(a) Trajectory with Velocity Difference

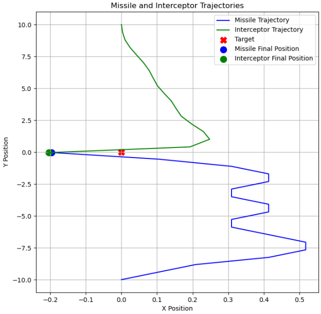


(b) Smoothed Rewards

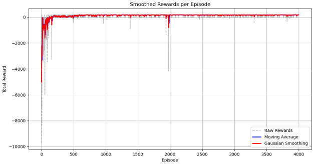
Figure: Missile velocity is higher than interceptor velocity.

Simulation Results (Continued)

- **Episodes:** 4000
  - Average reward per episode.
  - Number of successful intercept.



(a) Successfully intercept by interceptor

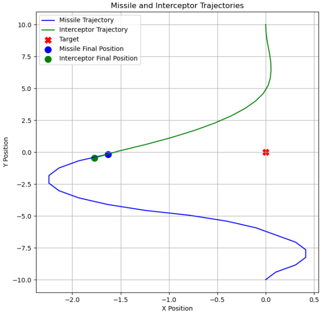


(b) 4k Rewards

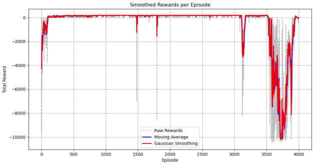
Figure: Missile velocity is higher than interceptor velocity at 4000 episodes.

Simulation Results (Continued)

- **Episodes:** 4000
  - Average reward per episode.
  - Number of successful avoid to hit target



(a) New Trajectory



(b) New Rewards

Figure: Missile and interceptor same velocity at 4000 episodes but interceptor is highly penalized.

Conclusion

- Successfully applied deep reinforcement learning to a missile evasion task.
- DQN proved to be effective in training the interceptor for evasion.
- Promising results, but further improvements can be made.
- After 4000 epochs of training under different environmental conditions and velocities, the interceptor successfully learned to predict and adapt to the missile's trajectory.

Future Work

- **Enhance the environment:** Add more complex terrains or multiple interceptors,weather condition, visibility, landscape, horizontal acceleration, lateral acceleration, longitudinal acceleration, position of target.
- **Improve training:** Use Double DQN or Dueling DQN for improved stability.
- **Experiment with other RL algorithms:** Actor-Critic methods for better performance, Proximal policy optimization (PPO) is a reinforcement learning (RL)

Refernces

1. Yan, M., Yang, R., Zhang, Y., Yue, L., & Hu, D. (2024). A Hierarchical Reinforcement Learning Method for Missile Evasion and Guidance. *Journal Name*, X(Y), Z-Z.
2. Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT Press.
3. Mnih, V., Kavukcuoglu, K., Silver, D., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529-533