PENGUIN-ML-AGENT HUNGER SYSTEM TECHNICAL REPORT

Name: Anette Hardy Sosa
Course: Advanced AI
GitHub Repo: https://github.com/AnetteHardyUCLL/penguins-ml-agent

---

1. Introduction

This project explores reinforcement learning using Unity's ML-Agents toolkit. The goal was to create an AI-controlled penguin capable of balancing self-preservation with parental responsibilities, specifically managing hunger while feeding a baby penguin. The project expands upon the "Penguin AI" tutorial by Immersive Limit, adding custom logic inspired by real-life biological behaviours to introduce complexity and realism into the AI training scenario.

2. Data

The project generates synthetic training data dynamically through agent interactions with the Unity environment, eliminating the need for external datasets.

2.1 Preprocessing & Setup:

The Unity environment simulates dynamic fish spawning, hunger depletion for both penguins, and interactive behaviours. The agent observes various states, including whether it carries a fish, its distance and orientation relative to the baby penguin, its own direction, and current hunger levels. Visual detection of objects like fish and the baby penguin is achieved using Unity's RayPerceptionSensor3D with sphere-casts.

Preprocessing primarily involved tuning the observation vector and training parameters defined in Penguin.yaml. Challenges included ensuring accurate hunger state resets and managing environmental consistency between episodes.

3. Model & Methods

The Proximal Policy Optimization (PPO) algorithm from ML-Agents Toolkit (v1.1.0) forms the basis of the model. Key training parameters include a batch size of 128, buffer size of 2048, learning rate of 0.0003, entropy regularization (beta) at 0.01, clipping threshold (epsilon) of 0.2, and lambda (GAE) at 0.95. The network architecture consists of two layers with 256 hidden units each, and an extrinsic reward gamma of 0.99.

The discrete action space comprises four branches: forward movement, rotation, deciding whether to eat carried fish, and deciding whether to feed the baby. Rewards are structured to encourage fish collection (+1), discourage prolonged fish carrying (-0.1 per second), incentivize movement toward the baby while holding fish (+0.1 per reduced distance), penalize self-eating (-0.5), and highly reward successful feeding (+10), alongside penalties related to urgency from low hunger.

4. Results & Evaluation

The agent's performance plateaued around a mean reward of -1.0 after approximately 50,000 steps. It adopted a sub-optimal strategy, favouring repeatedly eating fish selfishly over feeding the baby. Diagnostic testing revealed that PPO's limited exploration due to low entropy prevented sufficient exploration of the rewarding feed action. Manual heuristic testing confirmed the correct functionality of the feeding mechanism, indicating an exploration issue rather than a bug in the code implementation.

5. Contributions

Original work included fully implementing the hunger management system with depletion, thresholds, and refill logic, along with conditional actions for eating and feeding. In addition, discrete action space design, reward structure aiming for biological realism, debugging, heuristic testing, and training configurations.

The base environment setup was adapted from Immersive Limit's tutorial: (https://www.youtube.com/watch?v=axF_nHHchFQ).

GenAI tools (ChatGPT) provided assistance for debugging, structuring logic, code reviews, and proofreading. All custom implementations and modifications were personally developed.

Disclaimer: I have prior experience in academic writing; thus, although AI was used to assist with structuring and proofreading this report, its style and content remain intuitive and reflective of my own work.

6. Challenges & Future Work

Reward imbalance led the agent to prefer selfish actions, and PPO's entropy (beta = 0.01) might have been insufficient for exploring the beneficial feeding action. Complex debugging emphasized careful management of episodic resets and state persistence.

6.1 Lessons learned

The presence of large rewards alone is insufficient if actions are rarely sampled during training. Proper state resets between episodes are critical, as overlooked resets (e.g., feedRequested flag) lead to subtle and challenging bugs. Furthermore, agent performance significantly deteriorates without appropriate ray-based perception.

6.2 Future Work & Improvements

- Adjust reward shaping by increasing the feeding bonus and lowering selfish-eating penalties to encourage exploration.
- Implement curriculum learning to progressively increase challenge.
- Introduce memory modules (e.g., LSTM or GRU layers) to help the agent retain past states and improve decision-making.
- Develop a visual debug interface within Unity to track hunger and action selection more intuitively.