Progressive Robotics

George Kesoglidis

October 20, 2025

Machine Learning Engineer Assignment

Design Analysis

Environment Design

The environment represents the agent's observation as a 12-dimensional vector containing:

1. **Distance to walls:** the number of steps from the agent to the boundaries of the grid in each of the four directions: up, down, left, right.
2. **Distance to nearest obstacle:** computed using a BFS algorithm, returning the number of steps to the closest obstacle in each direction.
3. **Distance to nearest target:** also computed using BFS, returning the number of steps required to reach the nearest bonus position or the final target, along each direction.

Reward Function: The reward function is composed of several components:

- Positive reward for collecting bonuses and reaching the final target.
- Small reward proportional to progress towards the target, calculated as the change in Manhattan distance from the agent to the current target or bonus.

These choices encourage the agent to learn a policy that efficiently navigates the environment, reaches targets in the correct order, avoids obstacles, and reaches the target in minimal steps. By encoding distances in directional steps rather than raw coordinates, the agent learns a spatially-informed representation that is invariant to absolute positions. Thus, it can generalize better in random environments.

Algorithm Comparison

The performance of on-policy (A2C) and off-policy (DQN) algorithms were evaluated in a 6x6 grid environment with 5 obstacles and varying numbers of bonus positions (0, 2, and 5), on 1000 random episodes. The main metrics considered were average steps per episode, average total reward, success rate, bonus locations reached, obstacle hits, and timelimits.

- **Simple environment (no bonuses):** DQN outperformed A2C in both success rate (77% vs 62%) and average reward (1.26 vs 1.05). This is likely because off-policy DQN is more sample-efficient and can quickly learn optimal paths in simple environments with sparse rewards. A2C took more steps per episode (10.55 vs 3.31), indicating slower convergence.
- **Expanded environment with 2 bonuses:** A2C slightly outperformed DQN in success rate (79% vs 68%) and total reward (2.715 vs 2.34), while taking fewer average steps per episode than DQN (12.82 vs 17.85). DQN required more than double the total number of episodes to converge in this case, likely because exploration of the two bonus positions required more planning, and DQN's off-policy nature struggles with sequential multi-goal objectives. A2C benefits from richer observation spaces, allowing more efficient planning and bonus collection.
- **Expanded environment with 5 bonuses:** Both algorithms performed similarly in success rate (A2C 72%, DQN 74%), average reward (2.585 vs 2.565) and steps per episode (11.48 vs 12.86). Interestingly, DQN converged faster here than with 2 bonus parameters , possibly because with five bonuses the environment allowed more natural exploration and shorter paths between unvisited bonuses. A2C maintained slightly better obstacle avoidance and consistent bonus collection. This suggests that as the number of bonuses increases, sequential planning becomes easier, reducing the difference between on-policy and off-policy performance.

Algorithm	Environment	Avg Steps	Avg Reward	Success %	Bonus	Obstacles	Timelimits
A2C	no bonus	10.55	1.05	62	0	16	22
DQN	no bonus	3.31	1.26	77	0	22	1
Expanded_b2 A2C	2 bonuses	12.82	2.715	79	182	14	7
Expanded_b2 DQN	2 bonuses	17.85	2.34	68	170	7	25
Expanded_b5 A2C	5 bonuses	12.86	2.585	72	165	17	10
Expanded_b5 DQN	5 bonuses	11.48	2.565	74	172	22	4

Summary:

1. **Off-policy DQN** excels in simple tasks with sparse rewards due to its sample efficiency, achieving higher success and shorter paths in simple settings.
2. **On-policy A2C** benefits from richer observations and sequential planning requirements, performing better when multiple bonuses must be collected before reaching the final target.
3. Increasing the number of bonus makes exploration easier and sequential planning more straightforward, narrowing the performance gap between A2C and DQN in richer environments.

Challenges

Several challenges were encountered during development:

- **Environment Solvability:** Random placement of obstacles and bonuses could create unsolvable configurations. This was addressed by checking path existence with BFS before finalizing the layout during environment reset.
- **Bonus positions:** Introducing bonus positions that the agent must reach before the final target significantly increased the complexity of the environment. To handle this, a mechanism was implemented to prioritize bonus locations, and reward shaping was used so that the agent values collecting bonuses while still pursuing the final target.
- **Observation Representation:** Initially, using a dictionary to represent the grid environment was enough for the simple case, but when adding bonus positions, the agent could not find meaningful solutions. This was solved by implementing a representation for obstacles and targets relative to the agent, using a BFS-based directional step. This more accurate navigation information helped the agent achieve similar results in the expanded grid environment

These design decisions drastically improved the agent's learning efficiency, stability, and overall performance in the grid-world environment.