Abstract
In the 1980s, Michael Toy and Glenn Wichman released a new game that would change the ecosystem of Unix games forever: Rogue. Rogue was a dungeon crawler—a game in which the objective is to explore a dungeon, typically in search of an artifact—which would result in many spinoffs, including the open-source NetHack, enjoyed by many today. NetHack is a game both beloved and reviled for its incredible difficulty—finishing a game is a rare accomplishment.

In this project, we explore the creation of a new NetHack reinforcement learning (RL) framework, Quixote, and the creation of a Q-learning agent from this framework. Additionally, we explore a domain-specific state representation and Q-learning tweaks, allowing progress without involvement of deep architectures.

Possible future directions include reorganization and packaging of Quixote into a bona fide Python package for public use, and the incorporation of deep RL into Quixote for higher performance.

Objective
NetHack: a “roguelike” Unix game
• Permadeath, low win rate
• Sparse “rewards,” large action space
• Massive number of game mechanics
• Unpredictable NPCs

Difficult but interesting environment for RL
• Focus on navigation (10 actions)
• Maximize points at end of game: go deeper

Framework
Model 1: Basic Q-Learning
Simple Q-learning model:
\[ Q \left( s_t, a_t \right) = \lambda \cdot Q \left( s_{t-1}, a_{t-1} \right) + \gamma \cdot \max_{a'} Q \left( s_{t+1}, a' \right) + \gamma \cdot \left( R(s_t, a_t) - Q \left( s_t, a_t \right) \right) f(s_t, a_t) \]

- Simple epsilon-greedy strategy
- Primary reward is delta score
- Retracing penalty, exploration bonus
- State representation:

Model 2: Approximate Q-Learning
Ensure that model is able to generalize between states w/ LFA:
\[ \theta = \theta + \alpha \cdot \sum_{(s, a)} \left[ R(s, a) + \gamma \cdot \max_{a'} Q(s', a') - Q(s, a) \right] f(s, a) \]

- Same hyperparameters, rewards, etc.

Model 3: \( \varepsilon \) Scheduling
Random works well—bootstrap off it:
• Hard to capture informative enough state
• Much exploration needed, then exploitation

Results

<table>
<thead>
<tr>
<th>Rand.</th>
<th>QL</th>
<th>QL + ( \varepsilon )</th>
<th>AQL</th>
<th>AQL + ( \varepsilon )</th>
</tr>
</thead>
<tbody>
<tr>
<td>45.60</td>
<td>44.92</td>
<td>51.96*</td>
<td>32.53*</td>
<td>49.52</td>
</tr>
</tbody>
</table>

Discussion
While shallow QL appeared to perform at random performance, the modifications made improved QL over baseline.

- Hard to say how useful state rep. was
- Hard to say if linear model underfit
- Bias from restarting when RL stalled
- Variance between runs too high to conclude

In any case, the Quixote framework despite several early bugs was rugged and has great potential as an RL environment.

Future Directions
Both for framework and for agent:
• Much, much more testing!
• Implement deep RL architectures
• Explore more expressive state reps.
• Directions to landmarks
• Finer auxiliary rewards
• Use “meta-actions”: go to door, fight enemy

References