Quixote: A NetHack Reinforcement Learning Framework and Agent

Chandler Watson[1]

[1]*Department of Mathematics, Stanford University*

Stanford
CS 229, Spring 2019

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 the game is seen as a great 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

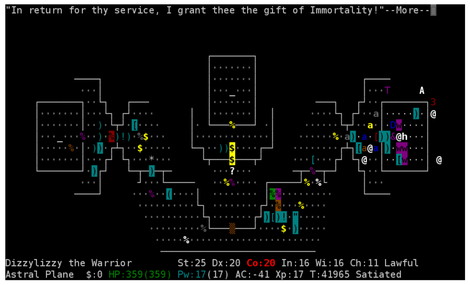


Figure 1: *A screenshot of the end of a game of NetHack.*
(https://www.reddit.com/r/nethack/comments/3ybcce/yay_my_first_360_ascension/)

Difficult but interesting environment for RL

- Focus on navigation (10 actions)
- Maximize points at end of game: go deeper

Framework

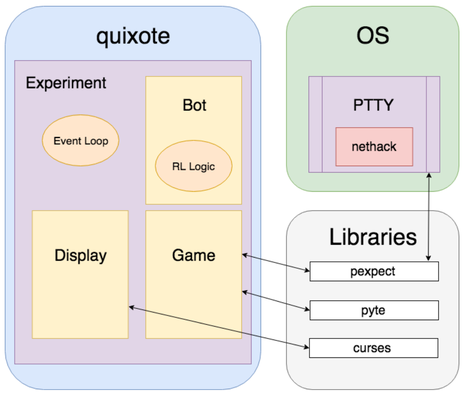


Figure 2: *The layout of the Quixote framework.*

Original framework for NetHack RL

- Interactivity allows rapid prototyping, debugging

Model 1: Basic Q-learning

Simple Q-learning model:

$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha)\cdot \underbrace{Q(s_t,a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1},a)}_{\text{estimate of optimal future value}}\right)}^{\text{learned value}}$$

(https://wikimedia.org/api/rest_v1/media/math/render/svg/47fa1e5cf8cf75996a777c11c7b9445dc96d4637)

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

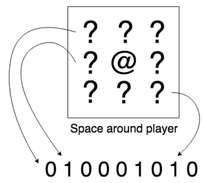


Figure 3: *The state representation for Model 1.*

- Choose relatively high ε = 0.1
- Try both per-state/action learning rates (opted for constant learning rate later)

$$\alpha_{s,a} := \frac{\alpha_0}{\#\text{ of times (s, a, s') is observed for any s'}}$$

- Validate over 20 episodes for discount factor

Model 2: Approximate Q-Learning

Ensure that model is able to generalize between states w/ LFA:

$$\theta := \theta + \alpha_{s,a}\left(R(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a)\right) f(s,a)$$

$$Q(s,a) = \theta^\top f(s,a)$$

- Same hyperparameters, rewards, etc.

Model 3: ε Scheduling

Random works well—bootstrap off it:

- Hard to capture informative enough state
- Much exploration needed, then exploitation

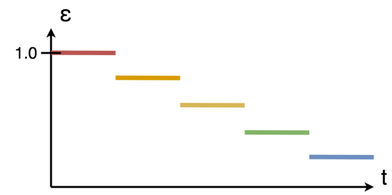


Figure 4: Stepwise epsilon scheduling.

Results

Rand.	QL	QL + ε	AQL	AQL + ε
45.60	44.92	51.96	32.53*	49.52

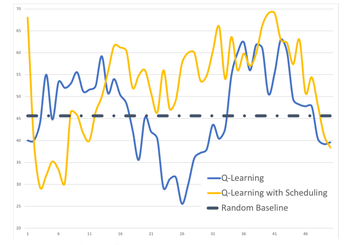


Figure 5: Results of running *Quixote* on 50 episodes (mean scores given). *Based on 30 episodes

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

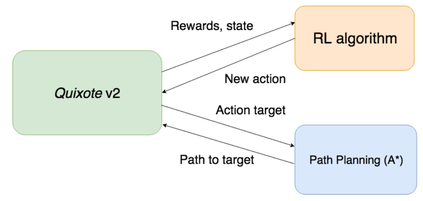
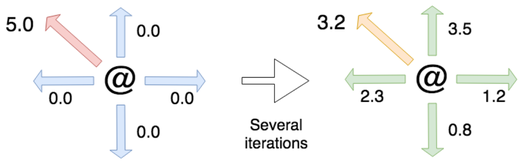


Figure 6: Possible ”meta-actions” for *Quixote*.

- Randomness as action:



$$Q(s,a) = \lambda \cdot 1\{a = \text{“random”}\}$$

Figure 7: *The “random as action” model.*

- Q-learning decreases randomness
- Blend policy and random

References

A. Fern, *RL for Large State Spaces. Available:* https://oregonstate.instructure.com/files/67025084/download

D. Takeshi, *Going Deeper into Reinforcement* Learning, 2019. Available: https://danieltakeshi.github.io/2016/10/31/going-deeper-into-reinforcement-learning-understanding-q-learning-and-linear-function-approximation/

Q-learning, Wikipedia, 2019. Available: https://en.wikipedia.org/wiki/Q-learning.

Y. Liao, K. Li, Z. Yang, *CS229 Final Report: Reinforcement Learning to Play Mario*, 2012. Available: http://cs229.stanford.edu/proj2012/LiaoYiYang-RLtoPlayMario.pdf

Stanford University