REINFORCE and Actor Critic Agents with Softmax Policy

Matthew Groholski, Neha Shaik, Rodney Staggers Jr, Anjali Mohanthy

***Abstract*—This endeavor seeks to utilize Pacman as an informed guide on the various algorithms of reinforcement learning; more specifically, REINFORCE, Actor-Critic, and Approximate Q-learning. These are fundamental algorithms within the realm of reinforcement learning providing varying techniques and methods to update the policy and predict future rewards. Within this report, we provide statistics about the different convergence times within different layouts.**

I. Introduction

PACMAN is an internationally recognized game that has captured the hearts of millions of people. While Pacman has been the source of entertainment for millions during its forty year existence, the layman's game can serve as the entry point for understanding the complexity algorithms modeled in artificial intelligence.

Reinforcement Learning compels an agent to interact with an environment an iterative amount of times in order to maximize its reward. Unlike supervised learning, which relies on labeled data, Reinforcement Learning involves learning through trial and error, receiving feedback in the form of rewards or penalties based on the actions taken.

An Actor-Critic algorithm relies on a criterion to evaluate prospective actions, optimizing itself by determining the path that guarantees the best outcome. By updating its parameters within a feedback loop.

Q-learning is a model-free Reinforcement Learning algorithm that prioritizes the optimal state-action pair to maximize its value function.

II. Technical Approach

WITHIN this section we will outline the technical approach for the REINFORCE and Actor Critic agents.

A. *Softmax Policy*

Each of the agents utilize the Softmax policy:

$$\pi(a|s,\boldsymbol{\theta}) = \frac{e^{\boldsymbol{\theta}^\intercal \mathbf{x}(s,a)}}{e^{\boldsymbol{\theta}^\intercal \mathbf{x}(s,\cdot)}} \tag{1}$$

, where $\cdot$ means all possible actions and $\mathbf{x}(s,a)$ is the feature vector of state s with action a . The policy uses $\boldsymbol{\theta}$ as weights for the features and normalizes the computation with the sum of all possible actions. Furthermore, both agents require the gradient of the softmax policy for updates. We will be using the gradient of the log for simplicity:

$$\nabla_{\boldsymbol{\theta}} \ln\, \pi(a|s,\boldsymbol{\theta}) = \mathbf{x}(s,a) - E_{\pi_{\boldsymbol{\theta}}}[\mathbf{x}(s,\cdot)] \tag{2}$$

, where E is the expected value function. The expected value function, in this context, is defined as $E_{\pi_{\boldsymbol{\theta}}}[\mathbf{x}(s,\cdot)] = \pi(s,a_1)\mathbf{x}(s,a_1) + ... + \pi(s,a_n)\mathbf{x}(s,a_n)$. Notice E is the weighted sum of the feature vectors.

B. *Feature Vector*

Our features for both the REINFORCE and Actor-Critic algorithms were inspired by the Stanford paper *"Reinforcement Learning in Pacman"*. Within the paper, five features are outlined: the number of scared ghosts, active ghosts one and two steps away, eating food if there are no active ghosts nearby, and the distance to the closest food [1]. However, their environment approaches the problem with a neural network which does not accurately translate into our environment. As such, we have refined these features with the Softmax policy. Within the scope of this project, the features that were outlined for the basic reflex Pacman were determined by:

1) The agent's distance to either food or a capsule.
2) The amount of ghosts within n proximity.
3) A boolean value representing whether to eat food.
4) The number of scared ghosts.

These features provide the agent with enough information about it's local surroundings without causing over fitting. Therefore, the agent is able to learn an optimal policy with enough exploration.

C. *REINFORCE*

We will outline the evaluation and update process within REINFORCE and give an in-depth discussion of the features.

Algorithm 1 REINFORCE [2]

```
   INPUT: A differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$.
1: Initialize policy parameter $\boldsymbol{\theta} \in R^{d'}$.
2: while forever do
3:     Generate an episode $S_0, A_0, R_1, ..., S_{T-1}, A_{T-1}, R_T$
       following $\pi(\cdot|\cdot,\boldsymbol{\theta})$.
4:     for each step of the episode $t = 0, ..., T-1$ do
5:         $G \leftarrow$ return from step $t$
6:         $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\gamma^t G \nabla_{\boldsymbol{\theta}} \ln\, \pi(a|s,\boldsymbol{\theta})$
```

REINFORCE is an algorithm that finds optimal feature weights using gradient descent with a differentiable stochastic policy. We begin with a randomly initialized weight vector $\boldsymbol{\theta}$. The training process consists of two steps: episode generation and policy update. For each episode, the current policy is followed to generate a sequence of states, actions, and rewards until a terminal state is found. We then use the episode data to update the weight vector according to line 6 of algorithm 2. Notice the update equation utilizes learning rate α and reward discount γ parameters to allow for larger leaps and prioritization of immediate rewards.

D. ACTOR-CRITIC

We will outline the evaluation and update process within ACTOR-CRITIC.

Algorithm 2 ACTOR-CRITIC ALGORITHM

```
    INPUT: A differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$,
    a differentiable state-value parameterization $\hat{v}(s,\mathbf{w})$.
 1: Initialize policy parameter $\theta \in \mathbb{R}^d$ and state-value weights $\mathbf{w} \in \mathbb{R}^d$.
 2: while forever do
 3:     Initialize $S$ (first state of episode)
 4:     $I \leftarrow 1$
 5:     while $S$ is not terminal do
 6:         $A \sim \pi(\cdot|S,\theta)$
 7:         $\delta \leftarrow R + \gamma v(S',w) - \hat{v}(S,w)$ (if $S'$ is terminal, then $\hat{v}(S',w) := 0$)
 8:         $\mathbf{w} \leftarrow \mathbf{w} + \alpha^w I\delta\nabla_w \hat{v}(S,\mathbf{w})$
 9:         $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^\theta I\delta\nabla_\theta \ln\pi(A|S,\boldsymbol{\theta})$
10:         $I \leftarrow \gamma I$
11:         $S \leftarrow S'$
```

The actor critic algorithm finds an optimal policy by updating after each observed action. The update is split into two steps: a critic and actor update. Each style provides a different benefit to the agent within its learning process.

1) Critic Implementation: The critic is responsible for exploitation. At its core, is a differentiable approximate state-value function $\hat{v}(s,\mathbf{w})$, where s is the state and $\mathbf{w}$ is a weight vector. The state-value function used within our implementation creates an expected feature vector and linearly adds together the values weighted by $\mathbf{w}$:

$$\hat{v}(s,\mathbf{w}) = \mathbf{w}^\intercal E_{\pi_\theta}[\mathbf{x}(\mathbf{s},\cdot)] \tag{3}$$

Therefore, throughout the algorithm the critic is learning how to weight each of the expected values. This approach to the state-value function can be likened to one-step look ahead. The state-value gradient is

$$\nabla_\mathbf{w} \hat{v}(s,\mathbf{w}) = E_{\pi_\theta}[\mathbf{x}(\mathbf{s},\cdot)] \tag{4}$$

2) Actor Implementation: The actor is responsible for exploration. We use the policy outlined in section IV-D to update the $\boldsymbol{\theta}$ vector.

An "actor" that chooses an action based on a policy and a "critic" that assesses the action using a value function make up the ACTOR CRITIC Reinforcement Learning approach. The temporal difference error, which shows how the action varied from expected rewards, is computed by the critic and used by the actor to adjust its policy in response. The algorithm is able to efficiently balance exploration (trying new actions) and exploitation (using known successful actions) thanks to this dual approach. Over time, decision-making is improved as a result of the algorithm's dynamic adjustment of the policy based on continuous learning and feedback from the environment. Notice the update equation utilizes learning rate $\alpha_{\boldsymbol{\theta}}$, $\alpha_\mathbf{w}$ and reward discount γ parameters to allow for larger learning leaps and prioritization of immediate rewards.

E. Experiments

Within our system of environments, there were technically 13 layouts. These individual layout encompass differing complexities that impede the Pacman agent's success. Each environment provides a unique challenge for the agent with varying wall structures, map sizes, food placement, and capsule amount. Furthermore, each layout can be configured for a certain amount of ghosts. Thus, our experimental setup choose three environments and ghost amounts ($[1,5]$) uniformly at random. We compare the convergence rates of three agents within the selected configuration: REINFORCE, Actor Critic, and Approximate Q-Learning. For each of the tests ran, the parameters are set to:

TABLE I
AGENT PARAMETERS

	$\alpha_{\boldsymbol{\theta}}$	$\alpha_\mathbf{w}$	γ	Feature Extractor
REINFORCE	0.2	-	0.8	-
Actor Critic	0.25	0.15	0.9	-
Approx. Q-Learning	-	-	-	Simple Feature Extractor

For the large convergence test, we averaged the 300 episodes over 200 runs of each agent. For the randomized convergence tests, we averaged the 100 episodes over 48 runs of each agent. During each run, our script randomly chose a layout; each one consisted of differing amounts of ghosts within different environmental dimensions.

III. RESULTS

We have provided results for four convergence experiments ran (three randomized and a large-scale convergence). See section II-E for experiment set up.

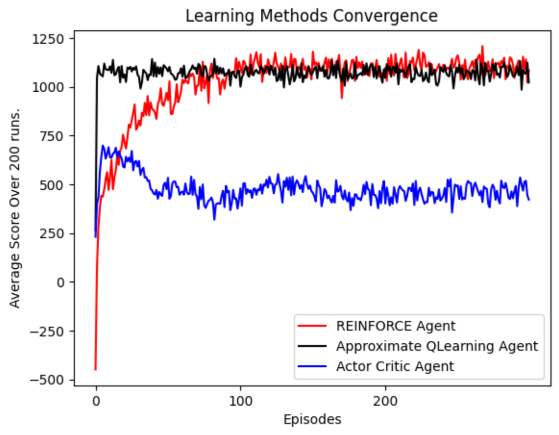

Fig. 1. Completed on $'mediumClassic'$ layout with 2 Ghosts, for 300 episodes averaged over 200 runs.

The t-test statistics of the dataset for the 2-Ghost $'mediumClassic'$ layout identified that the QLearning and

Actor-Critic algorithms share zero correlation amongst their average scores. More specifically, the t-statistic and p-values of the two algorithms are 116.600 and 0.000, respectively. Amongst the REINFORCE and Actor-Critic algorithms, the t-statistic and p-values are 47.951 and $4.603*10^{-207}$, respectively. Finally, the REINFORCE and QLearning algorithms has a t-statistic value of -4.150 and a p-value of $3.809*10^{-5}$.

Over the course of this dataset, there are some interesting anomalies that appear. For starters, the Actor-Critic agent seems to undergo the most fluctuations throughout the episode count. After the first 50 episodes, the Actor-Critic method expeditiously drops to an average of 400 (from around 700) for another 60-70 episodes (as shown in Figure 1). Overall, the t-tests determine that the QLearning algorithm performed the best for this layout.

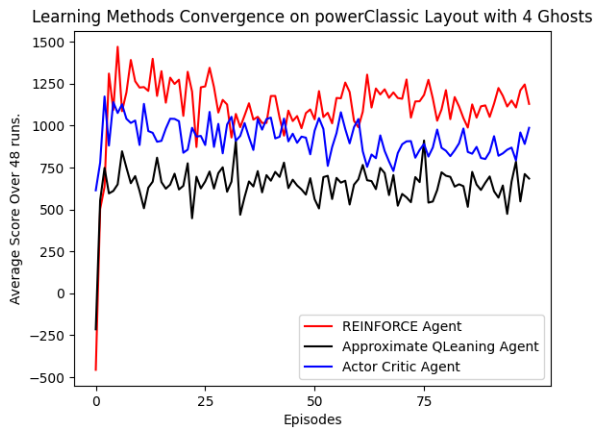

Fig. 2. The convergence in the average reward of the Learning Methods, after 48 runs, for the $'powerClassic'$ layout containing 4 Ghosts.

As shown in Figure 2, our algorithm performed the three machine learning methods for a duration of 48 runs (each run containing 100 episodes). With a threshold p-value of 0.05, the t-test statistics identified that none of the methods had significantly similar average scores. The REINFORCE and QLearning Methods acquired a t-statistic and p-value of 19.639 and $2.286*10^{-48}$. Given the fact that the t-statistic value was a positive rational number and the p-value was approximately zero, this is a significant sign that the average score for REINFORCE Learning outclasses the QLearning results. By utilizing the graphics of Figure 2, one can see that the REINFORCE Learning scored approximately 1150 on average, whereas the QLearning model scored around 600 on average. With a t-statistic and p-value of -17.707 and $1.118*10^{-42}$, the correlation between the QLearning and Actor-Critic methods mirrors the former statistic. Additionally, the negative value of the t-statistic shows that the QLearning Method's average score was significantly lower than the Actor-Critic's. While significantly different from each other, the t-statistic and p-value between the REINFORCE and Actor Critic dataset (8.473 and $5.418*10^{-15}$, respectively) showed that the techniques were closer in relation with each other than with the QLearning dataset.

The analytics for this $'powerClassic'$ layout clearly favor the utilization of the REINFORCE Learning algorithm. For starters, all three algorithms were able to converge to a positive score within 10 episodes. While further analytics could be administered to register the Success rate of all three systems, the current statistical data acknowledges that all three algorithms encouraged a positive score throughout each run.

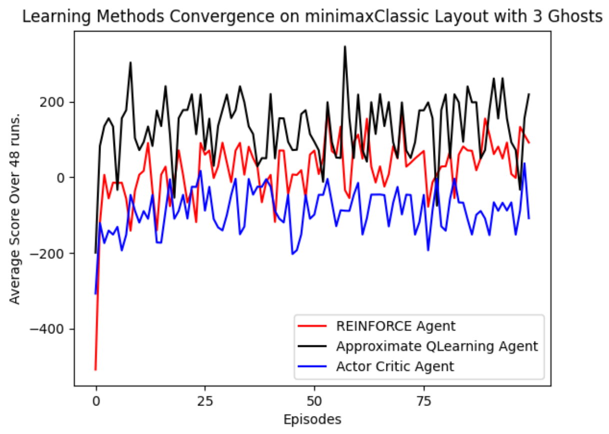

Fig. 3. The convergence in the average reward of the Learning Methods, after 48 runs, for the $'minimaxClassic'$ layout containing 4 Ghosts.

As in the previous layout, the t-test statistics identified that none of the methods had significantly similar average scores. The most severe deviation amongst algorithms occurred between the QLearning and Actor-Critic methods, which resulted in a t-statistic and p-value of 21.605 and $5.478*10^{-54}$, respectively. These conclusions (along with the graphical data of Figure 3) show that QLearning algorithm performed significantly better than the Actor-Critic algorithm. The second t-test statistic showed that the REINFORCE and QLearning Methods acquired a t-statistic and p-value of -9.137 and $7.707*10^{-17}$. Finally, with a t-statistic and p-value of 10.742 and $1.679*10^{-21}$, the correlation between the REINFORCE and Actor-Critic methods mirrors the former statistic.

The analytics for this $'minimaxClassic'$ layout favor the utilization of the QLearning algorithm. However, unlike the previous layout, this domain resulted in meager scores across the board. These minuscule averages can probably be attributed to the 3 Ghost agents intentionally striving to minimize the Pacman's score (as is the modus operandi of the minimax game). On average, the QLearning algorithm was the only method to have a positive average (approximately 100-150), whereas the REINFORCE Learning averages (approximately 0), and Actor-Critic averages (approximately -100) were abysmal.

The REINFORCE and QLearning Methods acquired a t-statistic and p-value of 28.747 and $1.356*10^{-72}$. By utilizing the graphics of Figure 4, one can see that the REINFORCE Learning scored approximately 1300 on average, whereas the QLearning model scored around 500 on average.

While significantly different from each other, the t-statistic and p-value between the REINFORCE and Actor Critic dataset (17.552 and $3.243*10^{-42}$, respectively) showed that the tech-

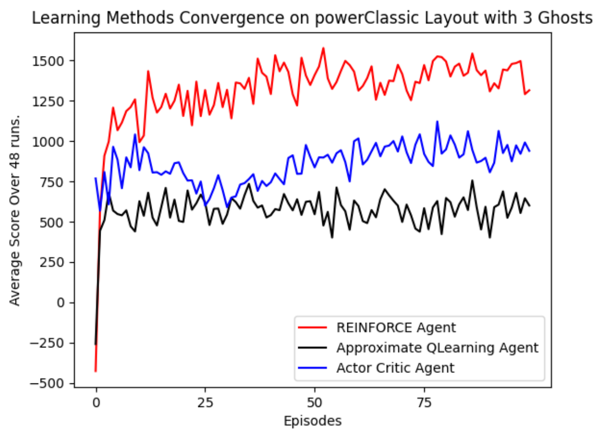

Fig. 4. The convergence in the average reward of the Learning Methods, after 48 runs, for the $'powerClassic'$ layout containing 3 Ghosts.

niques were closer in relation with each other than with the QLearning dataset. With a t-statistic and p-value of -17.703 and 1.152*10^{-42}, the correlation between the QLearning and Actor-Critic methods mirrors the former statistic. Additionally, the negative value of the t-statistic shows that the QLearning Method's average score was significantly lower than the Actor-Critic's.

The analytics for this $'powerClassic'$ layout with 4 Ghosts favor the utilization of the REINFORCE Learning algorithm. Although all algorithms converged to a positive score, unlike the $'powerClassic'$ layout with 3 Ghosts, it took approximately 30 episodes to do so. Additionally, the REINFORCE agent performed far better here with an additional ghost in the environment, whereas the other two agents resulted with worse score averages.

IV. Analysis

A. Analysis of learning methods on 'mediumClassic' Layout with 2 Ghosts:

The REINFORCE algorithm consistently performed well, with the Actor-Critic method exhibiting the most fluctuation. The QLearning algorithm was identified as the best performer for this layout based on the t-test results. Additionally, this algorithm converged to its average almost instantaneously, whereas the REINFORCE algorithm took 100 episodes and the Actor-Critic took 30-50 episodes to do the same. The Actor-Critic agent experienced a significant drop early on but later stabilized to some extent.

B. Analysis of learning methods on 'powerClassic' Layout with 4 Ghosts:

In this layout, The t-tests indicated that all methods performed differently from one another. The REINFORCE algorithm outperformed the QLearning algorithm with a higher average score and maintained a significantly higher average score, followed by the Actor-Critic, with QLearning performing the least effectively.

-

C. Analysis of learning methods on 'minimaxClassic' Layout with 3 Ghosts:

All methods scored lower on average compared to other layouts, likely due to the minimax strategy of the ghosts which minimizes the player's score. The QLearning algorithm performed better than the other two methods, being the only one with a positive average score. The REINFORCE and Actor-Critic algorithms had scores around zero and negative averages, respectively.

D. Analysis of learning methods on 'powerClassic' Layout with 3 Ghosts:

In this environment, all algorithms eventually reached a positive score after about 30 episodes. The REINFORCE algorithm showed superior performance, scoring approximately 1300 on average. The REINFORCE algorithm's performance improved with the addition of another ghost, whereas the others declined in performance.

As mentioned above, from the multiple experiments we've run so far, it appears that the **Reinforcement** algorithm seems to be the most robust across different layouts and numbers of ghosts, consistently achieving high average scores.Of the four layouts that were showcased in this report, REINFORCE had an average score around 950, whereas QLearning averaged 693.62, and Actor-Critic averaged a score around 650. The **QLearning** algorithm's performance varied significantly across different layouts, indicating it may be more sensitive to changes in the environment. The **Actor-Critic** method showed considerable fluctuations and, in some layouts, scored lower than the other methods.

V. Conclusion & Discussion

Comparing the results of Reinforcement Learning, Q-learning, and Actor-Critic methods is crucial for several reasons. Each approach represents a different paradigm in Reinforcement Learning, with unique advantages and challenges. **Q-learning**, a value-based method, focuses on estimating the optimal action-value function, providing a straightforward way to derive optimal policies but often faces challenges with stability and convergence in complex environments. **Reinforce learning**, a type of policy gradient method that directly optimizes the policy based on entire episodes, stands out in the class of reinforcement learning techniques for its straightforward approach to improving policy based on the rewards obtained. Actor-Critic methods combine policy-based and value-based approaches, potentially leading to more stable learning with lower variance in policy updates, making them suitable for continuous action spaces and complex state representations. These differences can significantly impact a model's performance, robustness, and scalability. By comparing the results across these methods, researchers and practitioners can identify which approach is best suited for a given task, understand the trade-offs in terms of convergence, sample efficiency, and computational complexity, and develop more effective reinforcement learning algorithms. Ultimately, these comparisons drive innovation in Reinforcement Learning, leading to more robust and efficient solutions for real-world applications like robotics, game AI, and autonomous systems.

Software Artifacts

Source code used for REINFORCE and Actor Critic in Section II is available at https://github.com/mgroholski/REINFORCE-and-actor-critic-pacman.

References

[1] J. A. Abeynaya Gnanasekaran, Jordi Feliu Faba, "Reinforcement learning in pacman," Standford University, https://cs229.stanford.edu/proj2017/final-reports/5241109.pdf, Tech. Rep.

[2] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3, pp. 229–256, 1992. [Online]. Available: https://doi.org/10.1007/BF00992696