Reinforcement learning for PACMAN

Jing An, Jordi Feliu and Abeynaya Gnanasekaran

{jingan, jfeliu, abeynaya}@stanford.edu

Abstract

We apply reinforcement learning on the classical game PACMAN; we study and compare Q-learning, Approximate Q-learning and Deep Q-learning based on the total rewards and win-rate. While Q-learning proved to be quite effective on `smallGrid`, it was too expensive for larger grids. In Approximate Q-learning, we handcraft 'intelligent' features which are fed into the game. To ensure that we don't miss important features, we implement a Convolutional Neural Network (CNN) that can implicitly 'extract' the features.

Methods

We can try to minimize the following loss function:

$$L(s,a) = (r + \gamma \max_{a'} Q(s',a') - Q(s,a))^2$$

We start by implementing Q-Learning for Pacman based on the Berkeley's Pacman project [1].

1. **Approximate Q-Learning**
   The Q function is approximated by a combination of features $f_i(s,a)$ extracted from the game states

$$Q(s,a) = \sum_{i=1}^{n} f_i(s,a) w_i$$

   where the weights are updated by:

$$w_i \leftarrow w_i + \alpha \cdot (r + \gamma \max_{a'} Q(s',a') - Q(s,a)) \cdot f_i(s,a)$$

2. **Deep Q-Learning**
   The Q-values for different state and action pairs are computed with a Convolutional Neural Network (CNN) that can implicitly "extract" the features using the pixel data from a downsampled image of a state s. The neural network architecture comprises three convolutional layers followed by a fully connected hidden layer.

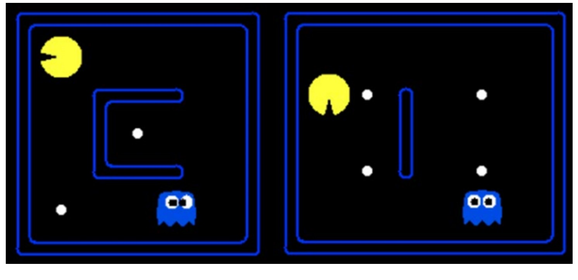

Figure 1: Pacman with `smallGrid` and `mediumGrid` layout

Feature Selections in Approximate Q-Learning

We perform ablative analysis to select the most important features in terms of total reward, win-rate and conclude that the best features are: number of scared, active ghosts one step away, eat food if there are no active ghosts nearby. The number of active ghosts one step away and eat food features are essential for winning a game and hence we do not include them in the analysis.

Table: Ablation Analysis

Component	Average Score	Win Rate
Overall System	1608	92%
inv. min. distance active ghost	1583	91.6%
min. distance active ghost	1594	91.8%
min. distance capsule	1591	90.2%
no. scared ghosts 2 steps away	1545	92.2%
no. scared ghosts 1 step away	**1438**	91.2%
distance to closest food	**1397**	90%

Deep Q-learning with Experience Replay

We use TensorFlow to implement a Convolutional Neural Network (Q-network), to generate the Q-values corresponding to the five possible directions at each state: North, South, East, West, Stop.

1. Create **equivalent images** of frames with each pixel representing objects in the Pacman grid.
2. Use **Replay memory** to store the last N experiences (s,a,r,s',a').
3. Sampling a minibatch randomly from replay memory.
4. Use an additional **target network** ($\hat{Q}$-network) to generate the target values for Q-network. Once every C iterations the target network is updated with the learned parameters from Q-network by minimizing the loss

$$L = \frac{1}{T}\sum_{i=1}^{T}(r_i + \gamma \max_{a_i'} \hat{Q}(s_i', a_i') - Q(s_i, a_i))^2$$

5. **Epsilon-greedy** strategy for exploration. We gradually reduce ϵ from 1 to 0.1 during training.

DQN Pipeline

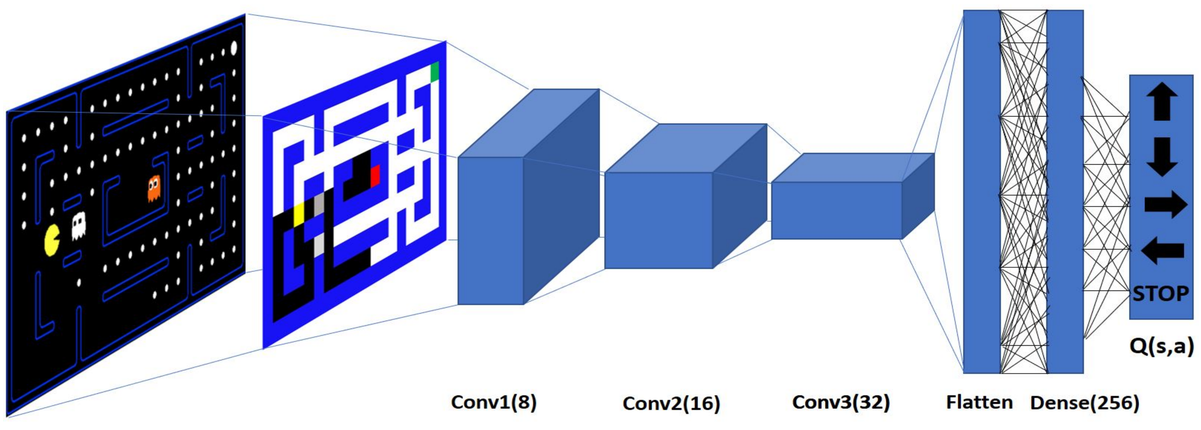

Figure 2: DQN pipeline with an equivalent image, three convolutional layers, a flattening layer and a fully connected hidden layer

Results of DQN

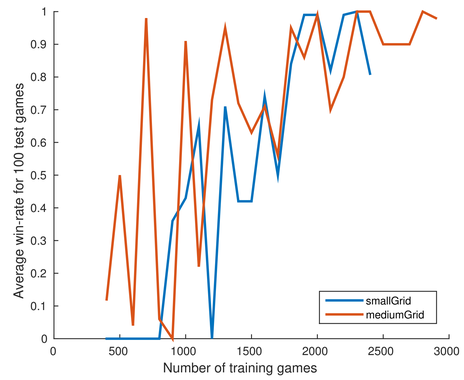

Figure 3: Average win-rate on `smallGrid` and `mediumGrid` with the number of training games

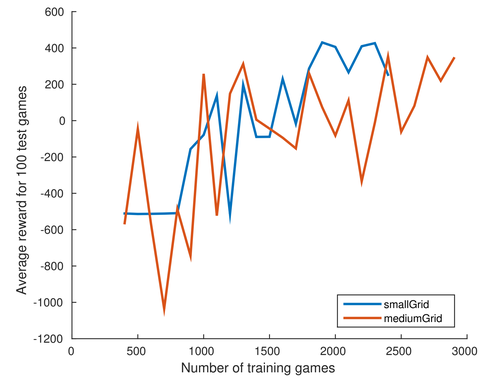

Figure 4: Average rewards on `smallGrid` and `mediumGrid` with the number of training games

Future Work

- Increasing training for better performance in medium grids
- Adding prioritized replay by sampling from distribution according to rewards or choosing the samples from which the agent can learn the most
- Adding auxiliary rewards
- Adding asynchronous gradient descent for optimization of the deep neural network

References

[1] Berkeley Pacman Project.
http : //ai.berkeley.edu/project_overview.html.

[2] Volodymyr Mnih et al.
Playing atari with deep reinforcement learning.

[3] Volodymyr Mnih et al.
Human-level control through deep reinforcement learning.

Conclusion

- Use of intelligent features for Approximate Q-learning works well for small and medium grids
- The developed architecture for DQN gives a reasonably good performance for small and medium grids on which it was tested. Also the computational cost was reasonable without the use of GPUs.