Albert Zhang, Neuro 240

Impact of Changing Transition Function in Deep RL

Project Hypothesis

- Question: With regards to deep reinforcement learning (RL), we are curious to study the impact of changing the transition function between training and testing on an RL agent's performance, specifically in the case of an RL agent that has been trained to play the video game Pac-Man where we change the probabilities of enemy ghost movements between training and testing.
- Expected answer: We expect that changing this transition function will decrease the RL agent's performance, and that a larger change in this transition function will lead to a larger decrease in that performance.

Dataset/Environment

- Description: because our project uses reinforcement learning rather than supervised learning, we do not need a dataset to train our RL agent to play Pac-Man. Instead, we plan on using deep reinforcement learning, specifically Deep Q-learning (DQN), to train our agent by having it learn Q-values for state-action pairs, which should capture how good making certain actions in certain states are based on the expected reward. Our equivalent of a dataset is the implementation of Pac-Man in Python, since this implementation will provide us with the game environment that we need to train using DQN, so the Pac-Man environment acts as our "dataset".
- Publicly available? Yes, the Pac-Man implementation in Python is publicly available at http://ai.berkeley.edu/project_overview.html
- Did you find any code-bases using this environment? Yes, an example project on using deep RL to train Pac-Man is publicly available at https://github.com/tychovdo/PacmanDQN
- Do you have the compute power to train/test on this environment? Yes, I have already tried using DQN to try training a Pac-Man agent, and my computer was able to train relatively successfully (the agent did not have great performance, but I'm confident that me working on this project will yield better results). Pac-Man is not as complex as Chess or Go, so I should be able to train and test using either my computer or affordable cloud tools like Google Colab.

Model Input

Model will be fed in the Pac-Man environment, which includes the states, actions, rewards, transition functions, and more, that our agent is then trained on using DQN.

Model Output

Albert Zhang, Neuro 240

Output of training will be a Pac-Man agent that attempts to play (and hopefully beat) the Pac-Man game. This Pac-Man agent will be able to pick actions in each state of the environment in order to play the game and earn rewards.

Model Architecture

We will use a Deep Q Learning architecture using TensorFlow available at https://github.com/mrkulk/deepQN_tensorflow that is based on the reputable Arcade Learning Environment (ALE) found at https://github.com/mgbellemare/Arcade-Learning-Environment.

Evaluation Metric

Model performance will be measured as the total reward earned by the trained agent in an episode, which corresponds to one game, averaged over multiple games (to reduce variability), where the reward is defined in the Pac-Man environment, where higher reward is better. We may also use the win rate (the percentage of games that the trained agent wins) to evaluate model performance, where higher win rate is better.