Investigating the effects of Policy Network structure

Andrew Zhang (andrewdu)
CS 221 (Artificial Intelligence), Stanford University

Abstract

- Numerous research in deep reinforcement learning (RL) has been centered on speeding up agent training through techniques such as improving sampling inefficiency or policy loss [1].
- The interest of this study is to investigate the simple inquiry of whether network depth (number of neurons per layer, number of hidden layers) and choice of activation function directly impact overall agent training performance.
- While the complexity is not advanced, most people tend to overlook these factors when conducting deep RL research since things such as configuring the policy loss is (as this study demonstrates) more potent.

Methods

- We ran trials of 100 episodes each with 0.01 learning rate and 0.98 discount factor to test three independent variables: hidden neuron count, hidden layer count, and activation function.
- The hidden neuron count, started from two and doubled up to 128.
- The layer increased linearly from zero to four.
- We chose three different activation functions: tanh, ReLU, and sigmoid.
- PPO, our choice of policy gradient method, optimizes a clipped surrogate objective to ensure that the ratio between the new and old policy stays within $1 - \epsilon$ and $1 + \epsilon$ (A_t denotes the advantage at each timestep):

$$L^{PPO}(\theta) = E_t[\min(r_t(\theta)A_t, clip(r_t(\theta), 1 - \varepsilon, 1 + \epsilon)A_t)]$$

Data

➢ CartPole-v0 (balance a pole on a cart):
- Observations – 4-tuple containing cart position, velocity, angle, and tip velocity
- Actions – 0 or 1 denoting a push to the left or right
- Rewards – 1 for every step taken

Results

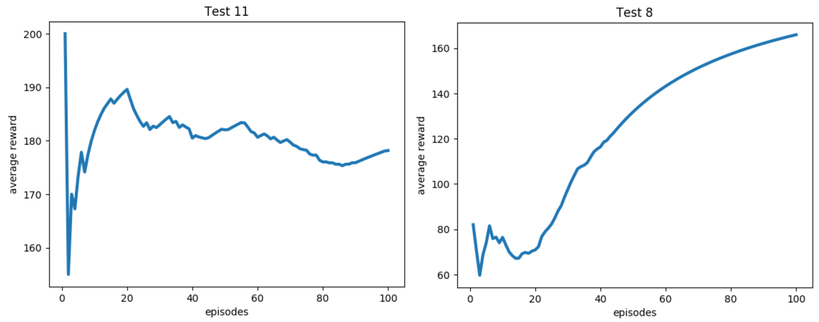

Figure 1: Effect of additional hidden neurons in hidden layers

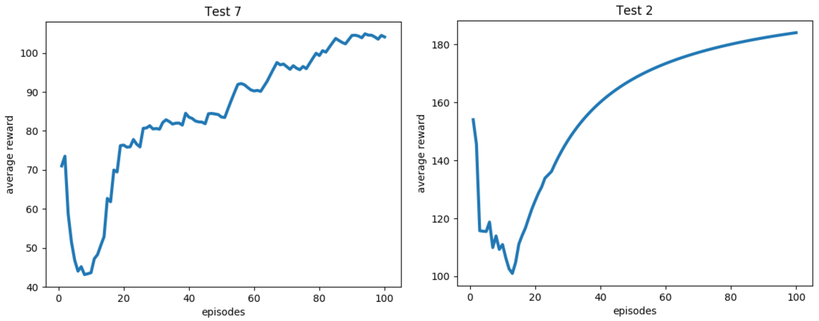

Figure 2: Effect of additional hidden layers

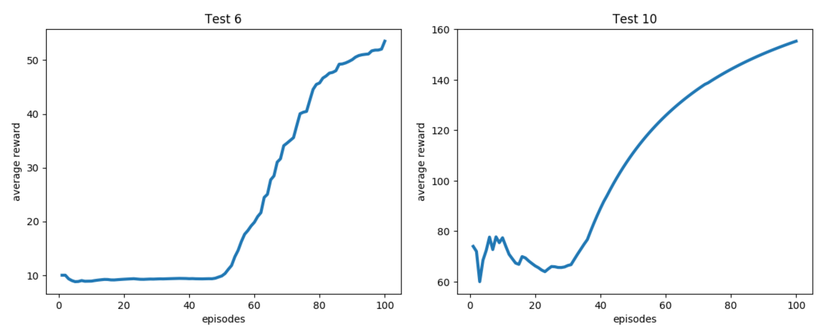

Figure 3: Effect of activation function (sigmoid vs ReLU/tanh

Conclusions

- When the number of hidden neurons is relatively small (four or less), the agent's trajectory is generally not as smooth as compared to one using a higher hidden neuron count (at least eight).
- Similarly, the effect of adding additional layers is also logarithmic in the sense that the effect slopes off and is more apparent when the hidden layer count is near zero. We found that adding additional hidden layers after two or three does not drastically improve performance and only serves to increase runtime.
- For PPO and classic control environments, we see although any of sigmoid, ReLU, or tanh can perform, sigmoid is not an ideal activation function simply because it is not zero-centered and encounters the vanishing gradient issue (hence why it is better for binary classification). In terms of general usage, it seems ReLU is the optimal choice for deep RL since it does not vanish gradients at saturation and is also relatively computationally inexpensive.
- Overall, the results are pretty linear and expected. That being said, it is still advantageous to peel back the hood to compare and contrast similarities of deep RL to more standard ML tasks.
- This is extendable to different agent environments and policy gradient methods (although the runtime tradeoff will be higher).

Future Work

- Adapt to original minimax idea for further optimized game playing.
- Currently, the learning paradigm based on a human is to iteratively improve on a black state by "teaching" the model what is right and wrong – this project has piqued an interest in constructing a potentially different a learning archetype.

References

[1] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov, "Proximal Policy Optimization Algorithms," *arXiv*, 2017