Review: Introduction to RL

Week 6 Tuesday paper 1: Proximal Policy Optimization

Supplemental: PPO Algorithms

Questions:

1. **What are the types of stochastic policies in Reinforcement Learning?**

Ans: Stochastic policies in Reinforcement Learning refer to policies that produce actions based on a probability distribution, as opposed to deterministic policies that map states to specific actions. A few types of stochastic policies include,

Categorical Policies: They are used in discrete action spaces, where the policy outputs a probability distribution over a finite set of possible actions. The action to be taken is selected based on this distribution.

Gaussian Policies: They are used in continuous action spaces. The policy outputs mean and variance parameters of a Gaussian distribution, and actions are sampled from this distribution. This allows for exploration in the action space.

Softmax Policies: Often used for discrete action spaces, similar to categorical but the probabilities are computed using the softmax function over the action values, allowing for a more stable selection process that's influenced by the relative differences between the action values.

Epsilon-Greedy Policies: While primarily used in the context of exploration, epsilon-greedy policies can be considered stochastic. In this approach, there is a fixed probability (epsilon) of choosing an action at random, and a 1-epsilon probability of choosing the best action according to the current policy. This ensures that the agent explores the environment by not always selecting what it currently believes is the best action.

2. **How is loss defined for RL algorithms? Is it an accurate measure of the performance of the agent**

Ans: In RL, the concept of loss is not as straightforward as in supervised learning, primarily because there isn't always a clear target or label for every situation. However, various forms of loss functions are used in RL, particularly in value-based and policy-based methods, as well as actor-critic methods. Here's how loss is defined in different contexts within RL:

Value-Based Methods: In these methods, the loss is often defined as the difference between the currently estimated state-action values (Q-values) and the target Q-values, which are derived from the Bellman equation. The loss function often used is the MSE between the target and the estimated Q-values. This is known as the Temporal Difference error in classical RL, and it measures how far the current value estimates are from the expected values.

Policy-Based Methods: In policy-based methods, the loss is usually defined through the policy gradient, which aims to maximize the expected return. The objective function often involves the logarithm of the policy multiplied by the advantage or return. In this case, the loss is actually a negative log-likelihood, which is minimized when the policy leads to higher returns.

Actor-Critic Methods: These methods combine elements of both value-based and policy-based learning. The actor updates the policy based on the policy gradient, similar to policy-based methods, while the critic updates value function estimates, similar to value-based methods. Here, there are typically two different loss functions- one for the actor and one for the critic.

Distributional RL: In some advanced RL algorithms, the loss function involves comparing entire distributions of returns rather than just expected values. This can provide a more nuanced understanding of the possible outcomes and their probabilities.

The loss in RL, as defined above, provides a measure of how well the agent is expected to perform based on its current policy or value function estimates. However, it is not always a direct or comprehensive measure of an agent's performance because of the following reasons,

Exploration vs. Exploitation: An agent might incur high loss in the short term because it is exploring the environment, which is essential for learning optimal policies. High loss in this context does not necessarily indicate poor performance.

Delayed Rewards: In many RL problems, rewards and the true consequences of actions are delayed. An agent might appear to perform poorly according to immediate loss metrics but could be optimizing for long-term gains.

3. **How are Policy Gradients computed? How is it different from gradient descent?**

Ans: Policy Gradients are computed as the gradient of the expected return (total accumulated rewards) with respect to the policy parameters. This involves computing the derivative of the log of the policy times the reward or advantage (how much better an action is compared to the average or baseline). This method is different from standard gradient descent primarily because it optimizes expected returns rather than minimizing a loss function. Furthermore, policy gradient methods directly parameterize and update the policy, whereas gradient descent is typically used to minimize error or loss.

4. **What is the primary motivation behind PPO? How does it affect the policy objective function? Can you think of another function that might be suitable?**

Ans: The primary motivation behind PPO is to avoid large updates to the policy that can lead to performance collapse, ensuring more stable and reliable learning. PPO modifies the policy objective function to penalize changes to the policy that move too far from the previous policy, effectively limiting the size of the policy update at each step. This is typically achieved through a clipped objective function or a penalty on large changes in policy.

Another function that might be suitable for ensuring stable updates in reinforcement learning is Trust Region Policy Optimization, which also seeks to limit the step size of policy updates but does so through a more complex mechanism involving trust regions and second-order optimization methods.

5. **What are some advantages and potential drawbacks of using Proximal Policy Optimization in practical applications?**

Ans: Advantages of using Proximal Policy Optimization include,

Stability and Reliability: By limiting the size of policy updates, PPO tends to offer more stable and consistent improvement compared to other methods.

Simplicity: PPO is simpler to implement and tune compared to other methods like TRPO.

Efficiency: PPO can be more sample-efficient than vanilla policy gradient methods.

Disadvantages of using Proximal Policy Optimization include,

Hyperparameter Sensitivity: While simpler than some alternatives, PPO still requires careful tuning of hyperparameters to achieve optimal performance.

Computational Resources: Depending on the environment and the implementation, PPO can be computationally intensive, particularly in terms of memory and CPU/GPU time.

Suboptimality: The constraints on policy updates can lead to slower convergence or suboptimal policies if not properly managed.

6. **Do you have any questions?**

Ans: When I find the time, I am really interested in trying out the Doom agent training as it looks really interesting, especially the implementation of the deathmatch because it seems a bit more complicated.

Week 6 Tuesday Paper 2: Reinforcement Learning from Human Feedback

Supplemental: Deep Reinforcement Learning from Human Preference

Questions:

1. **Describe the action space, observation space, and reward function in the formulation of the fine-tuning task as a reinforcement learning problem in RLHF.**

Ans: From the article, the **action space** of this policy is all the tokens corresponding to the vocabulary of the language model (often on the order of 50k tokens) and the **observation space** is the distribution of possible input token sequences, which is also quite large given previous uses of RL (the dimension is approximately the size of vocabulary ^ length of the input token sequence). The **reward function** is a combination of the preference model and a constraint on policy shift.

2. **Explain the concept of reward model training in RLHF and how human preferences are integrated into the system.**

Ans: In RLHF, a reward model is trained to predict the human-provided reward based on the input (observation) and the model's output (action). This involves collecting a dataset of (input, output, reward) triples, where the rewards are provided by humans based on their preferences or evaluations of the outputs. The reward model learns to approximate these human judgments, allowing the reinforcement learning process to scale by providing estimated rewards for a vast number of input-output pairs without direct human evaluation.
Human preferences are integrated into the system through the collection of feedback that is used to train the reward model. This feedback can be gathered through various methods, such as pairwise comparisons(Elo system), direct ratings, or other forms of evaluation. The trained reward model then serves as a proxy for human judgment, guiding the language model towards generating text that aligns with human preferences.

3. **What metrics are generally used to rank the generated text from the model? Can you think of your own ranking metric?**

Ans: This feedback can be gathered through various methods, such as pairwise comparisons(Elo system), direct ratings, or other forms of evaluation. The trained reward model then serves as a proxy for human judgment, guiding the language model towards generating text that aligns with human preferences.

I am really biased towards the Elo system because I play a lot of games that make use of it to rank players.

4. **What is the purpose of the KL divergence term in the reward function, and how does it contribute to the optimization process?**

Ans: The KL divergence term penalizes the RL policy from moving substantially away from the initial pretrained model with each training batch, which can be useful to make sure the model outputs reasonably coherent text snippets. Without this penalty the optimization can start to generate text that is gibberish but fools the reward model to give a high reward. In practice, the KL divergence is approximated via sampling from both distributions

5. **What questions do you have?**

Ans: What exactly is the Elo system? I have always heard of the Elo system but I have never thought of learning more about it. When I saw the article mention the Elo system to compare two language models' performance, I thought I really have to google it within this week.