Optimisation In Reinforcement Learning

The following article contains the following:

1 - Commonly-Used Optimisation Algorithms

1. **A2C (Advantage Actor-Critic):**
    - Explanation: A reinforcement learning algorithm that combines elements of policy and value-based methods.
2. **PPO (Proximal Policy Optimization):**
    - Explanation: A policy optimization algorithm for reinforcement learning, focusing on stable and efficient training.
3. **DDPG (Deep Deterministic Policy Gradient):**
    - Explanation: An algorithm for learning continuous actions in reinforcement learning, combining deep neural networks with deterministic policies.
4. **SAC (Soft Actor-Critic):**
    - Explanation: A deep reinforcement learning algorithm that aims for optimal policy learning in continuous action spaces.
5. **TD3 (Twin Delayed DDPG):**

- Explanation: An improvement over DDPG, addressing overestimation issues in Q-learning through the use of multiple critics.

6. **HER (Hindsight Experience Replay):**
   - Explanation: A technique in reinforcement learning that replays and learns from experiences, even if the actual goal was not achieved.
7. **DQN (Deep Q Network):**
   - Explanation: A model-based reinforcement learning algorithm that combines Q-learning with deep neural networks.
8. **SB3 (Stable Baselines3):**
   - Explanation: A set of high-quality implementations of reinforcement learning algorithms based on OpenAI Baselines.
9. **QR-DQN (Quantile Regression DQN):**
   - Explanation: A variation of DQN that estimates the distribution of the return, providing a more comprehensive understanding of uncertainty.
10. **TQC (Twin Q-Consistency):**
    - Explanation: A deep reinforcement learning algorithm that combines the benefits of both actor-critic and Q-learning methods.

2 - Explanation of PPO

Overview:
Proximal Policy Optimization (PPO) is a reinforcement learning algorithm that belongs to the family of policy optimization methods. It was introduced by OpenAI as a solution to address some of the issues associated with traditional policy gradient methods, such as high variance and instability during training. PPO is designed to provide a more stable and sample-efficient approach for training policy-based models.

Key Concepts:

1. **Objective Function:**
   - PPO aims to optimize the policy by maximizing the expected cumulative reward. The objective function involves finding a policy that maximizes

the expected return over a set of trajectories.

2. **Trust Region Optimization:**
    - PPO introduces a "trust region" to limit the size of policy updates during optimization. This prevents the policy from changing too rapidly, ensuring more stable training.
3. **Surrogate Objective Function:**
    - PPO uses a surrogate objective function that constrains the policy update. The objective is to maximize the probability of actions that have higher returns while penalizing actions that deviate too far from the current policy.
4. **Clipping:**
    - One of the distinctive features of PPO is the use of a clipping mechanism. The surrogate objective is clipped to a certain range, preventing large policy updates. This helps maintain stability during training.

Training Procedure:

1. **Collect Trajectories:**
    - Generate trajectories by executing the current policy in the environment. Collect observations, actions, and rewards over multiple episodes.
2. **Compute Surrogate Objective:**
    - Calculate the ratio of the probability of the new policy to the old policy for each action taken in the trajectory. Formulate the surrogate objective function using this ratio.
3. **Clipping:**
    - Apply the clipping mechanism to the surrogate objective function. This constrains the policy update to a specified range, preventing overly aggressive updates.
4. **Optimization:**
    - Perform optimization to update the policy parameters using techniques such as stochastic gradient ascent or other optimization algorithms.
5. **Repeat:**

- Iterate through multiple epochs, collecting new trajectories and optimizing the policy parameters, until convergence or a predefined number of iterations.

Advantages of PPO:

1. **Stability:**
   - PPO is designed to provide more stable training by constraining policy updates and preventing large deviations.
2. **Sample Efficiency:**
   - The algorithm is relatively sample-efficient compared to some other policy gradient methods, allowing faster learning with fewer interactions with the environment.
3. **Versatility:**
   - PPO can be applied to a variety of reinforcement learning problems, including both discrete and continuous action spaces.

Use Cases:
PPO has been successfully applied to a wide range of tasks, including robotic control, game playing, and simulated environments. Its stability and ease of implementation make it a popular choice for researchers and practitioners in the field of reinforcement learning.

Implementation Note:
PPO is often implemented using deep neural networks to represent the policy. The algorithm's success has led to its adoption in various reinforcement learning applications and benchmarks.

3 - Surrogate Objective Function

The Proximal Policy Optimization (PPO) algorithm involves a surrogate objective function that is optimized to update the policy. The objective is to maximize the expected cumulative reward while constraining the policy update to a certain range. The key formula for PPO is as follows:

Let's denote the old policy's probability distribution for an action aa given an observation ss as πold(a ∣ s)$\pi_{old}(a \mid s)$, and the new policy's probability distribution as πnew(a ∣ s)$\pi_{new}(a \mid s)$.

The ratio of the new policy to the old policy is defined as:

$$Ratio(\theta) = \pi new(a \mid s) / \pi old(a \mid s)$$

where:

- θ represents the parameters of the policy (neural network weights, for example).
- A is the action.
- S is the observation.

The surrogate objective function is given by:

$$L(\theta) = E[min(Ratio(\theta) \cdot A, clip(Ratio(\theta), 1 - \epsilon, 1 + \epsilon) \cdot A)$$

where:

- A is the advantage function, representing how much better an action is compared to the average action in a given state.
- ε is a hyperparameter, typically a small constant, that determines the range of the policy update.

The objective is to maximise L(θ) with respect to the policy parameters θ. This is typically done using an optimization algorithm such as stochastic gradient ascent.

In summary, the PPO algorithm involves calculating the ratio of probabilities, constructing a surrogate objective function, and optimizing it while applying a clipping mechanism to ensure a stable policy update. This helps prevent overly aggressive changes to the policy during training.

4 - Types of Policy Optimisation Methods:

1. **Vanilla Policy Gradient (VPG):**
    - Basic policy gradient method that directly optimizes the expected return by adjusting the policy parameters.
2. **Trust Region Policy Optimization (TRPO):**
    - Introduces a constraint on the policy update to ensure that the new policy does not deviate too far from the old policy.

3. **Proximal Policy Optimization (PPO):**
    - A family of policy optimization algorithms that use a clipped surrogate objective to prevent large policy updates.
4. **Actor-Critic Methods:**
    - Combines policy optimization with value estimation. The actor (policy) is updated based on policy gradient methods, and the critic (value function) is updated to reduce the difference between predicted and actual returns.
5. **Deep Deterministic Policy Gradient (DDPG):**
    - Extends policy gradient methods to continuous action spaces by using a deterministic policy.
6. **Soft Actor-Critic (SAC):**
    - A soft actor-critic algorithm introduces an entropy term in the objective function, encouraging the policy to be stochastic and more exploratory.
7. **Twin Delayed DDPG (TD3):**
    - An improvement over DDPG that addresses overestimation bias by using multiple critics.
8. **Twin Q-Consistency (TQC):**
    - A policy optimization algorithm that combines elements of actor-critic and twin Q-learning methods.

5 - Types of Value Optimisation Methods:

1. **Q-Learning:**
    - Classic temporal difference learning algorithm that estimates the state-action values (Q-values) and updates the Q-function iteratively.
2. **Deep Q Network (DQN):**
    - Combines Q-learning with deep neural networks to handle high-dimensional state spaces.
3. **Double Q-Learning:**
    - Addresses overestimation bias in Q-learning by maintaining two separate Q-value estimates.

4. **Dueling DQN:**
    - Splits the Q-network into two streams to separately estimate the value of being in a state and the advantage of taking different actions.
5. **Quantile Regression DQN (QR-DQN):**
    - Estimates the distribution of returns, providing a more comprehensive understanding of uncertainty compared to point estimates.
6. **Twin Delayed DDPG (TD3):**
    - Not only an actor-critic method but also a value optimization method, as it involves updating the Q-function (value function).

Hybrid Methods:

1. **Actor-Critic Methods:**
    - These methods inherently involve both policy optimization (actor) and value optimization (critic).
2. **Proximal Policy Optimization (PPO):**
    - While primarily a policy optimization method, it also involves estimating advantages and updating a value function.
3. **Soft Actor-Critic (SAC):**
    - Combines value optimization with policy optimization, using both a value function and an entropy term in the objective.

These methods offer a range of approaches to address the challenges in reinforcement learning, and the choice of method often depends on the specific characteristics of the problem at hand.

6 - Policy Optimisation and Value Optimisation in Reinforcement Learning:

1. **Definition:**

- **Policy Optimization:** Policy optimization methods focus on directly adjusting the policy to maximize the expected cumulative reward. The policy defines

the agent's behavior, specifying the probability distribution over actions given a particular state.

- **Value Optimization:** Value optimization methods aim to estimate the value of states or state-action pairs, representing the expected cumulative reward under a given policy. These methods use value functions to guide the agent's decision-making.

2. Objective:

- **Policy Optimization:** The primary objective is to find a policy that maximizes the expected return. This is typically done by adjusting the parameters of the policy in the direction that increases the likelihood of actions leading to higher rewards.
- **Value Optimization:** The goal is to estimate the expected cumulative reward associated with being in a particular state or taking a specific action in a state. Value functions provide a measure of the long-term desirability of different states or state-action pairs.

3. Representation:

- **Policy Optimization:** The policy is directly parameterized and adjusted. This can be represented as a neural network, a set of parameters for a probability distribution, or other functional forms.
- **Value Optimization:** Value functions are used to represent the expected return. This includes state-value functions (V-function) for policies and action-value functions (Q-function) for state-action pairs.

4. Exploration vs. Exploitation:

- **Policy Optimization:** Focuses on exploring the action space to find actions that lead to higher rewards. The policy is adjusted to increase the probability of selecting actions with higher expected returns.
- **Value Optimization:** Balances exploration and exploitation by estimating the value of different actions or states. The agent may choose actions with the highest estimated value.

5. Actor-Critic Methods:

- **Policy Optimization:** Often involves actor-critic architectures where the actor (policy) is updated using policy gradient methods, and the critic (value function) is updated to reduce the difference between predicted and actual returns.

- **Value Optimization:** Q-learning and other value-based methods estimate value functions directly, updating them to improve the accuracy of value predictions.

6. Uncertainty:

- **Policy Optimization:** Typically focuses on finding a deterministic or stochastic policy that maximizes expected returns. May involve introducing exploration through stochastic policies.
- **Value Optimization:** Can provide estimates of uncertainty by representing distributions over returns (e.g., in QR-DQN) or by using methods like bootstrapping in ensemble Q-learning.

7. Stability and Convergence:

- **Policy Optimization:** The stability of training can be a challenge due to the high variance of policy gradient estimates. Algorithms like PPO and TRPO introduce mechanisms to enhance stability.
- **Value Optimization:** Methods like DQN and its variants often benefit from experience replay and target networks to improve stability during training.

In practice, some algorithms, like actor-critic methods, combine aspects of both policy and value optimization. The choice between these approaches often depends on the nature of the problem, the structure of the state and action spaces, and the desired properties of the learned policy.

7 - Stochastic Gradient Ascent (SGA) and Stochastic Gradient Descent (SGD):

Stochastic Gradient Descent (SGD):

1. **Objective:**
    - **Minimization Problem:** SGD is commonly used for solving optimization problems, particularly in the context of machine learning. It is employed to minimize a cost or loss function by adjusting the parameters of a model.
2. **Batch Gradient Descent vs. Stochastic Gradient Descent:**

- **Batch Gradient Descent (BGD):** In BGD, the entire dataset is used to compute the gradient of the cost function, and the parameters are updated based on this average gradient.
- **Stochastic Gradient Descent (SGD):** In SGD, only a single randomly chosen data point (or a small batch of data points) is used to compute the gradient and update the parameters. This introduces randomness and efficiency.

3. **Update Rule:**

   The update rule for the parameters θ in SGD is given by:

$$\theta t + 1 = \theta t - \eta \nabla J(\theta t; xi, yi)$$

- Where $J(\theta t; xi, yi)$ is the cost function for the ith data point, and η is the learning rate.

4. **Pros and Cons:**
   - **Pros:** SGD is computationally efficient, particularly when dealing with large datasets, as it processes only one or a few data points at a time.
   - **Cons:** The randomness introduced by using only a subset of data points at each iteration can result in noisy updates. This noise can cause the optimization process to exhibit more oscillations, but it can also help escape local minima.

Stochastic Gradient Ascent (SGA):

1. **Objective:**
   - **Maximization Problem:** SGA is used for maximizing an objective function, typically in the context of reinforcement learning or certain probabilistic models where the goal is to find parameter values that maximize the likelihood of observed data.
2. **Update Rule:**

   The update rule for the parameters θ in SGA is given by:

$$\theta t + 1 = \theta t + \eta \nabla J(\theta t; xi, yi)$$

- Where $J(\theta t; xi, yi)$ is the objective function for the ith data point, and η is the learning rate.

3. **Direction of Update:**
    - Unlike SGD, where the goal is to minimize a cost or loss function, SGA is used when the objective is to maximize a likelihood or utility function. Therefore, the update is performed in the direction of the gradient ascent.
4. **Pros and Cons:**
    - **Pros:** Similar to SGD, SGA is computationally efficient, especially when dealing with large datasets. It is well-suited for problems where the goal is to maximize a likelihood or utility.

Note:
In practice, the terms "gradient descent" and "gradient ascent" are often used interchangeably, as maximizing an objective is equivalent to minimizing its negative. Therefore, the update rules for SGD and SGA are conceptually similar, with the primary difference lying in the objective they address (minimization vs. maximization).