TRPO

Trust Region Policy Optimization

Advantage

- Advantage (A): $A(s, a) = Q(s, a) - V(s)$
- Intuitive: How good an action is compared to the average action for a specific state.

- TRPO is an on-policy algorithm.
- TRPO updates policies by taking the largest step possible to improve performance
- It does this while satisfying a special constraint on how close the new and old policies are allowed to be.
- The constraint is expressed in terms of KL-Divergence, a measure of (something like, but not exactly) distance between probability distributions.

Trust Region

Well known method in optimization

You have a function to optimize and you have a local approximation of this function - which works in a boundary called the trust region

But it gets really inaccurate if you go away from your starting point

Trust Region - Space where we trust our local approximation

Making sure that any action that is performed is well within the region that we Trust to be a good action.

- This is different from normal policy gradient, which keeps new and old policies close in parameter space.
- But even seemingly small differences in parameter space can have very large differences in performance—so a single bad step can collapse the policy performance.
- This makes it dangerous to use large step sizes with vanilla policy gradients, thus hurting its sample efficiency.
- TRPO nicely avoids this kind of collapse, and tends to improve performance.

```python
import pandas as pd
import yfinance as yf


class YahooDownloader:
    """Provides methods for retrieving daily stock data from
    Yahoo Finance API
    Attributes
    ----------
        start_date : str
            start date of the data (modified from config.py)
        end_date : str
            end date of the data (modified from config.py)
        ticker_list : list
            a list of stock tickers (modified from config.py)
    Methods
    -------
    fetch_data()
        Fetches data from yahoo API
    """
```

Compared to A2C

- We can use a modified version of the loss function given below:

$$\mathrm{L}_{\theta_{old}} = \mathbb{E}_t \left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{old}}(a_t \mid s_t)} A_t \right]$$

- Instead of the loss function in A2C:

$$L(\theta) = \mathbb{E}_t \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) A_t \right]$$

Let π_θ denote a policy with parameters θ. The theoretical TRPO update is:

$$\begin{aligned}\theta_{k+1} = \arg\max_{\theta} \; & \mathcal{L}(\theta_k, \theta) \\ \text{s.t. } & \bar{D}_{KL}(\theta || \theta_k) \leq \delta\end{aligned}$$

where $\mathcal{L}(\theta_k, \theta)$ is the *surrogate advantage*, a measure of how policy π_θ performs relative to the old policy π_{θ_k} using data from the old policy:

$$\mathcal{L}(\theta_k, \theta) = \mathop{\mathrm{E}}_{s,a \sim \pi_{\theta_k}} \left[\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s,a) \right],$$

and $\bar{D}_{KL}(\theta || \theta_k)$ is an average KL-divergence between policies across states visited by the old policy:

$$\bar{D}_{KL}(\theta || \theta_k) = \mathop{\mathrm{E}}_{s \sim \pi_{\theta_k}} \left[D_{KL}\left(\pi_\theta(\cdot|s) || \pi_{\theta_k}(\cdot|s) \right) \right].$$