Trust Region Policy Optimization Demo

Rishikesh Vaishnav

July 11, 2018

Code

- The code for this project is available at: .

Implementation Details

Single Path Executable Pseudocode:

- Initialize policy parameter θ.
- Iterate until convergence:
  - Initialize/clear list S of $\{\frac{G_\theta(s,a)}{\pi_\theta(s,a)}, s, a\}$.
  - Generate N_τ trajectories $\{\tau\}$.
  - For each trajectory $\tau \in \{\tau\}$:
    - For each $\{s, a\} \in \tau$:
      - Calculate discounted return $G_\theta(s, a)$ from this time to end of episode.
      - Calculate $\pi_\theta(s, a)$ at this (s, a).
      - Store $\{\frac{G_\theta(s,a)}{\pi_\theta(s,a)}, s, a\}$ in S.
  - Use constraint optimizer to yield θ' by solving the problem:
    - Objective (to maximize): objective(S, θ).
    - Constraint: constraint(S, θ).
  - $\theta = \theta'$.

objective(S, θ') Pseudocode:

- Initialize $L = 0$.
- For $\{\frac{G_\theta(s,a)}{\pi_\theta(s,a)}, s, a\} \in S$:
  - Add $\pi_{\theta'}(s, a)\frac{G_\theta(s,a)}{\pi_\theta(s,a)}$ to L.
- Return L.

constraint(S, θ') Pseudocode:

- Initialize $D = 0$.
- For $s \in S$:

- Add $D_{KL}(\pi_\theta(\cdot|s)||\pi_{\theta'}(\cdot|s))$ to D.

- Return $\frac{D}{|S|}$.

Parameter Settings:

- $N_\tau = 4$ (adjusted to balance between empirical runtime and performance)
- $\gamma = 1$ (adjusted to maximize empirical performance)
- $\delta = 0.01$ (following Schulman et. al.)

Policy Function Encoding:

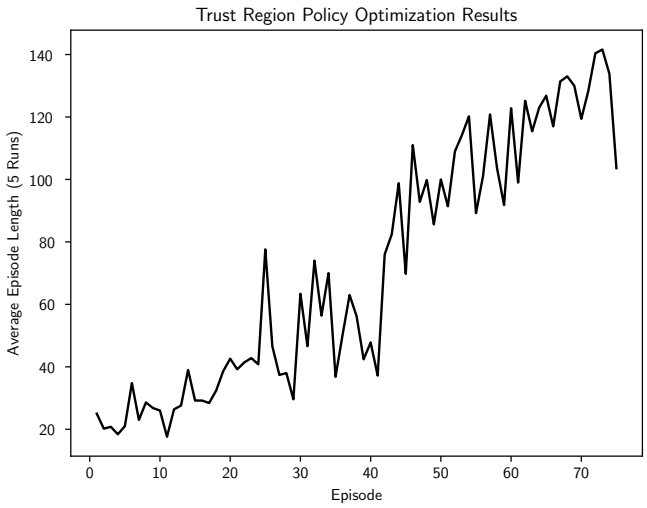
- Each state-action pair is converted to the feature vector $x(s, a)$. Letting S_{obs} and S_{act} be the size of the observation and action spaces, respectively, the size of the vector is $S_{obs} \times S_{act}$, where all features are 0 except for the S_{obs} features starting at index $S_{obs} \times a$, which are set to the environment's parameterization of s.
  - In this case, $S_{obs} = 4$ and $S_{act} = 2$.
- The policy function $\pi(a|s, \theta)$ performs the softmax on a parameterized linear mapping of feature vectors:

$$\pi(a|s, \theta) = \frac{e^{\theta^T x(s,a)}}{\sum_b e^{\theta^T x(s,b)}}$$

Constraint Optimization Method:

- I used scipy's optimize.minimize function, with the "trust-constr" method. All gradients and hessians were automatically calculated. A tolerance of 1×10^{-2} and a maximum iteration limit of 100 were used.

Results



- The results can be summarized as follows:
  -