Reinforcement Learning
Tutorial for Lecture 8

Georg Martius

Distributed Intelligence / Autonomous Learning Group, Uni Tübingen, Germany

December 3, 2024

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Advantage Estimation

In the lecture we had a Quiz on how to do Advantage estimation.
Why is the following answer wrong?

- train net $\hat{v}$ using TD and take the current reward:
  $\hat{A}(s, a) = r(s, a) - \hat{v}(s)$

Advantage Estimation

In the lecture we had a Quiz on how to do Advantage estimation.
Why is the following answer wrong?

- train net $\hat{v}$ using TD and take the current reward:
  $\hat{A}(s, a) = r(s, a) - \hat{v}(s)$

- r is the momentary reward, and does not contain the value of the state and action

Write down the correct estimation of Advantage:

Advantage Estimation

In the lecture we had a Quiz on how to do Advantage estimation.
Why is the following answer wrong?

- train net $\hat{v}$ using TD and take the current reward:
  $\hat{A}(s, a) = r(s, a) - \hat{v}(s)$

- r is the momentary reward, and does not contain the value of the state and action

Write down the correct estimation of Advantage:

$$\hat{A}(s, a) = q(s, a) - \hat{v}(s)$$

Trust Region

What is the idea behind trust region?

Trust Region

What is the idea behind trust region?

- optimize objective function but do not move too far
- typically phrased as a contraint problem:

$$\begin{array}{ll}\max_{\Delta\theta} & J(\theta + \Delta\theta) \\ \text{s.t.} & \|\Delta\theta\| \leq \delta\end{array}$$

where δ is the trust region parameters

Trust Region

What is the idea behind trust region?

- optimize objective function but do not move too far
- typically phrased as a contraint problem:

$$\begin{array}{ll} \max_{\Delta\theta} & J(\theta + \Delta\theta) \\ \text{s.t.} & \|\Delta\theta\| \leq \delta \end{array}$$

where δ is the trust region parameters

Why is it needed / a good idea for policy gradient?

Trust Region

What is the idea behind trust region?

- optimize objective function but do not move too far
- typically phrased as a contraint problem:

$$\begin{array}{ll} \max_{\Delta\theta} & J(\theta + \Delta\theta) \\ \text{s.t.} & \|\Delta\theta\| \leq \delta \end{array}$$

  where δ is the trust region parameters

Why is it needed / a good idea for policy gradient?

- We make a very local estimate of our gradient
- The gradient estimate is noisy (noise in environment and policy)
- We cannot guarantee that a big step actually leads to an improvement

TRPO – Computing the Trust Region

How is the difference between new and old policy measured?

TRPO – Computing the Trust Region

How is the difference between new and old policy measured?

- Difference in action probabilities for states visited during a rollout:

$$\mathrm{dist}(\theta, \theta^{\mathrm{old}}) = \mathbb{E}_{s \sim \pi_{\theta^{\mathrm{old}}}} D_{\mathrm{KL}}(\pi_\theta \| \pi_{\theta^{\mathrm{old}}})$$

How is the difference between new and old policy measured?

- Difference in action probabilities for states visited during a rollout:

$$\mathrm{dist}(\theta, \theta^{\mathrm{old}}) = \mathbb{E}_{s \sim \pi_{\theta^{\mathrm{old}}}} D_{\mathrm{KL}}(\pi_\theta \| \pi_{\theta^{\mathrm{old}}})$$

Why not measuring the distance in changes to the network parameter?

TRPO – Computing the Trust Region

How is the difference between new and old policy measured?

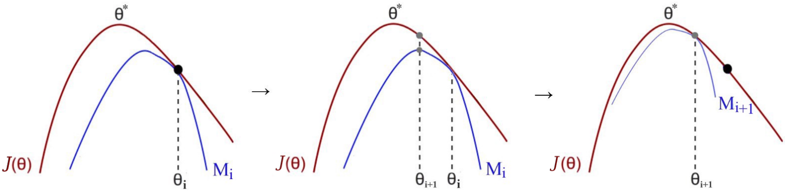
- Difference in action probabilities for states visited during a rollout:

$$\operatorname{dist}(\theta, \theta^{\text{old}}) = \mathbb{E}_{s\sim\pi_{\theta^{\text{old}}}} D_{\text{KL}}(\pi_\theta \| \pi_{\theta^{\text{old}}})$$

Why not measuring the distance in changes to the network parameter?

- a change in network weights can have drastic effects on the output / actions

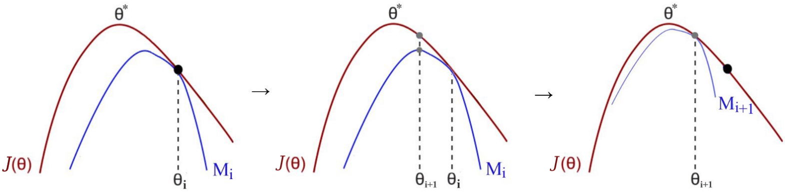
TRPO uses the Minorization Maximization technique:



Why can we evaluate the surrate objective M efficiently?
$M(\theta') = \nabla_\theta J(\theta)^\top \cdot (\theta - \theta') - b \cdot (\theta - \theta')^2$

TRPO

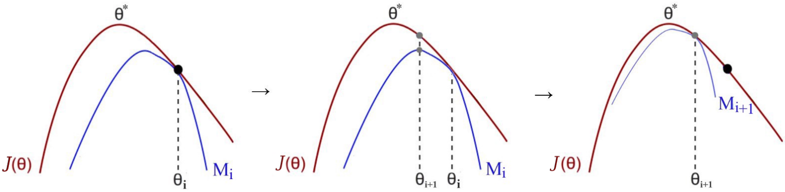
TRPO uses the Minorization Maximization technique:



Why can we evaluate the surrate objective M efficiently?
$M(\theta') = \nabla_\theta J(\theta)^\top \cdot (\theta - \theta') - b \cdot (\theta - \theta')^2$

- Because J is approximated by a line! The gradient / tangent is considered at point θ only

TRPO uses the Minorization Maximization technique:

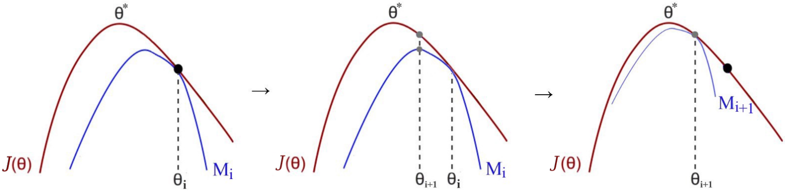


Why can we evaluate the surrate objective M efficiently?
$M(\theta') = \nabla_\theta J(\theta)^\top \cdot (\theta - \theta') - b \cdot (\theta - \theta')^2$

- Because J is approximated by a line! The gradient / tangent is considered at point θ only

So when is M actually a lower bound if J?

TRPO uses the Minorization Maximization technique:



Why can we evaluate the surrate objective M efficiently?
$M(\theta') = \nabla_\theta J(\theta)^\top \cdot (\theta - \theta') - b \cdot (\theta - \theta')^2$

- Because J is approximated by a line! The gradient / tangent is considered at point θ only

So when is M actually a lower bound if J?

- We perform a Tailor expansion if J: if b is larger than the |second derivative| of J (it is negative as we are considering concav functions)

PPO

Let's assemble the PPO algorithm!
What do we need?

PPO

Let's assemble the PPO algorithm!
What do we need?

- Parametrized policy: π_θ
- Parametrized value function: $\hat{v}(s, \mathbf{w})$
- Importance weights: $\rho_t(\theta) = \frac{\pi_\theta(s_t, a_t)}{\pi_{\theta^{\text{old}}}(s_t, a_t)}$
- Monte Carlo Returns: $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} r_k$
- Loss function for value function: $L^{\text{VF}}(\mathbf{w}) = (\hat{v}(s_t, \mathbf{w}) - G_t)^2$
- Loss function for policy:
  $L^{\text{CLIP}}(\theta) = \underset{\pi_{\theta^{\text{old}}}}{\mathbb{E}} \left[\min\left(\rho_t(\theta) A_t, \text{clip}\left(\rho_t(\theta), 1-\epsilon, 1+\epsilon\right) A_t\right)\right]$
- Entropy regularizer: $\mathcal{H}[\pi_\theta] = -\sum_a \pi_\theta(a \mid s) \log(\pi_\theta(a \mid s))$

PPO

$$L^{\text{CLIP}}(\theta) = \mathop{\mathbb{E}}_{\pi_{\theta \text{old}}} \left[\min\left(\rho_t(\theta) A_t, \text{clip}\left(\rho_t(\theta), 1-\epsilon, 1+\epsilon\right) A_t\right)\right]$$

Let's assemble the PPO algorithm!

PPO

Initialise $\theta, \mathbf{w}$ arbitrarily
for each iteration:

1. $\theta_{\text{old}} \leftarrow \theta$
2. $\mathcal{D} = \emptyset$
3. for $e = 1$ to E episodes:
   3.1 $\tau^e \leftarrow \{s_1^e, a_1^e, r_2^e, \ldots, s_{T-1}^e, a_{T-1}^e, r_T^e\} \sim \pi_\theta$
   3.2 for $t = 1$ to $T-1$ timesteps:

PPO

$$L^{\text{CLIP}}(\theta) = \mathop{\mathbb{E}}_{\pi_{\theta\text{old}}} \left[\min\left(\rho_t(\theta) A_t, \text{clip}\left(\rho_t(\theta), 1-\epsilon, 1+\epsilon\right) A_t\right)\right]$$

Let's assemble the PPO algorithm!

PPO

Initialise $\theta, \mathbf{w}$ arbitrarily
for each iteration:

1. $\theta_{\text{old}} \leftarrow \theta$
2. $\mathcal{D} = \emptyset$
3. for $e = 1$ to E episodes:
   - **3.1** $\tau^e \leftarrow \{s_1^e, a_1^e, r_2^e, \ldots, s_{T-1}^e, a_{T-1}^e, r_T^e\} \sim \pi_\theta$
   - **3.2** for $t = 1$ to $T-1$ timesteps:
     - **3.2.1** $G_t^e \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} r_k^e$

PPO

$$L^{\text{CLIP}}(\theta) = \mathop{\mathbb{E}}_{\pi_{\theta\text{old}}} \left[\min\left(\rho_t(\theta) A_t, \text{clip}\left(\rho_t(\theta), 1-\epsilon, 1+\epsilon\right) A_t\right)\right]$$

Let's assemble the PPO algorithm!

PPO

Initialise $\theta, \mathbf{w}$ arbitrarily
for each iteration:

1. $\theta_{\text{old}} \leftarrow \theta$
2. $\mathcal{D} = \emptyset$
3. for $e = 1$ to E episodes:
   - **3.1** $\tau^e \leftarrow \{s_1^e, a_1^e, r_2^e, \ldots, s_{T-1}^e, a_{T-1}^e, r_T^e\} \sim \pi_\theta$
   - **3.2** for $t = 1$ to $T-1$ timesteps:
     - **3.2.1** $G_t^e \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} r_k^e$
     - **3.2.2** $A_t^e \leftarrow G_t^e - \hat{v}(s_t^e, \mathbf{w})$
     - **3.2.3** $\mathcal{D} = \mathcal{D} \cup (s_t^e, a_t^e, G_t^e)$
4. $\theta \leftarrow \theta + \alpha \nabla_\theta \mathbb{E}_{(s,a,G)\sim\mathcal{D}}\left[L^{\text{CLIP}}(\theta) + c_2 \mathcal{H}[\pi_\theta]\right]$
5. $\mathbf{w} \leftarrow \mathbf{w} + \alpha \nabla_{\mathbf{w}} \mathbb{E}_{(s,G)\sim\mathcal{D}}\left[(G - \hat{v}(s, \mathbf{w}))^2\right]$ value function loss: L^{VF}

return $\theta, \mathbf{w}$