Policy gradient

Policy gradient

$$\frac{1}{N} \sum_{\tau \sim P_{\pi,T}} R(\tau) \nabla \log P_{\pi,T}(\tau)$$

- REINFORCE on steroids
  - lower variance
    - baseline
  - off-policy
    - reuse rollouts

Vanilla policy gradient algorithm

$$\frac{1}{N}\sum_{\tau\sim P_{\pi,T}} R(\tau)\,\nabla\log P_{\pi,T}(\tau)$$

- For i iterations
  - Collect rollouts
  - Estimate the sample gradient
  - Take a gradient step

Variance of REINFORCE

$$\frac{1}{N}\sum_{\tau \sim P_{\pi,T}} R(\tau)\,\nabla \log P_{\pi,T}(\tau)$$

- What happens if all rewards are positive?
  - Only learn to do "more" things in τ
  - SGD zig-zags
- RL worst best of we have positive and negative returns

Baselines

- Gradient for constant return is zero

$$\frac{1}{N} \sum_{\tau \sim P_{\pi,T}} R(\tau) \nabla \log P_{\pi,T}(\tau)$$

  - $\mathbb{E}_{\tau \sim P_{\pi,T}}[b \nabla \log P_{\pi,T}(\tau)] = 0$
- Reduces variance
  - Positive and negative returns
- Unbiased gradient estimate

On- vs off-policy

$$\frac{1}{N}\sum_{\tau\sim P_{\pi,T}} R(\tau)\,\nabla\log P_{\pi,T}(\tau)$$

- REINFORCE is on-policy
  - Trajectories (rollouts) need to come from current policy
  - No reuse of trajectories between gradient update

Off-policy

$$\frac{1}{N}\sum_{\tau\sim Q}\frac{P_{\pi,T}(\tau)}{Q(\tau)}R(\tau)\,\nabla\log P_{\pi,T}(\tau)$$

- Importance sampling
  - Many variants

Policy gradient algorithm

- For i iterations
  - Collect rollouts
    - Add to **replay buffer**
  - Update baseline network
  - For j batches
    - Estimate the sample gradient on **replay buffer**
    - Take a gradient step

Policy gradient

- REINFORCE with many tricks

$$\frac{1}{N}\sum_{\tau\sim P_{\pi,T}} R(\tau)\,\nabla\log P_{\pi,T}(\tau)$$

  - Not very sample efficient
  - Gradient estimate by sampling from an exponential trajectory space