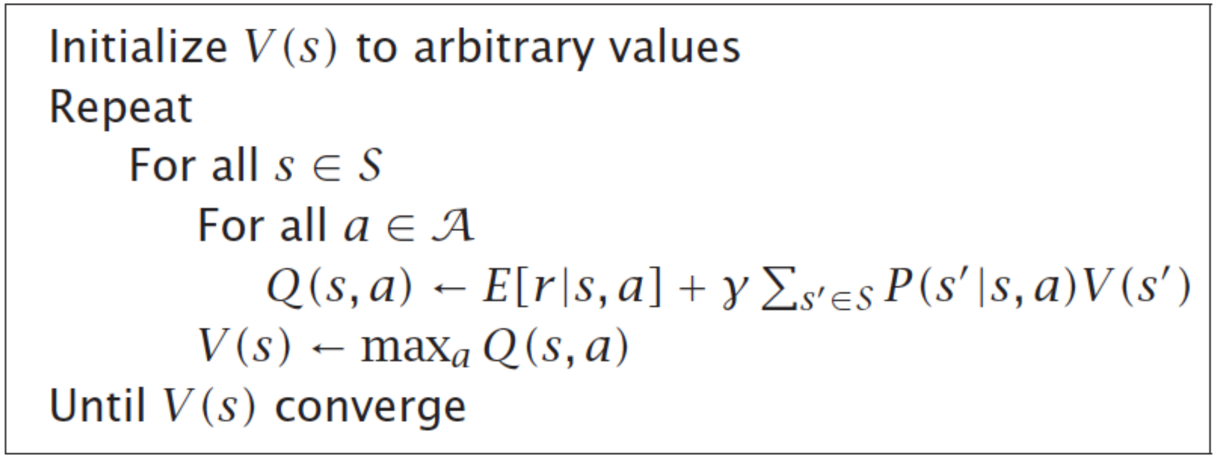
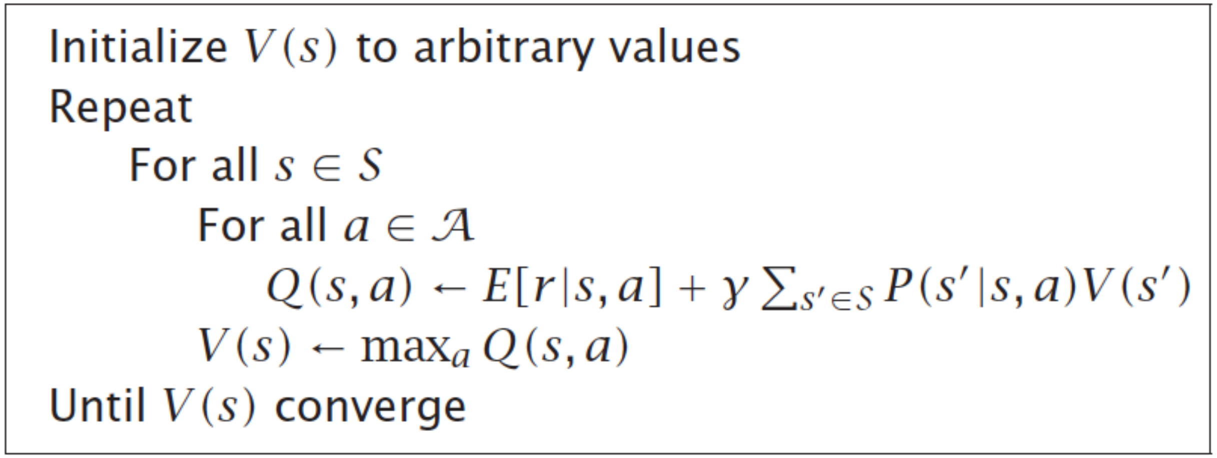
# Trust region policy optimization (TRPO)

# Value Iteration



```
Initialize $V(s)$ to arbitrary values
Repeat
    For all $s \in S$
        For all $a \in \mathcal{A}$
            $Q(s,a) \leftarrow E[r|s,a] + \gamma \sum_{s' \in S} P(s'|s,a)V(s')$
        $V(s) \leftarrow \max_a Q(s,a)$
Until $V(s)$ converge
```

# Value Iteration

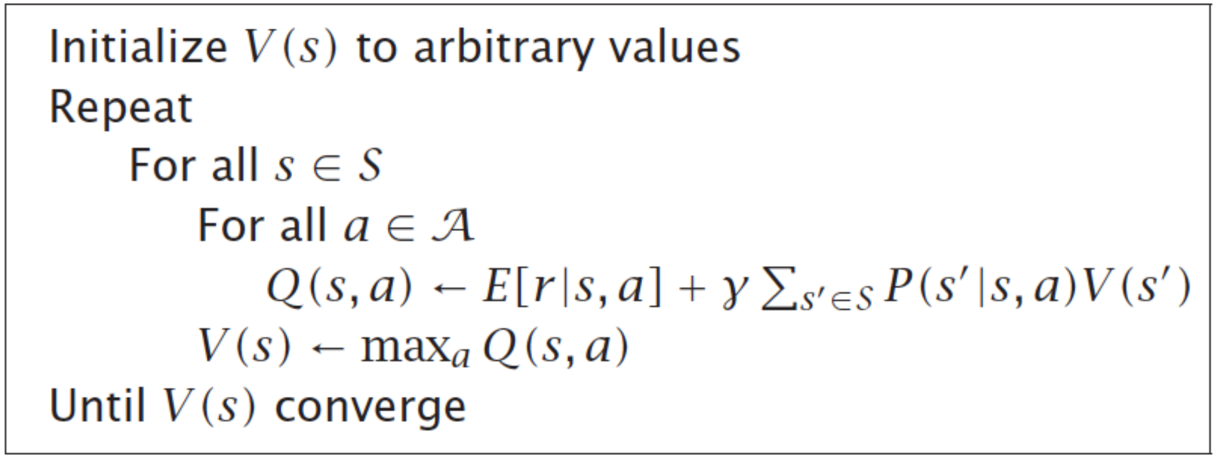


```
Initialize $V(s)$ to arbitrary values
Repeat
    For all $s \in S$
        For all $a \in \mathcal{A}$
            $Q(s,a) \leftarrow E[r|s,a] + \gamma \sum_{s' \in S} P(s'|s,a)V(s')$
        $V(s) \leftarrow \max_a Q(s,a)$
Until $V(s)$ converge
```

- This is what we similar to what Q-Learning does, the main difference being that we we might not know the actual expected reward and instead explore the world and use discounted rewards to model our value function.

# Value Iteration

Model-based

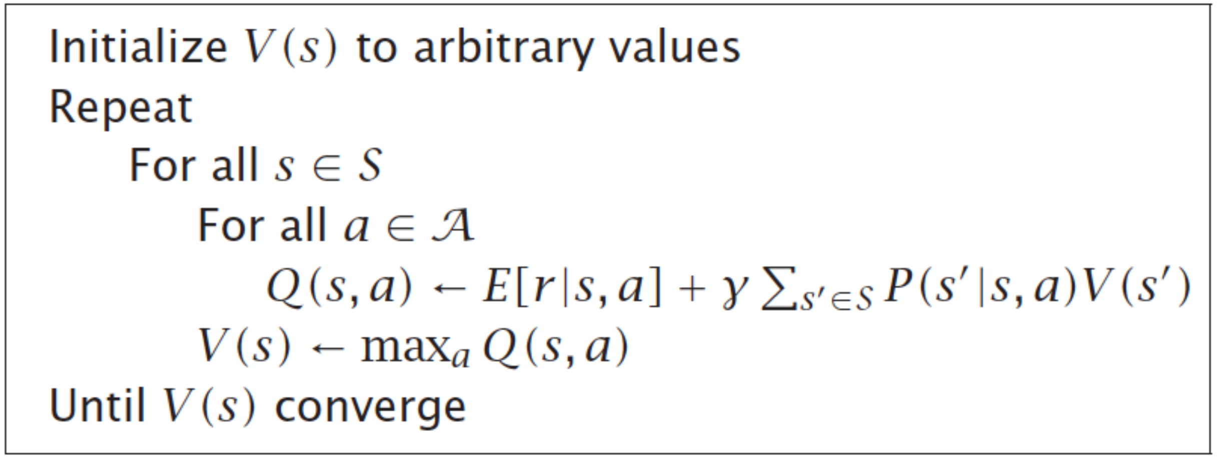


```
Initialize $V(s)$ to arbitrary values
Repeat
    For all $s \in S$
        For all $a \in \mathcal{A}$
            $Q(s,a) \leftarrow E[r|s,a] + \gamma \sum_{s' \in S} P(s'|s,a)V(s')$
        $V(s) \leftarrow \max_a Q(s,a)$
Until $V(s)$ converge
```

odel-free

- This is what we similar to what Q-Learning does, the main difference being that we we might not know the actual expected reward and instead explore the world and use discounted rewards to model our value function.

# Value Iteration



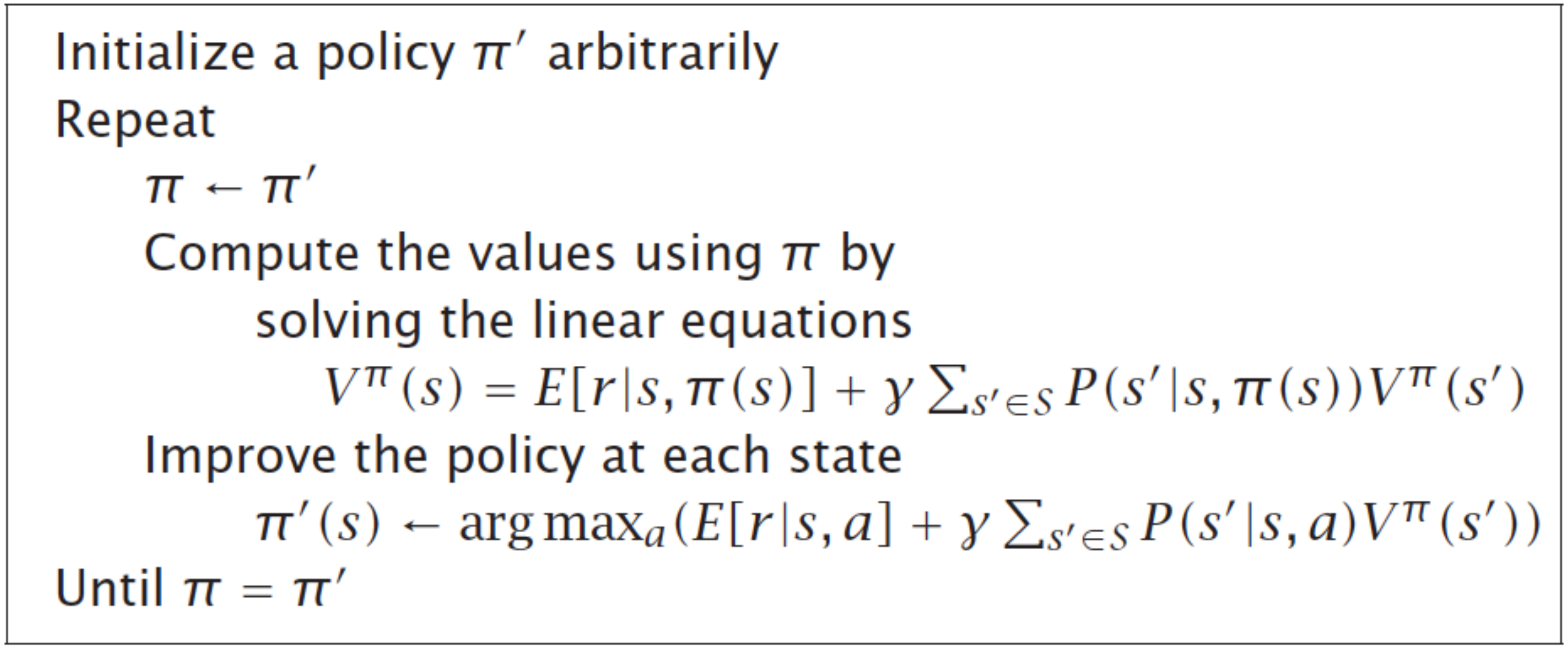
```
Initialize $V(s)$ to arbitrary values
Repeat
    For all $s \in S$
        For all $a \in \mathcal{A}$
            $Q(s,a) \leftarrow E[r|s,a] + \gamma \sum_{s' \in S} P(s'|s,a)V(s')$
        $V(s) \leftarrow \max_a Q(s,a)$
Until $V(s)$ converge
```

- This is what we similar to what Q-Learning does, the main difference being that we we might not know the actual expected reward and instead explore the world and use discounted rewards to model our value function.
- Once we have Q(s,a), we can find optimal policy π* using:

$$\pi^*(s) = arg\max_a Q(s,a)$$

# Policy Iteration

- We can directly optimize in the policy space.

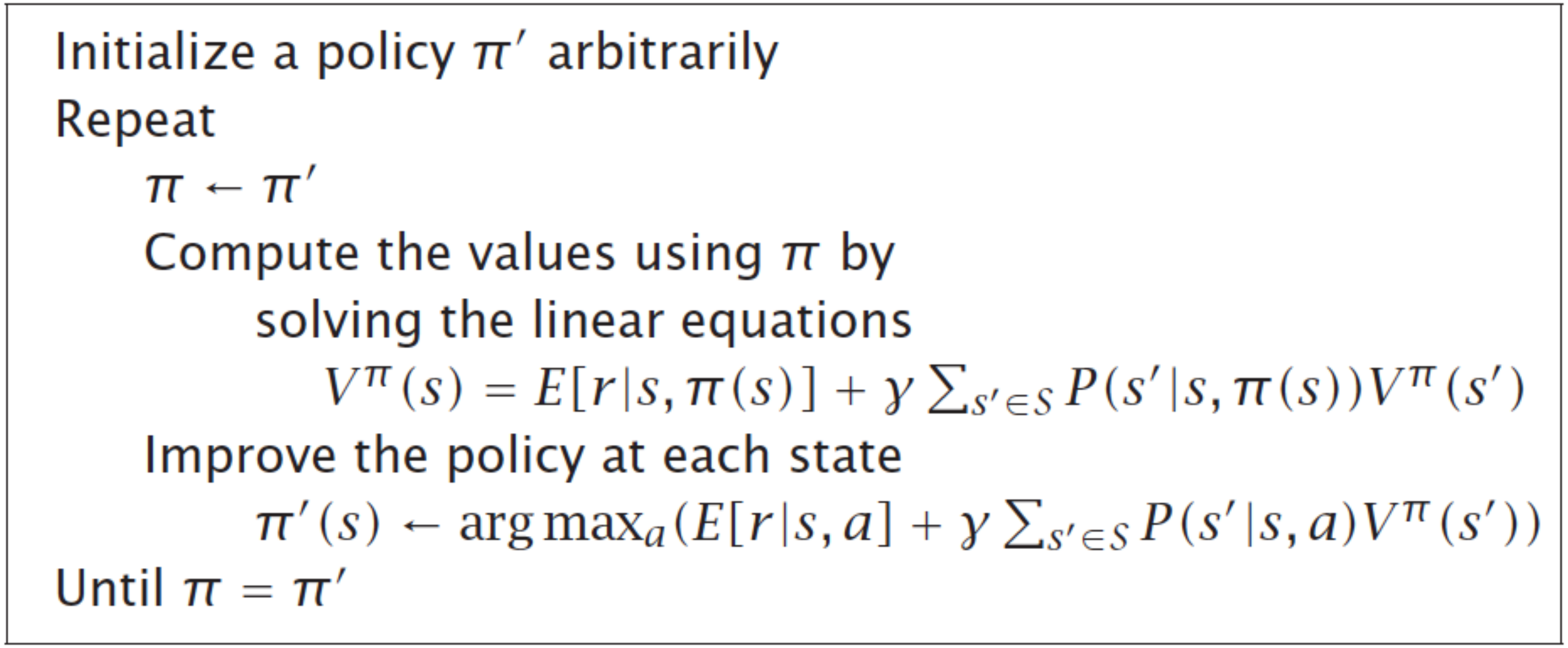


```
Initialize a policy $\pi'$ arbitrarily
Repeat
    $\pi \leftarrow \pi'$
    Compute the values using $\pi$ by
        solving the linear equations
            $V^{\pi}(s) = E[r|s, \pi(s)] + \gamma \sum_{s' \in S} P(s'|s, \pi(s)) V^{\pi}(s')$
    Improve the policy at each state
        $\pi'(s) \leftarrow \arg\max_a (E[r|s, a] + \gamma \sum_{s' \in S} P(s'|s, a) V^{\pi}(s'))$
Until $\pi = \pi'$
```

# Policy Iteration

- We can directly optimize in the policy space.

Smaller than Q-function space



```
Initialize a policy $\pi'$ arbitrarily
Repeat
    $\pi \leftarrow \pi'$
    Compute the values using $\pi$ by
        solving the linear equations
            $V^{\pi}(s) = E[r|s,\pi(s)] + \gamma \sum_{s' \in S} P(s'|s,\pi(s))V^{\pi}(s')$
    Improve the policy at each state
        $\pi'(s) \leftarrow \arg\max_a (E[r|s,a] + \gamma \sum_{s' \in S} P(s'|s,a)V^{\pi}(s'))$
Until $\pi = \pi'$
```

# Preliminaries

Following identity expresses the expected return of another policy $\tilde{\pi}$ in terms of the advantage over π, accumulated over time steps:

$$\begin{aligned}\eta(\tilde{\pi}) &= \eta(\pi) + \mathbb{E}_{s_0,a_0,\dots\sim\tilde{\pi}}\left[\sum_{t=0}^{\infty}\gamma^t A_\pi(s_t,a_t)\right] \\ &= \eta(\pi) + \sum_s \rho_{\tilde{\pi}}(s)\sum_a \tilde{\pi}(a|s)A_\pi(s,a)\end{aligned}$$

Where $A_\pi$ is the advantage function:

$$A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s)$$

And $\rho_{\tilde{\pi}}$ is the visitation frequency of states in policy $\tilde{\pi}$

$$\rho_{\tilde{\pi}}(s) = P(s_0 = s) + \gamma P(s_1 = s) + \gamma^2 P(s_2 = s) + \dots$$

# Preliminaries

To remove the complexity due to $\rho_{\tilde{\pi}}$ , following local approximation is introduced:

$$L_\pi(\tilde{\pi}) = \eta(\pi) + \sum_s \rho_\pi(s) \sum_a \tilde{\pi}(a|s) A_\pi(s,a)$$

If we have a parameterized policy $\pi_\theta$, where $\pi_\theta(a|s)$ is a differentiable function of the parameter vector $\theta$, then $L_\pi$ matches $\eta$ to first order. i.e.,

$$\begin{aligned} L_{\pi_{\theta_0}}(\pi_{\theta_0}) &= \eta(\pi_{\theta_0}), \\ \nabla_\theta L_{\pi_{\theta_0}}(\pi_\theta)\big|_{\theta=\theta_0} &= \nabla_\theta \eta(\pi_\theta)\big|_{\theta=\theta_0} \end{aligned}$$

This implies that a sufficiently small step that improves $L_{\pi_{\theta_\text{old}}}$ will also improve $\eta$, but does not give us any guidance on how big of a step to take.

# Preliminaries

- To address this issue, Kakade & Langford (2002) proposed conservative policy iteration:

$$\pi_{\text{new}}(a|s) = (1-\alpha)\pi_{\text{old}}(a|s) + \alpha\pi'(a|s)$$

where,

$$\pi' = \arg\max_{\pi'} L_{\pi_{\text{old}}}(\pi')$$

- They derived the following lower bound:

$$\eta(\pi_{\text{new}}) \geq L_{\pi_{\text{old}}}(\pi_{\text{new}}) - \frac{2\epsilon\gamma}{(1-\gamma)^2}\alpha^2$$

$$\text{where } \epsilon = \max_s \left| \mathbb{E}_{a\sim\pi'(a|s)}\left[A_\pi(s,a)\right]\right|$$

# Preliminaries

- Computationally, this $\alpha$-coupling means that if we randomly choose a seed for our random number generator, and then we sample from each of $\pi$ and $\pi_{new}$ after setting that seed, the results will agree for at least fraction 1-$\alpha$ of seeds.
- Thus $\alpha$ can be considered as a measure of disagreement between $\pi$ and $\pi_{new}$

# Theorem 1

- Previous result was applicable to mixture policies only. Schulman showed that it can be extended to general stochastic policies by using a distance measure called "Total Variation" divergence between π and $\tilde{\pi}$ as :

  $D_{TV}(p \parallel q) = \frac{1}{2}\sum_i |p_i - q_i|$ for discrete probability distributions p; q

- Let $D_{\mathrm{TV}}^{\max}(\pi, \tilde{\pi}) = \max_s D_{TV}(\pi(\cdot|s) \parallel \tilde{\pi}(\cdot|s))$

- They proved that for $\alpha = D_{\mathrm{TV}}^{\max}(\pi_{\mathrm{old}}, \pi_{\mathrm{new}})$, following result holds:

$$\eta(\pi_{\mathrm{new}}) \geq L_{\pi_{\mathrm{old}}}(\pi_{\mathrm{new}}) - \frac{4\epsilon\gamma}{(1-\gamma)^2}\alpha^2$$

$$where\ \epsilon = \max_{s,a} |A_\pi(s,a)|$$

# Theorem 1

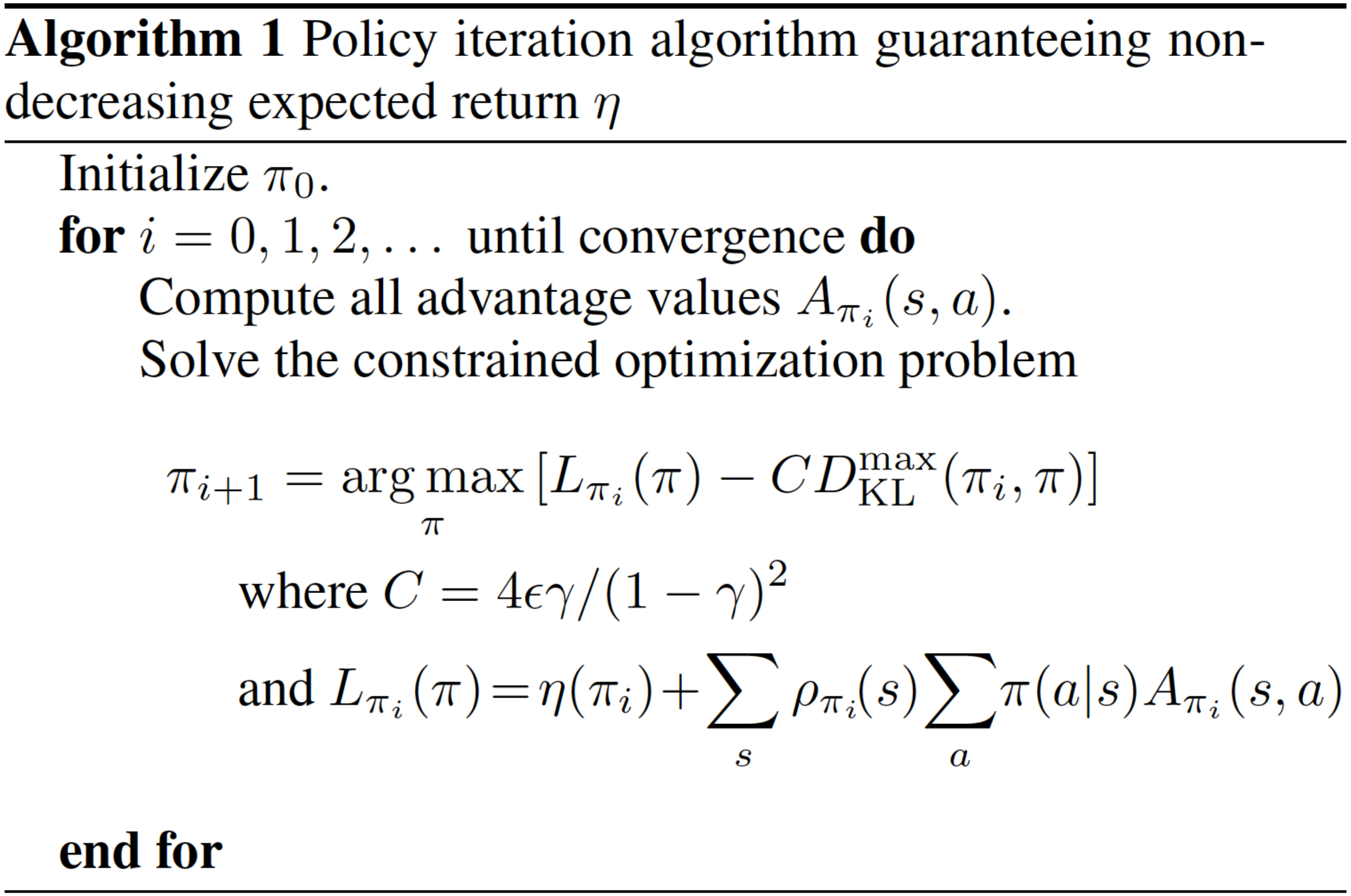
- Note the following relation between Total Variation & Kullback–Leibler:

$$D_{TV}(p \parallel q)^2 \leq D_{\mathrm{KL}}(p \parallel q)$$

- Thus bounding condition becomes:

$$\eta(\tilde{\pi}) \geq L_{\pi}(\tilde{\pi}) - C D_{\mathrm{KL}}^{\max}(\pi, \tilde{\pi}),$$
$$\text{where } C = \frac{4\epsilon\gamma}{(1-\gamma)^2}.$$

# Algorithm 1



**Algorithm 1** Policy iteration algorithm guaranteeing non-decreasing expected return $\eta$

```
Initialize $\pi_0$.
**for** $i = 0, 1, 2, \dots$ until convergence **do**
    Compute all advantage values $A_{\pi_i}(s, a)$.
    Solve the constrained optimization problem
        $$\pi_{i+1} = \arg\max_{\pi} \left[ L_{\pi_i}(\pi) - C D_{\mathrm{KL}}^{\max}(\pi_i, \pi) \right]$$
            where $C = 4\epsilon\gamma/(1-\gamma)^2$
            and $L_{\pi_i}(\pi) = \eta(\pi_i) + \sum_s \rho_{\pi_i}(s) \sum_a \pi(a|s) A_{\pi_i}(s, a)$
**end for**
```

# Trust Region Policy Optimization

- For parameterized policies with parameter vector, we are guaranteed to improve the true objective by performing following maximization:

$$\underset{\theta}{\text{maximize}}\left[L_{\theta_{\text{old}}}(\theta) - CD_{\text{KL}}^{\max}(\theta_{\text{old}}, \theta)\right]$$

- However, using the penalty coefficient like above results in very small step sizes. One way to take larger steps in a robust way is to use a constraint on the KL divergence between the new policy and the old policy, i.e., a **trust region constraint**:

$$\begin{aligned}&\underset{\theta}{\text{maximize}}\, L_{\theta_{\text{old}}}(\theta)\\&\text{subject to } D_{\text{KL}}^{\max}(\theta_{\text{old}}, \theta) \leq \delta.\end{aligned}$$

# Trust Region Policy Optimization

- The constraint is bounded at every point in state space, which is not practical. We can use the following heuristic approximation:

$$\overline{D}_{\mathrm{KL}}^{\rho}(\theta_1, \theta_2) := \mathbb{E}_{s\sim\rho}\left[D_{\mathrm{KL}}(\pi_{\theta_1}(\cdot|s) \parallel \pi_{\theta_2}(\cdot|s))\right]$$

- Thus, the optimization problem becomes:

$$\begin{aligned} &\underset{\theta}{\text{maximize}}\, L_{\theta_{\text{old}}}(\theta) \\ &\text{subject to } \overline{D}_{\mathrm{KL}}^{\rho_{\theta_{\text{old}}}}(\theta_{\text{old}}, \theta) \leq \delta \end{aligned}$$

# Trust Region Policy Optimization

- In terms of expectation, previous equation can be written as:

$$\begin{aligned} &\underset{\theta}{\text{maximize}}\, \mathbb{E}_{s\sim\rho_{\theta_{\text{old}}}, a\sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)} Q_{\theta_{\text{old}}}(s,a)\right] \qquad (14) \\ &\text{subject to } \mathbb{E}_{s\sim\rho_{\theta_{\text{old}}}}\left[D_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot|s) \parallel \pi_\theta(\cdot|s))\right] \leq \delta. \end{aligned}$$

where, *q* denotes the sampling distribution

- This sampling distribution can be calculated in two ways:
  - a) Single Path Method
  - b) Vine Method

# Final Algorithm

- Step 1: Use the single path or vine procedures to collect a set of state-action pairs along with Monte Carlo estimates of their Q -values
- Step 2: By averaging over samples, construct the estimated objective and constraint in Equation (14)
- Step 3: Approximately solve this constrained optimization problem to update the policy’s parameter vector