信赖域算法报告

TRPO

孟响

November 21, 2018

optimization

almost every optimization algorithms can be formed as:

$$x_{k+1} = x_k + \alpha_k d_k$$

where α_k is the step size and d_k is direction. Ways to determine α_k and d_k vary from method to method, and they have their own flaws and merits.

line-search method: determine d_k first, then α_k
trust-region method: determine α_k first, then d_k
first-order method: use gradient of point x_k to determine d_k
second-order method: use gradient and hessian to determine d_k

trust-region method

Trust-region methods define a region around the current iterate x_k within which they trust the quadratic model to be an adequate representation of the objective function, and then choose the step to be the approximate minimizer of the model in this trust region.

i.e. for each iteration, it solves the subproblem:

$$\text{minimize } m_k(t) = f(x_k) + \nabla f(x_k)^T t + \frac{1}{2} t^T H(x_k) t \tag{1}$$

$$\text{subject to } ||t||_2 \leq \triangle_k$$

trust-region method

Algorithm 1

Given $M > 0$, $\triangle_0 \in (0, M)$, and $\eta \in [0, \frac{1}{4})$

for k=0,1,2... **do**

obtain t_k by solving subproblem (1),evaluate $\rho_k = \frac{f(x_k) - f(x_k + t_k)}{m_k(0) - m_k(t_k)}$

if $\rho_k < \frac{1}{4}$ **then**

$\triangle_{k+1} = \frac{1}{4}||t_k||_2$

else

if $\rho > \frac{3}{4}$ and $||t_k||_2 = \triangle_k$ **then**

$\triangle_{k+1} = \min(2\triangle_k, M)$

else

$\triangle_{k+1} = \triangle_k$

end if

end if

$x_{k+1} = x_k + t_k$(if $\rho_k < \eta$, we often let $x_{k+1} = x_k$)

end for

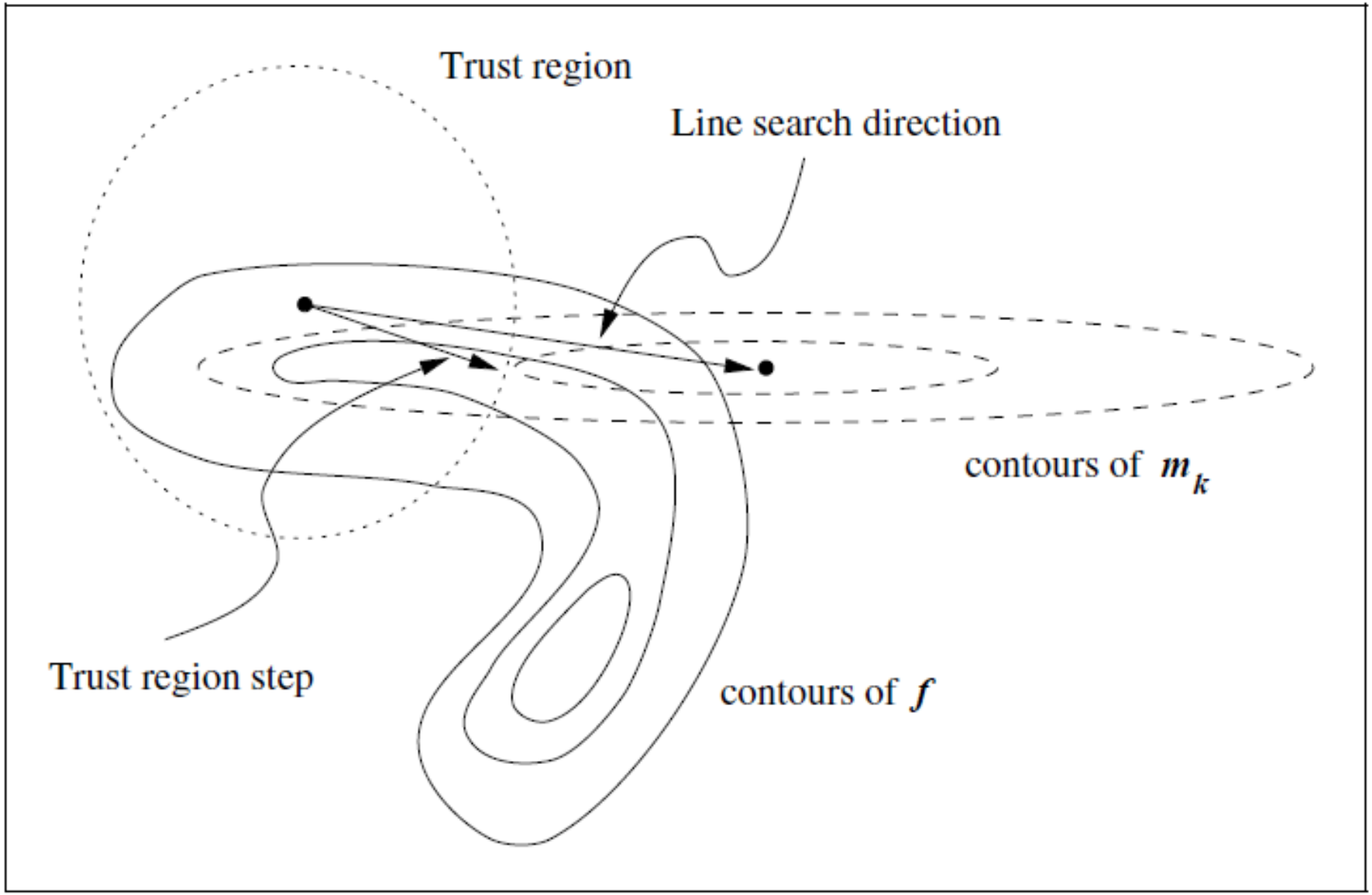


Figure 4.1 Trust-region and line search steps.

notation

- $\{S, A, R, P(s, a, s'), \pi, \gamma\}$ is defined as usual
- $V_\pi(s), Q_\pi(s, a)$ is the value(actioin-value) under policy π
- $A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s)$ is advantage function
- π_θ is the policy under parameter θ
- $\rho_\pi(s) = \frac{1}{1-\gamma} \sum_{n=0}^{\infty} \gamma^n P(s_t = s|\pi)$ is the discounted vistation frequencies
- $J(\pi_\theta) = \mathbb{E}_{s\sim\mu}[V_{\pi_\theta}]$ is the (usually used) perfomance measure of policy π_θ, where μ is the starting distribution.

problems posed by approximation methods

Value iteration and policy iteration method could converge exponentially in tabular case. However, approximate methods usually suffer from a lack of strong performance guarantees. We desire a method which could address following demands.

- Monotonicity. The method should be guaranteed to improve under some performance measure after every step. Bad update may cause policy degradation and divergence.
- Data efficiency. The method should attain a good performance in a reasonable time.
- Robustness (successful on a variety of problems)

First we would like to show the weakness of traditional methods. (approximate value function and policy gradient)

approximate value function

Approximate vallue function methods suffer from a paucity of theoretical results on the profermance of a policy based on the approximate values. We only have the following gurantee:(easy to prove)

$$V_{\pi'}(s) \geq V_{\pi}(s) - \frac{2\gamma\epsilon}{1-\gamma}$$

where $\epsilon = \max_s |\tilde{V}(s) - V_{\pi}(s)|$ is the l_∞ error of approximater $\tilde{V}$ and γ the discount rate.
So these methods don't assure improvement and only have asymptotic convergence results.

policy gradient methods

Traditional policy gradient methods ($\nabla J(\theta) = \mathbb{E}_{s\sim\pi_\theta}[A(s,a)\nabla ln\pi_\theta(s,a)]$) can assure convergence as it is a SGD method.
But several examples (**Approximately optimal approximate reinforcement learning, section 3.2**) show that policy gradient has the problem as a lack of exploration. As a result, it can't reach a satisfying performance in a reasonable time. To gain an insight of this, notice that in policy gradient method, we update $\pi_\theta(s,a)$ only if we visit state s. If a state s is important in the optimal policy π^*, but is visited less frequently under the current policy, it will take a long time for agent to learn a right policy at state s.
Besides, policy gradient method is of high variance, and it's difficult to choose an appropriate learning rate to assure monotonicity.

the origin of TRPO

To deal with these problems, Kakade and Landford provided two methods: **consevative policy iteration (CPI)**to ensure monotonicity and **natural policy gradient (natural PG)** to speed up learning.
TRPO gives a integrated modification of these two methods, makes it scalable to large, nonlinear, continuing models.
First, we will give a brief introduction of CPI and natural PG in the next few frames.

conservative policy iteration (CPI)

The following proposition is useful:
given two poilcy $\tilde{\pi}$, π,

$$J(\tilde{\pi}) = J(\pi) + \mathbb{E}_{\tau\sim\tilde{\pi}}[\gamma^t A_\pi(s_t, a_t)] \tag{2}$$

where $\mathbb{E}_{\tau\sim\tilde{\pi}}$ takes over trajectories $\tau := (s_0, a_0, s_1, a_1, \dots)$ under policy $\tilde{\pi}$

Proof.

$$\mathbb{E}_{\tau\sim\tilde{\pi}}[\sum_{t=0}^{\infty}\gamma^t A_\pi(s_t, a_t)] = \mathbb{E}_{\tau\sim\tilde{\pi}}[\sum_{t=0}^{\infty}\gamma^t(r(s_t) + \gamma V_\pi(s_{t+1}) - V_\pi(s_t))]$$

$$= \mathbb{E}_{\tau\sim\tilde{\pi}}[-V_\pi(s_0) + \sum_{t=0}^{\infty}\gamma^t r(s_t)] = -J(\pi) + J(\tilde{\pi})$$

□

conservative policy iteration (CPI)

However, under the current poilcy π,

$$J(\tilde{\pi}) = J(\pi) + \mathbb{E}_{\tau\sim\tilde{\pi}}[\gamma^t A_\pi(s_t, a_t)] = J(\pi) + \sum_s \rho_{\tilde{\pi}}(s) \sum_a \tilde{\pi}(a|s) A_\pi(s, a)$$

is difficult to optimize directly as a result of its complex dependency of $\rho_{\tilde{\pi}}(s)$
Instead, we use the local approximation to J:

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

It's easy to prove that $L_\pi(\pi) = J_\pi(\pi)$, $\nabla L_\pi(\pi) = \nabla J_\pi(\pi)$. It means a sufficiently small step which improves L will also improve J, but it doesn't tell us how big step we should take.

conservative policy iteration (CPI)

CPI constrained the update in the case of

$$\pi_{new}(s,a) = \alpha\pi(s,a) + (1-\alpha)\tilde{\pi}(s,a) \tag{3}$$

And this theroem guarantees a proper α could improve the poilcy:

$$J(\pi_{new}) - J(\pi) \geq \alpha\mathbb{A}_{\pi}(\tilde{\pi}) - \frac{2\alpha^2\epsilon\gamma}{(1-\gamma)^2} \tag{4}$$

where $\epsilon = \max_s \left|\mathbb{E}_{a\sim\tilde{\pi}}[A_{\pi}(s,a)]\right|$ and
$\mathbb{A}_{\pi}(\tilde{\pi}) = \sum_s \rho_{\pi}(s)\sum_a \tilde{\pi}(a|s)A_{\pi}(s,a)$
we will prove a stronger version of this theorem in the following section.

conservative policy iteration (CPI)

The conservative policy iteration algorithm is:

1. estimate $Q_\pi(s,a)$ and $V_\pi(s)$ under the current policy π to obtain a new policy $\tilde{\pi}$ which maxmizes $\mathbb{A}_\pi(\tilde{\pi})$
2. get an estimate $\mathbb{A}$ of $\mathbb{A}_\pi(\tilde{\pi})$
3. if $\mathbb{A}$ is small enough, stop and return π
4. else, update π according to equation (3), using proper α.

natural PG

Given θ, we want to find the steepest descent direction of $J(\pi_\theta)$, which means to solve the following optimization question:

$$\max \ J(\pi_{\theta+\Delta\theta})$$

$$s.t. \ |\Delta\theta| < \delta$$

However, it's better to use Fisher information matrix (FIM) to measure $\Delta\theta$ instead of euclidean metric.(suggested by Amari, see **this** to gain an insight).
For a given state s, the FIM of π_θ is

$$F_s(\pi_\theta) = \mathbb{E}_{a\sim\pi}[\frac{\partial ln\pi_\theta(s,a)}{\partial\theta_i}\frac{\partial ln\pi_\theta(s,a)}{\partial\theta_j}]$$

natural PG

As shown by Amari, FIM is an invariant metric in the sense that it defines same distance between two points regardless of the choice of coordinates. For all states, an intuitive choice of metric is

$$F(\pi_\theta) = \mathbb{E}_{s\sim\pi}[F_s(\pi_\theta)]$$

Applying lagrange duality, the steepest descent direction this gives is

$$\tilde{\nabla} J(\pi_\theta) = F(\pi_\theta)^{-1} \nabla J(\pi_\theta)$$

a proof

we first give a proof of theorem(4). As it is impractical to use an update like equation(3) in non-linear case, we will extend it to the following form:
Given policy π, $\tilde{\pi}$, define $D_{TV}(p||q) = \frac{1}{2}\sum_i |p_i - q_i|$,
$\alpha = \max_s D_{TV}(\pi(s,\cdot)||\tilde{\pi}(s,\cdot))$
we have:

$$J(\tilde{\pi}) \geq L_{\pi}(\tilde{\pi}) - \frac{4\epsilon\gamma}{(1-\gamma)^2}\alpha^2$$

where $\epsilon = \max_{(s,a)} |A_{\pi}(s,a)|$
(remark: it is straightforward to extend the theorem to continuing case by changing the sums to integrals)

Proof.

for all state s, we have:

$$\mathbb{E}_{\tilde{a}\sim\tilde{\pi}}[A_\pi(s,\tilde{a})] \leq 2\alpha \max_{(s,a)} A_\pi(s,a)$$

this is because

$$\mathbb{E}_{\tilde{a}\sim\tilde{\pi}}[A_\pi(s,\tilde{a})] = \mathbb{E}_{(a,\tilde{a})\sim(\pi,\tilde{\pi})}[A_\pi(s,\tilde{a}) - A_\pi(s,a)] \ (\mathbb{E}_{a\sim\pi}[A_\pi(s,a)] = 0)$$

$$= P(a \neq \tilde{a}|s)\mathbb{E}_{(a,\tilde{a})\sim(\pi,\tilde{\pi})|a\neq\tilde{a}}[A_\pi(s,\tilde{a}) - A_\pi(s,a)]$$

$$\leq \alpha \cdot 2 \max_{s,a} |A_\pi(s,a)|$$

$$(because\ \alpha = \max_s D_{TV}(\pi(s,\cdot)||\tilde{\pi}(s,\cdot)))$$

Proof.

$$|J(\tilde{\pi}) - L_{\pi}(\tilde{\pi})| = \sum_{t=0}^{\infty} |\mathbb{E}_{\tau\sim\tilde{\pi}}[\gamma^t A(s_t, a_t)] - \mathbb{E}_{\tau\sim\pi}[\gamma^t A(s_t, a_t)]|$$

$$\leq \sum_{t=0}^{\infty} \gamma^t P_t(|\mathbb{E}_{\tau\sim\tilde{\pi}}[A(s_t, a_t)]| + |\mathbb{E}_{\tau\sim\pi}[A(s_t, a_t)]|)$$

(P_t is the probability that π, $\tilde{\pi}$ don't disagree before timestep t)

$$\leq \sum_{t=0}^{\infty} \gamma^t (1 - (1-\alpha)^t)(|\mathbb{E}_{\tau\sim\tilde{\pi}}[A(s_t, a_t)]| + |\mathbb{E}_{\tau\sim\pi}[A(s_t, a_t)]|)$$

$$\leq \sum_{t=0}^{\infty} \gamma^t (1 - (1-\alpha)^t) 4\epsilon\alpha \leq \frac{4\gamma\epsilon\alpha^2}{(1-\gamma)^2}$$

□

a theoretical non-decreasing algorithm

It's not difficult to prove that $D_{TV}(p||q)^2 \leq D_{KL}(p||q)$ (using adjustment method, we could assume that for all i which $p_i < q_i$, $\frac{p_i}{q_i}$ is same, the following part is trivial)
By the preceding theorem, we have:

$$J(\tilde{\pi}) \geq L_{\pi}(\tilde{\pi}) - CD_{KL}^{max}(\pi, \tilde{\pi}) \; where \, C = \frac{4\epsilon\gamma}{(1-\gamma)^2}$$

Starting with π_0, solve problem

$$\pi_{i+1} = argmax_{\pi}[L_{\pi_i}(\pi) - CD_{KL}^{max}(\pi_i, \pi)]$$

for every timestep i, it's easy to verify that we got a monotonically improving sequence of policies.
(remark: this algorithm is simliar to proximal method and mirror descent)

a practical algorithm

Notice that $C = \frac{4\epsilon\gamma}{(1-\gamma)^2}$ could become very large when γ is near to 1. In practice, it will lead to a small stepsize, besides, $D_{KL}^{max}(\pi, \tilde{\pi})$ is difficult to approximate.
To address these problems, we use the trust region constraint instead:

$$\max_{\theta} L_{\pi_{\theta old}}(\pi_\theta)$$

$$subject\ to\ \bar{D}_{KL}^{\rho_{\theta old}}(\pi_{\theta old}, \pi_\theta) \leq \delta$$

where

$$\bar{D}_{KL}^{\rho_{\theta old}}(\pi_{\theta old}, \pi) = \mathbb{E}_{s\sim\pi_{\theta old}}[D_{KL}(\pi_{\theta old}(s,\cdot), \pi_\theta(s,\cdot))]$$

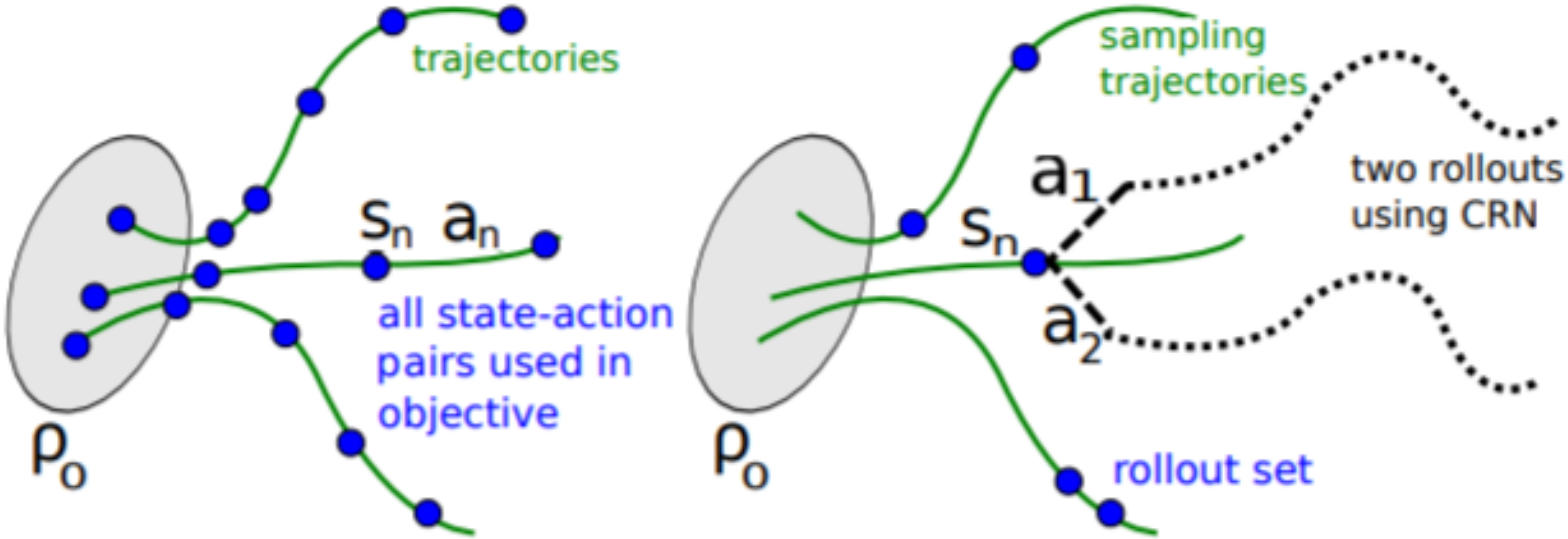
is the average KL divergence over states.

estimation of the objective and constraint

In practice, we optimize of following problem:

$$\text{maximize } \mathbb{E}_{s\sim\pi_{\theta old},a\sim q}[\frac{\pi_\theta(a|s)}{q(a|s)}Q_{\theta old}(s,a)] \tag{5}$$

$$\text{subject to } \mathbb{E}_{s\sim\pi_{\theta old}}[D_{KL}(\pi_{\theta old}(s,\cdot),\pi_\theta(s,\cdot))] \leq \delta$$



a practical algorithm

The actual algorithm we use repeatedly perform the following steps until convergence:

1. use sampled-based monte-carlo simulation to collect a set of state-action pairs (s, a) and their values $Q_\pi(s, a)$ with regard to the current policy.
2. Approximate objective and constraint by using following formula:

$$L = \sum_s \frac{\sum_i \frac{\pi_\theta(s,a_i)}{\pi_{\theta old}(s,a_i)} Q(s, a_i)}{\sum_i \frac{\pi_\theta(s,a_i)}{\pi_{\theta old}(s,a_i)}}$$

$$\bar{D}_{KL} = \sum_s \frac{1}{2}(\theta - \theta_{old})^T A(\theta - \theta_{old})$$

where $A = \frac{\partial}{\partial \theta_i} \frac{\partial}{\partial \theta_j} D_{KL}(\pi_{\theta old}(s, \cdot), \pi_\theta(s, \cdot))$

a practical algorithm

3. Use conjugate gradient algorithm to compute search direction $s = A^{-1}g$, where g is the gradient of function L. (A is the Hessian of KI divergence, so it is definite.)
4. Perform a line search in direction s to ensure improvement and update θ.

Observe that the update has a form which is the same as natural PG. In this way TRPO can be considered as a combination of CPI and natural PG.

some flaws of TRPO

TRPO achieves satisfying performance in continuing tasks and atari games. But it is relatively complicated and not compatible with parameter sharing(auxiliary tasks or policy and value funciton) and dropout.
In addition, TRPO doesn't totally solve the problem of lack of exploration.
Proximal policy Optimization (PPO) seeks to find a way to address these flaws to some extent.

Clipped objective

TRPO uses KL divergence as a constraint, which makes its update different to the original gradient. PPO proposes a clipped surrogate objective that aims to do the same thing as KL constraint does–penalize policy that move $\frac{\pi(s,a)}{\pi_{\theta old}(s,a)}$ far away from 1.
The clipped surrogate objective is:

$$L^{CLIP}_{\pi_{\theta old}}(\pi_\theta) =$$

$$\mathbb{E}_{(s,a)\sim\pi_{\theta old}}[\min(\frac{\pi(s,a)}{\pi_{\theta old}(s,a)}A_{\pi_{\theta old}}, clip(\frac{\pi(s,a)}{\pi_{\theta old}(s,a)}, 1-\epsilon, 1+\epsilon)A_{\pi_{\theta old}})]$$

where

$$clip(x, 1-\epsilon, 1+\epsilon) = \begin{cases} 1-\epsilon & x < 1-\epsilon \\ x & 1-\epsilon \le x \le 1+\epsilon \\ 1+\epsilon & 1+\epsilon < x \end{cases}$$

PPO

The PPO builds a network architecture that share parameters between policy and value function.
The ultimate loss function is:

$$L_{\pi_{\theta old}}(\pi_\theta) = \mathbb{E}_{(s,a)\sim\pi_{\theta old}}[L^{CILP}(\theta) - c_1 L^{VF}(\theta) + c_2 S_s(\theta)]$$

where c_1, c_2 is coefficients, L^{VF} is the value prediction error and S_s is entropy at state s in order to assure sufficient exploration.
PPO use a n-step bootstrap to estimate $A_{\pi_{\theta old}}(s, a)$, update θ for several minibatch and then update loss function L.

some material

There are some materials I've read in order to comprehend TRPO:

- **Proximal policy optimization paper**
- **Trust region policy optimization paper**
- **Natural policy gradient**
- **conservative policy iteration**
- **Fisher Information**
- **open source of PPO**
- **a introduction of opensource**