# Constrained Policy Optimization笔记

**写于2022.10.22，更新于2022.11.18**

**王荣荣**

参考文献：Joshua Achiam, David Held, Aviv Tamar, Pieter Abbeel. **Constrained Policy Optimization**. Proceedings of the 34th International Conference on Machine Learning, PMLR 70:22-31, 2017.

主要记录一下CPO这篇论文中各个公式的推导，都是咋得出来的。

## 3. Preliminaries

这是一堆符号定义，有个印象：

A Markov decision process (MDP) is a tuple 

*S* is the set of states

*A* is the set of actions

 is **the reward function**

 is the transition probability function, where  is the probability of transitioning to state s’ given that the previous state was s and the agent took action a in s.

 is the starting state distribution

A stationary policy  is a map from states to probability distributions over actions, with  denoting the probability of selecting action a in states s. The set of all stationary policies by .

In RL, we aim to select a policy  which maximizes **a performance measure, **, which is typically taken to be the infinite horizon discounted total return ,  is the discount factor

 denotes a trajectory (),  is shorthand for indicating that the distribution over trajectories depends on ， denote the discounted return of a trajectory

we press the on-policy value function as 

插一句：另一种表述。若，则



the on-policy action-value function as 

the advantage function is 

the discounted **future** state distribution , defined by 

插一句：为啥要有一个(1-*γ*)，别的将(1-*γ*)解释为normalizing factor。这是因为有了(1-*γ*)之后，*d*就可以称作概率分布了，没有(1-*γ*)的*d*不能叫做概率分布，因为对*s*求和之后*d*不等于1.

推导一下：



之前用的这种形式实际上并不是概率分布，对s求和之后



上述用到等比数列求和，公比为，则，当，

这个结论是2002年Kakade & Langford给出来的“Approximately optimal approximate reinforcement learning”

the difference in performance between two policies ,  as  (1)

问题是这么多文章并没有用到这个结论，只是和2002年这篇的结论进行比较，说明我们所提的界限更紧。

## 4. Constrained Markov Decision Processes

a set *C* of auxiliary cost functions, , with each one a function  **mapping transition tuples to costs**, like the usual reward, and limits 

 denote the expected discounted return of policy  with respect to cost function 

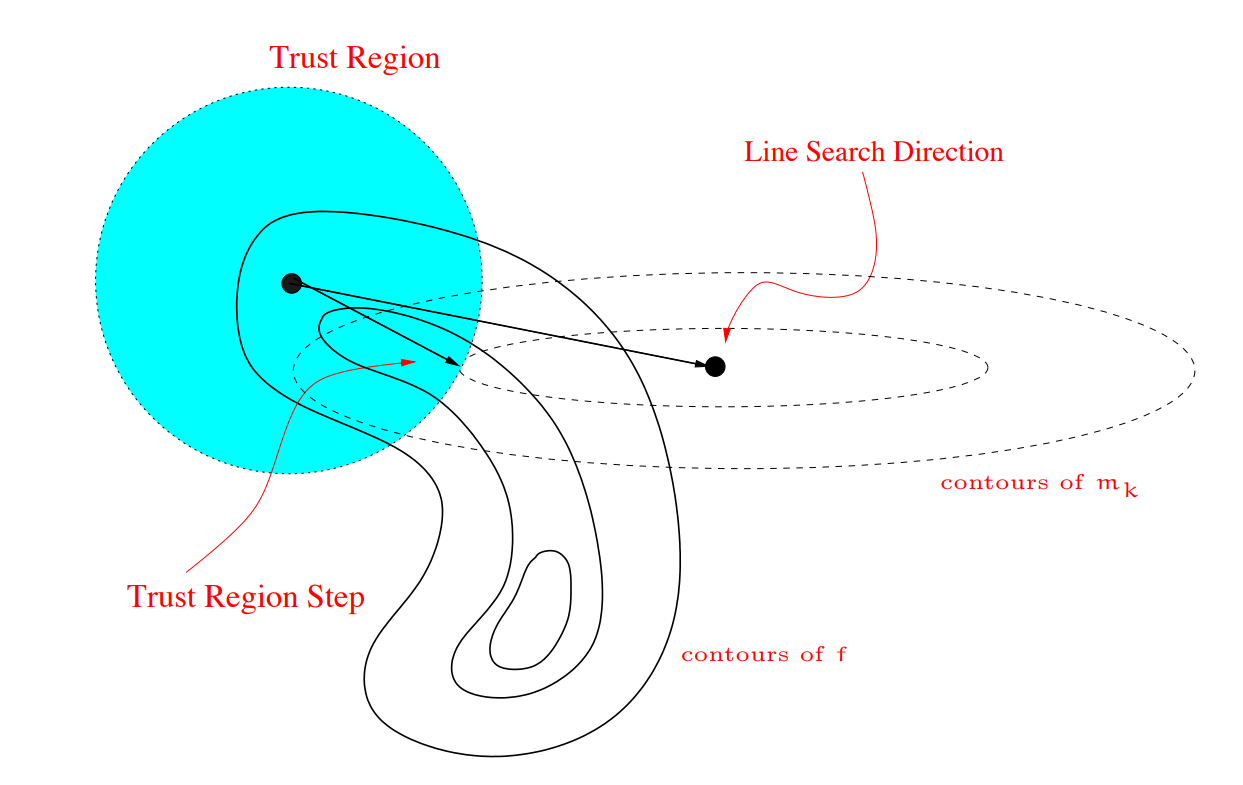
the set of feasible stationary policies for a CMDP is then (*d*界限)

the reinforcement learning problem in a CMDP is 

## 5. Constrained Policy Optimization

Policy search algorithms approach this problem by searching for the optimal policy within a set  of parameterized policies with parameters .

补充信赖域法(本身是一种求解无约束优化问题的方法)：



首先, 给定一个所谓的"信赖域半径"作为位移长度的上界, 并以当前迭代点为中心以此"上界"为半径"画地为牢"确定一个称之为"信赖域"的闭球区域.

然后, 通过求解这个区域内的"信赖域子问题"(目标函数的二次近似模型)的最优点来确定"候选位移".

若候选位移能使目标函数值有充分的下降量, 则接受该候选位移作为新的位移, 并保持或扩大信赖域半径, 继续新的迭代;

否则, 说明二次模型与目标函数的近似度不够理想, 需要缩小信赖域半径, 再通过求解新的信赖域内的子问题得到新的候选位移.

如此重复下去, 直到满足迭代终止条件.

这里还没有考虑约束项，只是传统的强化学习的信赖域求解的目标函数：

In the local policy search, the policy is iteratively updated by maximizing  over a local neighborhood of the most recent iterate : (前一次迭代与当前迭代的策略之间的距离应在信赖域半径范围内)

 (2)

其中*D* is some distance measure， is a step size.

when the objective is estimated by linearizing around  as , *g* is the policy gradient, 这里目标函数用的是一阶泰勒展开来近似，信赖域约束度量用的二范数。

下面考虑了安全约束，在约束条件基础上添加cost的约束项：

we optimize over 

 (3)

这里假设有*m*个约束项

### 5.1. Policy Performance Bounds

a new bound on the different in returns between two arbitrary policies

providing tighter bounds. 相比于TRPO与2002年那篇文章

又一次回顾一下这个结论，2002年Kakade & Langford给出来的：



CPO这篇文章给出了一个新边界，与有没有安全约束没有关系，都可以用：

**推论1：** (5)

下面这个是针对安全约束，CPO这篇文章也给出了cost的边界：

推论2： (6)

其中，。

这两个结论都是从定理1推过来的，关于定理1的证明，放在了最后，这里先默认定理1成立。

补充：**定理1：**

定义

以及，则边界满足：





其中。

推论1与2的具体推导：

令，则：，

由贝尔曼方程，

结合上面两个式子，有，

因此：



注意：

，，

因此：



(这里由于与*a*无关，常数的期望还是本身，

因此。)

这样就和推论1与2对应起来了。得证。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

下面为了得到推论3，引入了一些基本概念：

**Pinsker’s inequality**

，其中

**Jensen’s inequality**

若*f* (*x*)在[*a*, *b*]上是凸函数，则，如*f* (*x*)=| *x* |，1范数(绝对值)函数为凸函数。

但*f* (*x*)为对数函数时，由于对数函数是凹函数，所以有

因此，下面CPO公式(7)第一个不等式用的Pinsker’s inequality，第二个不等式用的Jensen’s inequality(这里是凹函数)。

~~(疑问：为什么这些文章都是用TV散度(the total variation divergence，总变差散度)来衡量分布之间的关系呢？它相比于其他度量方法有什么好处呢？最后还是用KL散度来近似了啊，为啥不直接就用KL散度呢？就因为KL散度是不对称吗？那为啥不用JS散度呢？这个是对称的，但不好求。一般理论推导用TV散度，对称，性质好，但实际操作时用KL散度，好求。补充：在CPO里面(其他文章可能也有类似的情况)，推导时TV散度与1范数存在关系，因此1范数直接用2倍TV散度来替代，后面定理1证明时就会用到)~~

 (7)

推论3：In bounds (4), (5), and (6), make the substitution



The resulting bounds hold.

推论3意思是直接可以用右边KL散度那个边界来替代左边的TV散度。

### 5.2. Trust Region Methods

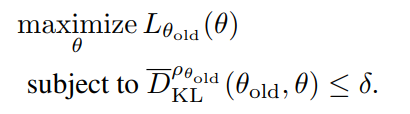
(这里还没涉及到cost，只涉及到信赖域方法，这个(8)就是TRPO里面的公式)

policy updates

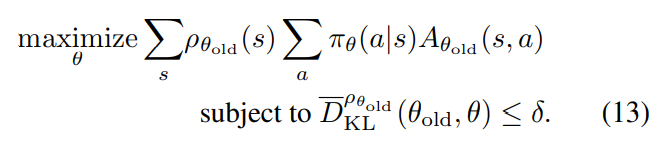
 (8)

其中, the set  is called the trust region,  is the step size.

补充：TRPO里面的公式：



又因为，第一项与*θ*无关，



~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**命题1： Trust Region Update Performance**

 (9)

其中 

这个命题在未考虑安全约束条件的情况下，只是采用信赖域方法能得到这个性能边界。

This result is useful for two reasons: 1) it is of independent interest, as it helps tighten the connection between theory and practice for deep RL, and 2) the choice to develop CPO as a trust region method means that CPO inherits this performance guarantee.

(我先前的疑问：命题1右边这一项是个负数，这不就意味着迭代时*J*一次比一次小吗？和最大化*J*目标相违背？

后来想到的：不是这样的，这个只是一个下界，实际中未必达到，只是说我们找到了这样一个界限，和数理统计里面的一致最小方差无偏估计C-R下界类似，实际新旧*J*之差未必能达到这个下界。

请教张佳志的：命题1右边和*δ*相关，也就是说*δ*越接近于0性能越能够得到稳定提升。*δ*就是新旧策略差异。至于负数，他们之间的关系就是这样的，*δ*越小越能够得到稳定提升，这样理解就好。这个是用了trust region之后的边界，和最开始那个目标函数不一样了)

证明：

这个命题的证明用到了：

推论1：

推论3：



信赖域那个约束条件 ，则以及

因此，

命题1得证。

### 5.3. Trust Region Optimization for Constrained MDPs

最初CMDP的公式：



现在变为：



**目标函数比原先小了，约束比原先更紧了。**

第一项目标项的推导：用的推论1和3。

由推论1：



又推论3：

所以：

又定义符号

因此，

先前是，现在把目标函数变小了：



由于待求的是*π*，因此第一项为常数，与*π*没关系，可以舍掉。

这样。

第二项约束条件：用到了推论2和3。



原本应该是，又因为

以及推论3：

因此：



这样，因此

举个例子(当然不太严谨，只是了解大概意思)：原先*x*≤10，现在*x*≤*x*+2≤10，*x*≤8。*x*的范围更小了，如果这个约束*x*≤8都满足，那么*x*≤10不是轻而易举了。

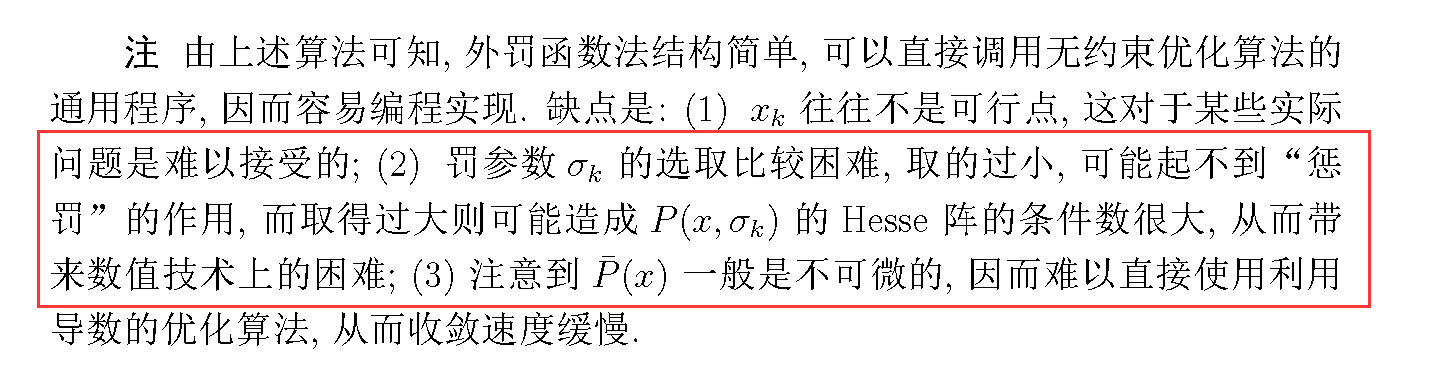
~~(先前的理解：这种约束变得更强了，原本只是，现在在左边又新加了两项，这种约束更紧了，这样一定能确保。)~~

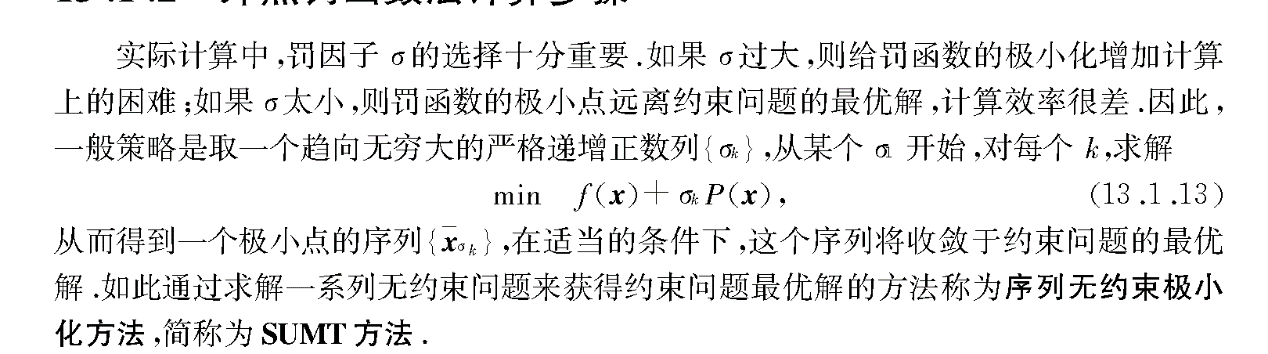
前面一顿操作猛如虎，终于经过一堆推导后得到了CPO：

Inspired by trust region methods, we propose CPO, which uses a trust region instead of penalties on policy divergence to **enable larger step sizes**.

 (10)

(请教张佳志的：CPO为啥又把信赖域约束放在约束条件下面了，*α*或*β*里面需要从当前的策略采样，没法做，或者*γ*取值很大(如0.999)导致*α*或*β*值很大，值太大的话，取最优就会变慢。) 就像外点法内点法里面的罚因子，如果罚因子太大的话，收敛慢。所以，又用信赖域方法来求解了，把惩罚项放在约束条件那块了。软约束转换为硬约束，常用的一种处理手段！和先前讲的CVPO套路一样(当然CVPO本身就是借鉴CPO的思路)，将拉格朗日惩罚项转变为约束项。





**命题2：CPO Update Worst-Case Constraint Violation**



其中

(这个是已知了CPO目标(10)的情况下得到的最差约束上界，意思是安全约束目标函数值最大也不会超过右边那个数值，有界。主要用到(10)的两个约束项。)

证明：由推论2以及(10)的约束项：



因此

又因为，所以

命题2得证。

前述推论1与2都用到定理1，没有定理1，也出不来后续的那一堆结论。而这个定理1是通用的，和CPO本身没关系，任何算法都适用，甚至未必是强化学习算法，只要满足条件，都能用这个定理。

为了证明定理1，前期需要一堆工作来做铺垫：

首先，定义the discounted future state distribution：

It allows us to express the expected discounted total reward compactly as

 (17)

默认了，放弃了，这只是一种定义，不想再纠结它怎么来的了，找不到。

(更新(我目前能想到的)：如果写到(17)右边，与前面的正好抵消)

设， denote the transition matrix, , ，(利用数学归纳法得到这个*t*表示*t*次方，而不是*t*时刻。大写的*P*对应的*t*是*t*次方，而小写的*p*对应的*t*是*t*时刻，这玩意定义的符号都一模一样难道不容易混淆吗？)

这就是状态转移概率的表达式*t-*1时刻的状态乘以状态转移矩阵之后得到当前的状态分布。

 is the starting state distribution 表示初始状态分布

 (18)

上述仍然用的等比数列求和公式

引理1：

 (19)

引理1的证明：(18)两边同乘，然后与*f* (*s*)做内积



左边：

右边：

因此得到  (19)

引理1得证。

引理1得到后，是为了推出(20)这个公式，在原先的*J*基础上加了引理1左边的为0的这一串公式：

(19) 先除以(1-*γ*)，然后再加上(17)，即(加(19)0这一项不影响，和原来的不变)

因此，

所以得到  (20)

引理1是为了得到引理2做铺垫，引理2中间会用到公式(20)

### 10.1. Proof of Policy Performance Bound

引理2：定义 (22)

以及，则边界满足：

 (23)

 (24)

LHS --> Left Hand Side，不等式左边，RHS --> Right Hand Side，不等式右边。

说的是这个约束(23)(24)是tight，当*π*’=*π*时，LHS与RHS等于0。

证明：For notational convenience, let 

用到了等式(20)：



上面式子再将*π*换成*π*’，。

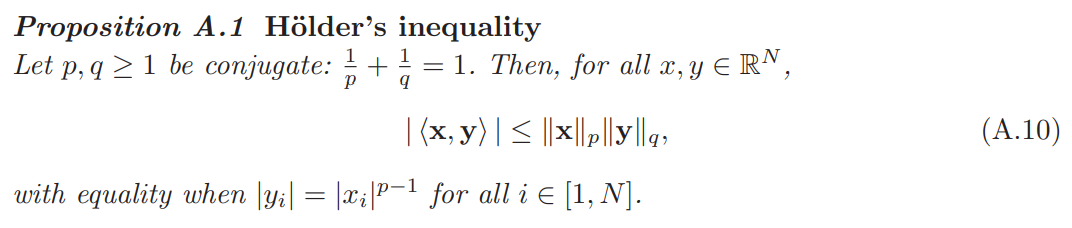
因此，

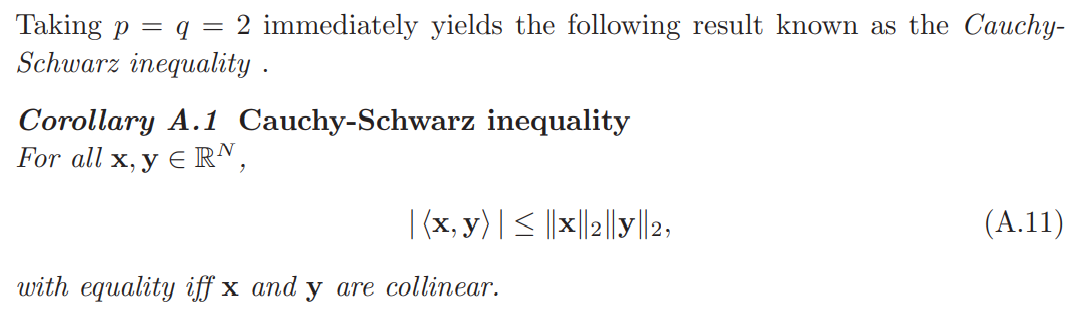
(上面*J*(*π*)与*J*(*π’*)的第一项都一样，于是就消掉了。)

令，

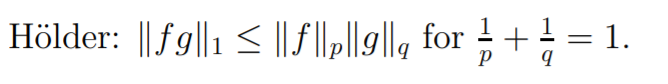
则(\*\*\*)

**Holder’s inequality** (来源：Foundations of Machine Learning (P342))





另一个关于Holder不等式的写法：



for any  such that , we have (\*\*\*)的第二项：



去掉绝对值：





而

因此，

令*p*=1, *q*=∞，(一)别忘了TV散度的公式：

(补充：这里解释为啥用TV散度，这是由于1范数与TV散度之间的关系，因此推导时1范数直接用2倍的TV散度来代替)

若Ω是一个可数状态空间，则，则

(二)还有：，

(引理2中的定义)

，则

因此上面不等式变成：



又因为

不等式太长了，拆开看：

先看右边：(这个在后续标蓝的地方用到了)

同理，若*s*从*dπ*中采样：

再看左边：

同理，若*s*从*dπ*中采样：

* 

(第一个等式是左边既≥右边，又≤右边，因此就等于右边，第二个等式是用了重要性采样。)

(思考：为啥不直接利用*s*从*dπ*中采样，*a*从*π*中采样的公式呢，因为这样的话右边是而不是，就和这个出来的没法消掉了，妙啊。)

因此，

(红色箭头上面那个式子把右边的重要性采样权重挪到左边去。严格意义上写法可能不对，得写成求和或期望的形式，理解意思就行，其中用到)



所以，

其中



因此，

因此得到(24)。同理(23)也能得出来，没必要重复了。

得证。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

介绍引理3之前先定义一波：

**定义**，因此

若左乘*G*右乘，则 (\*)

若左乘右乘*G*，则 (\*\*)

注意矩阵乘法满足结合律，(AB)C=A(BC)。由(\*)与(\*\*)，则。

所以，

 (21)

引理3：  (25)

其中

证明：

用到了(21)其中用到范数不等式||AB||≤||A||||B||

用到了(18)



其实就是等比数列的反推。因此，

上面不等式那里还用到了Jensen不等式，1范数是凸函数。若*f* (*x*)在[*a*, *b*]上是凸函数，则。

回顾符号的定义， denote the transition matrix, ，

下面把与*d*写具体了，并利用Jensen不等式，其中还是用到绝对值是1范数是凸函数



下面将写详细了，*a*也包含进去



再次利用Jensen不等式，仍然是绝对值1范数是凸函数



接下来，*s*’那个先合并，由于状态转移概率是一个概率密度函数，因此求和为1。



又用到TV散度的定义：若Ω是一个可数状态空间，则

因此，，所以

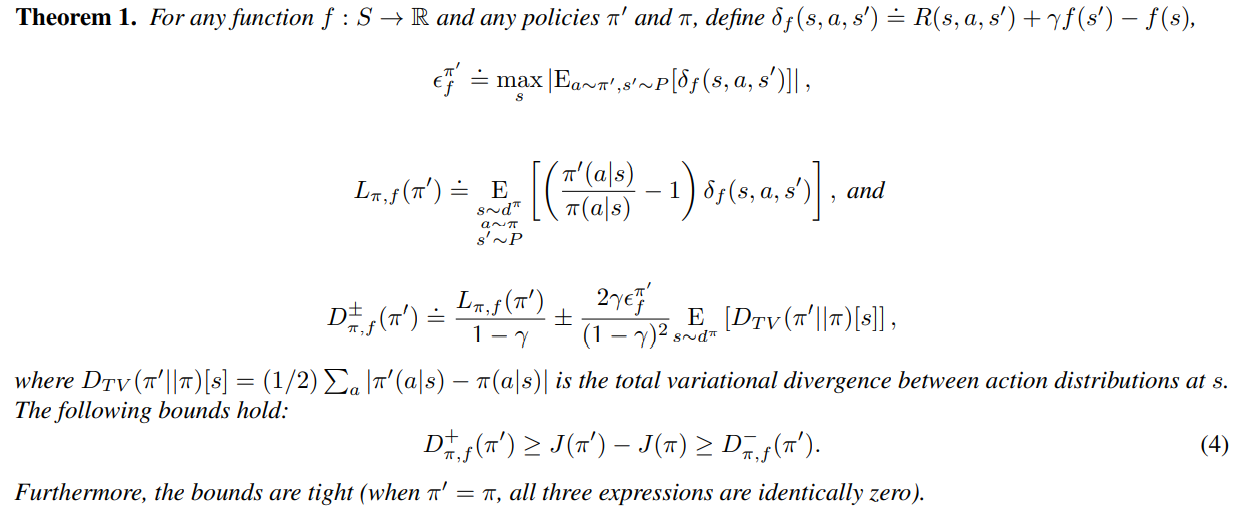
又，因此

引理3得证。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

经过前面那一堆的铺垫，定理1终于千呼万唤始出来了！**The new policy improvement bound follows immediately.**

定理1：



有了上述的引理2与引理3，这个定理自然就成立了。

引理2与定理1差的部分是第二项中的：与

又因为，由引理3：

因此：

因此：

再将这个不等式代入引理2，即可得到定理1。

得证。

**到这里才开始正式讲如何具体实现**

## 6. Practical Implementation

### 6.1. Approximately Solving the CPO Update

这一块内容假设所提模型是可行的，能找到最优解，若不可行，请移步6.2节。

回顾一下公式(10)

 (10)

For policies with high-dimensional parameter spaces like neural networks, (10) can be impractical to solve directly because of the computational cost.

对于具有高维参数空间的策略，如神经网络，(10)由于计算成本高，直接求解是不切实际的。(这里*θ*是*π*的参数)

因此KL散度用二阶泰勒展开来近似，而其余目标函数与第一个约束用一阶泰勒展开进行线性近似。公式(10)变为：



其中，，。

令，上述变为

 (11)

(令，上式变为：)



(定理2用的是类似于上面的这个式子)

或者如果令，则变为：



Because the **Fisher information matrix (FIM)** *H* is always positive semi-definite (and we will assume it to be positive-definite in what follows) (Fisher信息矩阵H本身是半正定的，这里假设正定)

this optimization problem is convex and, when feasible, can be solved efficiently using duality. (若可行，则为凸优化问题)

(We reserve the case where it is not feasible for the next subsection.) ，(11)的对偶：

 (12)

where .

(这个对偶公式是根据定理2得到的，首先构造拉格朗日函数，对*x*求偏导，把得到的*x*\*代回原来的拉格朗日函数里面，消掉*x*后，即可得到关于对偶变量的目标函数，即把带回拉格朗日函数里面)

This is a convex program in *m*+1 variables; when the number of constraints is small by comparison to the dimension of *θ*, this is much easier to solve than (11).

*m*是cost约束条件的个数，还有一个是信赖域的那个条件，因此一共有*m*+1个约束条件。

意思是如果约束条件的个数(对偶问题)小于*θ*的维度(原问题)，则求解对偶问题(12)比求解原问题(11)更容易。

如果*λ*∗, *ν*∗是对偶问题(12)的最优解，则原问题(11)的解为  (13)

For the special case where there is only one constraint, we give an analytical solution in the supplementary material (Theorem 2) which removes the need for an inner-loop optimization. 意思是如果只有一个安全约束的话，*m*=1时，问题(13)能够得到解析解。实验都采用的是一个约束条件，因此利用了定理2中的解析解的形式。(定理2具体见下一部分10.2的内容)

定理2给出的结论是：，

(注意：定理2和我们真正求的对应关系是：，)

由于近似误差，提出的更新可能不满足(10)中的约束条件;使用[回退线搜索](https://www.cnblogs.com/kailugaji/p/16567557.html" \l "_label3_0_1_0)(backtracking line search)来确保代理约束的满足。高维动作空间，H的逆不好求，用共轭梯度法来近似计算得到。

### 10.2. Proof of Analytical Solution to LQCLP

这一部分内容就是6.1的补充证明部分。

定理2：

**Theorem 2 (Optimizing Linear Objective with Linear and Quadratic Constraints)**. Consider the problem

 (26)

其中. When there is at least one strictly feasible point, the optimal point *x*∗ satisfies ，where *λ*∗, *ν*∗ are defined by ，



其中

令. The value of *λ*∗ satisfies





Proj(*a*, *S*) is the projection of a point *x* on to a set *S*. 

证明：This is a convex optimization problem. When there is at least one strictly feasible point, strong duality holds by Slater’s theorem. We exploit strong duality to solve the problem analytically.

这是一个凸优化问题。当存在至少一个严格可行点时，根据Slater定理，强对偶性成立。我们利用强对偶性来解析求解问题。针对问题(26)：

 (26)

将(26)不等约束转化为拉格朗日乘数法求解，转化为无约束问题：

令



首先，拉格朗日函数对*x*求偏导为0，得到*x*的求解公式，里面当然是含有拉格朗日参数的，因为现在还没对拉格朗日乘子进行求解。



把得到的*x*\*代回原来的拉格朗日函数里面，得到：



这里用到了Fisher信息矩阵的性质：根据定义可知，Fisher信息矩阵是a positive semidefinite symmetric matrix(半正定对称矩阵)，即，因此在合并计算第一个公式时，中间的就可以等于了。



接下来，用到之前定义的那些符号了

还用到矩阵转置的性质



这里需要注意的是*r*是一个数，自然。因为，*g*与*b*与*x*一样，是*n*维列向量，所以求出来的*r*是一个数！



Optimizing single-variable convex quadratic function over . 这个函数表示：

把求出的*v*也带回拉格朗日函数里面，



**分两种情况**，当，需要计算，而时，。

因此，(主要是为了求使目标最大化的*λ*)



其中，

这里又根据不同的*c*的取值，讨论了两种情况下的取值范围。



Notes on interpreting the coefficients in the dual problem:

**第一大块内容是：** (求使目标最大化的*λ*)



这里就要讨论的正负了。

(一)先看，主要判断括号里面是正是负，转化为*r*2-*sq*的正负。这里最后得出*r*2-*sq*≤0。

(这里巧妙地借助了Cauchy-Schwarz inequality，这是人能想到的吗？是提前就凑好的吗？妙啊！)

**Cauchy-Schwarz inequality**

Let  be two sequences of real numbers, then



with equality if and only if the sequences  are proportional, i.e., there is a constant *λ* such that *ak* = *λbk* for each *k*∈{1, 2, . . . , *n*}

在数理统计里面，

再次用到之前定义的那些符号



根据Cauchy-Schwarz inequality可知道，*r*2≤*sq*，也就是说这个不等式是恒成立的。

(二)再看，主要判断括号里面是正是负，转化为的正负。

若(26)约束条件里面，注意*x*是一个列向量，*c*与*s*是一个数，*H*是一个对称矩阵。

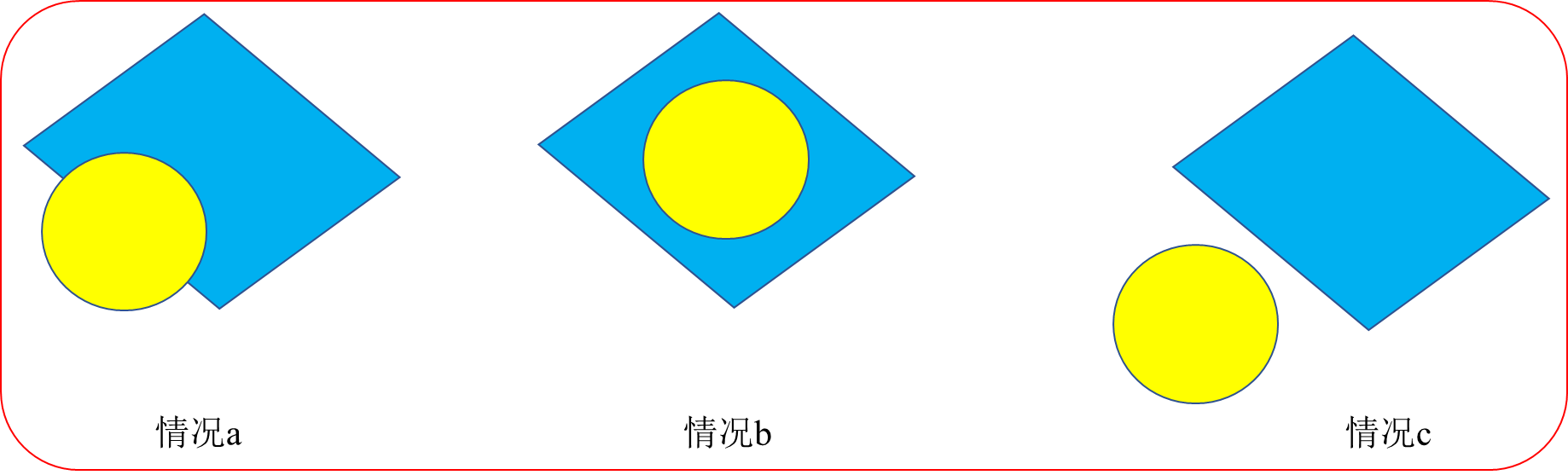


也就是说，如果的话，，这样的判断正负就转化为与*δ*之间判断正负。问题转化为：



判断*x*是否满足公式(26)的第二个约束条件就是。

**分三种情况讨论：**



**a. 如果，**则the plane intersects the trust region，意思是两个约束条件都满足的话，线性约束与信赖域约束的区域之间存在交集。

**b. 如果，**then the quadratic trust region lies entirely within the linear constraint-satisfying halfspace, and we can remove the linear constraint without changing the optimization problem，翻译为：二次信赖域完全位于满足线性约束的半空间内，我们可以在不改变优化问题的情况下去掉线性约束。意思是线性约束条件的覆盖范围大，二次约束的覆盖区域小，但完全包含在线性约束的范围内。因此，线性约束就可以去掉，只要求二次约束条件满足即可。

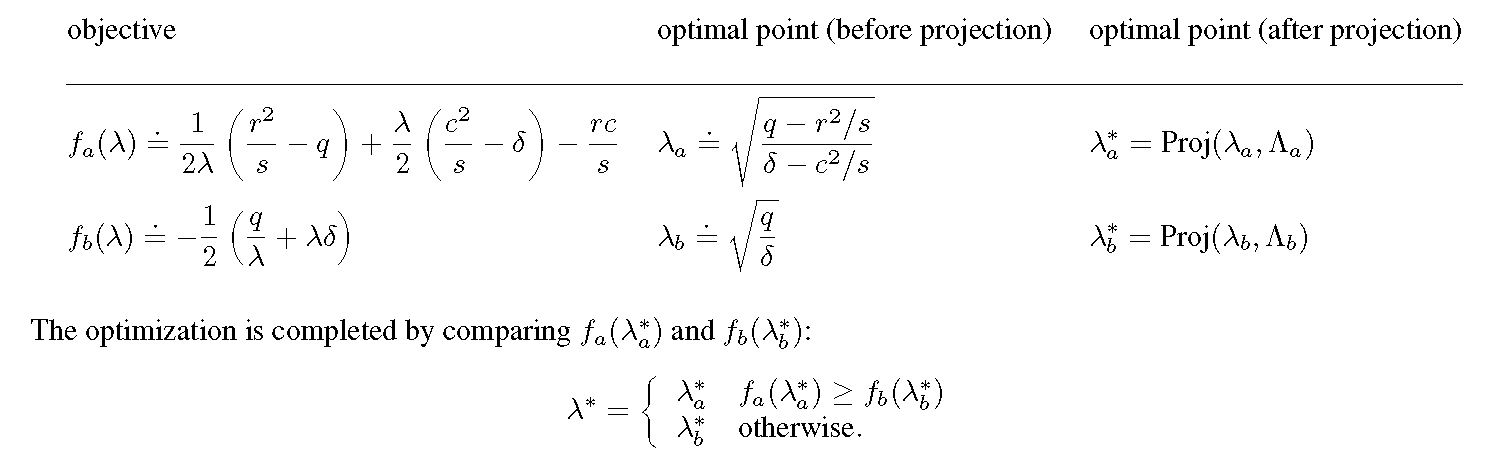
**c. 如果，the problem is infeasible** (the intersection of the quadratic trust region and linear constraint-satisfying halfspace is empty). 翻译为：二次信赖域与线性约束满足半空间的交点为空，也就是说两个约束条件之间完全没有交集，此时*x*无解。Otherwise, we follow the procedure below. (这时不可行，就用到6.2恢复方法了)

**第二大块内容是：**

求使目标值最大的*λ*：带回原来的*f* (*λ*)，则函数的最大值为

(这类只有在*q*与*δ*都大于等于0的情况下才满足)

两大块情况都讨论完了，因此总结：



定理2得证。

### 6.2. Feasibility

 (11)

Sometimes (11) will still be feasible and CPO can automatically recover from its bad step, but **for the infeasible case (第三种(c)情况), a recovery method is necessary**.

这里讨论如果安全约束条件得不到满足，则采用恢复策略，即对于下述问题(仅考虑一个安全约束条件，即*m*=1)，



*c*>0，表示没有满足安全约束，此时。

参考<https://zhuanlan.zhihu.com/p/408925264>，此时将最小化约束当作目标(仅考虑一个约束条件)，问题转化为：



再将其转化为拉格朗日函数：



设已通过共轭梯度法获得了近似搜索方法：，更新公式为：，现在确定步长*β*，



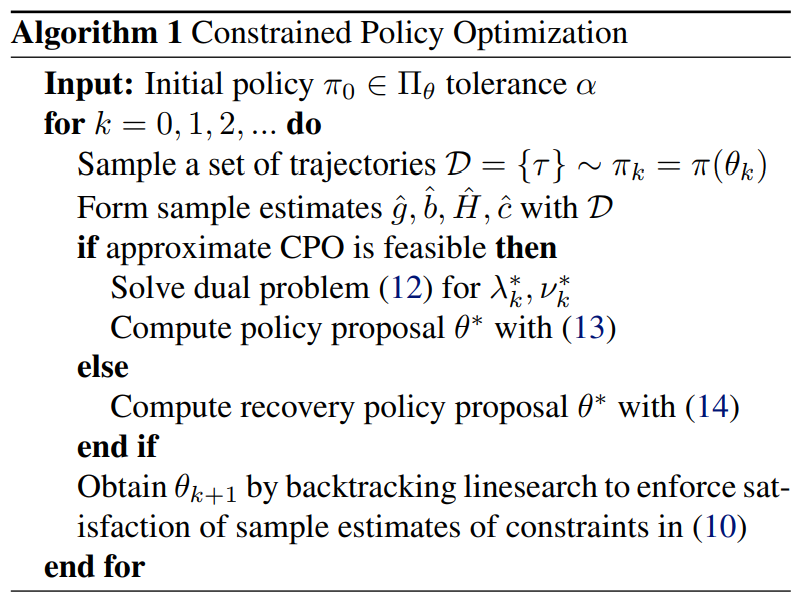
因此，，其中，最大步长为(当*α*=0时)。

In our experiments, where we only have one constraint, **we recover by proposing an update to purely decrease the constraint value:**

**** (14)

As before, this is followed by a line search. This approach is principled in that it uses the limiting search direction as the intersection of the trust region and the constraint region shrinks to zero. We give the pseudocode for our algorithm (for the single-constraint case) as Algorithm 1. 和以前一样，这之后是线搜索。 这种方法的原则性在于，它使用限制性的搜索方向，因为信任区域和约束区域的交点缩小到了零。 我们在算法1中给出了我们的算法（对于单约束的情况）的伪码。

参考[https://www.cnblogs.com/kailugaji/p/16567557.html#\_label3\_0\_1\_0](https://www.cnblogs.com/kailugaji/p/16567557.html" \l "_label3_0_1_0)，回退线搜索取的步长是，因此，迭代公式变为：****，其中，*j*={0,1,…,*K*}。(为啥叫回退法，是因为*α*的取值是从大到小)



### 6.3. Tightening Constraints via Cost Shaping

Because of the various approximations between (3) and our practical algorithm, it is important to build a factor of safety into the algorithm to minimize the chance of constraint violations. 由于（3）和我们的实际算法之间的各种近似，在算法中建立一个安全系数以尽量减少违反约束的机会是很重要的。

To this end, we choose to **constrain upper bounds on the original constraints**, , instead of the original constraints themselves. We do this by cost shaping:



也就是说，实际做的时候约束时比原先的cost的值要稍微大些。(这样不就可能会出现违反实际约束*Ci*的情况吗)

In our experiments, where we have only one constraint, we partition states into safe states and unsafe states, and the agent suffers a safety cost of 1 for being in an unsafe state. We choose  to be **the probability of entering an unsafe state within a fixed time horizon**, according to a learned model that is updated at each iteration. This choice confers the additional benefit of smoothing out sparse constraints.

在我们的实验中，我们只有一个约束条件，我们将状态划分为安全状态和不安全状态，智能体在不安全状态下会承受1的安全成本。我们根据每次迭代更新的学习模型，选择作为在固定时间范围内进入不安全状态的概率。这一选择带来了平滑稀疏约束的额外好处。

## 7. Connections to Prior Work

(说的是如果用原始-对偶法来优化的话，不能保证训练时安全，只能做到部署时安全，而CPO能同时确保训练时安全与部署时安全。这个原始-对偶方法他也用到了信赖域约束，想到之前将修改学习目标这类方法分为拉格朗日法与信赖域法好像并不对，按照之前的划分，这里既属于信赖域法又属于拉格朗日法，但他做不到训练时安全，只能归为拉格朗日法那类中。)

(CPO里面安全约束条件那一项也通过一阶泰勒展开来近似，而原始-对偶算法并未做这样的近似，而是直接用拉格朗日法对对偶变量进行更新这样只关注最终的代价函数，而不是每一步都满足约束)

我们的方法与Chow等人(2015, CVaR)提出的原始-对偶方法有类似的策略更新，但关键的是，我们在计算对偶变量(约束条件的拉格朗日乘子)方面有所不同。在原始-对偶优化(PDO)中，对偶变量是有状态的，并且与原始变量同时学习(Boyd等，2003)。在求解(3)

 (3)

的PDO算法中，对偶变量将根据更新

 (16)

其中，为学习率。在这种方法中，不能保证中间策略满足约束条件，只有收敛策略才能满足约束条件。相比之下，CPO 在每次更新时从头开始计算新的对偶变量，以精确地强制实施约束。

(*v*是关于cost function的对偶变量，用了这个函数是不是意味着*v*值单调不减)

### 10.3.3. Primal-Dual Optimization Implementation

这里就是TRPO+cost约束，其中目标函数线性近似，信赖域那项二次近似，与TRPO一样，cost约束项用原始-对偶问题求解(10.3.3给出的是这样)，不是线性近似求解的(CPO是这样的)。

Our primal-dual implementation is intended to be as close as possible to our CPO implementation. The key difference is that the dual variables for the constraints are stateful, learnable parameters, unlike in CPO where they are solved from scratch at each update.

我们的原始-对偶实现旨在尽可能接近我们的CPO实现。关键的区别在于，约束的对偶变量是有状态的、可学习的参数，这与CPO不同，在CPO中，它们在每次更新时都要从头求解。

The update equations for our PDO implementation are



第一项*θ*的更新和TRPO一模一样，只不过更新梯度的选取不同，用了。第二项是通过原始-对偶求的，也采用迭代更新，这里与CPO的更新不同，CPO直接得到解析解，而不需要通过迭代求解来近似(这是因为CPO本身在安全约束条件那里已经通过一阶泰勒展开来近似了，在此基础上能得到解析解。) (提醒：*v*是关于cost那项约束条件的拉格朗日乘子。)

(回顾公式(13)：，这里还假设的是有*m*个安全约束，因此用的矩阵*B*，实际上只用了一个安全约束条件。)

## 8. Experiments

We consider two tasks, and train multiple different agents (robots) for each task:

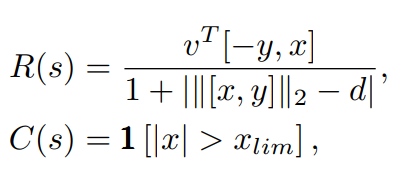
**(1) Circle:** The agent is rewarded for running in a wide circle, but is constrained to stay within a safe region smaller than the radius of the target circle.

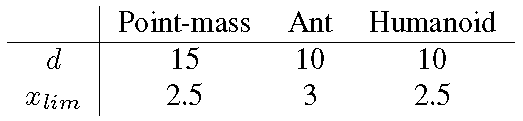
In the Circle task, reward is maximized by moving along the green circle. The agent is not allowed to enter the blue regions, so its optimal constrained path follows the line segments AD and BC. (翻译：在圆圈任务中，玩家通过沿着绿色圆圈移动而获得最大奖励。智能体不允许进入蓝色区域，因此其最优约束路径遵循线段AD和BC。)

Note that there are no physical walls: the agent only interacts with boundaries through the constraint costs. In each of these tasks, we have only one constraint; we refer to it as C and its upper bound from (15) as C+. (翻译：注意，这里没有物理墙:智能体仅通过约束成本与边界进行交互。在这些任务中，我们只有一个约束; 我们称它为C，上限为C+。)

In the Circle environments, the reward and cost functions are: (奖励与代价设置)



where *x*, *y* are the coordinates in the plane, *v* is the velocity, and *d*, *xlim* are environmental parameters. We set these parameters to be (翻译：其中*x*, *y*为平面内坐标，*v*为速度，*d*与 *xlim*为环境参数。我们把这些参数设为)



**(2) Gather:** The agent is rewarded for collecting green apples, and constrained to avoid red bombs.



In Point-Gather, the agent receives a reward of +10 for collecting an apple, and a cost of 1 for collecting a bomb. Two apples and eight bombs spawn on the map at the start of each episode. (翻译：在Point-Gather中，智能体收集苹果将获得+10的奖励，收集炸弹将获得1的成本。每局游戏开始时，地图上会出现两个苹果和八个炸弹。)

In Ant-Gather, the reward and cost structure were the same, except that the agent also receives a reward of -10 for falling over (which results in the episode ending). Eight apples and eight bombs spawn on the map at the start of each episode. (翻译：在Ant-Gather中，奖励和成本结构是相同的，除了智能体也会因为跌倒而获得-10的奖励(这将导致这局游戏结束)。在每一局游戏开始时，地图上会出现8个苹果和8个炸弹。)

Results:

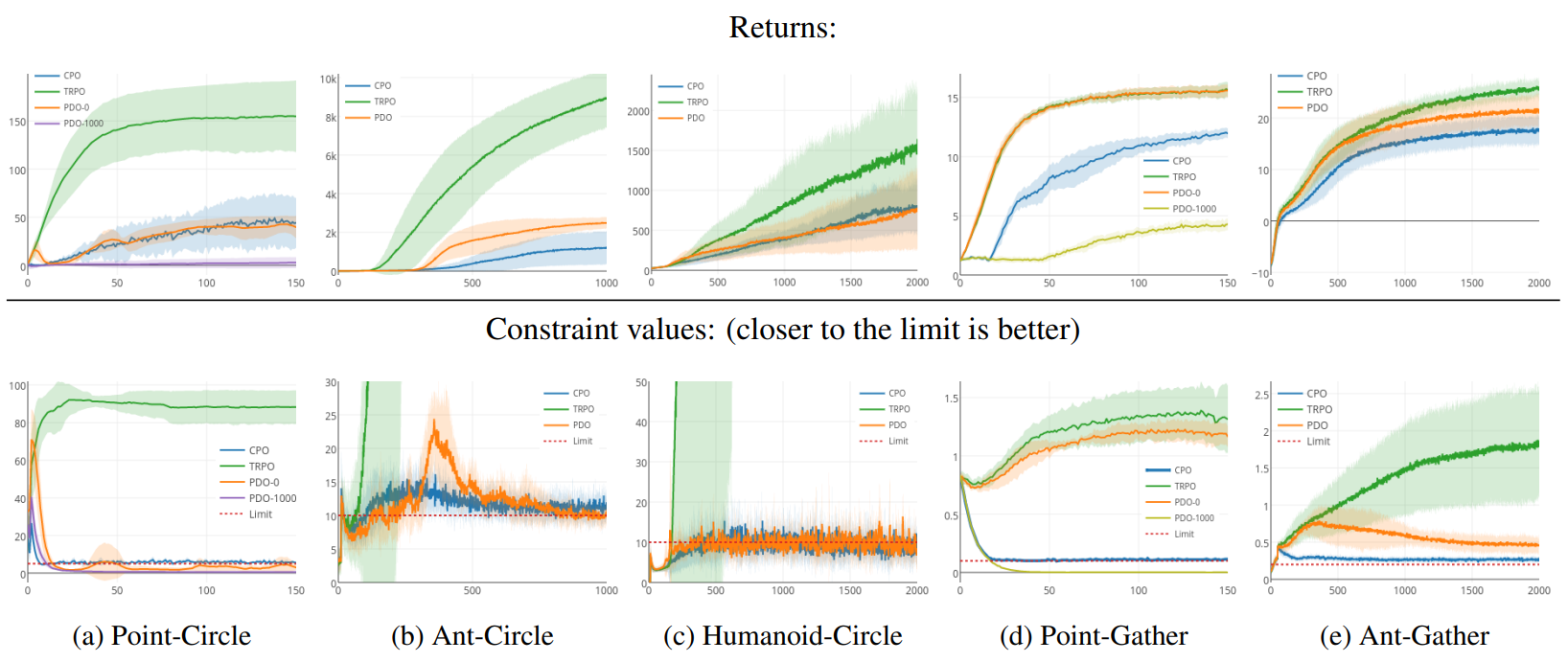


Figure 1. Average performance for CPO, PDO, and TRPO over several seeds (5 in the Point environments, 10 in all others); the x-axis is training iteration. CPO drives the constraint function almost directly to the limit in all experiments, while PDO frequently suffers from over- or under-correction. TRPO is included to verify that optimal unconstrained behaviors are infeasible for the constrained problem. (翻译：CPO、PDO和TRPO在多个种子上的平均性能(Point环境种子数为5，其他环境为10); *x*轴是训练迭代次数。在所有的实验中，CPO几乎直接将约束函数驱动到极限，而PDO经常出现过校正或欠校正。引入TRPO来验证最优无约束行为对于有约束问题是不可行的。)