How not to loose $100.000

Alex Vigneron

Constrained Cross-Entropy Method for Safe Reinforcement-Learning (2018)

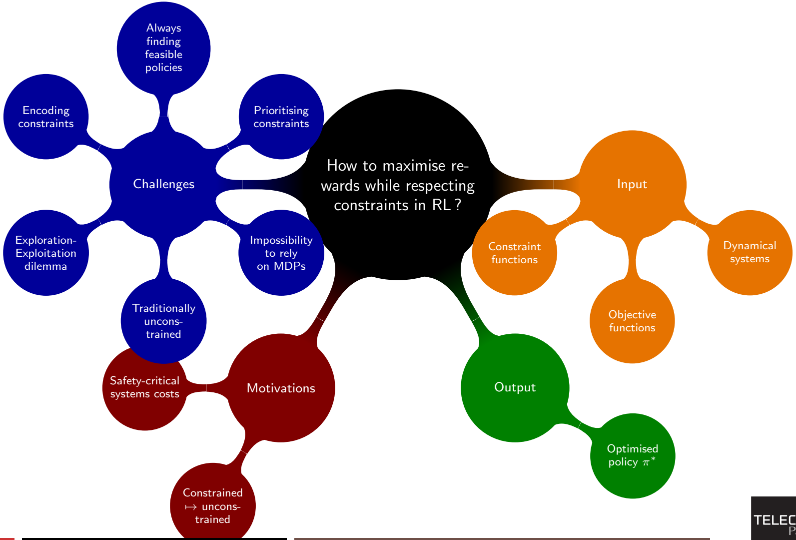
NEURAL INFORMATION PROCESSING SYSTEMS

Min Wen
University of Pennsylvania

Ufuck Topcu
University of Texas Austin

Menu 👍

Problem Overview



Applications : safety-critical systems

Definition

Safety-critical systems are those systems whose failure could result in loss of life, significant property damage, or damage to the environment. (Knight *et al.* 2002)

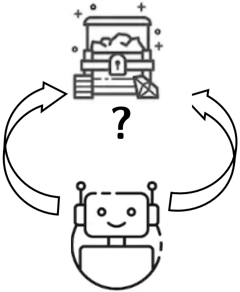
Safety Critical Systems using RL

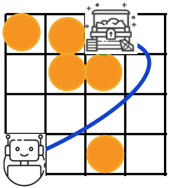
- Cooling systems (Li *et al.* 2018)
- Autonomous-driving vehicles (survey by Kiran *et al.* 2020)
- Industry robots Sarcos' Humanoïd DB used RL to learn a pole-balancing task (Schaal, 1996).
  Today Sarcos' Guardian suits are rented \$ 150.000 a year

Source : Brain controlled robots (Kawato, 2008)

Imagine...

A brave robot having to look for a marvelous treasure kept in a solid chest inside volcanic caves filles with lava pits...

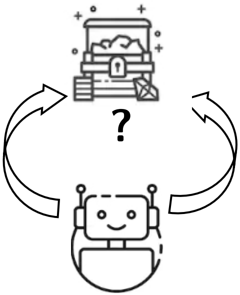




It needs to **get the treasure** while **avoiding lava**. **Where should it tread** ?

- Get treasure = objective
- Avoid lava = constraint
- Where to tread = policy

Reinforcement Learning Paradigm



Policy π

A probability distribution parameterised by a vector θ over actions given states.
$\pi_\theta(s_i) \mapsto \{(a_{i0}, p_{i0})...(a_{in}, p_{in})\}$

Concept

Agent interacts with environment looking for optimal behaviour, receives rewards, modifies its actions to maximise reward

- **Input** : MDP
- **Output** : Policy $\Pi = max(\sum_{r\in\mathcal{R}} \mathcal{R}(s, a))$

Markov Decision Process (Sutton & Barto 2017)

MDP characterised as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$

- $\mathcal{S}$, set of states $\mathcal{S} = \{s_0, s_1...s_n\}$
- $\mathcal{A}$, set of actions $\mathcal{A} = \{a_0, a_1...a_n\}$
- $\mathcal{R}$, reward function $\mathcal{R}(s, a) \mapsto \mathbb{R}$
- $\mathcal{P}$, transition probability function $\mathcal{P}(s, a) \mapsto \mathbb{R}^+$ where for a fixed s, $\sum_a \mathcal{P}(s, a) = 1$
- Initial distribution state

Constraints in traditional RL : an example

Problem

Avoiding lava while obtaining the treasure

Solution

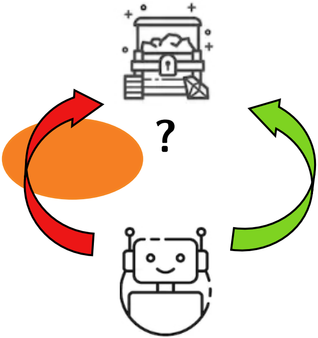
Lava is encoded as a (major) negative reward

Effect

First iterations : agent explores any state (including lava states)
Following iterations : agent learns it *should* not tread lava

Problem

Lava **is** still an option... $\longrightarrow$ no guarantee on lava constraint



Constrained Cross-Entropy

Source : Medium/Udacity Serrano

Entropy (Shannon's)

Entropy is a measure of our lack of information about the microstate of a system (Machta *et al.* 1999)).

Cross-Entropy

Cross-entropy is commonly used to quantify the difference between two probability distribution.

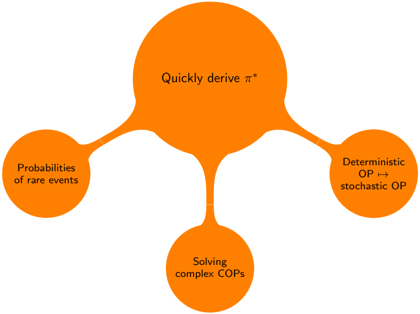
Constrained Cross-Entropy

This functions incorporates the constraints on the system in a generic fashion, irrespective of the form or even the number of the constraints (Niven *et al.*,

CE : an intuition

Idea

Probabilistic approach to maximise possibility of an unlikely event happening.



Algorithm

1. Sample from a distribution of policies
2. Select a set of elite samples and use them to update policy disribution.

Example

Unlikely event =
having a policy which both maximises rewards and respects all constraints.

CCE Algorithm

Input: G, H, upper bound b, set of parameterised policies Π_θ,
NEF family F

Output: Π^*

Initialise *(random, feasible, unfeasible)*

while *stopping rule is not satisfied* **do**

    Sample over parameters θ of policy

    for *each policy* $\pi \in \Pi$ *with* θ *parameters* **do**

        Simulate π_θ

        Compute and store $G(\pi_\theta)$ and $H(\pi_\theta)$

    end

    Sort θ in ascending order of H-value

    Let ψ be the first k elements

    if $H_{\theta\psi} \leq b$ **then**

        Sort θ_π with $H_{\pi\theta} b$ in descending order of G

    end

    Estimate through CE

    Ensure new p.d.f. still $\in$ intitial NEF family

    Update

end

L-function

The L function

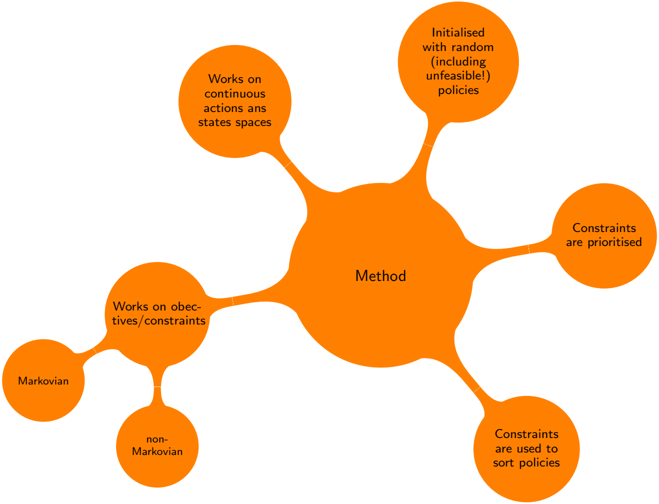
Optimised through learning !

$$L(\mathbf{v}, \rho) = \begin{cases} if\, \xi_H(\rho, \mathbf{v}) > d \longrightarrow \mathbb{E}_{\theta \sim f_v}[\mathcal{G}(\pi_\theta) \;\; \delta(\mathcal{H}(\pi_\theta) \leq \xi_H(\rho, \mathbf{v}))] \\ else \longrightarrow \mathbb{E}_{\theta \sim f_v}[\mathcal{U}(\pi_\theta) \;\; \delta(\mathcal{U}(\pi_\theta) \geq \xi_U(1-\rho, \mathbf{v}))] \end{cases} \tag{1}$$

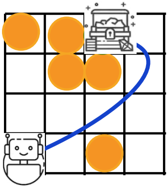
- $\mathbf{v}$, parameter for probability distribution
- $f_{\mathbf{v}}(\theta)$, probability distribution over parameter θ
- d : acceptability threshold
- $\mathcal{G}(\pi_\theta)$, expected gain with regards to policy π with parameters θ
- $\mathcal{H}(\pi_\theta)$ expected cost of constraints with regards to π with parameters θ
- $\xi_F(\rho_i, \mathbf{v})$, p-quantile of the p.d.f. parameterised by $\mathbf{v}$
- $\delta(F)$... maps result to a boolean based
- $\mathcal{U}(\pi_\theta)$: expected reward $\mathcal{G}(\pi_\theta)$ if respects constraints, 0 otherwise

Source : theproducemoms

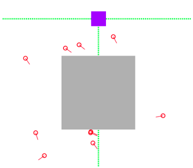
CCE recap 👍

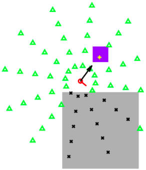

Method
Works on continuous actions ans states spaces
Initialised with random (including unfeasible!) policies
Constraints are prioritised
Constraints are used to sort policies
Works on obectives/constraints
Markovian
non-Markovian

The problem at hand



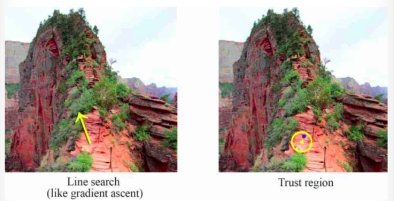
Task : robot navigation with only local sensors
Setting : compact goal region $\mathcal{G}$ and a non-overlapping compact bad region $\mathcal{B}$





- $\mathcal{S}, \mathcal{A}, \Pi$ sets of states, actions and policies
- Objective function $\mathcal{O} : (\mathcal{S} \times \mathcal{A})^n \longrightarrow \mathbb{R}^+$
- Cost function $\mathcal{C} : (\mathcal{S} \times \mathcal{A})^n \longrightarrow \mathbb{R}$
- Deterministic Transition function
- $\mathcal{G} : \Pi \longrightarrow \mathbb{R}^+$ expected value of rewards over Π
- $\mathcal{H} : \Pi \longrightarrow \mathbb{R}$ expected value of constraints over Π

Trust Region Policy Optimisation 👍



Source : MC.AI

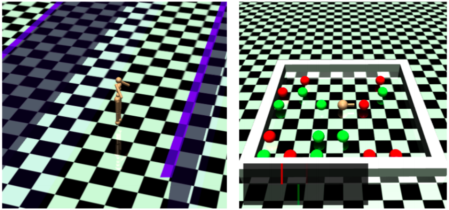
TRPO (Schulman *et al.* 2015)

- Unconstrained RL
- Used as a safeguard against policy gradient disasters (especially in high-curvature functions).
- Restricts policy moves so changes are not too agressive.
- Allows one policy update per iteration.

Constrained Policy Optimisation 👍

CPO (Achiam *et al.* 2017)

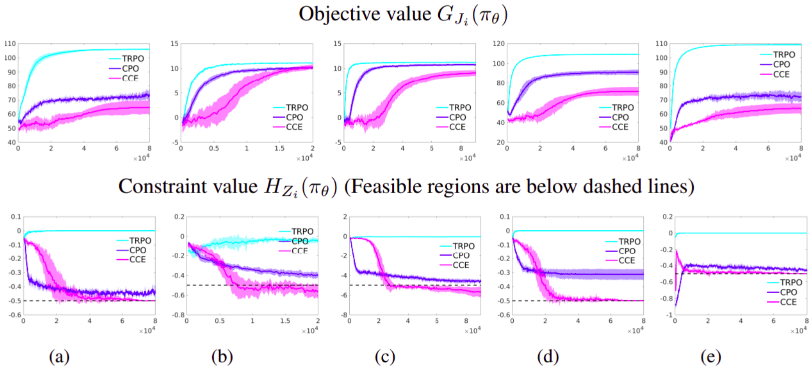
- Constrained RL
- Defines a risk function to ensure agent's security
- Both reward and risk are taken into account at each step
- Promotes safe exploration



(a) Humanoid-Circle (b) Point-Gather

Source : Achiam (2017)

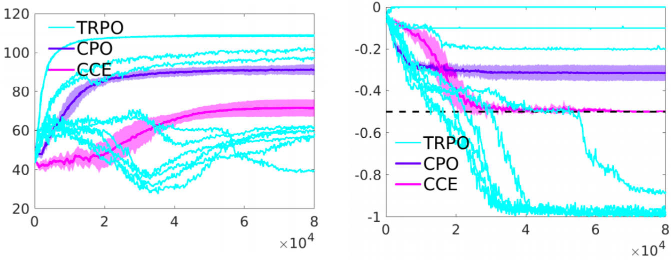
Baseline



x-axis : number of sample trajectories

y-axis : expectation of cumulated reward/punishment for CPO/TRPO and learned policy distribution for CCE

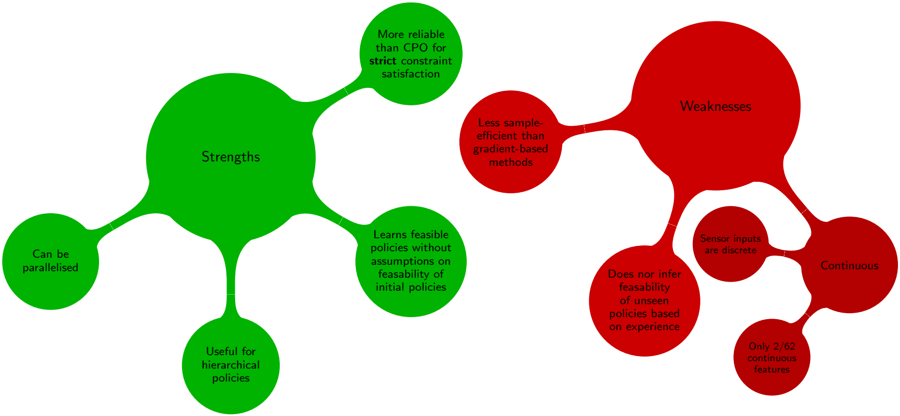
Baseline 👍



TRPO's unconstrained approach is insufficient, merely optimising objectives is not enough to output feasible policy.

CPO needs more samples to find a feasible policy or converges to infeasible policy (especially if constraint is non-Markovian).

Performance analysis



Limitations of the paper

Perspective

Postulate that it is very hard to encode constraints as reward functions, but though hard, this is feasible (Geibel *et al.* 2005, Abe *et al.* 2010, Tong *et al.* 2000)

Experimental setting

Both goal and danger region are compact. This simplification does not take into account the difficulty of proper exploration in sparse settings.

Thank you for your time

In a nutshell

- CCE uses a probabilistic approach to solve safe RL problems.
- CCE can work while initialised with unfeasible policy, which is often not the case in safe RL.
- The key function is $\mathcal{L}(\mathbf{v}, \rho)$ since it takes into account both $\mathcal{G}$ the objective function and $\mathcal{H}$ the constraint function.

Alex Vigneron
alex.vigneron@ip-paris.fr

This presentation together with the bibtex file of references are available on github at
github.com/Narmondil/CCE_for_safe_RL

Constrained Cross-Entropy

Entropy (Shannon's)

Skewed Probability Distribution (unsurprising) : Low entropy. Balanced Probability Distribution (surprising) : High entropy.

$$\mathcal{H}(X) = -\sum_{i=1}^{n} P(x_i) log(P(x_i))$$

X $= (x_1, ..., x_n)(2)$

Cross-Entropy

$$\mathcal{H}(p, q) = -\mathbb{E}_p[log(q)] = \mathcal{H}(p) + \mathcal{D}_{KL}(p||q) \tag{3}$$

Constrained Cross-Entropy

This functions incorporates the constraints on the system in a generic fashion, irrespective of the form or even the number of the constraints (Niven *et al.*, 2003).

TELECOM Paris

Remembering why we're doing CCE 👍

FIGURE – RL-based self-driving car