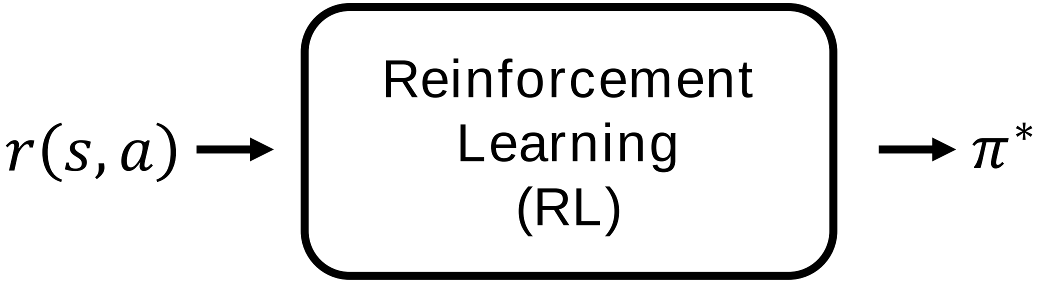
# Learning Soft Constraints from Constrained Expert Demonstrations

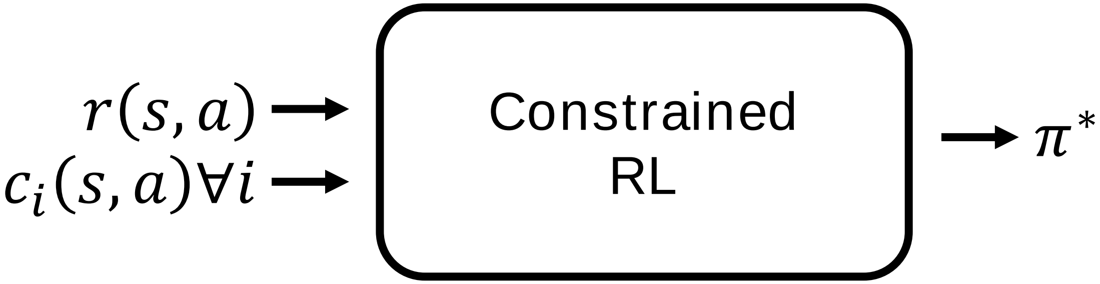
Ashish Gaurav, Kasra Rezaee, Guiliang Liu, Pascal Poupart

ICLR 2023

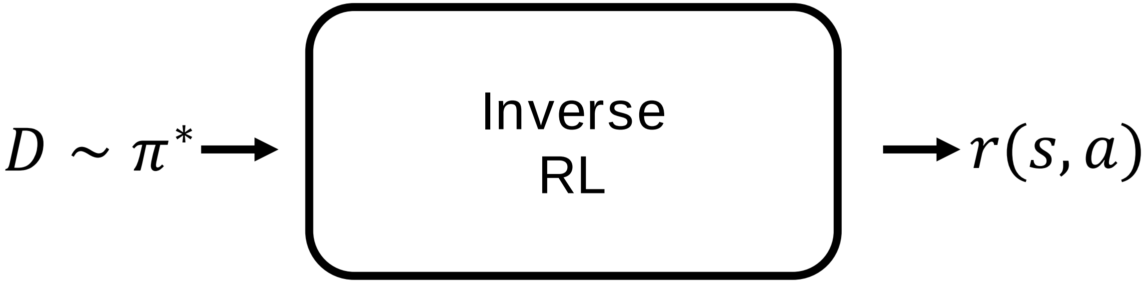
University of Waterloo
Vector Institute
Huawei
香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen



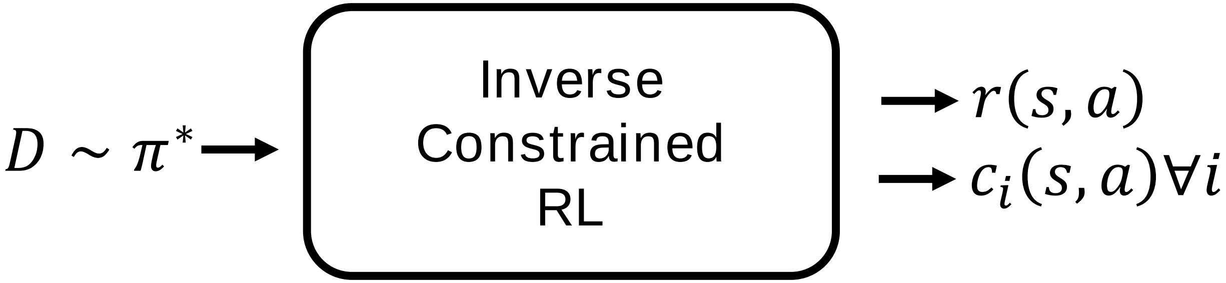

$$\mathbf{RL}(r) \coloneqq \underset{\pi}{\operatorname{argmax}}\, \mathbf{E}_{\pi}[\Sigma_t \gamma^t r(s_t, a_t)]$$



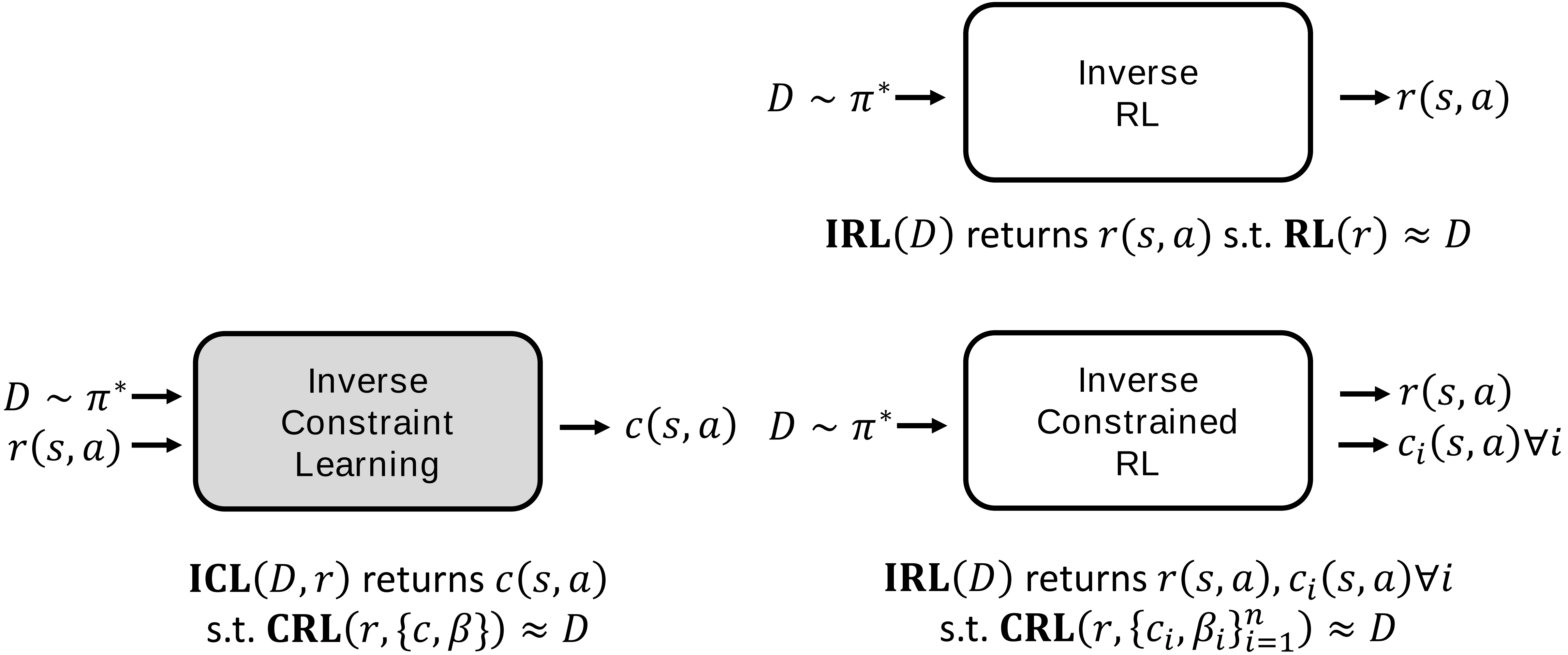

$$\mathbf{CRL}(r, \{c_i, \beta_i\}_{i=1}^{n}) \coloneqq \underset{\pi}{\operatorname{argmax}}\, \mathbf{E}_{\pi}[\Sigma_{\mathrm{t}} \gamma^t r(s_t, a_t)]$$
$$\text{s.t.}\ \mathbf{E}_{\pi}[\Sigma_t \gamma^t c_i(s_t, a_t)] \leq \beta_i\ \forall i$$




$\mathbf{IRL}(D)$ returns $r(s, a)$ s.t. $\mathbf{RL}(r) \approx D$



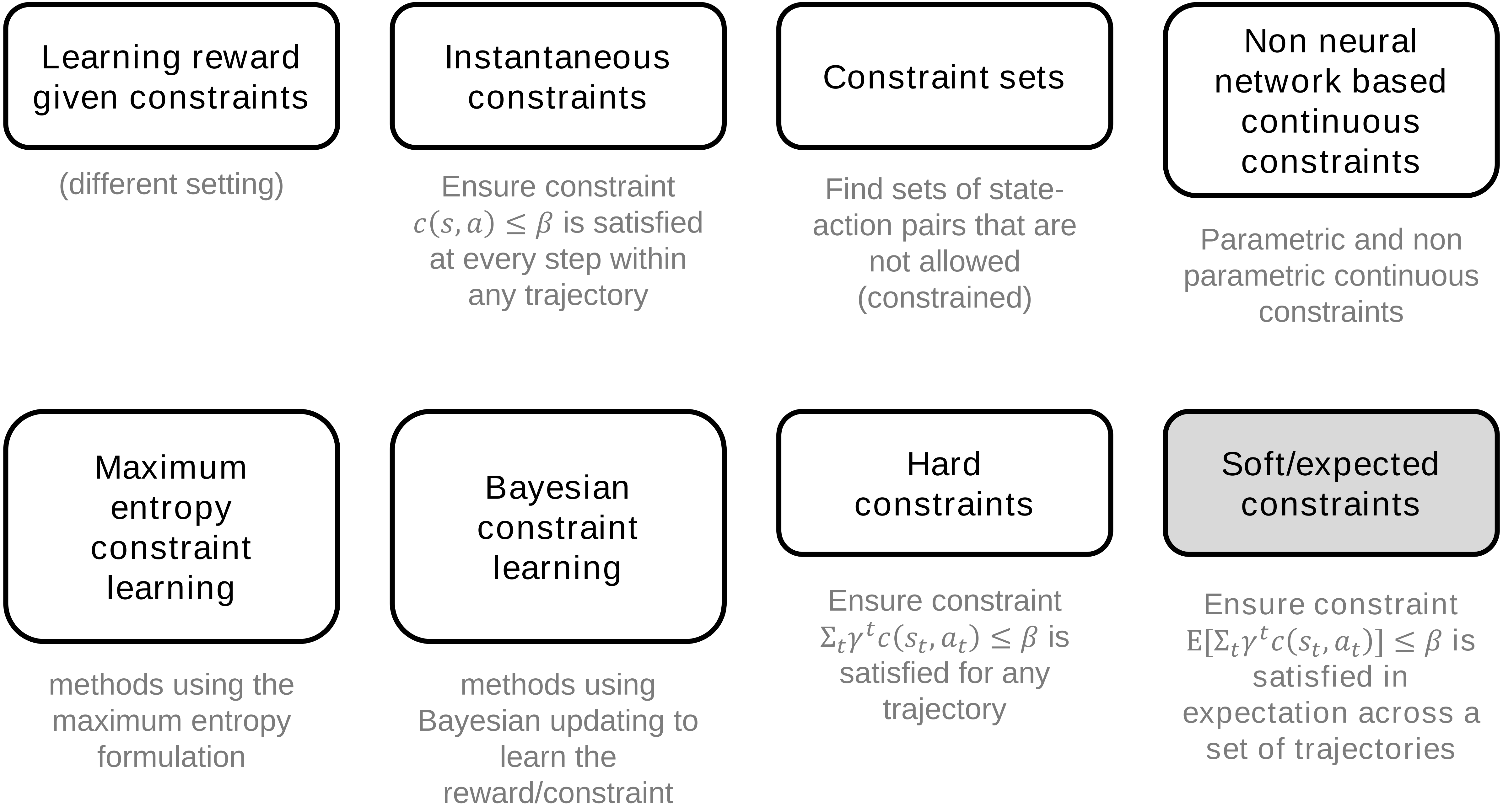

$\mathbf{IRL}(D)$ returns $r(s, a), c_i(s, a) \forall i$
s.t. $\mathbf{CRL}(r, \{c_i, \beta_i\}_{i=1}^{n}) \approx D$

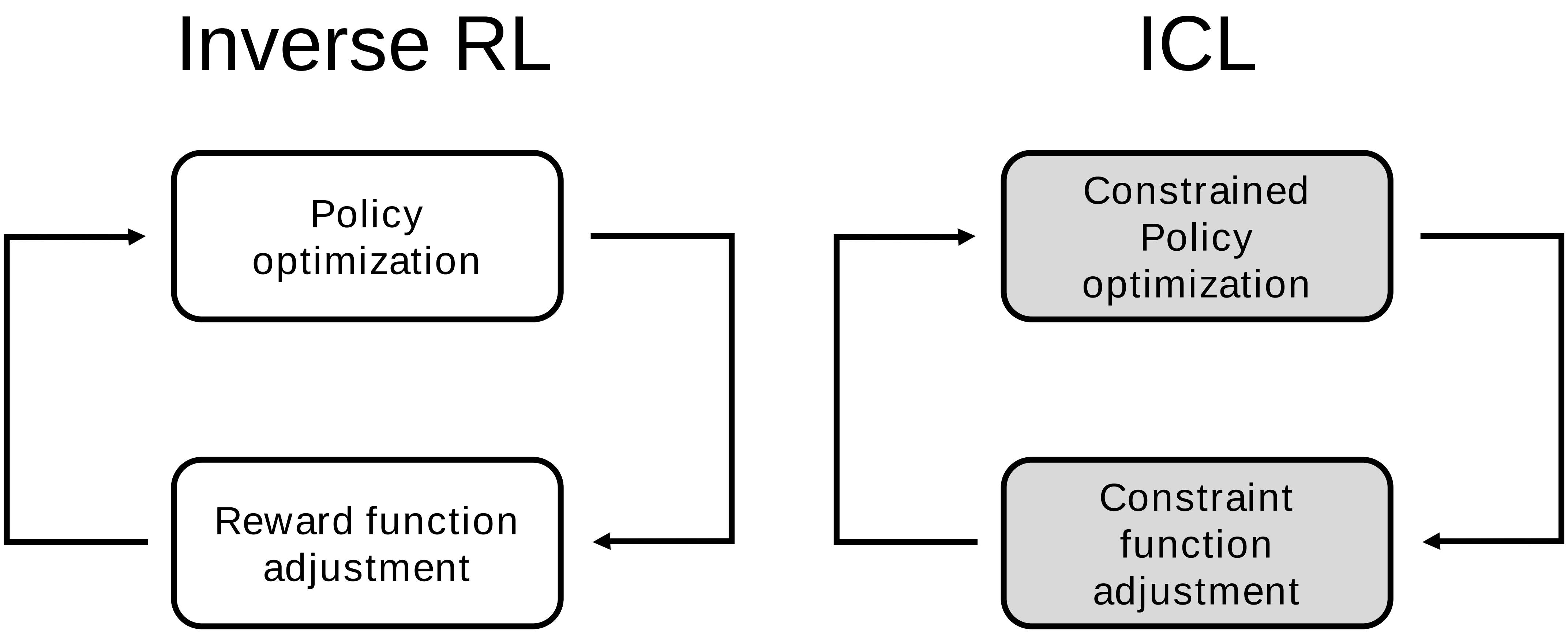

$D \sim \pi^*$
Inverse RL
$r(s,a)$
$\mathbf{IRL}(D)$ returns $r(s,a)$ s.t. $\mathbf{RL}(r) \approx D$
$D \sim \pi^*$
$r(s,a)$
Inverse Constraint Learning
$c(s,a)$
$\mathbf{ICL}(D,r)$ returns $c(s,a)$ s.t. $\mathbf{CRL}(r,\{c,\beta\}) \approx D$
$D \sim \pi^*$
Inverse Constrained RL
$r(s,a)$
$c_i(s,a) \forall i$
$\mathbf{IRL}(D)$ returns $r(s,a), c_i(s,a) \forall i$ s.t. $\mathbf{CRL}(r,\{c_i,\beta_i\}_{i=1}^{n}) \approx D$

- Novel formulation that can learn an arbitrary constraint function from optimal constrained demonstrations (ICL)
- First method that can learn a soft/expected constraint
- Experiments on synthetic environments, robotics environments and with real world driving scenarios

"do not use more than 3 units of energy"

| Hard constraints | Soft constraints |
| --- | --- |
| satisfy for any individual trajectory | satisfy on average across a set of trajectories |


Learning reward given constraints
(different setting)
Instantaneous constraints
Ensure constraint $c(s, a) \leq \beta$ is satisfied at every step within any trajectory
Constraint sets
Find sets of state-action pairs that are not allowed (constrained)
Non neural network based continuous constraints
Parametric and non parametric continuous constraints
Maximum entropy constraint learning
methods using the maximum entropy formulation
Bayesian constraint learning
methods using Bayesian updating to learn the reward/constraint
Hard constraints
Ensure constraint $\Sigma_t \gamma^t c(s_t, a_t) \leq \beta$ is satisfied for any trajectory
Soft/expected constraints
Ensure constraint $\mathrm{E}[\Sigma_t \gamma^t c(s_t, a_t)] \leq \beta$ is satisfied in expectation across a set of trajectories


Inverse RL
Policy optimization
Reward function adjustment
ICL
Constrained Policy optimization
Constraint function adjustment

Constrained Policy Optimization:

$$\pi^* \coloneqq \mathrm{argmax}_\pi\, \mathbf{J}^\pi(r)\ \mathrm{s.t.}\ \mathbf{J}^\pi(c) \leq \beta$$

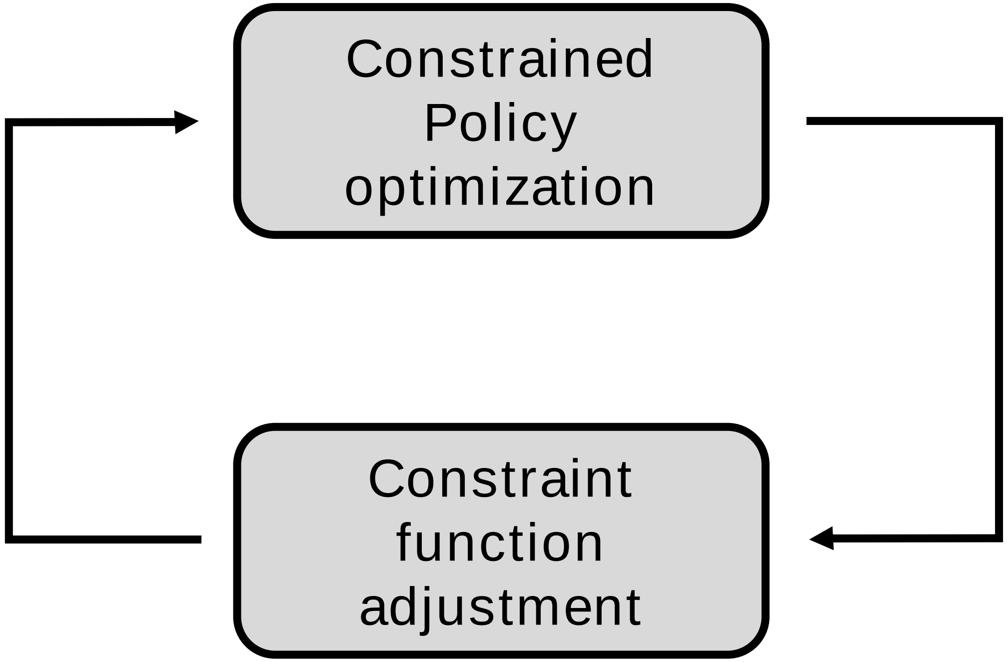
Add to set of optimal policies:

$$\Pi \leftarrow \Pi \cup \{\pi^*\}$$

Constraint function adjustment:

$$c^* \coloneqq \mathrm{argmax}_c \min_{\pi \in \Pi} \mathbf{J}^\pi(c)\ \ \mathrm{s.t.}\ \mathbf{J}^{\pi_E}(c) \leq \beta$$

# ICL




Define $\mathbf{J}^\pi(r) \coloneqq \mathbf{E}_\pi[\Sigma_{\mathrm{t}} \gamma^t r(s_t, a_t)]$

*(Theorem 1) Alternating between these optimization procedures converges in the sense that eventually $\pi^*$ becomes $\pi_E$*

Constrained Policy Optimization:

$$\pi^* \coloneqq \text{argmax}_\pi \, \mathbf{J}^\pi(r) \text{ s.t. } \mathbf{J}^\pi(c) \leq \beta$$

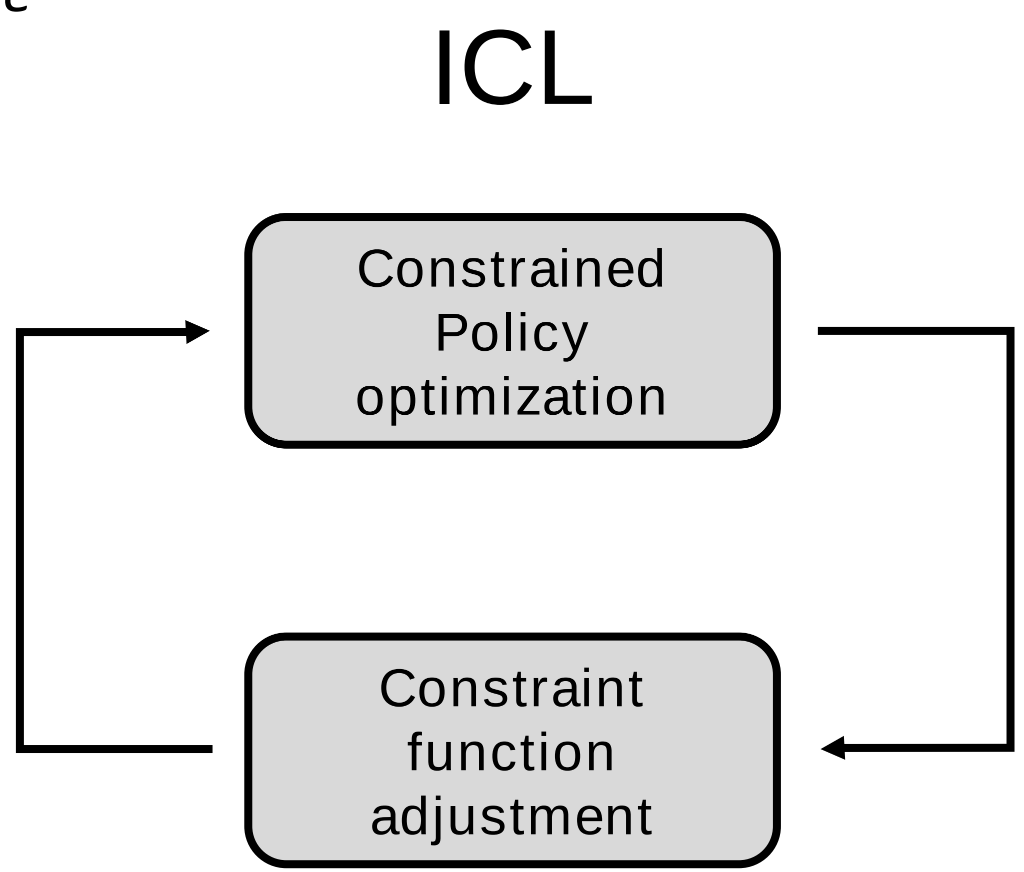
Add to set of optimal policies:

$$\Pi \leftarrow \Pi \cup \{\pi^*\}$$

Constraint function adjustment:

$$c^* \coloneqq \text{argmax}_c \min_{\pi \in \Pi} \mathbf{J}^\pi(c) \text{ s.t. } \mathbf{J}^{\pi_E}(c) \leq \beta$$

**Difficult to optimize!**




Define $\mathbf{J}^\pi(r) \coloneqq \mathbf{E}_\pi[\Sigma_t \gamma^t r(s_t, a_t)]$

*(Theorem 1) Alternating between these optimization procedures converges in the sense that eventually $\pi^*$ becomes $\pi_E$*

Constrained Policy Optimization:

$$\pi^* := \text{argmax}_\pi\, \mathbf{J}^\pi(r) \text{ s.t. } \mathbf{J}^\pi(c) \le \beta$$

Add to set of optimal policies:

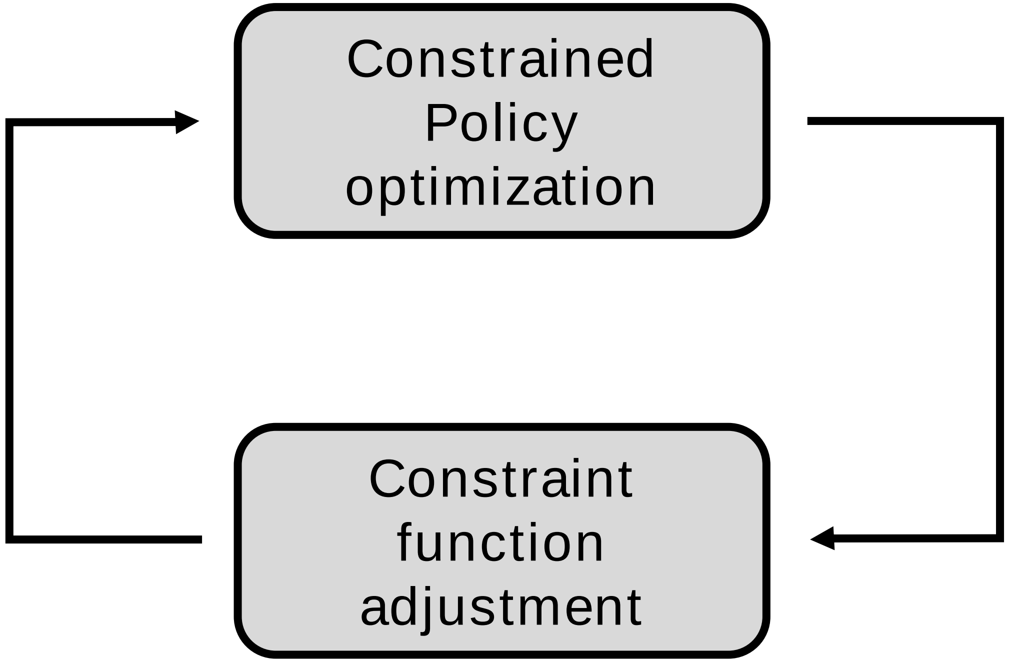
$$\Pi \leftarrow \Pi \cup \{\pi^*\}$$

Constraint function adjustment:

$$c^* := \text{argmax}_c \mathbf{J}^{\pi_{\text{mix}}}(c) \text{ s.t. } \mathbf{J}^{\pi_E}(c) \le \beta$$

**Simpler to optimize**

# ICL




Define $\mathbf{J}^\pi(r) := \mathbf{E}_\pi[\Sigma_t \gamma^t r(s_t, a_t)]$

**Algorithm 1** INVERSE-CONSTRAINT-LEARNING

**hyper-parameters:** number of ICL iterations $n$, tolerance $\epsilon$
**input:** expert dataset $\mathcal{D} = \{\tau\}_{\tau \in \mathcal{D}} := \{\{(s_t, a_t)\}_{1 \le t \le |\tau|}\}_{\tau \in \mathcal{D}}$

**initialize** normalizing flow $f$
**optimize** likelihood of $f$ on expert state action data: $\max_f \text{SUM}_{(s,a)\in\tau,\tau\in\mathcal{D}}(\log p_f(s,a))$
**initialize** constraint function $c$ (parameterized by $\phi$)
**for** $1 \le i \le n$ **do**
  **initialize** policy $\pi_i$ (parameterized by $\boldsymbol{\theta}_i$)
  **perform** $\pi_i := \text{CONSTRAINED-RL}(\pi_i, c)$
  **perform** $c := \text{CONSTRAINT-ADJUSTMENT}(\pi_{1:i}, c, \mathcal{D}, f)$
  **break** if $\text{NORMALIZED-ACCRUAL-DISSIMILARITY}(\mathcal{D}, \mathcal{D}_{\pi_i}) \le \epsilon$
  ▷ *See Section 5 for normalized accrual dissimilarity metric*
**end for**
**output:** learned constraint function $c$ (neural network with sigmoid output),
learned most recent policy $\pi_i$

**Algorithm 2** CONSTRAINED-RL

**hyper-parameters:** learning rates $\eta_1, \eta_2$, constraint threshold $\beta$, constrained RL epochs $m$
**input:** policy $\pi_i$ parameterized by $\boldsymbol{\theta}_i$, constraint function $c$

**for** $1 \le j \le m$ **do**
  **correct** $\pi_i$ to be feasible: (iterate) $\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i - \eta_1 \nabla_{\boldsymbol{\theta}_i} \text{RELU}(J_\mu^{\pi_i}(c) - \beta)$
  **optimize** expected discounted reward: $\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i - \eta_2 \nabla_{\boldsymbol{\theta}_i} \text{PPO-LOSS}(\pi_i)$
  ▷ *Proximal Policy Optimization* (Schulman et al., 2017)
**end for**
**output:** learned policy $\pi_i$

**Algorithm 3** CONSTRAINT-ADJUSTMENT

**hyper-parameters:** learning rate $\eta_3$, penalty wt. $\lambda$, constraint threshold $\beta$,
constraint adjustment epochs $e$
**input:** policies $\pi_{1:i}$, constraint function $c$, trained normalizing flow $f$,
expert dataset $\mathcal{D} = \{\tau\}_{\tau \in \mathcal{D}} := \{\{(s_t, a_t)\}_{1 \le t \le |\tau|}\}_{\tau \in \mathcal{D}}$
**given:** $c^\gamma(\tau) := \text{SUM}_{1 \le t \le |\tau|}(\gamma^{t-1} c(s_t, a_t))$,
$\text{SAMPLE}_\tau(\Pi, p)$ which generates $|\mathcal{D}|$ trajectories $\tau = \{(s_t, a_t)\}_{1 \le t \le |\tau|}$, where for each
$\tau$, we choose $\pi \in \Pi$ with prob. $p(\pi)$, then, $s_1 \sim \mu(\cdot), a_t \sim \pi(\cdot|s_t), s_{t+1} \sim p(\cdot|s_t, a_t)$

$\mu_E := \text{MEAN}_{(s,a)\in\tau,\tau\in\mathcal{D}}(-\log p_f(s,a))$
$\sigma_E := \text{STD-DEV}_{(s,a)\in\tau,\tau\in\mathcal{D}}(-\log p_f(s,a))$
$w(\tau) := \text{MEAN}_{(s,a)\in\tau}(\mathbf{1}(-\log p_f(s,a) > \mu_E + \sigma_E))$ ▷ *trajectory dissimilarity w.r.t. expert*
**construct** policy dataset $\mathcal{D}_{\pi_i} = \text{SAMPLE}_\tau(\Pi = \{\pi_i\}, p = \{1\})$
$\tilde{w}_i := \text{MEAN}_{\tau \in \mathcal{D}_{\pi_i}} w(\tau)$ ▷ *unnormalized policy weights*
**construct** agent dataset $\mathcal{D}_A = \text{SAMPLE}_\tau(\Pi = \pi_{1:i}, p(\pi_i) \propto \tilde{w}_i)$ ▷ *policy reweighting*
**for** $1 \le j \le e$ **do** ▷ *constraint function adjustment*
  **compute** $J_\mu^{\pi_{mix}}(c) := \text{SUM}_{\tau\in\mathcal{D}_A} \frac{w(\tau) c^\gamma(\tau)}{\text{SUM}_{\tau\in\mathcal{D}_A} w(\tau)}$ ▷ *trajectory reweighting*
  **compute** $J_\mu^{\pi_E}(c) := \text{MEAN}_{\tau\in\mathcal{D}}(c^\gamma(\tau))$
  **compute** soft loss $L_{\text{soft}}(c) := -J_\mu^{\pi_{mix}}(c) + \lambda \text{RELU}(J_\mu^{\pi_E}(c) - \beta)$
  **optimize** constraint function $c$: $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \eta_3 \nabla_{\boldsymbol{\phi}} L_{\text{soft}}(c)$
**end for**
**output:** constraint function $c$

Loop alternates between Constrained RL and Constraint adjustment

**Algorithm 1** INVERSE-CONSTRAINT-LEARNING

**hyper-parameters:** number of ICL iterations $n$, tolerance $\epsilon$
**input:** expert dataset $\mathcal{D} = \{\tau\}_{\tau \in \mathcal{D}} := \{\{(s_t, a_t)\}_{1 \le t \le |\tau|}\}_{\tau \in \mathcal{D}}$
**initialize** normalizing flow $f$
**optimize** likelihood of $f$ on expert state action data: $\max_f \text{SUM}_{(s,a) \in \tau, \tau \in \mathcal{D}} (\log p_f(s, a))$
**initialize** constraint function $c$ (parameterized by $\phi$)
**for** $1 \le i \le n$ **do**
  **initialize** policy $\pi_i$ (parameterized by $\boldsymbol{\theta}_i$)
  **perform** $\pi_i :=$ CONSTRAINED-RL$(\pi_i, c)$
  **perform** $c :=$ CONSTRAINT-ADJUSTMENT$(\pi_{1:i}, c, \mathcal{D}, f)$
  **break** if NORMALIZED-ACCRUAL-DISSIMILARITY$(\mathcal{D}, \mathcal{D}_{\pi_i}) \le \epsilon$
  ▷ *See Section 5 for normalized accrual dissimilarity metric*
**end for**
**output:** learned constraint function $c$ (neural network with sigmoid output),
learned most recent policy $\pi_i$

**Algorithm 2** CONSTRAINED-RL

**hyper-parameters:** learning rates $\eta_1, \eta_2$, constraint threshold $\beta$, constrained RL epochs $m$
**input:** policy $\pi_i$ parameterized by $\boldsymbol{\theta}_i$, constraint function $c$
**for** $1 \le j \le m$ **do**
  **correct** $\pi_i$ to be feasible: (iterate) $\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i - \eta_1 \nabla_{\boldsymbol{\theta}_i} \text{ReLU}(J_\mu^{\pi_i}(c) - \beta)$
  **optimize** expected discounted reward: $\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i - \eta_2 \nabla_{\boldsymbol{\theta}_i} \text{PPO-LOSS}(\pi_i)$
  ▷ *Proximal Policy Optimization* (Schulman et al., 2017)
**end for**
**output:** learned policy $\pi_i$

**Algorithm 3** CONSTRAINT-ADJUSTMENT

**hyper-parameters:** learning rate $\eta_3$, penalty wt. $\lambda$, constraint threshold $\beta$,
constraint adjustment epochs $e$
**input:** policies $\pi_{1:i}$, constraint function $c$, trained normalizing flow $f$,
expert dataset $\mathcal{D} = \{\tau\}_{\tau \in \mathcal{D}} := \{\{(s_t, a_t)\}_{1 \le t \le |\tau|}\}_{\tau \in \mathcal{D}}$
**given:** $c^\gamma(\tau) := \text{SUM}_{1 \le t \le |\tau|}(\gamma^{t-1} c(s_t, a_t))$,
$\text{SAMPLE}_\tau(\Pi, p)$ which generates $|\mathcal{D}|$ trajectories $\tau = \{(s_t, a_t)\}_{1 \le t \le |\tau|}$, where for each
$\tau$, we choose $\pi \in \Pi$ with prob. $p(\pi)$, then, $s_1 \sim \mu(\cdot), a_t \sim \pi(\cdot|s_t), s_{t+1} \sim p(\cdot|s_t, a_t)$
$\mu_E := \text{MEAN}_{(s,a) \in \tau, \tau \in \mathcal{D}}(-\log p_f(s, a))$
$\sigma_E := \text{STD-DEV}_{(s,a) \in \tau, \tau \in \mathcal{D}}(-\log p_f(s, a))$
$w(\tau) := \text{MEAN}_{(s,a) \in \tau}(\mathbf{1}(-\log p_f(s, a) > \mu_E + \sigma_E))$ ▷ *trajectory dissimilarity w.r.t. expert*
**construct** policy dataset $\mathcal{D}_{\pi_i} = \text{SAMPLE}_\tau(\Pi = \{\pi_i\}, p = \{1\})$
$\tilde{w}_i := \text{MEAN}_{\tau \in \mathcal{D}_{\pi_i}} w(\tau)$ ▷ *unnormalized policy weights*
**construct** agent dataset $\mathcal{D}_A = \text{SAMPLE}_\tau(\Pi = \pi_{1:i}, p(\pi_i) \propto \tilde{w}_i)$ ▷ *policy reweighting*
**for** $1 \le j \le e$ **do** ▷ *constraint function adjustment*
  **compute** $J_\mu^{\pi_{mix}}(c) := \text{SUM}_{\tau \in \mathcal{D}_A} \frac{w(\tau) c^\gamma(\tau)}{\text{SUM}_{\tau \in \mathcal{D}_A} w(\tau)}$ ▷ *trajectory reweighting*
  **compute** $J_\mu^{\pi_E}(c) := \text{MEAN}_{\tau \in \mathcal{D}}(c^\gamma(\tau))$
  **compute** soft loss $L_{\text{soft}}(c) := -J_\mu^{\pi_{mix}}(c) + \lambda \text{ReLU}(J_\mu^{\pi_E}(c) - \beta)$
  **optimize** constraint function $c$: $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \eta_3 \nabla_{\boldsymbol{\phi}} L_{\text{soft}}(c)$
**end for**
**output:** constraint function $c$

Loop alternates between Constrained RL and Constraint adjustment

**Algorithm 1** INVERSE-CONSTRAINT-LEARNING

**hyper-parameters:** number of ICL iterations $n$, tolerance $\epsilon$
**input:** expert dataset $\mathcal{D} = \{\tau\}_{\tau \in \mathcal{D}} := \{\{(s_t, a_t)\}_{1 \leq t \leq |\tau|}\}_{\tau \in \mathcal{D}}$
**initialize** normalizing flow $f$
**optimize** likelihood of $f$ on expert state action data: $\max_f \text{SUM}_{(s,a) \in \tau, \tau \in \mathcal{D}} (\log p_f(s, a))$
**initialize** constraint function $c$ (parameterized by $\phi$)
**for** $1 \leq i \leq n$ **do**
  **initialize** policy $\pi_i$ (parameterized by $\boldsymbol{\theta}_i$)
  **perform** $\pi_i :=$ CONSTRAINED-RL$(\pi_i, c)$
  **perform** $c :=$ CONSTRAINT-ADJUSTMENT$(\pi_{1:i}, c, \mathcal{D}, f)$
  **break** if NORMALIZED-ACCRUAL-DISSIMILARITY$(\mathcal{D}, \mathcal{D}_{\pi_i}) \leq \epsilon$
  ▷ *See Section 5 for normalized accrual dissimilarity metric*
**end for**
**output:** learned constraint function $c$ (neural network with sigmoid output), learned most recent policy $\pi_i$

Constrained RL algorithm can be replaced with any equivalent algorithm!

**Algorithm 2** CONSTRAINED-RL

**hyper-parameters:** learning rates $\eta_1, \eta_2$, constraint threshold $\beta$, constrained RL epochs $m$
**input:** policy $\pi_i$ parameterized by $\boldsymbol{\theta}_i$, constraint function $c$
**for** $1 \leq j \leq m$ **do**
  **correct** $\pi_i$ to be feasible: (iterate) $\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i - \eta_1 \nabla_{\boldsymbol{\theta}_i} \text{RELU}(J_\mu^{\pi_i}(c) - \beta)$
  **optimize** expected discounted reward: $\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i - \eta_2 \nabla_{\boldsymbol{\theta}_i} \text{PPO-LOSS}(\pi_i)$
  ▷ *Proximal Policy Optimization* (Schulman et al. 2017)
**end for**
**output:** learned policy $\pi_i$

**Algorithm 3** CONSTRAINT-ADJUSTMENT

**hyper-parameters:** learning rate $\eta_3$, penalty wt. $\lambda$, constraint threshold $\beta$, constraint adjustment epochs $e$
**input:** policies $\pi_{1:i}$, constraint function $c$, trained normalizing flow $f$, expert dataset $\mathcal{D} = \{\tau\}_{\tau \in \mathcal{D}} := \{\{(s_t, a_t)\}_{1 \leq t \leq |\tau|}\}_{\tau \in \mathcal{D}}$
**given:** $c^\gamma(\tau) := \text{SUM}_{1 \leq t \leq |\tau|} (\gamma^{t-1} c(s_t, a_t))$,
SAMPLE$_\tau(\Pi, p)$ which generates $|\mathcal{D}|$ trajectories $\tau = \{(s_t, a_t)\}_{1 \leq t \leq |\tau|}$, where for each $\tau$, we choose $\pi \in \Pi$ with prob. $p(\pi)$, then, $s_1 \sim \mu(\cdot), a_t \sim \pi(\cdot|s_t), s_{t+1} \sim p(\cdot|s_t, a_t)$
$\mu_E := \text{MEAN}_{(s,a) \in \tau, \tau \in \mathcal{D}} (-\log p_f(s, a))$
$\sigma_E := \text{STD-DEV}_{(s,a) \in \tau, \tau \in \mathcal{D}} (-\log p_f(s, a))$
$w(\tau) := \text{MEAN}_{(s,a) \in \tau} (\mathbf{1}(-\log p_f(s, a) > \mu_E + \sigma_E))$ ▷ *trajectory dissimilarity w.r.t. expert*
**construct** policy dataset $\mathcal{D}_{\pi_i} = \text{SAMPLE}_\tau(\Pi = \{\pi_i\}, p = \{1\})$
$\tilde{w}_i := \text{MEAN}_{\tau \in \mathcal{D}_{\pi_i}} w(\tau)$ ▷ *unnormalized policy weights*
**construct** agent dataset $\mathcal{D}_A = \text{SAMPLE}_\tau(\Pi = \pi_{1:i}, p(\pi_i) \propto \tilde{w}_i)$ ▷ *policy reweighting*
**for** $1 \leq j \leq e$ **do** ▷ *constraint function adjustment*
  **compute** $J_\mu^{\pi_{mix}}(c) := \text{SUM}_{\tau \in \mathcal{D}_A} \frac{w(\tau) c^\gamma(\tau)}{\text{SUM}_{\tau \in \mathcal{D}_A} w(\tau)}$ ▷ *trajectory reweighting*
  **compute** $J_\mu^{\pi_E}(c) := \text{MEAN}_{\tau \in \mathcal{D}} (c^\gamma(\tau))$
  **compute** soft loss $L_{\text{soft}}(c) := -J_\mu^{\pi_{mix}}(c) + \lambda \text{RELU}(J_\mu^{\pi_E}(c) - \beta)$
  **optimize** constraint function $c$: $\phi \leftarrow \phi - \eta_3 \nabla_\phi L_{\text{soft}}(c)$
**end for**
**output:** constraint function $c$

Constrained optimization using the penalty method:

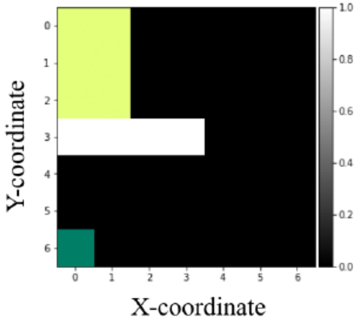
$$\min_y f(y) \ \text{s.t.} \ g(y) \leq 0$$

becomes

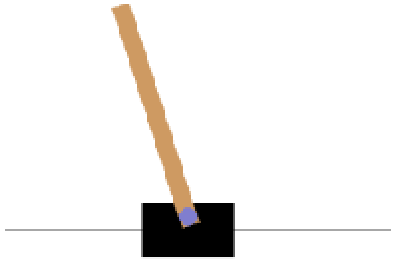
$$\min_y L(y) \coloneqq f(y) + \lambda \, \text{ReLU}\big(g(y)\big)$$

# Experiments

Synthetic

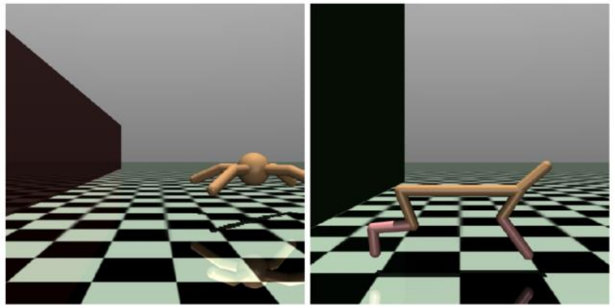



Gridworld



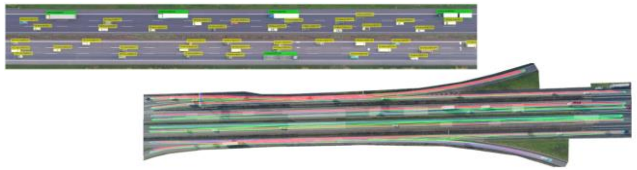
CartPole

Robotics



Mujoco

Real world



Highway driving

## Baselines

- GAIL Constraint: Ho & Ermon (2016)
- ICRL: Malik et al. (2021)

## Metrics

- Constraint MSE (recovered vs true)
- Similarity between policies (learned vs expert)

# Experiments

Synthetic

Robotics

Real world

## Constraint MSE (recovered vs true)

| Algorithm↓, Environment→ | Gridworld (A) | Gridworld (B) | CartPole (MR) | CartPole (Mid) | Ant-Constrained | HalfCheetah-Constrained |
|---|---|---|---|---|---|---|
| GAIL-Constraint | 0.31 ± 0.01 | 0.25 ± 0.01 | 0.12 ± 0.03 | 0.25 ± 0.02 | 0.17 ± 0.04 | 0.20 ± 0.03 |
| ICRL | 0.11 ± 0.02 | 0.21 ± 0.04 | 0.21 ± 0.16 | 0.27 ± 0.03 | 0.41 ± 0.00 | 0.35 ± 0.17 |
| ICL (ours) | **0.08 ± 0.01** | **0.04 ± 0.01** | **0.02 ± 0.00** | **0.08 ± 0.05** | **0.07 ± 0.00** | **0.05 ± 0.00** |

Recovered constraint is closest to the true constraint for our method

## Dissimilarity between policies (learned vs expert)

| Algorithm↓, Environment→ | Gridworld (A) | Gridworld (B) | CartPole (MR) | CartPole (Mid) | Ant-Constrained | HalfCheetah-Constrained |
|---|---|---|---|---|---|---|
| GAIL-Constraint | 1.76 ± 0.25 | 1.29 ± 0.07 | 1.80 ± 0.24 | 7.23 ± 3.88 | 8.02 ± 2.84 | 14.38 ± 2.36 |
| ICRL | 1.73 ± 0.47 | 2.15 ± 0.92 | 12.32 ± 0.48 | 13.21 ± 1.81 | 9.50 ± 2.84 | **7.50 ± 4.97** |
| ICL (ours) | **0.36 ± 0.10** | **1.26 ± 0.62** | **1.63 ± 0.89** | **3.04 ± 1.93** | **6.84 ± 1.29** | 10.16 ± 7.49 |

Learned policy is similar to the expert policy in 5/6 environments

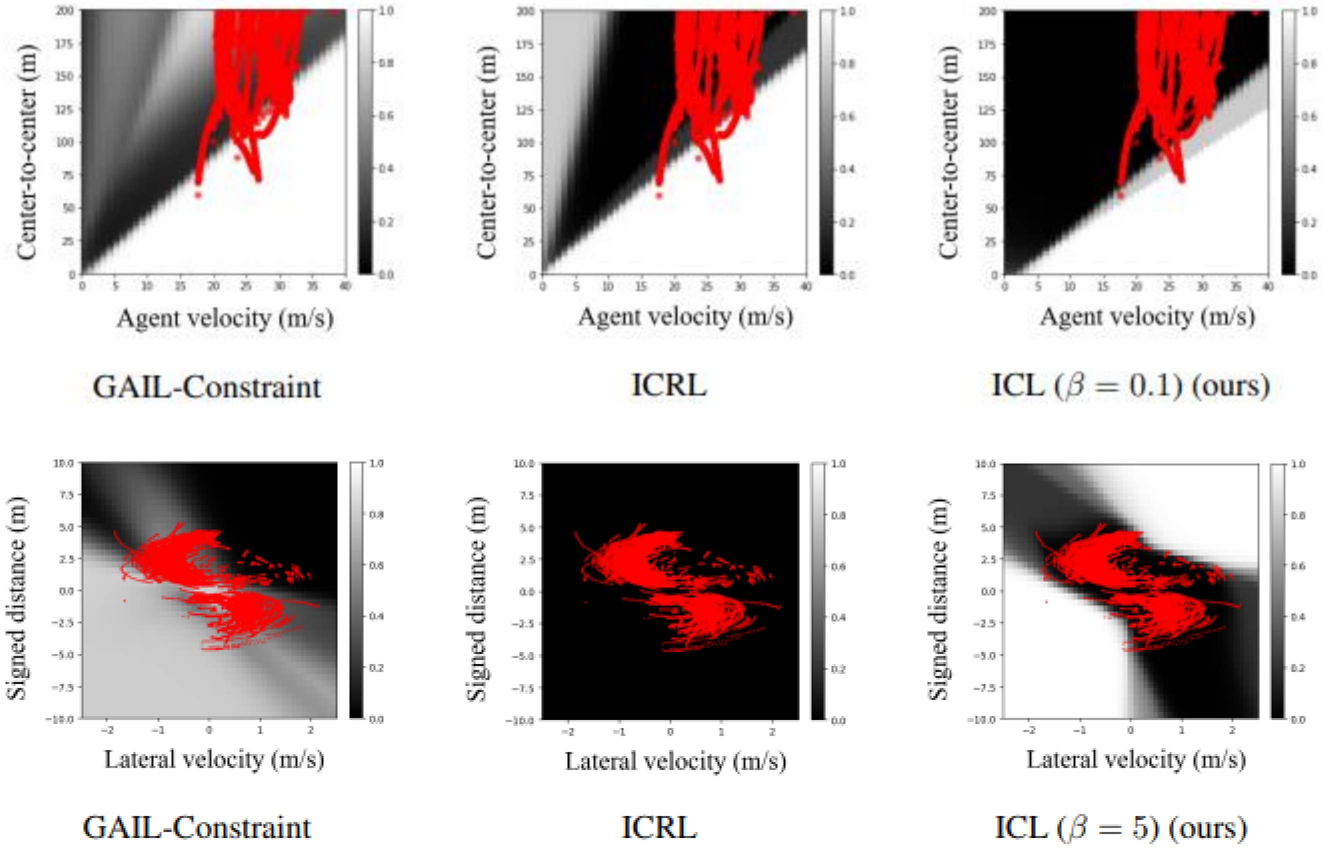
# Experiments

Synthetic | Robotics | Real world

Since these environments are based on real world autonomous driving datasets, ground truth constraints are not known!

Our method finds more reasonable constraints than the baselines.

## Recovered constraint functions




(HighD)

(ExiD)

Advantages:

- Accurate and sharp constraints
- Can learn complex constraints
- Learns a policy similar to the expert policy in most cases
- Any method can be used for constrained RL
- Works with stochastic dynamics

Future work:

- Learn multiple constraint functions?
  Or reward with constraints?
- Address unidentifiability
- Learn from suboptimal trajectories?

One-line summary:
“new technique to learn soft/expected constraint from expert demonstrations”

Please visit our poster!

Location: MH1-2-3-4 #104
Time: 11:30am – 1:30pm (today)