Sampling-based Inverse Reinforcement Learning Algorithms with Safety Constraints

Johannes Fischer[1,3], Christoph Eyberg[1], Moritz Werling[2], Martin Lauer[1]

Abstract—**Planning for robotic systems is frequently formulated as an optimization problem. Instead of manually tweaking the parameters of the cost function, they can be learned from human demonstrations by Inverse Reinforcement Learning (IRL). Common IRL approaches employ a maximum entropy trajectory distribution that can be learned with soft reinforcement learning, where the reward maximization is regularized with an entropy objective. The consideration of safety constraints is of paramount importance for human-robot collaboration. For this reason, our work addresses maximum entropy IRL in constrained environments. Our contribution to this research area is threefold: (1) We propose Constrained Soft Reinforcement Learning (CSRL), an extension of soft reinforcement learning to Constrained Markov Decision Processes (CMDPs). (2) We transfer maximum entropy IRL to CMDPs based on CSRL. (3) We show that using importance sampling in maximum entropy IRL in constrained environments introduces a bias and fails to achieve feature matching. In our evaluation we consider the tactical lane change decision of an autonomous vehicle in a highway scenario modeled in the SUMO traffic simulation.**

Index Terms— **Reinforcement Learning, Inverse Reinforcement Learning, Maximum Entropy, Constraints, Safety, SUMO.**

I. Introduction

The planning component of an autonomous system is often realized as an optimal control problem or a reinforcement learning agent. Both approaches require a cost or reward function, respectively, which encodes not only the mission goal but also has to weight competing objectives such as quickly achieving the goal while at the same time moving smoothly and with low control energy. Instead of arduously hand-tuning the parameters of the objective function until the desired behavior is achieved, reward functions can be learned from human demonstrations by Inverse Reinforcement Learning (IRL).

The dynamics of a real system are often too complex to formulate an accurate dynamics model, especially when they involve human interaction. For this reason, many IRL algorithms rely on sampled behavior instead of assuming knowledge of the dynamics model. To remain sample-efficient, they frequently use importance sampling. As described by Finn *et al.*, a particularly suitable sampling policy is given by soft reinforcement learning [1, p. 4], where the policy is regularized with an entropy term [2].

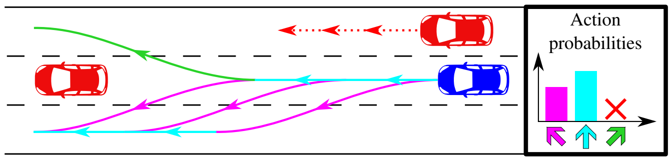


Figure 1: Constrained Soft Reinforcement Learning does not allow the unsafe action of changing lane to the right for the ego vehicle (blue). Changing to the left lane would allow driving faster, but the trajectory entropy is higher on the center lane, since more options remain available in the future, hence the lane change is most likely postponed.

The planning task is aggravated by vital safety constraints when interacting with humans which can not be adequately captured by a reward function, e.g. how exactly would you weight energy consumption against risking human life? Clearly the latter should be avoided no matter what the other reward terms are. Prior work has shown that constraining the trained policy upon execution time is not sufficient. Instead, the safety constraints have to be incorporated during reinforcement learning [3].

To address IRL problems in environments with safety constraints, our first contribution is Constrained Soft Reinforcement Learning (CSRL), illustrated in Fig. 1, which extends previous work on constrained reinforcement learning to soft reinforcement learning. Based on this advancement, our second contribution is to apply CSRL to solve IRL problems in constrained environments. In particular, we investigate the effect of importance sampling in presence of constraints as our third contribution.

We evaluate our algorithms in a highway driving environment, where an agent is trained to make optimal lane change decisions. The evaluation is conducted in the Simulation of Urban MObility (SUMO). This not only allows to compare the results with the ground truth expert, but also to compute the true expected value difference between the expert policy and an agent trained on the learned reward function, which is not available when using real world data.

II. Related Work

Prior work has studied the problem of safety in reinforcement learning extensively. Most approaches try to avoid collisions by penalizing them in the reward function [4]–[6]. This leads to a trade-off between safety and other reward terms

[1]Institute of Measurement and Control Systems, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
[2]BMWGroup, Unterschleissheim, Germany
[3]Corresponding author: johannes.fischer@kit.edu

and hence does not guarantee safety. Instead, the level of safety depends on the penalty. Another class of methods constrain the agent's actions upon execution time by masking unsafe actions and thus can provide safety guarantees [7]–[10]. However, in those approaches safety is only considered at execution time, not during the actual learning process. Since reinforcement learning primarily learns which action has the highest Q-value, there is no guarantee on the order of the other Q-values. For this reason, Kalweit *et al.* incorporate the constraints directly into the Q-learning update rule to train a safe and optimal policy [3]. In comparison, our proposed CSRL algorithm integrates constraints into an entropy-regularized policy.

There have been many sampling-based IRL approaches before, some of which use Importance Sampling (IS). Two particularly important approaches are described in more detail in Section III-C [1], [11]. Others use sampling in path integral IRL, but achieve improved performance by dropping the importance weights, similar to our findings [12], [13].

Kalweit *et al.* consider a similar problem setting where they also simultaneously learn the reward and the Q-function [14]. Further they reformulate the problem to not use importance sampling, but they employ a Boltzmann distribution with the energy given by the optimal value function, which only maximizes the entropy of the next action. In contrast, the maximum entropy policy we consider, maximizes the entropy of the whole trajectory [2, p. 3]. Additionally, Kalweit *et al.* consider the case that the recovered policy has to satisfy additional constraints that have not been imposed on the expert, a topic also investigated by Tschiatschek *et al.* [15]. The topic of additionally inferring constraints from demonstrations is considered by Scobee *et al.* [16].

III. Preliminaries

A. Reinforcement Learning

In a typical planning problem, an agent has to choose actions in an environment where the task is encoded in an objective function. Such a sequential decision-making problem can be modeled as a Markov Decision Process (MDP) and is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma)$. The agent sequentially chooses actions $a_t \in \mathcal{A}$ according to a policy π, where $\pi(a_t|s_t)$ denotes the probability distribution over actions when the agent finds itself in state $s_t \in \mathcal{S}$. The transition model $\mathcal{T}(s_{t+1}|s_t, a_t)$ encodes the distribution over successor states after executing an action. In each step the agent receives a reward $r_t = r(s_t, a_t)$ and its goal is to find a policy π^* that maximizes the expected cumulative reward (also called return) $\mathbb{E}_{a_t \sim \pi^*}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$, where the reward is discounted with $\gamma < 1$ in each step [17].

A trajectory τ is a finite sequence of state-action pairs (s_t, a_t). The probability that a policy follows a specific trajectory is given by $\pi(\tau) = \prod_t \pi(a_t|s_t)$ while the reward which is collected along a trajectory is $r(\tau) = \sum_t r(s_t, a_t)$.

The optimal value function is defined as $Q^*(s_0, a_0) = r_0 + \mathbb{E}_{a_t \sim \pi^*}\left[\sum_{t=1}^{\infty} \gamma^t r_t\right]$ and captures the expected return when choosing action a_0 in s_0 and following the optimal policy π^* thereafter. Reinforcement learning algorithms use interactions with the environment to sample the Bellman optimality equation and iteratively learn π^* [17].

In the popular Deep Q-Network (DQN) algorithm, the value function is approximated by a deep neural network [18]. The network parameters ϕ are updated to minimize the squared error between the current value function $Q^{\phi}(s_t, a_t)$ and the one-step temporal difference target $y_t = r_t + \gamma \max_{a_{t+1}} Q^{\hat{\phi}}(s_{t+1}, a_{t+1})$. Introducing separate parameters $\hat{\phi}$ for the target net that are periodically updated with $\hat{\phi} \leftarrow \phi$ stabilizes the training process [18].

B. Soft Reinforcement Learning

For modeling human-like behavior, it can be beneficial to induce sub-optimality by having the agent not always choose the best action. In deterministic environments, such behavior can be modeled with a policy where a trajectory becomes exponentially more likely, the higher its reward is [19]

$$p(\tau) \sim \exp(r(\tau)). \tag{1}$$

Another benefit of such a stochastic policy is that it implicitly solves the trade-off between exploration and exploitation [2]. Such behavior can be achieved by maximizing not only the expected return, but also the policy's entropy along the trajectory

$$\pi^*_{\text{MaxEnt}} = \arg\max_{\pi} \sum_{t=0}^{\infty} \mathbb{E}_{a_t \sim \pi}\left[r_t + \alpha \mathcal{H}(\pi(\,\cdot\,|s_t))\right], \tag{2}$$

where $\alpha > 0$ weights between reward and entropy maximization [2]. The optimal policy for Eq. (2) is given by $\pi^*_{\text{MaxEnt}} = \exp(\frac{1}{\alpha}(Q^*_{\text{soft}} - V^*_{\text{soft}})$, where the soft value functions can be found by iterating the soft Bellman equation

$$Q^*_{\text{soft}}(s_t, a_t) = r_t + \gamma \mathbb{E}\left[V^*_{\text{soft}}(s_{t+1})\right], \tag{3}$$

$$V^*_{\text{soft}}(s_t) = \alpha \log \int_{\mathcal{A}} \exp\left(\frac{1}{\alpha} Q^*_{\text{soft}}(s_t, a)\right) \mathrm{d}a\,. \tag{4}$$

Note that the conventional Bellman equation is obtained in the limit $\alpha \to 0$, where Eq. (4) approaches a maximum over actions [2].

C. Maximum Entropy Inverse Reinforcement Learning

The goal of IRL is to infer an unknown reward function from a set of N expert demonstrations $\mathcal{D}_{\text{demo}}$. For now, let us consider a reward model $r_\theta(s, a) = \theta^T f(s, a)$ given as a linear combination of features $f(s, a)$. In general, there exist infinitely many weights θ that explain the demonstrations, but a reasonable reward function can be expected to achieve matching feature expectations

$$\mathbb{E}_{\tau \sim \mathcal{D}_{\text{demo}}}[f(\tau)] = \mathbb{E}_{\pi_\theta}[f(\tau)] \tag{5}$$

between the demonstrations and a policy π_θ trained on the learned reward r_θ, where $f(\tau) = \sum_t f(s_t, a_t)$ [20]. Due to the linear reward model, matching feature expectations imply that the learned policy achieves the same reward as the expert.

The Maximum Entropy Inverse Reinforcement Learning (MaxEntIRL) framework solves the ambiguity in reward functions by employing the soft-optimal trajectory distribution in

Eq. (1), which implicitly results in matching feature expectations [19], [21]. The reward parameters θ are obtained by minimizing the negative log-likelihood of the demonstrations

$$\mathcal{L}(\theta) = -\frac{1}{N}\sum_{\mathcal{D}_{\text{demo}}} r_\theta(\tau) + \log Z(\theta), \tag{6}$$

where the normalizing constant of Eq. (1) is the partition function $Z(\theta) = \int \exp(r_\theta(\tau))\,\mathrm{d}\tau$. Computing the gradient requires to solve the forward problem with dynamic programming in each iteration, which requires knowledge of the transition model $\mathcal{T}$ [19].

Finn *et al.* developed a model-free formulation by approximating the partition function with Importance Sampling (IS) [1]. Furthermore, this enables their Guided Cost Learning (GCL) algorithm to work on samples from a policy, that is not yet optimally trained for the current reward model. In each iteration they sample M trajectories $\mathcal{D}_{\text{sample}}$ from a policy π_{sample} and approximate the loss as

$$\mathcal{L}(\theta) = -\frac{1}{N}\sum_{\mathcal{D}_{\text{demo}}} r_\theta(\tau_i) + \log \sum_{\mathcal{D}_{\text{sample}}} \frac{\exp(r_\theta(\tau_j))}{M\pi_{\text{sample}}(\tau_j)}. \tag{7}$$

The sampling policy is iteratively improved by training an agent on the current reward iterate r_θ.

A slightly different objective is used in Relative Entropy Inverse Reinforcement Learning (RelEntIRL), where the relative entropy between the trajectory distribution under a baseline policy π_{b} and the distribution induced by the reward function is minimized subject to the constraint of matching feature expectations [11]. They also use IS to approximate the partition function but sample from a uniform policy to obtain an offline algorithm. For a linear reward model the resulting loss gradient is given by

$$\frac{\partial \mathcal{L}}{\partial \theta_k} = -\overline{f}_k + \sum_{\mathcal{D}_{\text{sample}}} \frac{\frac{\pi_{\text{b}}(\tau)}{\pi_{\text{sample}}(\tau)} \exp\left(r_\theta(\tau)\right) f_k(\tau)}{\sum_{\mathcal{D}_{\text{sample}}} \frac{\pi_{\text{b}}(\tau)}{\pi_{\text{sample}}(\tau)} \exp\left(r_\theta(\tau)\right)}, \tag{8}$$

where $\overline{f}_k = \frac{1}{N}\sum_{\mathcal{D}_{\text{demo}}} f_k(\tau_i)$ denotes the average feature value.

IV. Concepts for Constrained Inverse Reinforcement Learning

In real-world problems, safety is often a paramount constraint, especially in domains where humans interact or collaborate with robots, such as industrial robots or autonomous driving. Our goal is to learn a soft-optimal policy to model the IRL problem under consideration of constraints. To this end, we assume a Constrained Markov Decision Process (CMDP), that is an MDP where the set of feasible actions $\mathcal{A}(s) = \{a \in \mathcal{A} \,|\, c_k(s,a) \leq 0 \;\; \forall k\}$ is restricted based on the current state s and arbitrary constraint signals $c_k : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ [3]. A requirement is that at least one action is safe to choose in each state, i.e. $\mathcal{A}(s) \neq \emptyset$.

Commonly, safety is guaranteed by safe policy extraction (e.g. [8]), where the policy is trained on the full action space but restricted to safe actions at execution time. However, this strategy can lead to undesired behavior as demonstrated by Kalweit *et al.* [3]: Unsafe state-action pairs that achieved a high Q-value during training can mislead the policy, since those high values cannot be attained with safe policy extraction. To resolve this issue, they propose to adapt the DQN target to $y_t = r_t + \gamma \max_{a \in \mathcal{A}(s_{t+1})} Q^{\hat{\phi}}(s_{t+1}, a)$, i.e. the maximum is only computed over the safe actions.

Algorithm 1 Deep Constrained Soft Reinforcement Learning

```
1: load transitions into replay memory
2: randomly initialize weights $\phi$ for $Q^\phi$
3: initialize weights $\hat{\phi}$   $\phi$ for target $Q^{\hat{\phi}}$
4: for $M$ episodes do
5:     sample random minibatch $(s_j, a_j, r_j, s_{j+1})_{1 \leq j \leq m}$
6:     $v_{j+1}$   $\alpha \log \int_{\mathcal{A}(s_{j+1})} \exp\left(\frac{1}{\alpha} Q^{\hat{\phi}}(s_{j+1}, a')\right) \mathrm{d}a'$
7:     $y_j$   $r_j + \gamma v_{j+1}$
8:     minimize MSE $\frac{1}{m}\sum_j (Q^\phi(s_j, a_j) - y_j)^2$ w.r.t. $\phi$
9:     soft target update $\hat{\phi} \leftarrow (1-\tau)\hat{\phi} + \tau\phi$
10: end for
```

A. Constrained Soft Reinforcement Learning

In this section we present Constrained Soft Reinforcement Learning (CSRL), a generalization of maximum entropy reinforcement learning to CMDPs. To learn the soft value function of the CMDP, we restrict the soft maximum over actions in Eq. (4) to the safe action space $\mathcal{A}(s)$. For a sampled transition (s_t, a_t, r_t, s_{t+1}) the target for updating $Q_{\text{soft}}(s_t, a_t)$ is then given by

$$y_t = r_t + \gamma\alpha \log \int_{\mathcal{A}(s_{t+1})} \exp\left(\frac{1}{\alpha} Q^{\hat{\phi}}_{\text{soft}}(s_{t+1}, a')\right) \mathrm{d}a', \tag{9}$$

and the corresponding soft policy is

$$\pi(a|s) = \begin{cases} \dfrac{\exp\left(\frac{Q_{\text{soft}}(s,a)}{\alpha}\right)}{\int_{\mathcal{A}(s)} \exp\left(\frac{Q_{\text{soft}}(s,a')}{\alpha}\right) \mathrm{d}a'}, & \text{if } a \in \mathcal{A}(s), \\ 0, & \text{else.} \end{cases} \tag{10}$$

Note that this is consistent with the idea of avoiding unsafe actions by using soft constraints and thus strongly punishing unsafe actions, since a vanishing probability corresponds to $Q_{\text{soft}}(s,a) = -\infty$ in this setting. Hence, this seamlessly integrates constraints into soft-optimal policies. Algorithm 1 presents the resulting CSRL method.

B. Constrained Maximum Entropy Inverse Reinforcement Learning

To incorporate constraints into the MaxEntIRL framework, we also use the trajectory distribution in Eq. (1), but restricted to feasible trajectories. Let

$$C = \{\tau = (s_t, a_t)_{0 \leq t \leq T} | a_t \in \mathcal{A}(s_t) \; \forall t\} \tag{11}$$

denote the set of feasible trajectories, then the partition function for constrained MaxEntIRL is given by $Z(\theta) = \int_C \exp(r_\theta(\tau))\,\mathrm{d}\tau$. The negative log-likelihood can be approximated by sampling from a feasible policy, i.e. a policy resulting in a trajectory distribution q satisfying $\operatorname{supp}(q) \subseteq C$.

Regarding constraints there is no difference between sampling for the regular MaxEntIRL, GCL or RelEntIRL algorithms, as long as the sampling policy remains feasible. As reported by Finn *et al.*, using soft reinforcement learning allows to recover a suitable sampling policy [1, p. 4]. For this reason, we propose to train a CSRL agent as a sampling policy during IRL.

In our evaluation we investigate the implications of CSRL to sampling-based IRL algorithms building on the maximum entropy framework. As it will turn out, using IS biases the estimate of the partition function and therefore leads to worse behavior in CMDPs. To resolve this issue, we propose to use a sampling-based approximation of MaxEntIRL without IS, which is derived in the following. The gradient of the negative log likelihood in Eq. (6) is given by

$$\frac{\partial \mathcal{L}}{\partial \theta_k} = -\frac{1}{N} \sum_{\mathcal{D}_{\text{demo}}} \frac{\partial r_\theta(\tau_i)}{\partial \theta_k} + \int p_\theta(\tau) \frac{\partial r_\theta(\tau)}{\partial \theta_k} \, \mathrm{d}\tau, \tag{12}$$

where p_θ is the maximum entropy distribution Eq. (1) corresponding to the current reward model r_θ. This distribution is learned by the CSRL agent, leading to the sampling-based approximation

$$\frac{\partial \mathcal{L}}{\partial \theta_k} = -\frac{1}{N} \sum_{\mathcal{D}_{\text{demo}}} \frac{\partial r_\theta(\tau_i)}{\partial \theta_k} + \frac{1}{M} \sum_{\mathcal{D}_{\text{sample}}} \frac{\partial r_\theta(\tau_j)}{\partial \theta_k}. \tag{13}$$

However, in our experiments it was not necessary to optimize the policy until convergence at each step. These findings are consistent with prior works suggesting dropping the importance weights [12], [13].

V. Problem description

In the following, we present the environment on which we evaluate CSRL and the IRL algorithms. We consider the tactical lane change decision on highways. First, the problem is formulated as a CMDP, then we present the employed traffic simulation and describe the training process.

A. Formulation of the CMDP

We model the lane change decision problem with three discrete high level actions $\mathcal{A} = \{a_{\text{keep}}, a_{\text{left}}, a_{\text{right}}\}$ for keeping the current lane and executing a lane change to either side. To guarantee safety, the action space is constrained to state specific sub-sets of safe actions $\mathcal{A}(s) \subseteq \mathcal{A}$ if necessary. To this end, a rule-based safety module such as presented in [22] is running in the background and asserts whether the lane change actions are safe to execute at each time instant. All other driving tasks like lateral control during a lane change or longitudinal control are executed by low level control modules that ensure that the action to stay on the current lane is always safe by following an ACC-like behavior.

To achieve a realistic modeling, we use a continuous state space representation containing the positions and velocities of the ego vehicle and the six neighboring vehicles, provided they drive within the sensor range of the ego vehicle. Hence, the state consists of the variables

$$s = \left(v_{\text{ego}}, \mathbb{1}_{\text{left}}, \mathbb{1}_{\text{right}}, d_1, \Delta v_1, \ldots, d_6, \Delta v_6\right), \tag{14}$$

similar to previous work [8], [10]. Here v_{ego} denotes the ego velocity, $\mathbb{1}_{\text{left}}$ and $\mathbb{1}_{\text{right}}$ indicate whether a neighboring lane exists to the left and right of the ego lane, respectively. Further, d_i denotes the (signed) distance and $\Delta v_i = v_i - v_{\text{ego}}$ the relative velocity, where i enumerates the direct leader and follower on each of the ego lane and the adjacent lanes in fixed order. If there exist no vehicles within sensor range on any of these positions, e.g. because the respective lane does not exist, this position is filled with a ghost vehicle that drives with the same velocity as the ego vehicle at the sensing range limit. Additionally, all velocities and distances are normalized with the maximum legal velocity of $v_{\max} = 24\,\frac{\mathrm{m}}{\mathrm{s}}$ and the sensor range of $80\,\mathrm{m}$, respectively. The optimal behavior in this environment is defined by the reward function

$$r(s, a) = \frac{v_{\text{ego}}}{v_{\max}} - p_{\text{lc}} \mathbb{1}_{\text{lc}}(a) \tag{15}$$

where $\mathbb{1}_{\text{lc}}(a)$ indicates whether a lane change was performed and $p_{\text{lc}} > 0$ is the lane change penalty. Hence, the agent drives as fast as possible while trying to avoid lane changes.

B. Simulation Environment

We perform our experiments in the traffic simulation software SUMO [23]. Similar to previous research the ego vehicle drives on a circular three-lane highway of $1000\,\mathrm{m}$ length [24]. Realistic highway traffic situations are simulated by randomly placing 30 to 60 cars with different driving preferences in the environment. At test time, the trained agents are simulated in scenarios with 60 to 90 cars, to evaluate the learned behavior in situations they have not seen during training.

C. Training Setup

Even tough the agents are trained on an infinite horizon setup, the trajectory length is limited to reduce the curse of horizon in IS. The IRL algorithms are supplied with $20\,000$ demonstrations with a length of 5 time steps each. We have also conducted experiments with a reduced number of demonstration trajectories, but as our main goal is to investigate the effect of IS on IRL algorithms, it is more important to ensure that failing convergence does not occur due to lack of demonstrations. After each iteration of the IRL algorithms the agents are retrained on the latest approximation of the reward function for 1000 iterations with CSRL. All agents are trained online with an experience replay buffer [18]. The relevant training parameters are listed in Table I.

A Behavior Cloning (BC) agent is trained with cross-entropy loss on the demonstration set. This agent is used as a baseline for the IRL agents and as the baseline policy in RelEntIRL.

VI. Results and Evaluation

We first evaluate our proposed CSRL algorithm to ensure its suitability for constrained IRL as described in Section IV-B. This is followed by an evaluation of the IRL algorithms introduced in the previous sections. Lastly, we present a study of the influence the reward model's expressiveness has on the resulting performance.

Reinforcement learning	
Number input neurons	15
Number hidden layers	2
Density hidden layers μ	100
Weight initialization	$\mathcal{U}(\frac{-1}{\sqrt{\mu}}, \frac{1}{\sqrt{\mu}})$
Number output neurons	3
Activation function	elu
Batch size	64
Optimizer	Adam [25]
Learning rate	1×10^{-4}
Soft target update rate	1×10^{-4}
Training iterations	2×10^{6}
Entropy weight α	0.1
Inverse reinforcement learning	
Batch size	500
Trajectory length T	5
Number sampled trajectories	400
Weight decay	0.01
Learning rate	1×10^{-4}

Table I: Parameters used for training RL and IRL agents.

A. Validation of Constrained Soft Reinforcement Learning

We evaluate CSRL on the CMDP described in Section V-A. As suggested in prior work, we choose a lane change penalty of $p_{\text{lc}} = 0.01$ to achieve comparable results [3], [24].

Fig. 2 shows the results of CSRL trained with different values α for weighting the entropy regularization. The performance is benchmarked against a Constrained DQN (CDQN) agent [3], the MOBIL lane change model [26], and two agents doing random lane changes and no lane changes, respectively. Results are shown for different traffic densities used during simulation. Note that there are no safety constraint violations by design of the algorithm.

The total reward in Fig. 2a is measured with respect to the original reward Eq. (15), not considering the entropy reward. For $\alpha \in \{0.01, 0.1, 1\}$, the performance of CSRL matches the performance of CDQN, it only decreases for $\alpha = 10$, which weights maximizing the entropy too much compared to the actual reward. The advantage over CDQN is that the parameter α can be used to tune the entropy of the resulting trajectory distribution without decreasing the actual return, as illustrated in Figs. 2a and 2c. Surprisingly, a higher trajectory entropy does not lead to an increased number of lane changes. As Fig. 2b shows, it is rather the other way around: A higher entropy weight leads to the agent visiting states with less expected constraints, such as following another vehicle on the center lane. Contrarily, frequent lane changes decrease the entropy, since more constraints are active while passing other vehicles.

B. The Effect of Importance Sampling in Constrained IRL

In the following, we use a CSRL agent with $p_{\text{lc}} = 1$ to generate expert demonstrations for the IRL algorithms. This increased penalty forces the agent to weight speed gain against lane changes more carefully. To compare the different algorithms we first consider the class of reward models

$$r_\theta(s,a) = \theta_{\text{speed}} \cdot \frac{v_{\text{ego}}}{v_{\max}} + \theta_{\text{lc}} \cdot \mathbb{1}_{\text{lc}}(a) \tag{16}$$

	$w(\tau)$	$\overline{f}^{w}_{\text{speed}}$	$\overline{f}^{w}_{\text{lc}}$
unweighted mean $\overline{f}$		0.72	0.069
MaxEntIRL	1	0.72	0.069
RelEntIRL (expert)	$\exp(r(\tau))$	0.85	0.061
RelEntIRL (BC)	$\frac{\pi_{\text{BC}}(\tau)}{\pi_{\text{sample}}(\tau)} \exp(r(\tau))$	0.86	0.059
GCL	$\frac{\exp(r(\tau))}{\pi_{\text{sample}}(\tau)}$	0.89	0.18

Table II: Feature weights and weighted mean feature values used by the algorithms.

with parameters $\theta_{\text{speed}}, \theta_{\text{lc}}$, which contains the ground truth reward Eq. (15). This linear reward model allows to directly observe artifacts of the IRL algorithms due to IS and to measure the error in matching feature expectations.

The resulting behavior of a policy trained on the learned reward r_θ is evaluated in Fig. 3. The average values of the two features are illustrated in Figs. 3b and 3c for different traffic densities. Furthermore, Fig. 3a shows the performance of the policy with respect to the ground truth reward function to assess the expected value difference between the learned agent and the expert.

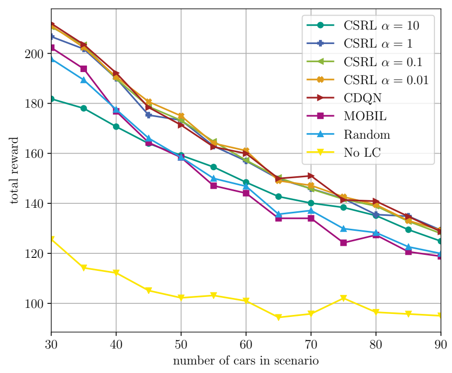
The MaxEntIRL-agent (blue) not only achieves the same performance as the expert (turquoise), but also achieves close feature matching in all traffic densities. The GCL-agent (red) also performs well in terms of reward, but when evaluating the feature values, its behavior is far off the expert with less average velocity and reduced lane change frequency. The RelEntIRL-agent (green) on the other hand achieves a slightly higher velocity than the expert, but at the cost of a very high number of lane changes, which leads to a very low overall reward. Since the RelEntIRL-agent depends on the baseline policy, which is learned with BC (purple), its performance could potentially be influenced by the suboptimal performance of the BC-agent. For this reason, we train an additional RelEntIRL-agent, supplied with the true expert policy as baseline policy (yellow), which achieves a lower number of lane changes but also a lower average velocity, leading to a reward comparable with the RelEntIRL-agent using BC.

This can be explained by examining where the objectives of the different algorithms converge. To this end, the loss gradient of the considered algorithms defined by Eqs. (7), (8) and (13) is written as

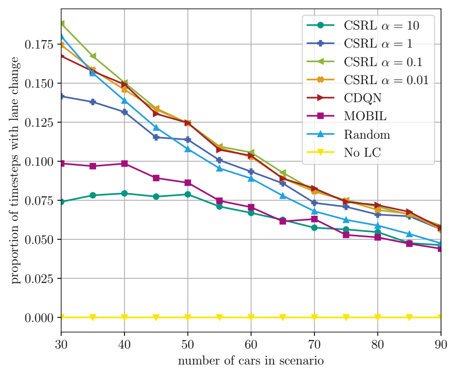
$$\frac{\partial \mathcal{L}}{\partial \theta_k} = -\overline{f}_k + \frac{\sum_{\mathcal{D}_{\text{sample}}} w(\tau_j) f_k(\tau_j)}{\sum_{\mathcal{D}_{\text{sample}}} w(\tau_j)} = -\overline{f}_k + \overline{f}^{w}_k, \tag{17}$$

where the algorithms differ in the sampling policy and in the weights $w(\tau)$ they use for weighting the trajectories. That is, the loss gradient is the difference between the average feature values $\overline{f}_k$ of the demonstrations and the weighted average feature values $\overline{f}^{w}_k$ of the samples. Table II summarizes the form of the weights used by each algorithm.

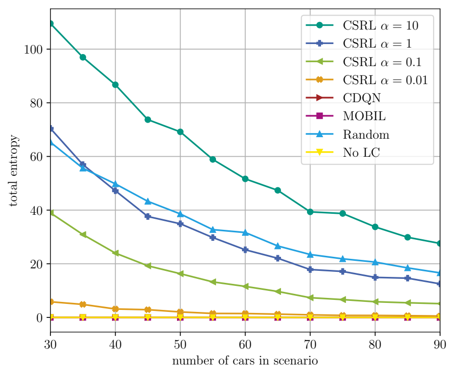
At convergence, the loss gradient vanishes. Eq. (17) illustrates that only the MaxEntIRL loss achieves an unbiased



(a) Average total reward

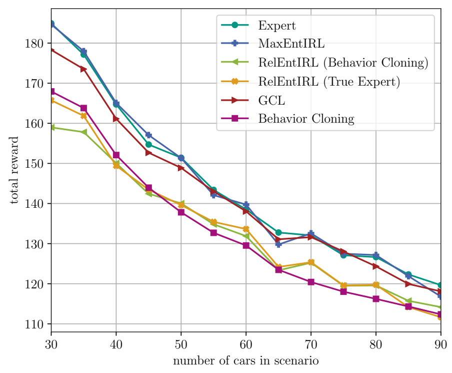


(b) Average proportion of lane changes

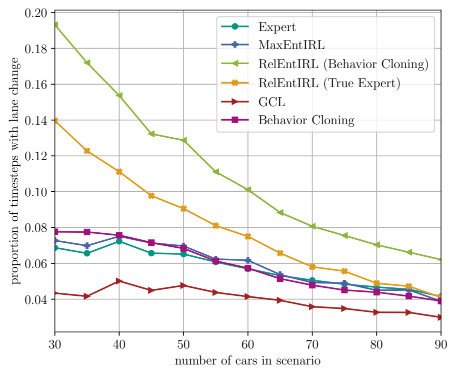


(c) Average trajectory entropy

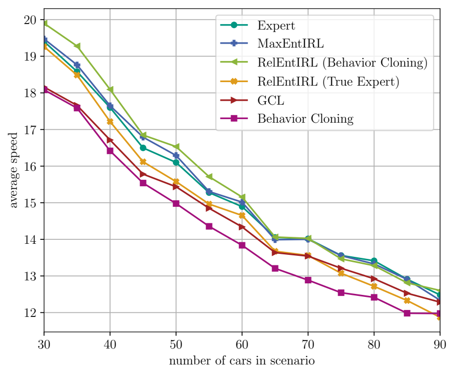
Figure 2: Performance of CSRL for varying values of entropy weight α.



(a) Average total reward



(b) Average proportion of lane changes



(c) Average velocity

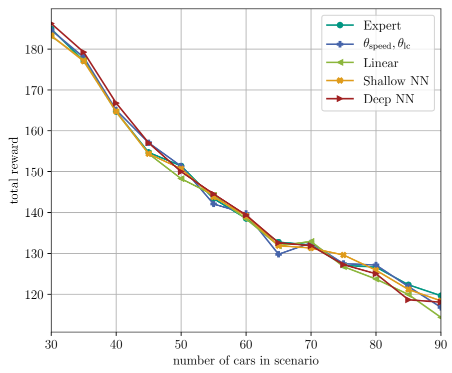
Figure 3: Performance of learned reward models.

feature matching at convergence. The GCL and RelEntIRL losses are biased by the weights of self-normalized IS [27].

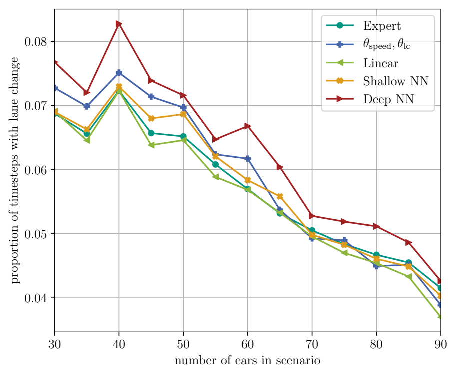
To evaluate this bias, we computed the weighted average features and compared them with the unbiased average feature. To this end, we assumed ideal prerequisites as they would occur at convergence, i.e. that all three algorithms were able to sample from the expert policy and have access to the ground truth reward function. The resulting weighted average features $\overline{f}^w_{\text{speed}}$ and $\overline{f}^w_{\text{lc}}$ are shown in Table II. It can be concluded that the MaxEntIRL-agent finds a reward that best explains the expert's behavior, as it only converges upon true feature matching. Using importance sampling, the other algorithms weight trajectories higher that obtain high rewards and therefore have more extreme feature values than the average trajectory, which leads to an overestimation of these feature values. These algorithms converge to reward functions that induce weighted feature values that coincide with the expert's feature values. As Table II shows, those are not the true average feature values, such that the algorithms are unable to recover the true expert policy.

Theoretically, this is compensated for by the importance weights also weighting unlikely trajectories higher. However, in the environment considered this leads to problems due to the constraints: In low traffic density, the agent can typically achieve a high reward, since it can drive very fast. However, since most of the time all actions are feasible and all share a similar expected return, the individual trajectories each have a relatively low probability. On the contrary, in high traffic density, the agent gets stuck in traffic jams frequently, which reduces the overall velocity and thus the reward. Since the action space is frequently constrained in such situations, the individual trajectories have a higher probability than those in low traffic density. Hence, the two effects of high reward and high probability do not balance out but amplify the effect, leading to very high or very low importance weights. In summary, the weighted average features reflect only trajectories in low traffic density, since they have high rewards as well as a low individual trajectory probability. This clearly illustrates, that only the unbiased MaxEntIRL algorithm can be expected to achieve feature matching. In particular, if considering the RelEntIRL-agent using the expert policy, high reward trajectories are weighted exponentially higher as can be seen in Table II.

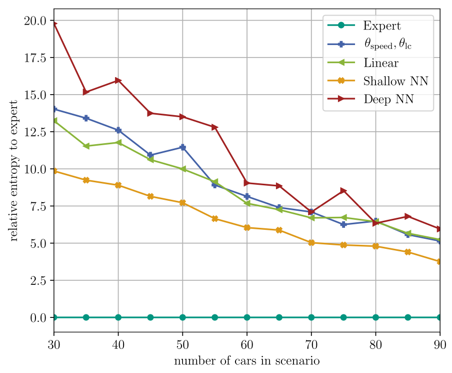
The argument is further supported by the empirical mean feature values of the converged policies in Table III. The MaxEntIRL agent achieves by far the least deviation from the expert feature counts.



(a) Average total reward



(b) Average proportion of lane changes



(c) KL-divergence to expert policy

Figure 4: Performance of learned reward models.

	$\overline{f}_{\text{speed}}$			$\overline{f}_{\text{lc}}$		
	μ	σ	η	μ	σ	η
Expert	15.29	1.64	0.0%	0.057	0.018	0.0%
MaxEntIRL	15.34	1.56	0.3%	0.059	0.017	3.1%
RelEntIRL	15.52	1.61	1.5%	0.110	0.028	93.0%
GCL	14.70	1.45	−3.9%	0.040	0.014	−30.0%

Table III: Empirical average feature values for learned agents in terms of the mean μ and standard deviation σ averaged over all traffic densities and the relative deviation η from the expert's feature values.

C. *Learning Nonlinear Reward Models*

Even though the ground truth reward is linear in its parameters, we investigate the effect of learning a nonlinear reward model. In this section we try to answer the question how well rewards can be learned with an increasingly complex parameterization of the reward model. To this end, we compare the following reward structures:

- The ground truth reward model in Eq. (16),
- a linear combination of all state-action features,
- neural networks with one (shallow NN) and two (deep NN) hidden layers, respectively.

Since the previous discussion revealed issues with using IS in CMDPs, we only consider the MaxEntIRL-agent in the following.

As Fig. 4b illustrates, the closest feature matching with respect to the lane change frequency is achieved by the linear and the shallow network rewards. However, the average velocity (not shown) and the resulting performance in terms of average ground truth reward in Fig. 4a are nearly identical.

Nevertheless, we would like to emphasize that feature matching can only be expected in the case of a linear reward model. For a nonlinear reward model, this translates to matching the features in the last hidden layer of the neural network. In particular, the networks might learn totally different features, and hence cannot be expected to match in the velocity and lane change frequency features. For this reason, the Kullback-Leibler-divergence between the expert policy and the learned policies was evaluated. Fig. 4c shows that the shallow network policy is closest to the expert policy.

These results indicate, that it can be beneficial to train a reward model with higher complexity than what is theoretically necessary. Even tough the ground truth reward could be exactly represented by the simplest reward model used, the increased number of parameters of the shallow network lead to a policy closer to the expert policy. A possible explanation is that this allows to capture long term reward more easily. However, the performance can also easily degrade when using too many parameters, as can be seen for the network with two hidden layers.

VII. Conclusions and Future Work

In this work, we presented the Constrained Soft Reinforcement Learning algorithm, an extension of soft reinforcement learning to constrained MDPs. This allowed us to model the concept of a maximum entropy trajectory distribution, which is commonly assumed in MaxEntIRL algorithms, also within constrained environments. This is particularly relevant for reinforcement learning in human-robot collaboration, where safety constraints are of the uttermost importance, We exemplarily demonstrated the utility of our approach for a high-level planning module of an automated vehicle.

Our findings indicate, that IRL algorithms using importance sampling are not well suited for constrained environments. Furthermore, we have shown that IRL can benefit from a reward model which is more complex than necessary: Even though the ground truth reward used in our experiments can be exactly represented by the linear reward model, it turned out to be beneficial to learn a more complex reward structure like a small neural network.

Promising directions for further research include incorporating long term constraints as suggested in [3] into CSRL. Moreover, it is an interesting question to further narrow down the requirements for importance sampling to work well in IRL algorithms. An orthogonal future research direction is to investigate our approach in other environments where robots have to safely interact with humans such as industry or healthcare robots.

Acknowledgements

This work has been supported by the BMWGroup, Germany. The authors also thank the German Research Foundation (DFG) for being funded within the priority program "SPP 1835 – Cooperative Interacting Automobiles (CoInCar)". Further, the authors would like to thank Maria Kalweit and Gabriel Kalweit for help with the implementation.

References

[1] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *Proceedings of the 33rd International Conference on Machine Learning - Volume 48*, 2016.

[2] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement Learning with Deep Energy-Based Policies," Jul. 2017. arXiv: 1702.08165 `[cs]`.

[3] G. Kalweit, M. Huegle, M. Werling, and J. Boedecker, "Deep Constrained Q-learning," Sep. 2020. arXiv: 2003.09398 `[cs, stat]`.

[4] C.-J. Hoel, K. Wolff, and L. Laine, "Automated Speed and Lane Change Decision Making using Deep Reinforcement Learning," *21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018.

[5] T. Tram, A. Jansson, R. Grönberg, M. Ali, and J. Sjöberg, "Learning Negotiating Behavior Between Cars in Intersections using Deep Q-Learning," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Nov. 2018.

[6] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, and K. Fujimura, "Navigating Occluded Intersections with Autonomous Vehicles Using Deep Reinforcement Learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018.

[7] D. Isele, A. Nakhaei, and K. Fujimura, "Safe Reinforcement Learning on Autonomous Vehicles," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018.

[8] B. Mirchevska, C. Pek, M. Werling, M. Althoff, and J. Boedecker, "High-level Decision Making for Safe and Reasonable Autonomous Lane Changing using Reinforcement Learning," in *International Conference on Intelligent Transportation Systems (ITSC)*, 2018.

[9] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *AAAI Conference on Artificial Intelligence*, 2018.

[10] S. Nageshrao, H. E. Tseng, and D. Filev, "Autonomous Highway Driving using Deep Reinforcement Learning," in *IEEE International Conference on Systems, Man and Cybernetics (SMC)*, Oct. 2019.

[11] A. Boularias, J. Kober, and J. Peters, "Relative entropy inverse reinforcement learning," in *14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, ser. Proceedings of Machine Learning Research, vol. 15, 2011.

[12] M. Kalakrishnan, P. Pastor, L. Righetti, and S. Schaal, "Learning objective functions for manipulation," in *2013 IEEE International Conference on Robotics and Automation*, May 2013.

[13] N. Aghasadeghi and T. Bretl, "Maximum Entropy Inverse Reinforcement Learning in Continuous State Spaces with Path Integrals," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[14] G. Kalweit, M. Huegle, M. Werling, and J. Boedecker, "Deep inverse q-learning with constraints," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020.

[15] S. Tschiatschek, A. Ghosh, L. Haug, R. Devidze, and A. Singla, "Learner-aware teaching: Inverse reinforcement learning with preferences and constraints," in *Advances in Neural Information Processing Systems*, vol. 32, Curran Associates, Inc., 2019.

[16] D. R. R. Scobee and S. S. Sastry, "Maximum Likelihood Constraint Inference for Inverse Reinforcement Learning," Sep. 2019. arXiv: 1909.05477 `[cs, eess, stat]`.

[17] R. S. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, Second edition. Cambridge, MA London: The MIT Press, 2018.

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, Feb. 2015.

[19] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, "Maximum Entropy Inverse Reinforcement Learning," in *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, AAAI Press, 2008.

[20] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Twenty-First International Conference on Machine Learning - ICML '04*, Banff, Alberta, Canada: ACM Press, 2004.

[21] E. T. Jaynes, "Information theory and statistical mechanics," *Phys. Rev.*, vol. 106, no. 4, May 1957.

[22] C. Pek, P. Zahn, and M. Althoff, "Verifying the safety of lane change maneuvers of self-driving vehicles based on formalized traffic rules," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2017.

[23] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y. Flötteröd, R. Hilbrich, *et al.*, "Microscopic Traffic Simulation using SUMO," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Nov. 2018.

[24] M. Huegle, G. Kalweit, B. Mirchevska, M. Werling, and J. Boedecker, "Dynamic Input for Deep Reinforcement Learning in Autonomous Driving," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Macau, China, Nov. 2019.

[25] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Dec. 2014. arXiv: 1412.6980 `[cs]`.

[26] A. Kesting, M. Treiber, and D. Helbing, “General Lane-Changing Model MOBIL for Car-Following Models,” *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1999, no. 1, Jan. 2007.

[27] C. P. Robert and G. Casella, *Monte Carlo Statistical Methods*, ser. Springer Texts in Statistics. New York: Springer-Verlag, 2004.