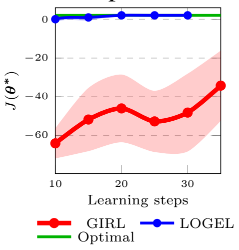
We thank the reviewers for insightful comments and suggestions. We hope the rebuttal will clarify all the doubts.

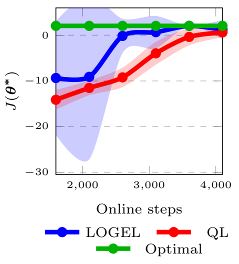
Motivation and real-world application In the LfL [1,2] setting, we do not ignore the near-optimal trajectories, but we can use all the training data to recover the reward function. In some cases, such as in multi-agent IRL, the agent is interested in learning the rewards of other agents before the other agents become experts, to adopt strategic behavior. Also in autonomous driving, when there is a new circuit, instead of waiting for the driver to become an expert, we can learn his intentions in the initial learning phase. In the appendix, we reported an experiment that uses the trajectories collected from an autonomous driving simulator (used in real-world problems [3]) to recover the reward function. We think this experiment may show a more **realistic application** of the LfL problem.

Gridworld In the gridworld experiment, we show LOGEL's performance when the *gradient-based learner assumption* is *violated*. We experiment with three no-gradient learners (Value Iteration, Q-Learning, and Soft Policy Iteration) and one gradient learner (GPOMDP). The results show that LOGEL works well even when the hypothesis is violated. In Figure 2) (10 runs) we show the performance of LOGEL using only the **online samples** of Q-Learning.

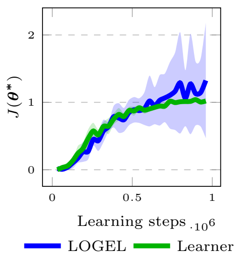
MuJoCo In these experiments, the learner uses the original reward function, but we construct the reward features for LOGEL from the state and action features. In the Hopper environment, we add the alive feature. The LfL algorithm uses, as in the original paper, a reward function which is very informative as they model the reward with a policy that has the same structure of the learner policy. In Figure 3) (6 runs) we show the behavior of LOGEL in the Hopper environment without the alive feature, using only the l2 distance between the current state and the next state and the norm, the squared norm, and the cubed norm of the action. The good behavior of the learned policies with the recovered reward weights is because the optimal reward function to learn a task can be different from the real one [5].



1)



2)



3)

R1 We answered questions 1 and 2 above. In the MuJoCo experiments, we use a different reward space than the learner one, so we cannot show convergence to the original reward weights. As suggested, we report another experiment with different reward features, Fig.3. Other questions: L138 the noise is due to the estimation of the gradient from trajectories; the intrinsic bias refers to the bias introduced by the estimated gradient that cannot be reduced with more learning steps; $\alpha_r > \epsilon$ because the learning rate is by assumption greater than 0; in gridworld experiment we used four learners: SVI, SPI, Q-learning and G(PO)MDP, we collected their trajectories and we estimate the reward function with LOGEL; L254 at the end of (paper) Fig.1 the weight difference is zero; L286 the Reacher and the Hopper experiments use the original reward functions for the learner, but different reward features for LOGEL created from the state and action information. In the appendix we compared LOGEL with TREX.

R2 We can think that in the MuJoCo environment the linearity assumption is violated as we use a different reward space for the recovered reward function. Behavioral cloning, as in LfL, is a requirement for running our algorithm. However, we think that by using the behavioral cloning of the previous iteration as a starting point, we can learn a good approximation of the current policy after a few learning steps. In the MuJoCo experiment, and the Gridworld experiment with the GPOMDP learner, we only use a subset of the demos that the learner uses to update her policy and we can assume that in a real-world scenario we have access to such demos. Thank you for pointing out another related work, we will add the citation.

R3 In the gridworld experiment, we use a value-based and a Q-Learning learner that violate the gradient-based assumption. In Figure 2) we show the behavior of LOGEL using only the online updates of a QLearning learner. In Figure 1) (10 runs), we compare the performance of LOGEL with GIRL [4], which is an IRL batch model-free algorithm. The experiment shows that the algorithm cannot recover the correct reward weights from suboptimal trajectories. We answer to the Hopper question in the MuJoCo section above.Ψ is bounded in l2-norm, $\lambda_{\min}$ is the smallest eigenvalue, the confidence interval is computed with t-distribution.

R4 In the Hopper domain, as in the original paper [1], the LfL baseline does not recover a correct reward function; this behavior is due to the starting behavior of the simulated robot which at first often falls to the ground; the LfL algorithm tends to consider these absorbing states as good ones. The domain tasks in which LOGEL outperforms LfL are tasks in which monotonous improvement is violated, in which there are absorbing states and where the learner is not a soft policy improvement algorithm (as the gridworld experiment shows). In the behavioral cloning, we *do not assume the knowledge of the policy* model of the learner: for example, in the gridworld environment, the Q-Learning and the SVI learner are ϵ-greedy actors, but we model their policies as Boltzman policies; also, in the appendix, in the autonomous driving experiment, the learner has a rule-based policy, while we use a linear layer to approximate the policy. In Figure 1) we show how a batch-model free IRL algorithm cannot learn using only learning trajectories. We think that the merits of LOGEL go beyond removing the monotonically improving assumption: we provide theoretical insights into the correctness of the proposed method and we empirically show that the proposed method works well also with learner using value-based methods (as value-iteration) and soft policy improvement algorithms, outperforming in most of the experiments the LfL baseline.

[1] Jacq, A., Geist, M., Paiva, A., Pietquin, O. (2019). Learning from a Learner. ICML
[2] Kubala, V., Konidaris, G., Greenwald, A. (2019). Inverse Reinforcement Learning from a Learning Agent, RLDM
[3] Likmeta, A., Metelli, A. M., Tirinzoni, A., Giol, R., Restelli, M., Romano D., (2020) Combining reinforcement learning with rule-based controllers for transparent and general decision-making in autonomous driving, Robotics and Autonomous Systems
[4] Pirotta, M. and Restelli, M. (2016). Inverse Reinforcement Learning through policy gradient minimization. AAAI
[5] Ng et al. "Policy invariance under reward transformations: Theory and application to reward shaping." ICML 1999