Versatile Inverse Reinforcement Learning via Cumulative Rewards

Niklas Freymuth * **Philipp Becker** **Gerhard Neumann**
Autonomous Learning Robots
Karlsruhe Institute of Technology
Karlsruhe, Germany

Abstract

Inverse Reinforcement Learning infers a reward function from expert demonstrations, aiming to encode the behavior and intentions of the expert. Current approaches usually do this with generative and uni-modal models, meaning that they encode a single behavior. In the common setting, where there are various solutions to a problem and the experts show versatile behavior this severely limits the generalization capabilities of these methods. We propose a novel method for Inverse Reinforcement Learning that overcomes these problems by formulating the recovered reward as a sum of iteratively trained discriminators. We show on simulated tasks that our approach is able to recover general, high-quality reward functions and produces policies of the same quality as behavioral cloning approaches designed for versatile behavior.

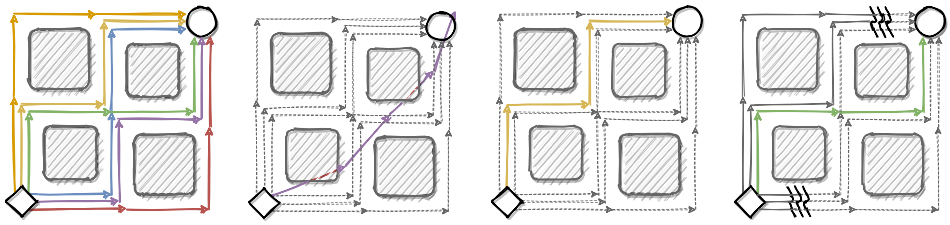
1 Introduction and Related Work

Reinforcement Learning shows great potential for tasks with clearly specified objectives, such as games [22, 29, 8]. Yet, in many real-world scenarios, manually specifying a suitable reward is complicated and can lead to unintended behavior. Thus, Inverse Reinforcement Learning (IRL) [27] aims at recovering a reward from expert demonstrations instead. Many tasks allow various solutions and different experts may chose different approaches, leading to multi-modal, versatile demonstrations. Properly capturing this versatility does not only better reflects the nature of the task but also naturally provides robustness to changes in the environment. See Figure 1 for an example. Many approaches [33, 34, 25, 10, 11] use uni-modal policies together with maximum likelihood objectives, which forces the model to average over modes of data that cannot be represented. Another recent line of work [14, 15] is closely connected to Generative Adversarial Networks [17]. They build on Generative Aderversarial Imitation Learning [19], but reparameterize the discriminator by inserting the learned policy. It can be shown that this modified discriminator then estimates the density of the expert demonstrations, which can be used as a reward. These methods implicitly optimize the reverse KL-divergence allowing them to focus on a single behavior. See e.g., [16, 4] for a detailed discussion. While this prevents averaging over multiple potential solutions, the uni-modal nature of the policy still prevents properly capturing the versatility.

We introduce Versatile Inverse Reinforcement Learning (V-IRL), an approach designed specifically for learning reward functions form highly-versatile demonstrations. We build on recent works in Varitional Inference [5, 6] and most notably on Expected Information Maximization (EIM) [7], a method for density estimation that trains a mixture model where each component is able to focus on a different mode of the given data. EIM is able to reliably model versatile behavior, but does not provide a reward function. We extend EIM to still produce a similar multi-modal policy while also

*correspondence to niklas.freymuth@kit.edu

NeurIPS 2021 Workshop on Robot Learning: Self-Supervised and Lifelong Learning, Virtual, Virtual



(a) (b) (c) (d)

Figure 1: **a)** A versatile path-panning task, where to goal is to find an efficient path from the diamond to the circle. **b)** Maximum likelihood methods average over modes, leading to poor solutions. **c)** Choosing a single behavior via the information projection works, but is not flexible. **d)** V-IRL reconstructs a versatile reward function that allows for good solutions even when some paths are obstructed.

making its reward function explicit, producing a fine-grained and versatile reward in the process. We experiment on diagnostic tasks designed to have versatile solutions. The results show that V-IRL is the only method that is able to consistently capture the multimodal nature of the tasks while recovering a fine-grained reward.

2 Algorithm

We consider sample trajectories $\boldsymbol{x}$ which follow an unknown distribution $p(\boldsymbol{x})$ and are given by the expert. Under the common maximum entropy assumption [33, 18] the expert's reward is given by $R(\boldsymbol{x}) = \log p(\boldsymbol{x}) - c$ for some constant offset c. Recovering the log density of the unknown distribution $p(\boldsymbol{x})$ thus also recovers a reward which explains the experts behavior.

Expected Information Maximization (EIM) [7] iteratively minimizes the reverse KL-Divergence $\mathrm{KL}(q(\boldsymbol{x}) \,||\, p(\boldsymbol{x}))$ between a latent variable policy $q(\boldsymbol{x}) = \int q(\boldsymbol{x}|\boldsymbol{z})\, q(\boldsymbol{z}) d\boldsymbol{z}$ and an unknown distribution $p(\boldsymbol{x})$ of which only samples are available. Similar to EM [13], EIM uses a variational bound to make the optimization tractable. This bound is given by a reformulation of the bound used in [5], $q^*_{t+1}(\boldsymbol{x}) =$

$$\arg\min_{q(\boldsymbol{x})} \mathbb{E}_{q(\boldsymbol{x}|\boldsymbol{z})q(\boldsymbol{z})}\left[-\log \frac{p(\boldsymbol{x})}{q^*_t(\boldsymbol{x})}\right] + \mathrm{KL}(q(\boldsymbol{z})||q^*_t(\boldsymbol{z})) + \mathbb{E}_{q(\boldsymbol{z})}\left[\mathrm{KL}\left(q(\boldsymbol{x}|\boldsymbol{z})\,||q^*_t(\boldsymbol{x}|\boldsymbol{z})\right)\right], \quad (1)$$

where $q^*_t(\boldsymbol{x}|\boldsymbol{z})$ and $q^*_t(\boldsymbol{z})$ denote the model from the previous iteration. Relating Equation 1 to IRL, $q_t(\boldsymbol{x})$ can be seen as an iteratively optimized behavioral cloning policy. The KL penalties enforce that consecutive $q_t(\boldsymbol{x})$ do not change too quickly, acting like trust regions [28]. To use this bound under the assumption that only samples of $p(x)$ are available, EIM employs density ratio estimation techniques [32] to approximate $\log\left(p(\boldsymbol{x})/q^*_t(\boldsymbol{x})\right)$ with a neural network $\phi_t(\boldsymbol{x})$. In the practical implementation of the approach the change of the model between iterations is limited and thus, in each iteration, the density ratio estimator can be reused after a few update steps.

At iteration t, $q^*_{t+1}(\boldsymbol{x})$ in Equation 1 is optimal for $q^*_{t+1}(\boldsymbol{x}) \propto \exp\left(\log q^*_t(\boldsymbol{x}) + \phi_t(\boldsymbol{x})\right)$ [2]. From there, induction over t shows that $\log q^*_{t+1}(\boldsymbol{x}) = \log q^*_0(\boldsymbol{x}) + \sum_{i=0}^{t} \phi_i(\boldsymbol{x}) + c$ for some arbitrary prior $q^*_0(\boldsymbol{x})$ and constant offset c. Assuming convergence at iteration T, plugging this into $R(\boldsymbol{x}) = \log p(\boldsymbol{x}) - c$ yields

$$R(\boldsymbol{x}) = \log q^*_0(\boldsymbol{x}) + \sum_{i=0}^{T-1} \phi_i(\boldsymbol{x}), \quad (2)$$

where we dropped constant offsets as they do not play a role in optimization. Intuitively, each $\phi_t(\boldsymbol{x})$ acts as a change of reward that refines the current recovered reward $q^*_t(\boldsymbol{x})$. By adding more $\phi_t(\boldsymbol{x})$s, this gradually produces a more and more precise estimate of the reward. At convergence, the target distribution $p(\boldsymbol{x}) = q^*_T(\boldsymbol{x})$ is recovered and $\phi_T(\boldsymbol{x}) = 0$ everywhere. Since each $q^*_t(\boldsymbol{x})$ is the accumulation of t different estimators $\phi_i(\boldsymbol{x})$, we call this a *cumulative reward*.

Comparing V-IRL to EIM, we see that V-IRL can represent an arbitrarily complex reward, as each additional $\phi_t(\boldsymbol{x})$ adds capacity to the model. Opposed to this, EIM only recovers a policy and is

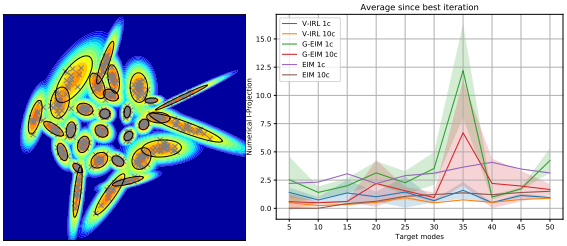
thus limited to the capacity of this policy by construction. V-IRL also acts in an off-policy setting, allowing for large update steps of the policy without destabilizing the training. In contrast, both EIM and generative approaches [14, 15] often need to take sufficiently small steps for the method to converge [4]. Formulating the IRL problem as a sum of changes of rewards is conceptually easier than having a generative reward, because estimating a ratio is easier than estimating a density [32]. Another benefit lies in the multi-modal structure of our approach. Since the log density ratio estimate $\phi_t(\boldsymbol{x})$ gives a strong signal in areas of uncovered expert modes by design, the recovered reward will eventually cover the relevant modes of the expert distribution and naturally prefer those with higher density over lower density ones. If some modes become unavailable at inference time due to changes in dynamics caused by e.g., obstructions, a policy trained on the recovered reward can still use the remaining ones as a reward signal.

Importance Sampling In general, $q_t^*(\boldsymbol{x})$ may become arbitrarily complex and can not easily be samples from. To work around this, we introduce a tractable sampling policy $\tilde{q}_t(\boldsymbol{x})$ that approximates $q_t^*(\boldsymbol{x})$. In our experiments, we use a recent method for variational inference, Variational Inference by Policy Search (VIPS) [5, 6] to train $\tilde{q}_t$ by minimizing $\mathrm{KL}\left(\tilde{q}_t(\boldsymbol{x}) \,||q_t^*(\boldsymbol{x})\right)$. Given $\tilde{q}_t(\boldsymbol{x})$, we employ importance sampling to train a discriminator between expert demonstrations and samples of $\tilde{q}_t(\boldsymbol{x})$ that are weighted by $q_t^*(\boldsymbol{x})$. These samples then act as the importance sampling estimate of samples of $q_t^*(\boldsymbol{x})$. To do this, we minimize the weighted binary cross entropy loss

$$\mathcal{L}_{BCE}(\phi_t, p, \tilde{q}_t, q_t^*) = \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\left[-\log\sigma\left(\phi_t(\boldsymbol{x})\right)\right] + \mathbb{E}_{\boldsymbol{x}\sim\tilde{q}_t(\boldsymbol{x})}\left[-w_t(\boldsymbol{x})\log\left(1-\sigma\left(\phi_t(\boldsymbol{x})\right)\right)\right] \quad (3)$$

with normalized importance weights $w_t(\boldsymbol{x}) = \frac{1}{\int q_t^*(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}} \frac{q_t^*(\boldsymbol{x})}{\tilde{q}_t(\boldsymbol{x})}$ and logits $\phi_t(\boldsymbol{x})$. It has been shown that Equation 3 causes the logits $\phi_t(\boldsymbol{x})$ of the network to recover a log density ratio estimate at convergence [32]. In other words, $\phi_t(\boldsymbol{x}) = \log\left(p(\boldsymbol{x})/q_t^*(\boldsymbol{x})\right)$, which is precisely the change of reward used in Equation 2. Finally, we add the newly obtained $\phi_t(\boldsymbol{x})$ to $q_t^*(\boldsymbol{x})$ to obtain a new reward estimate. We iterate over this until convergence at iteration T, and obtain $q_T^*(\boldsymbol{x})$ as the recovered reward and $\tilde{q}_T(\boldsymbol{x})$ as an optimal policy for this reward. Pseudocode for the approach can be found in Appendix A.

Kernel Density Estimation In practical settings with highly-versatile behavior, $\tilde{q}(\boldsymbol{x})$ is unlikely to cover all relevant modes of $q^*(\boldsymbol{x})$, causing high-variance estimates in the importance sampling procedure and thus a poor coverage of the density ratio estimators and subsequently the reward recovered from them. To prevent this, we follow prior work [14, 15] and enrich the sampling distribution $\tilde{q}_t(\boldsymbol{x})$ with an estimate $\tilde{p}(\boldsymbol{x})$ of the target distribution $p(\boldsymbol{x})$, resulting in a fusion distribution $\mu_t(\boldsymbol{x}) = 0.5\tilde{q}_t(\boldsymbol{x}) + 0.5\tilde{p}(\boldsymbol{x})$. Finn et al. [14] and Fu et al. [15] use a Gaussian for $\tilde{p}(\boldsymbol{x})$. Since this is ill-suited for versatile behavior, we instead resort to a Kernel Density Estimate [26, 24] of the expert samples, using a Gaussian kernel for our estimate. Thus, $\mu_t(\boldsymbol{x})$ covers all modes of $p(\boldsymbol{x})$ by construction. Intuitively, $\tilde{p}(\boldsymbol{x})$ has the purpose of roughly covering $p(\boldsymbol{x})$ to stabilize the training process, while $\tilde{q}_t(\boldsymbol{x})$ is used to explore and approximate some modes of $p(\boldsymbol{x})$ more closely.

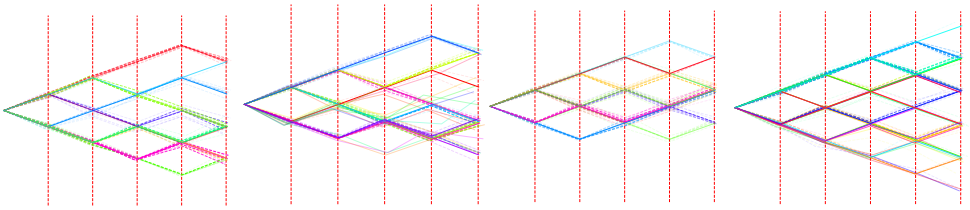


(a) (b)

Figure 2: **a)**: Visualization of a 30-component GMM. The heatmap shows the log-density of the GMM. The ellipsoids are 95%-covariances of the components. Expert demonstrations are shown with a grey '×'. **b)**: Results for different numbers of target components. V-IRL provides a stable solution and is able to represent a larger number of modes than the baselines for the same number of policy components.

3 Experiments

We conduct all our experiments in a non-contextual and episodic setting. We choose Gaussian Mixture Models (GMMs) for the policies of the different methods. Note that extensions to a contextual setting are readily available in the form of Gaussian Mixtures of Experts, which we leave as a promising direction for future work. For hyperparameter settings, refer to Appendix C. We use EIM [7] as a baseline for versatile behavioral cloning, using the log-density of its policy as



(a) G-EIM Training (b) G-EIM Inference (c) V-IRL Training (d) V-IRL Inference

Figure 3: Grid walker experiments. Each walk is made up of $d = 5$ line segments. The solid lines are policy component means, the dotted lines are samples from these components. Lines with a higher opacity correspond to a higher evaluation of the recovered reward. The intermediate targets are denoted by the vertical dotted lines. **a)** a 10-component sampling policy of G-EIM. **b)** a 25-component inference policy trained on the reward recovered by a). **c)** and **d)** repeat this for V-IRL. Both V-IRL and GEIM produce high-quality sampling policies during training. However, only V-IRL is able to also recover a reward function that extends to modes that were not seen during training.

a surrogate reward function. Additionally, we adapt Adversarial Inverse Reinforcement Learning (AIRL) [15] to versatile tasks by combining it with EIM. We use the EIM objective of Equation 1 for the general training procedure, but reparameterize the density ratio according to the AIRL formulation as $\phi_t(\boldsymbol{x}) = \log(\exp(R_t(\boldsymbol{x}))/q_t(\boldsymbol{x}))$. Here, $q_t(\boldsymbol{x})$ is the density of the current sampling policy and $R_t(\boldsymbol{x}) = \log p(\boldsymbol{x})$ is a learned function that recovers the reward up to a constant. We call this approach generative EIM (G-EIM), and also enrich it with the fusion distribution mentioned in Section 2. Note that G-EIM is a novel and interesting approach for versatile IRL in itself.

Gaussian Experiments We start with a versatile toy task to showcase the ability of VIRL to precisely capture highly multi-modal distributions. For this task, the reward is represented by the log-density of a 2-dimensional GMM with m randomly drawn and weighted components, leading to a complex and highly multi-modal target reward. An example for $m = 30$ is given in the left of Figure 2. We evaluate the reverse KL by computing the numerical I-Projection near the ground truth data, comparing EIM, G-EIM and V-IRL with 1 ('1c') and 10 ('10c') GMM components each. We optimize for $m = 50$ target components and evaluate on $m \in \{5, 10, \dots, 50\}$ components. We report the average results between the best evaluated iteration and the final iteration to account for instabilities in the training and evaluation. Results for 5 random seeds can be seen in the right of Figure 2. We find that V-IRL scales gracefully with the number of modes to be covered, comparing favorably to both baselines. Inference experiments on the recovered rewards can be found in Appendix B.

Grid Walker A more realistic highly-versatile task is the path-planning task of Figure 1. We model this task as a walk over d steps of uniform size, where each step is given by an angle. For optimal solutions, the walker can either go up or down in each step. As a result, all efficient solutions lie on a regular grid. Appendix B.2 presents a detailed construction of the task. We train V-IRL and G-EIM with 10-component policies for $d = 5$ path segments, using the negative ELBO (see e.g., [9]) as the optimization metric. The resulting hyperparameters can be seen in Table 3. We then perform inference on the recovered rewards for policies with 25 components. The results are shown in Figure 3. We find that the sampling policies for both V-IRL and G-EIM are versatile and precise. However, the inference policy of G-EIM is mostly limited to behaviors that were found during the training of the recovered reward. Opposed to this, the recovered reward of V-IRL allows for previously unexplored modes to be found by the inference policy. We explore this behavior in more detail in Appendix B.2.

4 Conclusions

We propose a novel approach for Inverse Reinforcement Learning for versatile behavior that recovers a reward by accumulating iteratively trained discriminative models. The key idea of our approach is that every discriminator is trained to represent an optimal change of the previous sum of discriminators, and thus this sum ultimately recovers an appropriate reward. We show that this cumulative reward formulation works well in a variety of versatile tasks, outperforms strong baselines and yields rewards that generalize beyond the capabilities of the sampling policy.

Acknowledgments

The authors acknowledge support by the state of Baden-Württemberg through bwHPC.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

[2] Abbas Abdolmaleki, Rudolf Lioutikov, Jan R Peters, Nuno Lau, Luis Pualo Reis, and Gerhard Neumann. Model-based relative entropy stochastic search. In *Advances in Neural Information Processing Systems*, pages 3537–3545, 2015.

[3] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[4] Oleg Arenz and Gerhard Neumann. Non-adversarial imitation learning and its connections to adversarial methods, 2020.

[5] Oleg Arenz, Gerhard Neumann, Mingjun Zhong, et al. Efficient gradient-free variational inference using policy search. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 234–243. Proceedings of Machine Learning Research, 2018.

[6] Oleg Arenz, Mingjun Zhong, and Gerhard Neumann. Trust-region variational inference with gaussian mixture models. *J. Mach. Learn. Res.*, 21:163–1, 2020.

[7] Philipp Becker, Oleg Arenz, and Gerhard Neumann. Expected information maximization: Using the i-projection for mixture density estimation. In *International Conference on Learning Representations*, 2020.

[8] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning, 2019.

[9] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

[10] Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International conference on machine learning*, pages 783–792. PMLR, 2019.

[11] Daniel S Brown, Wonjoon Goo, and Scott Niekum. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In *Conference on robot learning*, pages 330–359. PMLR, 2020.

[12] François Chollet et al. Keras. `https://keras.io`, 2015.

[13] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[14] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pages 49–58, 2016.

[15] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adverserial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018.

[16] Seyed Kamyar Seyed Ghasemipour, Richard Zemel, and Shixiang Gu. A divergence minimization perspective on imitation learning methods. In *Conference on Robot Learning*, pages 1259–1277. PMLR, 2020.

[17] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[18] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning*, pages 1352–1361. PMLR, 2017.

[19] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in neural information processing systems*, pages 4565–4573, 2016.

[20] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 448–456. JMLR.org, 2015.

[21] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[22] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[23] Robert Nishihara, Iain Murray, and Ryan P Adams. Parallel mcmc with generalized elliptical slice sampling. *The Journal of Machine Learning Research*, 15(1):2087–2112, 2014.

[24] Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.

[25] Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. Maximum margin planning. In *Proceedings of the 23rd international conference on Machine learning*, pages 729–736, 2006.

[26] Murray Rosenblatt. Remarks on some nonparametric estimates of a density function. *Ann. Math. Statist.*, 27(3):832–837, 09 1956. doi: 10.1214/aoms/1177728190. URL `https://doi.org/10.1214/aoms/1177728190`.

[27] Stuart Russell. Learning agents for uncertain environments. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 101–103, 1998.

[28] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.

[29] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[30] Bernard W Silverman. *Density estimation for statistics and data analysis*, volume 26. CRC press, 1986.

[31] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1): 1929–1958, January 2014. ISSN 1532-4435.

[32] Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. *Density ratio estimation in machine learning*. Cambridge University Press, 2012.

[33] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

[34] Brian D. Ziebart, J. Andrew Bagnell, and Anind K. Dey. Modeling interaction via the principle of maximum causal entropy. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 1255–1262. Omnipress, 2010.

Algorithm 1: Versatile Inverse Reinforcement Learning

Input: Expert Samples $\chi_p = \{\boldsymbol{x}_p^{(i)}\}_{i=1\ldots M}$
Input: Reward prior $q_0^*(\boldsymbol{x})$
Input: Initial sampling distribution $\tilde{q}_0(\boldsymbol{x})$
Output: Reward $R(\boldsymbol{x}) = \log p(\boldsymbol{x}) + c$
Output: Optimal policy $\tilde{q}(\boldsymbol{x})$ for this reward

for $t = 0 \ldots$ **do**
  Train $\phi(\boldsymbol{x})$ on $(\tilde{q}_t, q_t^*, \chi_p)$ using weighted binary cross-entropy // Eq. 3
  Set $q_{t+1}^*(\boldsymbol{x}) = q_t^*(\boldsymbol{x}) \cdot \exp(\phi(\boldsymbol{x}))$
  Update $\tilde{q}_{t+1}(\boldsymbol{x})$ by minimizing $\mathrm{KL}(\tilde{q}_t(\boldsymbol{x}) \,||\, q_t^*(\boldsymbol{x}))$ using e.g., VIPS
end
return $\log q_{t+1}^*(\boldsymbol{x})$, $\tilde{q}_{t+1}(\boldsymbol{x})$

A Pseudocode for V-IRL

Algorithm 1 provides pseudocode for V-IRL.

B Additional Experiments

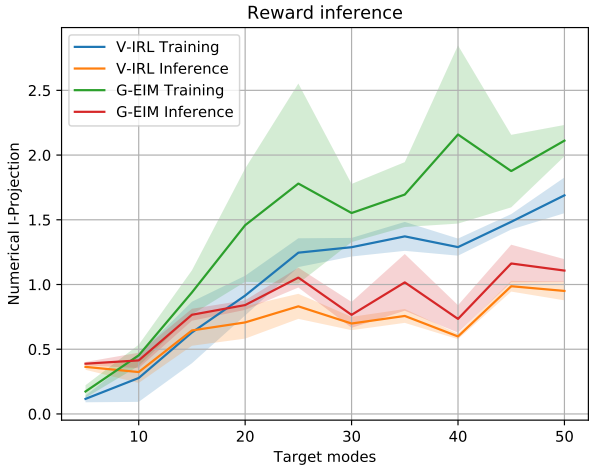


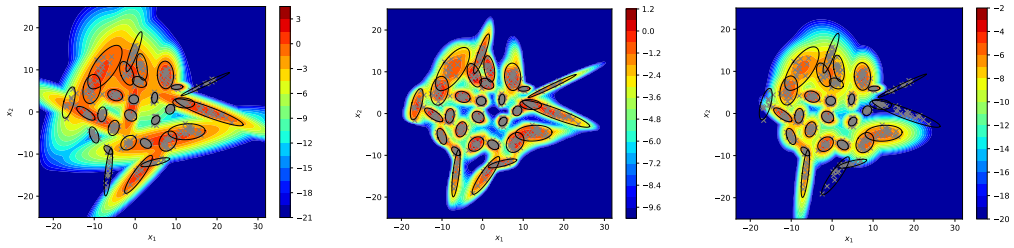
Figure 4: Numerical I-projection of inference policies trained on the best recovered rewards for V-IRL on the random Gaussian task. 'Inference' indicates VIPS with a number of components equal to the number of target modes trained on the final recovered reward. 'Training' refers to the 10-component learner policy.

B.1 Random Gaussians

We use the reward recovered by the 10-component versions of both V-IRL and G-EIM to train a newly initialized forward policy using VIPS with as many components as we have Gaussians in the task. We compare the numerical I-Projection between the 10-component learner policies of the best reward iteration and the newly trained m-component policies evaluated at their best iteration in Figure 4. Both methods generate a useful reward function that encodes more information than the policy that is used to produce it. However, V-IRL generally shows less variance in its solutions, and produces slightly better rewards overall. A qualitative comparison between the sampling policy, the recovered reward and the inference policy for V-IRL is given in Figure 5.

B.2 Grid Walker

Construction We model the task as an agent that takes steps of length 1 in a planar space, starting at $(0, 0)^T$. The action space is parameterized by an angle for each step, and the dimensionality



(a) (b) (c)

Figure 5: Comparison of contour plots of **a)** the log-density of the sampling policy used by V-IRL, **b)** the average reward recovered by V-IRL over 5 runs, and **c)** the log-density of a VIPS policy with 30 components that is fit on the recovered reward of b). For all figures, the black ellipsoids are covariances of the individual expert components scaled to include 95% of samples. 500 randomly chosen expert samples are marked with a grey '$\times$'.

of the task corresponds to the number of steps taken. We only consider the first half of the path planning example of Figure 1 for our environment. Note that this encodes most of the versatility in the behavior of the agent. To represent the choices of the agent more clearly, we rotate the example by $45°$, aligning the waypoints along equidistant vertical lines. In this representation, all efficient paths lie on a grid inside a $90°$ cone from the origin to the positive x-axis. The result is visualized on the left of Figure 6.

To construct the actual task, we define $\boldsymbol{x}$ to be the concatenation of absolute angles for the steps. That is, the position $h_i(\boldsymbol{x}) \in \mathbb{R}^2$ for a sample $\boldsymbol{x} = (x_1, \ldots, x_d)^T$ at step i is given by

$$h_i(\boldsymbol{x}) = h_{i-1}(\boldsymbol{x}) + \begin{pmatrix} \cos(x_i) \\ \sin(x_i) \end{pmatrix} = \begin{pmatrix} \sum_{j=1}^{i} (\cos(x_j)) \\ \sum_{j=1}^{i} (\sin(x_j)) \end{pmatrix},$$

where $h_0(\boldsymbol{x}) = (0, 0)^T$ for all $\boldsymbol{x}$. We then introduce d equidistant 1-dimensional Gaussians centered on each line and calculate the likelihood of a walk as the product of the likelihood of each of its steps being drawn from its respective Gaussian. This implicitly creates a grid-like structure, where for each segment the optimal behavior is to either 'go up' or 'go down'. We set the distance of two consecutive Gaussians to 0.8. This causes the waypoints of a step to be positioned at a relative difference of the y-axis of $\{+0.6, -0.6\}$ compared to the previous step. We use a variance of $1e-3$ for each Gaussian. The right side of Figure 6 shows 100 random samples for $d = 5$. The ground truth reward has 2^d distinct modes, each corresponding to one combination of going 'up' or 'down' d times. All solutions have equal probability due to the symmetry of the task.

Reward Comparison To explain the results of Figure 3, we analyse how well target modes that are unexplored by the sampling policy of V-IRL and G-EIM are represented by their recovered rewards. The steps have length 1 and two consecutive target lines a distance of 0.8. The centers of the modes are therefore given by

$$\chi^+ = \{-\cos^{-1}(0.8), \cos^{-1}(0.8)\}^d.$$

We use this to compare the evaluations of a recovered reward at the center of each mode with evaluations at random points. If the recovered reward represents these unexplored modes well, it will evaluate to higher values for the modes than it will for random samples. We evaluate 5 trials of V-IRL and G-EIM with sampling policies with 10 components each. We compare the 32 target centers and 100 negative samples randomly drawn from $\left[-1.2\cos^{-1}(0.8), 1.2\cos^{-1}(0.8)\right]^d$. The normalized results are shown in Figure 7. V-IRL is able to represent both explored and unexplored mode centers well. Opposed to this, G-EIM clearly prefers explored modes, failing to distinguish between them and unexplored ones. We hypothesize that this is due to V-IRL making use of the Kernel Density Estimate to roughly model the full reward distribution. While G-EIM also uses the Kernel Density Estimation, its structure causes expert demonstrations that are unexplored by the learner policy to be less important during the training of its reward function. This can be seen in the following equation

$$\phi_t(\boldsymbol{x}) = \log \frac{\exp(R_t(\boldsymbol{x}))}{q_t(\boldsymbol{x})},$$

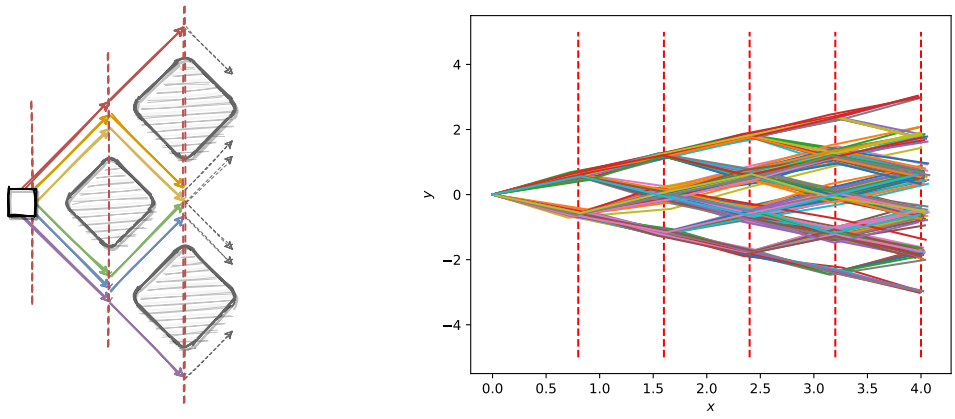


Figure 6: Grid walker construction. Left: Rotating the 'first half' of the introductory example causes all efficient waypoints to lie on equidistant vertical lines. Identical steps share the same waypoints. Right: 200 expert samples of the grid walker task for $d = 5$ steps. All samples follow one of 2^5 possible paths. These paths are implicitly encoded through 1-dimensional Gaussians, which are visualized via the red dotted lines.

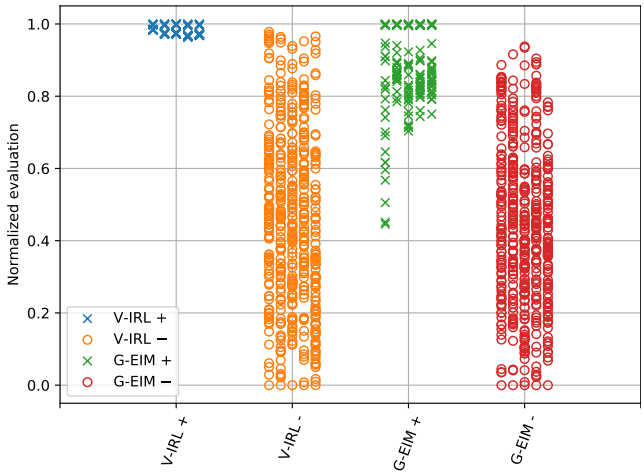


Figure 7: Normalized evaluation of the recovered reward on the grid walker task for $d = 5$ for V-IRL and G-EIM. The '$+$'/'$\times$' denotes mode centers, '$-$'/'$\circ$' randomly drawn samples. The i-th column of the same color corresponds to the i-th random trials. While V-IRL represents all target modes due to its KDE, G-EIM represents those explored by its sampling distribution and only approximates the others.

which states that the evaluation of G-EIM for a sample $\boldsymbol{x}$ negatively depends on the log density of its sampling policy. Linking this to the introductory example, V-IRL is able to provide paths for the path-planning task even if these paths were not explored by its sampling policy during training. If the explored paths were to become unavailable, the others would still provide valid solutions.

C Hyperparameters

All discriminators are feedforward neural networks and implemented using Tensorflow [1] and Keras [12]. The networks are trained using Adam [21], and we employ employing Dropout [31], L2 regularization and Batch Normalization [20] to avoid overfitting.

Each task uses 8000 expert demonstrations that are drawn from the ground truth reward of the task using Elliptical Slice Sampling [23]. We train on 8000 expert demonstrations for each task. Policy updates are performed using default hyperparameters for, and we resort to Optuna [3] for optimizing

the discriminator architecture for all methods. We only optimize the number of policy update steps for V-IRL, as the other methods have shown to be very unstable for larger numbers of update steps. Hyperparameter ranges are given in Table 1.

Table 1: Ranges of the optimized hyperparameters. The parameters are optimized independently for each task and method, unless noted otherwise.

Description	Range	Usage
Network layers	$[2, 4]$	Neural Network
Layer size	$2^{[3,8]}$	Neural Network
Batch normalization	[False, True]	Neural Network
Learning rate	$[5.0e-5, 1.0e-3]$	Neural Network
Dropout	$[0, 0.5]$	Neural Network
L2-Norm	$[0, 1]$	Neural Network
Bandwidth	$(0, 1]$	KDE
Policy update steps	$[1, 200]$	Learner Policy

Chosen Hyperparameters This section lists the hyperparameters chosen by Optuna for all experiments and optimized methods.

Table 2: Hyperparameters used for the random Gaussian experiments. Values with a '$*$' are not optimized and instead set to a default value. We fix the bandwidth according to Silverman's rule [30] for simplicity.

Parameters	V-IRL 1c	V-IRL 10c	G-EIM 1c	G-EIM 10c	EIM 1c	EIM 10c
Network Layers	4	4	4	3	4	4
Layer size	256	256	256	256	256	256
Batch norm	$True$	$False$	$True$	$False$	$False$	$True$
Learning Rate	$2.68e-4$	$3.81e-4$	$1.09e-4$	$6.22e-4$	$8.43e-4$	$3.28e-4$
Dropout	0.01	0.01	0	0	0.02	0.02
L2-Norm	0	$5.96e-8$	0	$4.77e-7$	$2.98e-8$	$9.54e-7$
Bandwidth	0.215^*	0.215^*	0.215^*	0.215^*	X	X
Update steps	1	162	5^*	5^*	1^*	1^*

Table 3: Hyperparameters used for the grid walker experiments. Values with a '$*$' are not optimized and instead set to a default value.

Parameters	V-IRL	G-EIM
Network Layers	3	3
Layer size	256	64
Batch norm	$True$	$False$
Learning Rate	$4.25e-4$	$4.38e-4$
Dropout	0.07	0
L2-Norm	$4.77e-7$	$2.44e-4$
Bandwidth	0.136	0.864
Update Steps	63	5^*