

Supplementary Materials

Anonymous Author(s)

Affiliation

Address

email

1 Further discussions

2 1.1 Candidate feature set

We presented an approach for learning interpretable reward models by automatically selecting non-linear functions as reward components. In this work, the candidate feature set was formed from the higher moments of state features. However, the candidate feature set could also incorporate other basis functions such as radial basis functions, Fourier series, etc. This allows for the incorporation of task-specific features that may be relevant to the reward function. Our proposed feature selection method for the reward model scales linearly with additional features. Additionally, the computation of feature expectations for a feature can be vectorized for the given data. Therefore, feature selection is computationally efficient and allows for an increase in the candidate feature set.

11 1.2 Comparison to Adversarial Methods

We have demonstrated the improvement of the policy trained with the proposed reward model compared to the adversarial inverse reinforcement learning method (AIRL) with a significantly smaller number of reward model parameters. Adversarial methods train a neural network to predict if samples are from one of two classes: expert or non-expert data. However, within expert data, adversarial methods are not trained to differentiate between expert samples if some are more probable or better than others. In contrast, using the proposed unsupervised feature selection, we identify the features that help to predict the probability of trajectories. Therefore, we anticipate that these features have stronger predictive power for the reward signal.

20 2 Limitations

One limitation of our work is the linear model for the reward function. However, our approach may be extended to nonlinear reward functions. In that case, we can incorporate mutual information between feature expectations and trajectories instead of a linear correlation metric for feature selection. We aim to address this extension in future works.

Another limitation of our work is the experimental design. We have not conducted experiments with real robots and human demonstrations. However, we validated our method using complex robotic environments. In future works, we plan to apply the method to real-world tasks and learn rewards from demonstrations.

29 3 Details of Experiments

Below are the short descriptions of the benchmark tasks and the corresponding *ground-truth* reward functions:

1. **Hopper-v4** The task of this one-legged robot is to move forward by applying torques to three hinges. The **true** reward is calculated using distance moved forward and high torque values are penalised.
2. **Ant-v4** The task of this robot consisting of three links is to move forward by applying torques to rotors. The **true** reward is calculated using distance moved forward and high torque values are penalised.
3. **Walker2d-v4** The task of this robot is to move forward by applying torques to its six hinges. The **true** reward is calculated using distance moved forward and high torque values are penalised.
4. **Half-Cheetah-v4** The task of this two-dimensional robot consisting of 9 body parts is to move forward by applying torques to joints, hinges, and feet. The **true** reward is calculated using distance moved forward and high torque values are penalised.

Tasks	$\dim(\mathcal{S})$	$\dim(\mathcal{A})$
Hopper	11	3
Walker	17	6
HalfCheetah	17	6
Ant	27	8

Table 1: State-action dimensionality of the tasks

An additional challenge in reward learning is that the features composing the *ground-truth* rewards are not directly available. For example, the direct components of the ground-truth reward, particularly the x -coordinate and torque values, are hidden from the states and, consequently, from the reward model. Instead, we infer the reward from the available indirect features like velocity or joint angles.

4 Missing Proofs

1. Computation of trajectory probability

The probability of trajectory $\tau = \{s_1, s_2, \dots, s_n\}$ is given as

$$P(\tau) = P(s_1)P(s_2|s_1)P(s_3|s_2) \dots P(s_n|s_{n-1}). \quad (1)$$

After applying the logarithm to both sides and simplifying the terms, we get

$$\log P(\tau) = \log P(s_1) + \sum_{t=1}^{n-1} (\log P(s_t, s_{t+1}) - \log P(s_t)) \quad (2)$$

Here, conditional state probability $P(s_{t+1}|s_t)$ can be written as $P(s_{t+1}|s_t) = P(s_t, s_{t+1})/P(s_t)$. Then:

$$P(\tau) = P(s_1) \frac{P(s_1, s_2)}{P(s_1)} \frac{P(s_2, s_3)}{P(s_2)} \dots \frac{P(s_{n-1}, s_n)}{P(s_{n-1})} \quad (3)$$

After applying the logarithm to both sides, we get:

$$\log P(\tau) = \log P(s_1) + \log \frac{P(s_1, s_2)}{P(s_1)} + \log \frac{P(s_2, s_3)}{P(s_2)} + \dots + \log \frac{P(s_{n-1}, s_n)}{P(s_{n-1})} \quad (4)$$

$$= \log P(s_1) + \sum_{t=1}^{n-1} \log \frac{P(s_t, s_{t+1})}{P(s_t)} \quad (5)$$

$$= \log P(s_1) + \sum_{t=1}^{n-1} (\log P(s_t, s_{t+1}) - \log P(s_t)) \quad (6)$$

$$(7)$$

56

2. Learning of reward weights:

$$\theta^* = \arg \max_{\theta} \sum_{\tau \in D} \log P(\tau|\theta) \quad (8)$$

57

To find the optimal weights θ , we take the derivative of log-likelihood that is given by:

$$\nabla L(\theta) = \nabla \sum_{\tau \in D} \log \left(\frac{e^{\theta^T \phi(\tau)}}{Z(\theta)} \right) \quad (9)$$

$$= \nabla \sum_{\tau \in D} (\theta^T \phi(\tau) - \log Z(\theta)) \quad (10)$$

$$= \sum_{\tau \in D} \phi(\tau) - \nabla \log Z(\theta) \quad (11)$$

$$= \sum_{\tau \in D} \phi(\tau) - \frac{1}{Z(\theta)} \sum_{\tau' \in \mathcal{T}} \phi(\tau') e^{\theta^T \phi(\tau')} \quad (12)$$

$$= \sum_{\tau \in D} \phi(\tau) - \sum_{\tau \in \mathcal{T}} p(\tau|\theta) \phi(\tau) \quad (13)$$

$$\approx \mu_e - \sum_{i=1}^m \sum_{\mathbf{s}_i \in \tau} \phi(\mathbf{s}_i) \quad (14)$$

58

5 Hyperparameters

	HalfCheetah	Walker	Hopper	Ant
Number of expert trajs	150	150	100	150
Epochs	50	50	50	50
Learning rate	0.05	0.03	0.05	0.03
Learning decay	0.985	0.99	0.99	0.985
Total RL timesteps	1e6	1e6	1e6	1e6
Discount factor	0.99	0.99	0.99	0.99
Number of parallel envs	4	4	4	4
Batch size	1024	256	512	256
Number of nodes (MLP)	256x2	256x2	256x2	256x2

Table 2: Training hyperparameters.