Appendix:

B-Pref: Benchmarking Preference-Based Reinforcement Learning

A Preliminaries: Reinforcement learning algorithms

Proximal policy optimization. Proximal policy optimization (PPO) [54] is a state-of-the-art on-policy algorithm for learning a continuous or discrete control policy, $\pi_\phi(\mathbf{a}|\mathbf{s})$. PPO forms policy gradients using action-advantages, $A_t = A^\pi(\mathbf{a}_t, \mathbf{s}_t) = Q^\theta(\mathbf{a}_t, \mathbf{s}_t) - V^\pi(\mathbf{s}_t)$, and minimizes a clipped-ratio loss over minibatches of recent experience (collected under $\pi_{\bar{\phi}}$):

$$\mathcal{L}_\pi^{\texttt{PPO}} = -\mathbb{E}_{\tau_t \sim \pi}\left[\min\left(\rho_t(\phi)A_t,\ \text{clip}(\rho_t(\phi), 1-\epsilon, 1+\epsilon)A_t\right)\right], \quad \rho_t(\phi) = \frac{\pi_\phi(\mathbf{a}_t|\mathbf{s}_t)}{\pi_{\phi_{old}}(\mathbf{a}_t|\mathbf{s}_t)}, \tag{4}$$

where $\bar{\phi}$ are the delayed parameters and ϵ is a clip ratio. Our PPO agents learn a state-value estimator, $V_\theta(\mathbf{s})$, which is regressed against a target of discounted returns and used with Generalized Advantage Estimation [53]:

$$\mathcal{L}_V^{\texttt{PPO}}(\theta) = \mathbb{E}_{\tau_t \sim \pi}\left[\left(V_\theta(\mathbf{s}_t) - V_{\bar{\theta}}(\mathbf{s}_t)\right)^2\right]. \tag{5}$$

PPO is more robust to the non-stationarity in rewards caused by online learning.

Soft actor-critic. Soft actor-critic (SAC) [27] is an off-policy actor-critic method based on the maximum entropy RL framework [77], which encourages exploration and greater robustness to noise by maximizing a weighted objective of the reward and the policy entropy. To update the parameters, SAC alternates between a soft policy evaluation and a soft policy improvement. At the soft policy evaluation step, a soft Q-function, which is modeled as a neural network with parameters θ, is updated by minimizing the following soft Bellman residual:

$$\begin{aligned} &\mathcal{L}_Q^{\texttt{SAC}} = \mathbb{E}_{\tau_t \sim \mathcal{B}}\Big[\left(Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - r_t - \bar{\gamma}\bar{V}(\mathbf{s}_{t+1})\right)^2\Big], \\ &\text{with} \quad \bar{V}(\mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t \sim \pi_\phi}\big[Q_{\bar{\theta}}(\mathbf{s}_t, \mathbf{a}_t) - \alpha \log \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)\big], \end{aligned} \tag{6}$$

where $\tau_t = (\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, r_t)$ is a transition, $\mathcal{B}$ is a replay buffer, $\bar{\theta}$ are the delayed parameters, and α is a temperature parameter. At the soft policy improvement step, the policy π_ϕ is updated by minimizing the following objective:

$$\mathcal{L}_\pi^{\texttt{SAC}} = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{B}, \mathbf{a}_t \sim \pi_\phi}\Big[\alpha \log \pi_\phi(\mathbf{a}_t|\mathbf{s}_t) - Q_\theta(\mathbf{s}_t, \mathbf{a}_t)\Big]. \tag{7}$$

SAC enjoys good sample-efficiency relative to its on-policy counterparts by reusing its past experiences. However, for the same reason, SAC is not robust to a non-stationary reward function.

Algorithm 2 `EXPLORE`: Unsupervised exploration

Initialize parameters of π_ϕ and a buffer $\mathcal{B} \leftarrow \emptyset$
for each iteration **do**
  for each timestep t **do**
    Collect $\mathbf{s}_{t+1}$ by taking $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$
    Compute intrinsic reward $r_t^{\texttt{int}} \leftarrow r^{\texttt{int}}(\mathbf{s}_t)$ as in (8)
    Store transitions $\mathcal{B} \leftarrow \mathcal{B} \cup \{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, r_t^{\texttt{int}})\}$
  end for
  for each gradient step **do**
    Sample minibatch $\{(\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}_{j+1}, r_j^{\texttt{int}})\}_{j=1}^B \sim \mathcal{B}$
    Optimize RL objective function with respect to ϕ
  end for
end for
return $\mathcal{B}, \pi_\phi$

Algorithm 3 Preference-based RL with reward learning

Require: frequency of teacher feedback K
Require: number of queries $N_{\texttt{query}}$ per feedback session
Initialize parameters of π_ϕ, $\widehat{r}_\psi$, a dataset of preferences $\mathcal{D} \leftarrow \emptyset$, and a buffer $\mathcal{B} \leftarrow \emptyset$
// EXPLORATION PHASE
$\mathcal{B}, \pi_\phi \leftarrow$ `EXPLORE`() in Algorithm 2
for each iteration **do**
  // REWARD LEARNING
  if iteration % $K == 0$ **then**
    for m in $1 \dots N_{\texttt{query}}$ **do**
      $(\sigma^0, \sigma^1) \sim$ `SAMPLE()` (see Section C) and query `SimTeacher` in Algorithm 1 for y
      Store preference $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\sigma^0, \sigma^1, y)\}$
    end for
    for each gradient step **do**
      Sample minibatch $\{(\sigma^0, \sigma^1, y)_j\}_{j=1}^{D} \sim \mathcal{D}$ and optimize $\mathcal{L}^{\texttt{Reward}}$ in (3) with respect to ψ
    end for
  end if
  // POLICY LEARNING
  for each timestep t **do**
    Collect $\mathbf{s}_{t+1}$ by taking $\mathbf{a}_t \sim \pi(\mathbf{a}_t|\mathbf{s}_t)$ and store transitions $\mathcal{B} \leftarrow \mathcal{B} \cup \{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, \widehat{r}_\psi(\mathbf{s}_t))\}$
  end for
  for each gradient step **do**
    Sample random minibatch $\{(\tau_j)\}_{j=1}^{B} \sim \mathcal{B}$ and optimize RL objective function with respect to ϕ
  end for
  Reset $\mathcal{B} \leftarrow \emptyset$ if on-policy RL algorithm is used
end for

B Preference-based reinforcement learning

Algorithm 3 summarizes the full procedure of preference-based RL methods that we consider in this paper. We also utilize unsupervised pre-training which encourages our agent to visit a wider range of states by using the state entropy $\mathcal{H}(\mathbf{s}) = -\mathbb{E}_{\mathbf{s} \sim p(\mathbf{s})}[\log p(\mathbf{s})]$ as an intrinsic reward [42, 29, 40, 56]. By following Lee et al. [39], we define the intrinsic reward of the current state $\mathbf{s}_t$ as follows:

$$r^{\texttt{int}}(\mathbf{s}_t) = \log(||\mathbf{s}_t - \mathbf{s}_t^k||), \tag{8}$$

where $\mathbf{s}_i^k$ is the k-NN of $\mathbf{s}_i$ within a set $\{\mathbf{s}_i\}_{i=1}^{N}$. The full procedure of unsupervised pre-training is summarized in Algorithm 2.

C Sampling schemes

We consider the following sampling schemes in this paper:

- *Uniform sampling*: We pick $N_{\texttt{query}}$ pairs of segments uniformly at random from the buffer $\mathcal{B}$.
- *Disagreement*: We first generate the initial batch of $N_{\texttt{inter}}$ pairs of segments $\mathcal{G}_{\texttt{init}}$ uniformly at random, measure the variance across ensemble of preference predictors $\{P_{\psi_i}[\sigma^1 \succ \sigma^0]\}_{i=1}^{N_{\texttt{en}}}$, and then select the $N_{\texttt{query}}$ pairs of segments with high uncertainty.
- *Entropy*: We first generate the initial batch of $N_{\texttt{inter}}$ pairs of segments $\mathcal{G}_{\texttt{init}}$ uniformly at random, measure the entropy of a single preference predictor $\mathcal{H}(P_\psi)$, and then select the $N_{\texttt{query}}$ pairs of segments with high uncertainty.
- *Coverage*: From the initial batch $\mathcal{G}_{\texttt{init}}$, we choose center points, which increase the dissimilarity between the selected queries. Specifically, we concatenate the states of segments, i.e., $\mathbf{s}_{\texttt{concat}} = \texttt{Concat}(\mathbf{s}_{k+1}^0, \cdots, \mathbf{s}_{k+H}^0, \mathbf{s}_{k+1}^1, \cdots, \mathbf{s}_{k+H}^1)$[7] and measure the Euclidean distance. Then, we choose $N_{\texttt{query}}$ center points such that the largest distance between a data point and its nearest center is minimized using a greedy selection strategy.

[7] Concatenating states would not be an optimal choice because it is not permutation-invariant, which is also not handled in the prior work [11]. However, we expect that an issue from permutation-variance is not significant because a probability to sample two segments in a different order is very low. However, it is an interesting future direction to explore to address this limitation.

- *Disagreement + Coverage*: We first select the $N_{\texttt{inter}}$ pairs of segments $\mathcal{G}_{\texttt{un}}$, using the disagreement sampling, where $N_{\texttt{init}} > N_{\texttt{inter}}$, and then choose $N_{\texttt{query}}$ center points from $\mathcal{G}_{\texttt{un}}$.
- *Entropy + Coverage*: We first select the $N_{\texttt{inter}}$ pairs of segments $\mathcal{G}_{\texttt{un}}$, using the entropy sampling, and then choose $N_{\texttt{query}}$ center points from $\mathcal{G}_{\texttt{un}}$.

D Experimental Details

Training details. We use PEBBLE and PrefPPO[8] with a full list of hyperparameters in Table 1 and Table 2, respectively. We pre-train an agent for 10K timesteps and 32K timesteps for PEBBLE and PrefPPO, respectively.

Hyperparameter	Value	Hyperparameter	Value
Initial temperature	0.1	Hidden units per each layer	1024 (DMControl), 256 (Meta-world)
Segment of length	50 (DMControl), 25 (Meta-world)	# of layers	2 (DMControl), 3 (Meta-world)
Learning rate	0.0003 (Meta-world)	Batch Size	1024 (DMControl), 512 (Meta-world)
	0.0001 (Quadruped), 0.0005 (Walker)	Optimizer	Adam [33]
Critic target update freq	2	Critic EMA τ	0.005
(β_1, β_2)	$(.9, .999)$	Discount $\bar{\gamma}$	.99
Frequency of feedback	5000 (Meta-world), 20000 (Walker)	Maximum budget /	$2000/200, 1000/100, 500/50$ (DMControl)
	30000 (Quadruped)	# of queries per session	$20K/100, 10K/50$ (Meta-world)

Table 1: Hyperparameters of the PEBBLE algorithm.

Hyperparameter	Value	Hyperparameter	Value
GAE parameter λ	0.9 (Quadruped), 0.92 (otherwise)	Hidden units per each layer	256
Segment of length	50 (DMControl), 25 (Meta-world)	# of layers	3
Learning rate	0.0003 (Meta-world)	Batch Size	64 (Walker), 256 (Sweep Into)
	$5e^{-5}$ (DMControl)		128 (Quadruped, Button)
Discount $\bar{\gamma}$	.99	Frequency of feedback	32000 (DMControl)
# of environments per worker	8 (Button), 16 (Quadruped),	PPO clip range	0.4
	32 (Walker, Sweep Into)	Entropy bonus	0.0
# of timesteprs per rollout	500 (DMControl)	Maximum budget /	$2000/200, 1000/100$ (DMControl)
	250 (Meta-world)	# of queries per session	

Table 2: Hyperparameters of the PrefPPO algorithm.

Reward model. For the reward model, we use a three-layer neural network with 256 hidden units each, using leaky ReLUs. To improve the stability in reward learning, we use an ensemble of three reward models, and bound the output using tanh function. Each model is trained by optimizing the cross-entropy loss defined in (3) using ADAM learning rule [33] with the initial learning rate of 0.0003.

Simulated human teachers. For all experiments, To evaluate the robustness, we evaluate against the following simulated human teachers with different hyperparameters:

- Oracle: $\texttt{SimTeacher}\Big(\beta \to \infty\,, \gamma = 1, \epsilon = 0, \delta_{\texttt{skip}} = 0, \delta_{\texttt{equal}} = 0\Big)$
- Stoc: $\texttt{SimTeacher}\Big(\beta = 1\,, \gamma = 1, \epsilon = 0, \delta_{\texttt{skip}} = 0, \delta_{\texttt{equal}} = 0\Big)$
- Mistake: $\texttt{SimTeacher}\Big(\beta \to \infty\,, \gamma = 1,\ \epsilon = 0.1\,, \delta_{\texttt{skip}} = 0, \delta_{\texttt{equal}} = 0\Big)$
- Skip: $\texttt{SimTeacher}\Big(\beta \to \infty\,, \gamma = 1, \epsilon = 0,\ \delta_{\texttt{skip}} = \delta_{\texttt{adapt}}(\epsilon_{\texttt{adapt}}, t)\,, \delta_{\texttt{equal}} = 0\Big)$
- Equal: $\texttt{SimTeacher}\Big(\beta \to \infty\,, \gamma = 1, \epsilon = 0, \delta_{\texttt{skip}} = 0,\ \delta_{\texttt{adapt}} = \delta_{\texttt{skip}}(\epsilon_{\texttt{adapt}}, t)\Big)$
- Myopic: $\texttt{SimTeacher}\Big(\beta \to \infty\,,\ \gamma = 0.9\,, \epsilon = 0, \delta_{\texttt{skip}} = 0, \delta_{\texttt{equal}} = 0\Big)$

Because each environment has a different scale of ground truth rewards, it is hard to design standardized Skip and Equal teachers using hard threshold. To address this issue, we use an adaptive

[8] For Meta-world, the frequency is chosen from $\{8K, 16K, 32K, 64K\}$ and # of queries per session is chosen from $\{50, 100, 250, 500, 1000\}$.

threshold, which is defined as follows:

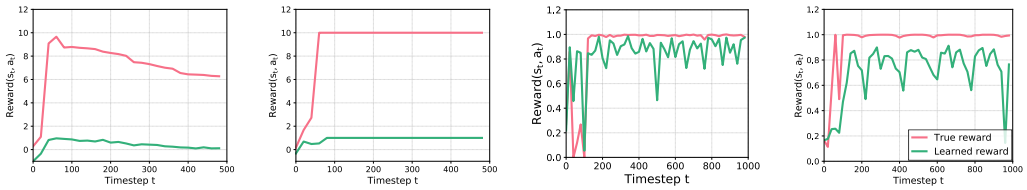
$$\delta_{\texttt{adapt}}(\epsilon_{\texttt{adapt}}, t) = \frac{H}{T} R_{\texttt{avg}}(\phi_t)\epsilon_{\texttt{adapt}}, \tag{9}$$

where t is the current timestep, $\epsilon_{\texttt{adapt}} \in [0, 1]$ is hyperparameters to control the threshold, T is the episode length, H is a length of segment and $R_{\texttt{avg}}(\pi_t)$ is the average returns of current policy with the parameters π_t. By adaptively rescaling the threshold based on the performance of agent, a teacher skips queries or provides uniform labels (i.e., $y = (0.5, 0.5)$. For all experiments, we choose $\epsilon_{\texttt{adapt}} = 0.1$.

E Additional experimental results

Reward analysis. Figure 6 shows the learned reward function optimized by PEBBLE on the oracle teacher in all tested environments. Because we bound the output using tanh function, the scale is different with the ground truth reward but the learned reward function is reasonably well-aligned.

Regularization for handling corrupted labels. To improve the robustness to corrupted labels, we apply the label smoothing [64]. By following Christiano et al. [19], Ibarz et al. [31], we use a soft label $\widehat{y} = 0.9 * y + 0.05$ for the cross-entropy computation. As shown in Figure 7, we find that the gains from label smoothing are marginal.



(a) Button Press (b) Sweep Into (c) Quadruped (d) Walker

Figure 6: Time series of learned reward function (green) and the ground truth reward (red) using rollouts from a policy optimized by PEBBLE.

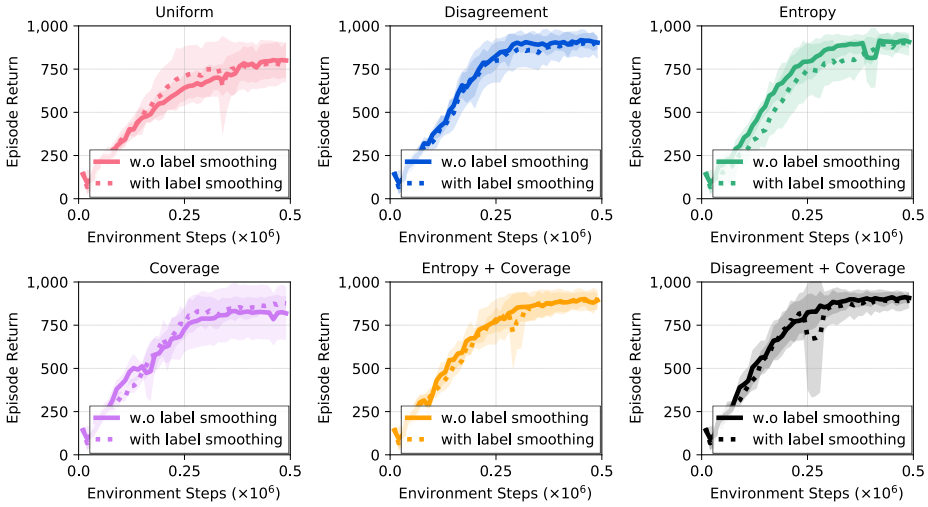


Figure 7: Learning curves of PEBBLE with 500 queries on Walker on the Mistake teacher. The solid line and shaded regions represent the mean and standard deviation, respectively, across five runs.

F Learning curves

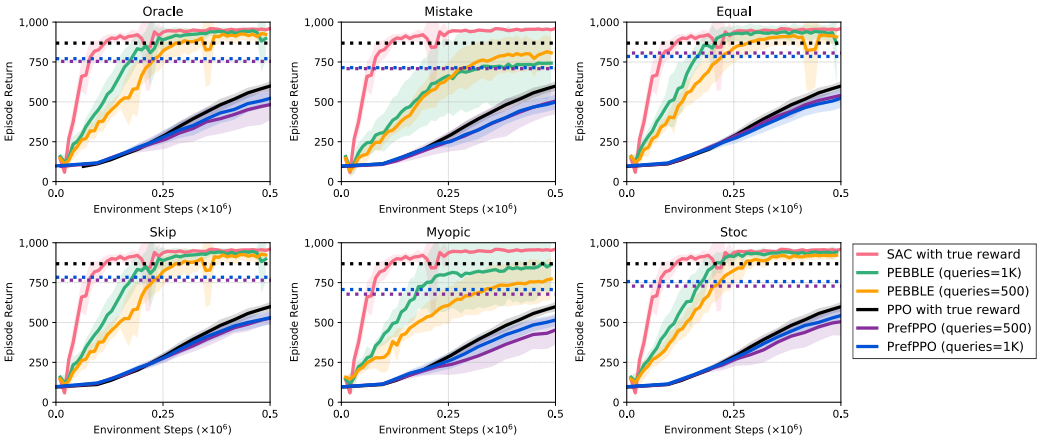


Figure 8: Learning curves of PEBBLE and PrefPPO on Walker-walk as measured on the ground truth reward. The solid line and shaded regions represent the mean and standard deviation, respectively, across ten runs. Asymptotic performance of PPO and PrefPPO is indicated by dotted lines of the corresponding color.

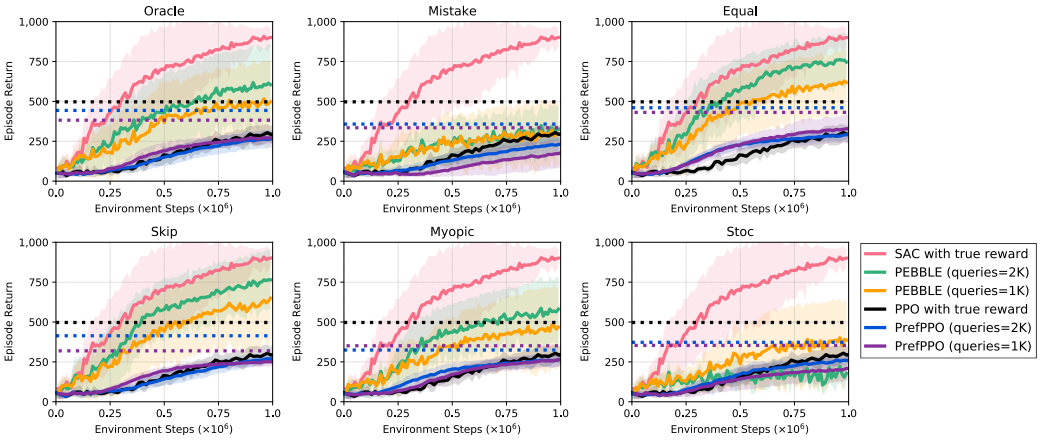


Figure 9: Learning curves of PEBBLE and PrefPPO on Quadruped-walk as measured on the ground truth reward. The solid line and shaded regions represent the mean and standard deviation, respectively, across ten runs. Asymptotic performance of PPO and PrefPPO is indicated by dotted lines of the corresponding color.

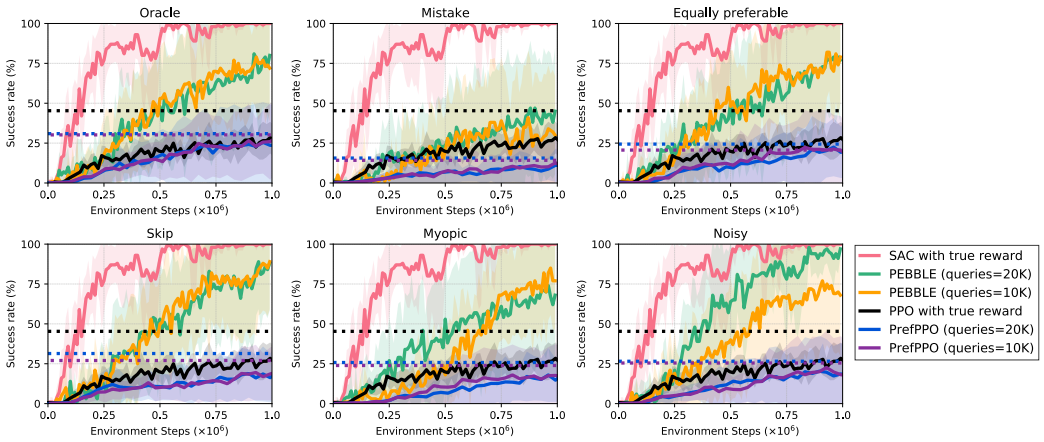


Figure 10: Learning curves of PEBBLE and PrefPPO on Sweep Into as measured on the success rate. The solid line and shaded regions represent the mean and standard deviation, respectively, across ten runs. Asymptotic performance of PPO and PrefPPO is indicated by dotted lines of the corresponding color.

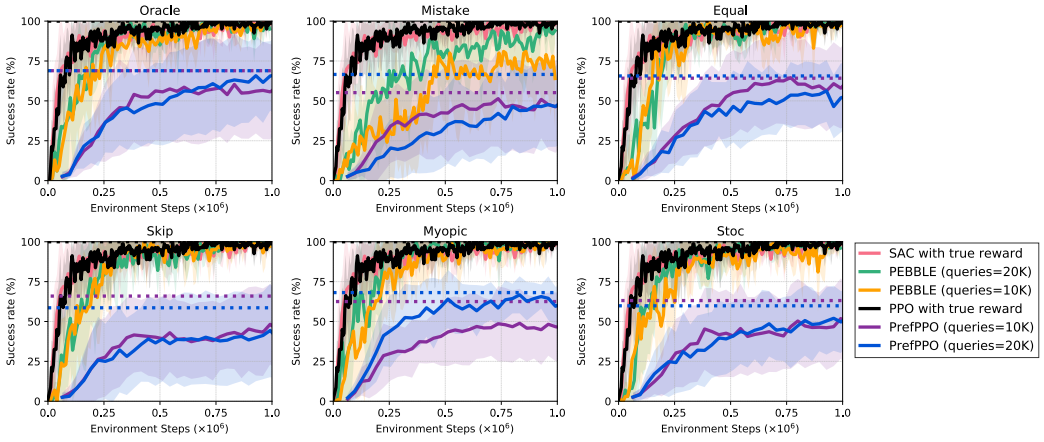


Figure 11: Learning curves of PEBBLE and PrefPPO on Button Press as measured on the success rate. The solid line and shaded regions represent the mean and standard deviation, respectively, across ten runs. Asymptotic performance of PPO and PrefPPO is indicated by dotted lines of the corresponding color.

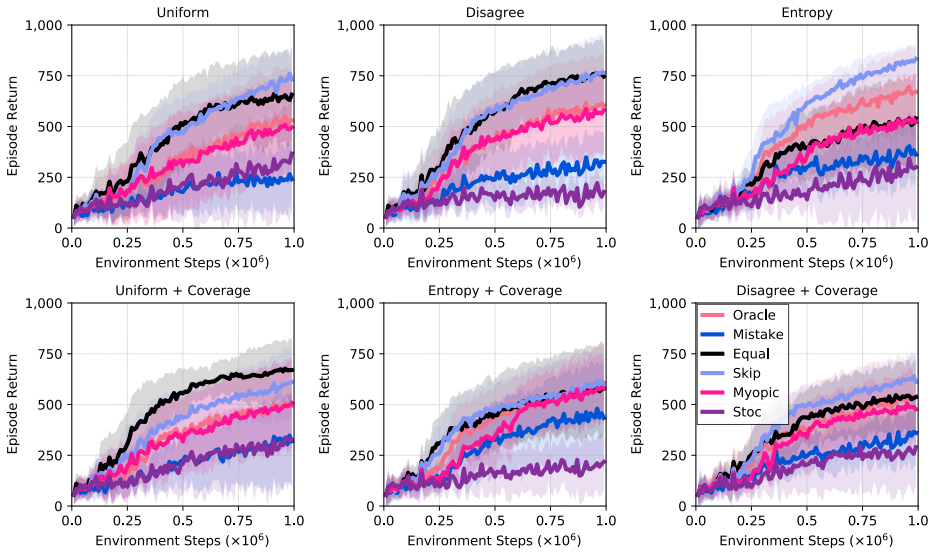


Figure 12: Learning curves of PEBBLE with 2000 queries on Quadruped-walk as measured on the ground truth reward. The solid line and shaded regions represent the mean and standard deviation, respectively, across ten runs.

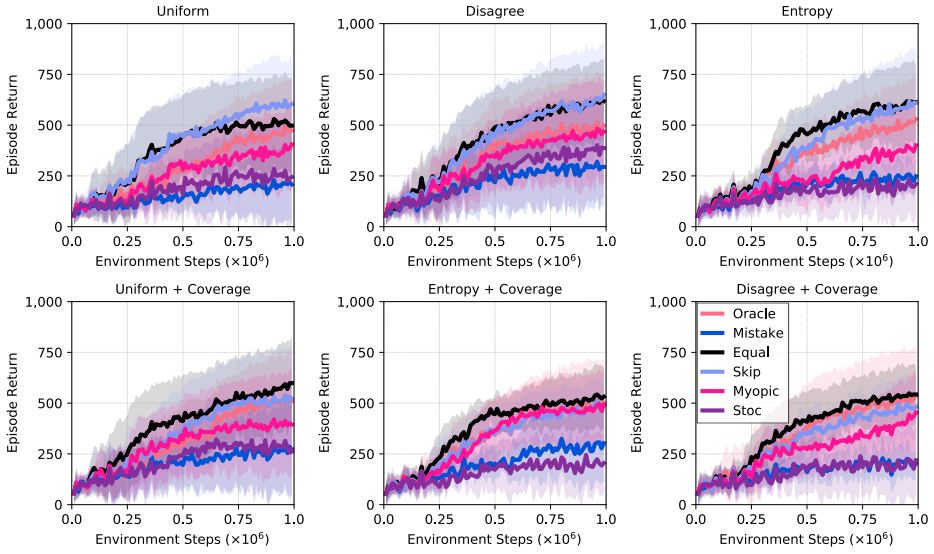


Figure 13: Learning curves of PEBBLE with 1000 queries on Quadruped-walk as measured on the ground truth reward. The solid line and shaded regions represent the mean and standard deviation, respectively, across ten runs.

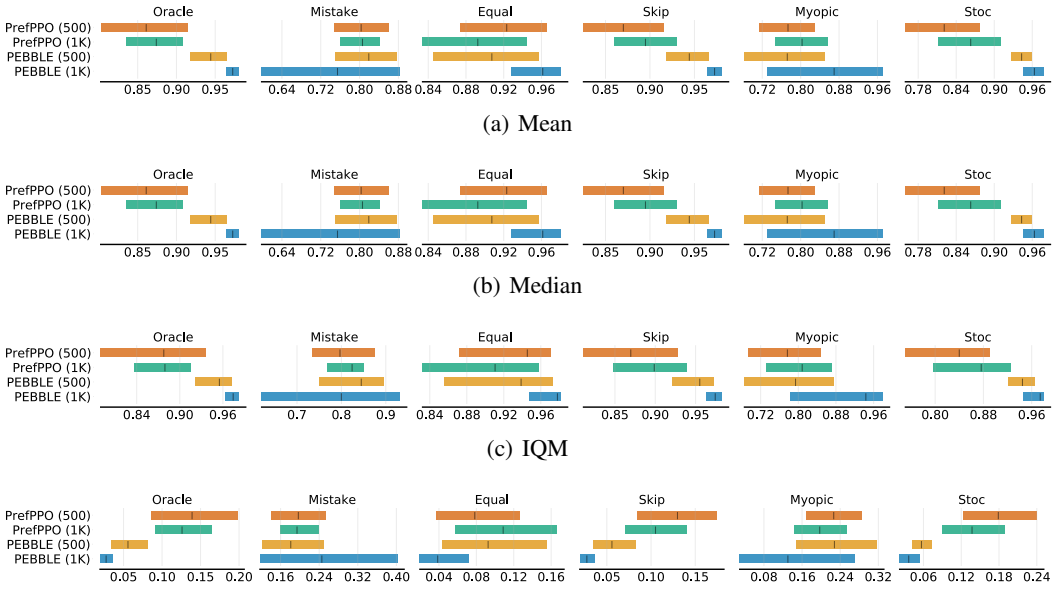


Figure 14: Aggregate metrics on Walker with 95% confidence intervals (CIs) across ten runs. Higher mean, median and IQM scores and lower optimality gap are better. The CIs are estimated using the percentile bootstrap with stratified sampling.

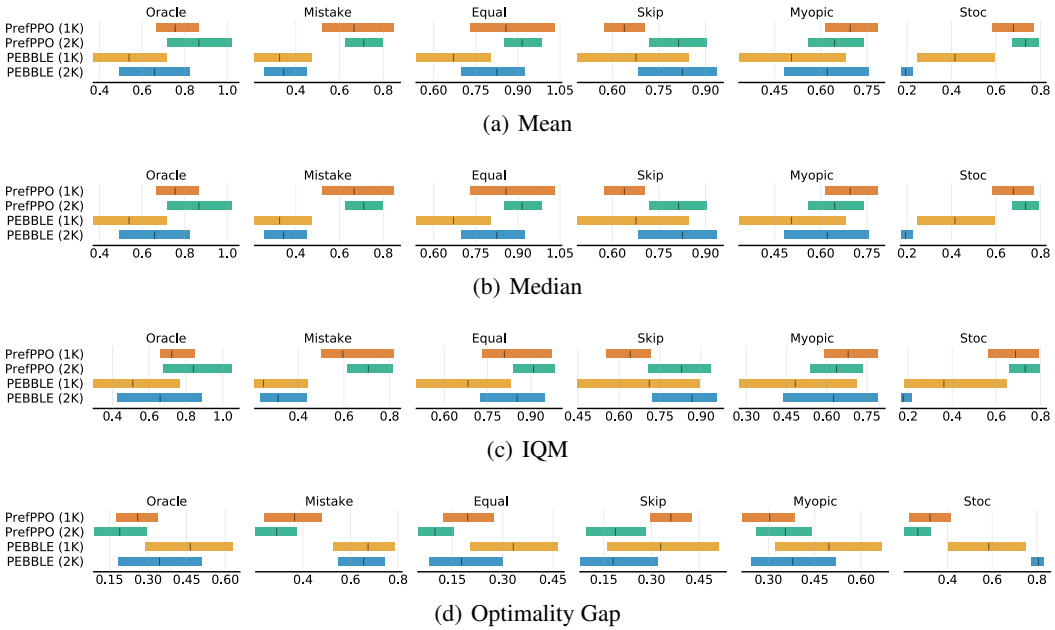
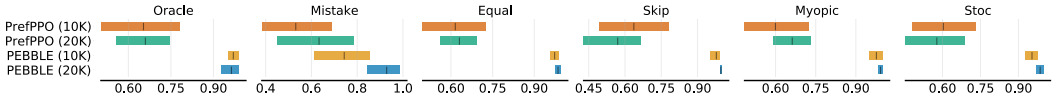
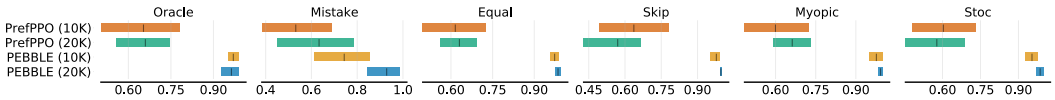


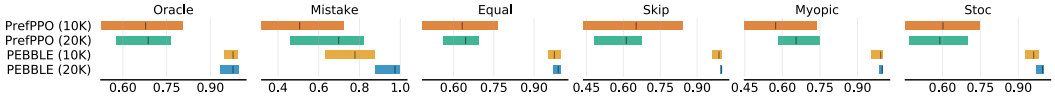
Figure 15: Aggregate metrics on Quadruped with 95% confidence intervals (CIs) across ten runs. Higher mean, median and IQM scores and lower optimality gap are better. The CIs are estimated using the percentile bootstrap with stratified sampling.



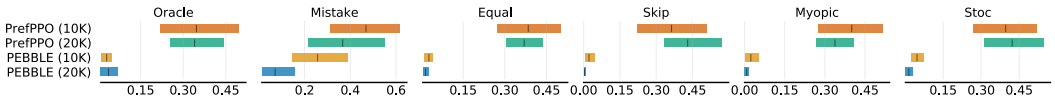
(a) Mean



(b) Median

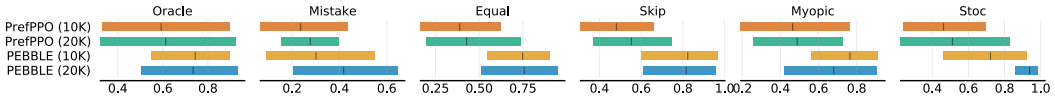


(c) IQM

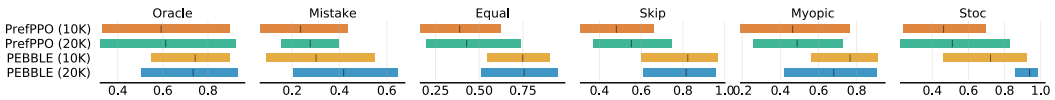


(d) Optimality Gap

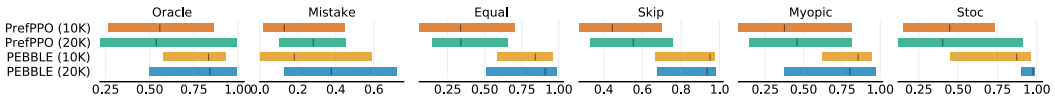
Figure 16: Aggregate metrics on Button Press with 95% confidence intervals (CIs) across ten runs. Higher mean, median and IQM scores and lower optimality gap are better. The CIs are estimated using the percentile bootstrap with stratified sampling.



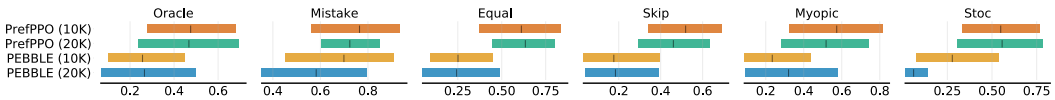
(a) Mean



(b) Median

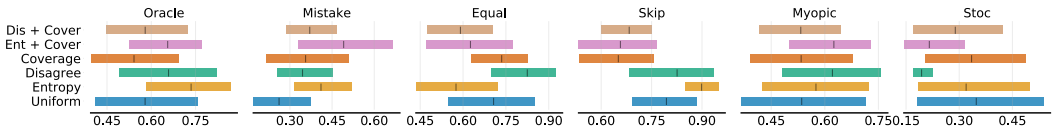


(c) IQM

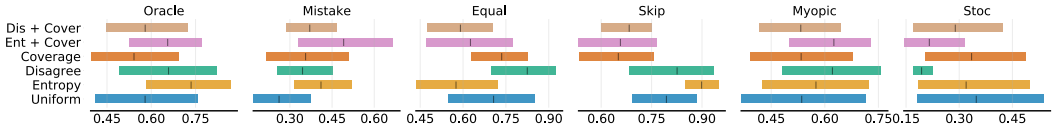


(d) Optimality Gap

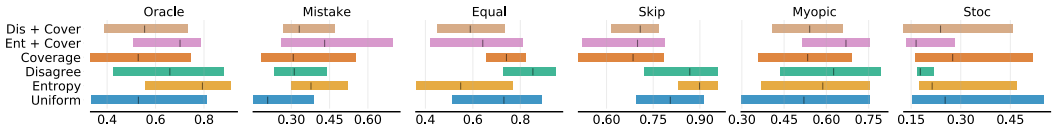
Figure 17: Aggregate metrics on Sweep Into with 95% confidence intervals (CIs) across ten runs. Higher mean, median and IQM scores and lower optimality gap are better. The CIs are estimated using the percentile bootstrap with stratified sampling.



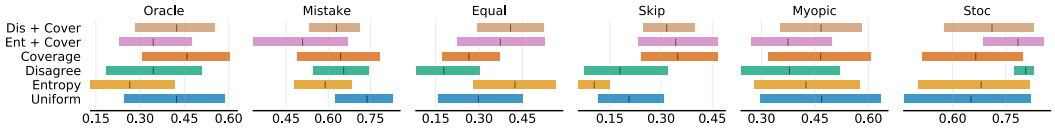
(a) Mean



(b) Median

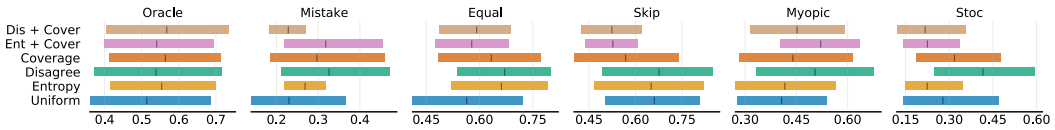


(c) IQM

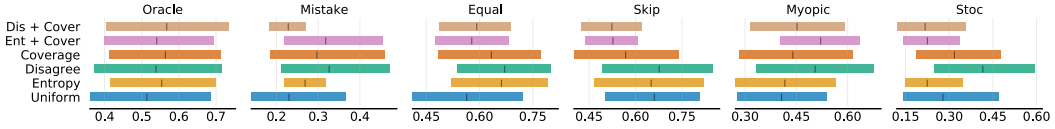


(d) Optimality Gap

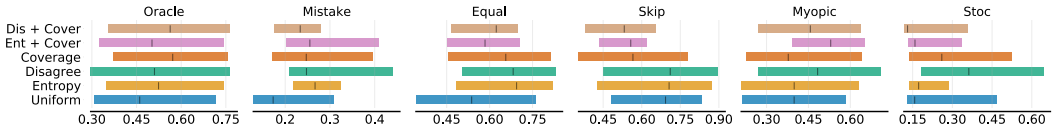
Figure 18: Aggregate metrics of PEBBLE on Quadruped with 2000 queries across ten runs. Higher mean, median and IQM scores and lower optimality gap are better. The CIs are estimated using the percentile bootstrap with stratified sampling.



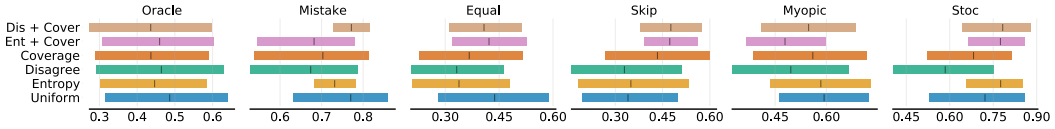
(a) Mean



(b) Median



(c) IQM



(d) Optimality Gap

Figure 19: Aggregate metrics of PEBBLE on Quadruped with 1000 queries across ten runs. Higher mean, median and IQM scores and lower optimality gap are better. The CIs are estimated using the percentile bootstrap with stratified sampling.