Behavior Alignment via Reward Function Optimization (Supplemental Material)

Table 1: Notations

Symbol	Description
θ	Parameters for policy π
ϕ	Parameters for reward function
φ	Parameters for learned γ
$\pi_\theta, r_\phi, \gamma_\varphi$	Functional form of policy, reward and γ with their respective parameters
$\alpha_\theta, \alpha_\phi, \alpha_\varphi$	Step sizes for the respective parameters
$\lambda_\theta, \lambda_\phi, \lambda_\varphi$	Regularization for policy, reward and γ function
δ	Number of on-policy samples collected between subsequent updates to ϕ, φ
η	Neumann Approximator Eigen value scaling factor
n	Number of loops used in Neumann Approximation
`optim`	Any standard optimizer like Adam, RMSprop, SGD, which takes input as gradients and outputs the appropriate update
E	Total Number of episodes to sample from the environment
N_i	Number of updates to be performed for updating the π by `Alg`
N_0	Number of initial updates to be peformed

A Proofs for Theoretical Results

In this section, we provide proofs for Property 1, Property 2, and Property 3. For the purpose of these proofs, we introduce some additional notation. To have a unified MDP notation for goal-based and time-based tasks, we first consider that in the the time-based task, time is a part of the state such that Markovian dynamics is ensured.

The (un-normalized) discounted and undiscounted visitation probability is denoted as,

$$d^\pi_\gamma(s,a) := \sum_{t=0}^{T} \gamma^t \Pr(S_t = s, A_t = a; \pi), \tag{9}$$

$$\bar{d}^\pi(s,a) := \sum_{t=0}^{T} \Pr(S_t = s, A_t = a; \pi). \tag{10}$$

We can normalize so that it is a distribution as follows :

$$d^\pi(s,a) := \frac{\bar{d}^\pi(s,a)}{\sum_{s' \in \mathcal{S}, a' \in \mathcal{A}} \bar{d}^\pi(s',a')}$$

Property 1. $\Delta(\theta, \tilde{r}) = \Delta(\theta, r_p)$.

Proof.

$$\begin{aligned}
\Delta(\theta, \tilde{r}) &= \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \psi_\theta(S_t, A_t) \sum_{j=t}^{T} \gamma^{j-t} \tilde{r}(S_j, A_j)\right] \\
&= \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \psi_\theta(S_t, A_t) \left(\sum_{j=t}^{T} \gamma^{j-t} \left(r_p(S_j, A_j) + \gamma\Phi(S_{j+1}) - \Phi(S_j)\right)\right)\right] \\
&= \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \psi_\theta(S_t, A_t) \sum_{j=t}^{T} \gamma^{j-t} r_p(S_j, A_j)\right] + \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \psi_\theta(S_t, A_t) \sum_{j=t}^{T} \gamma^{j-t}(\gamma\Phi(S_{j+1}) - \Phi(S_j))\right] \\
&= \Delta(\theta, r_p) + \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \psi_\theta(S_t, A_t) \sum_{j=t}^{T} \gamma^{j-t}(\gamma\Phi(S_{j+1}) - \Phi(S_j))\right] \\
&\overset{(a)}{=} \Delta(\theta, r_p) + \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \psi_\theta(S_t, A_t)(\gamma^{T-t+1}\Phi(S_{T+1}) - \Phi(S_t))\right] \\
&\overset{(b)}{=} \Delta(\theta, r_p) + \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \psi_\theta(S_t, A_t)(\gamma^{T-t+1}c - \Phi(S_t))\right] \\
&\overset{(c)}{=} \Delta(\theta, r_p) + \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} (\gamma^{T-t+1}c - \Phi(S_t))\mathbb{E}_{\pi_\theta}\left[\psi_\theta(S_t, A_t)|S_t\right]\right] \\
&\overset{(d)}{=} \Delta(\theta, r_p),
\end{aligned}$$

where (a) holds because on the expansion of future return, intermediate potential values cancel out, (b) holds because S_{T+1} is the terminal state and potential function is defined to be a fixed constant c for the terminal state [42], (c) holds from the law of total expectation, and (d) holds because,

$$\mathbb{E}_{\pi_\theta}\left[\psi_\theta(S_t, A_t)|S_t\right] = \sum_{a \in \mathcal{A}} \pi_\theta(S_t, a) \frac{\partial \ln \pi_\theta(S_t, a)}{\partial \theta} = \sum_{a \in \mathcal{A}} \frac{\partial \pi_\theta(S_t, a)}{\partial \theta} = \frac{\partial}{\partial \theta} \sum_{a \in \mathcal{A}} \pi_\theta(S_t, a) = 0.$$

In the stochastic setting, i.e., when using sample average estimates instead of the true expectation, $\gamma^{T-t+1}c - \phi(S_t)$ is analogous to a state dependent baseline for the sum of discounted future primary rewards. It may reduce or increase the variance of $\Delta(\theta, r_p)$, depending on this baseline's co-variance with $\sum_{j=t}^{T} \gamma^{j-t} r_p(S_j, A_j)$. □

Note: As we encountered the potential at the terminal state to be c, as it is a constant, we will use the value of $c = 0$ in accordance with [42].

$\mathbb{V}_{\pi_\theta}\left[\hat{\Delta}(\theta, \tilde{r})\right]$ can be more than $\mathbb{V}_{\pi_\theta}\left[\hat{\Delta}(\theta, r_p)\right]$.

Proof. We will first look at a different variant of the update, i.e., $\Delta(\theta, \tilde{r}) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \psi_\theta(S_t, A_t) \sum_{j=t}^{T} \gamma^{j-t} \tilde{r}(S_j, A_j)\right]$ can also be written as follows:

$$\Delta(\theta, \tilde{r}) = \mathbb{E}_{d^\pi, \pi}[\psi_\theta(S, A)(q^\pi(S, A) - \Phi(S))] \,.$$

This is proved as part of the proof of Property 2. Similarly the baseline update (using r_p can be written as,

$$\Delta(\theta, r_p) = \mathbb{E}_{d^\pi, \pi}[\psi_\theta(S, A)(q^\pi(S, A))] \,.$$

Let's put $Y = \psi_\theta(S, A)(q^\pi(S, A) - \Phi(S))$, $X = \psi_\theta(S, A)q^\pi(S, A)$ and $Z = \psi_\theta(S, A)q^\pi(S)$ as our random variables and try to understand when the variance of $\hat{\Delta}(\theta, \tilde{r})$ can be higher than $\hat{\Delta}(\theta, r_p)$. Also note that $\mathbb{E}[Y] = \mathbb{E}[X]$

$$
\begin{aligned}
\mathbb{V}_{d^\pi}[Y] =& \mathbb{E}_{d^\pi}\Big[Y^T Y\Big] - \mathbb{E}[Y]^T\,\mathbb{E}[Y] \\
=& \mathbb{E}_{d^\pi}\Big[(\psi_\theta(S,A)(q^\pi(S,A) - \Phi(S)))^T(\psi_\theta(S,A)(q^\pi(S,A) - \Phi(S)))\Big] - \mathbb{E}[X]^T\,\mathbb{E}[X] \\
=& \mathbb{E}_{d^\pi}\Big[\psi_\theta(S,A)^T\psi_\theta(S,A)((q^\pi(S,A) - \Phi(S)))^2\Big] - \mathbb{E}[X]^T\,\mathbb{E}[X] \\
=& \mathbb{E}_{d^\pi}\Big[\psi_\theta(S,A)^T\psi_\theta(S,A)\Phi(S)^2\Big] - 2\mathbb{E}_{d^\pi}\Big[\psi_\theta(S,A)^T\psi_\theta(S,A)(q^\pi(S,A)\Phi(S))\Big] + \\
& \mathbb{E}_{d^\pi}\Big[\psi_\theta(S,A)^T\psi_\theta(S,A)q^\pi(S,A)^2\Big] - \mathbb{E}[X]^T\,\mathbb{E}[X]
\end{aligned}
$$

The last term in above is the variance of the method with primary reward i.e. $\mathbb{E}_{d^\pi}\big[\psi_\theta(S,A)^T\psi_\theta(S,A)q^\pi(S,A)^2\big] - \mathbb{E}[X]^T\,\mathbb{E}[X] = \mathbb{V}_{d^\pi}[X]$

$$\mathbb{V}_{d^\pi}[Y] - \mathbb{V}_{d^\pi}[X] = \mathbb{E}_{d^\pi}\Big[\psi_\theta(S,A)^T\psi_\theta(S,A)\Phi(S)^2\Big] - 2\mathbb{E}_{d^\pi}\Big[\psi_\theta(S,A)^T\psi_\theta(S,A)(q^\pi(S,A)\Phi(S))\Big]$$

Hence variance of Y will be higher than X if $\mathbb{E}_{d^\pi}\big[\psi_\theta(S,A)^T\psi_\theta(S,A)\Phi(S)^2\big] - 2\mathbb{E}_{d^\pi}\big[\psi_\theta(S,A)^T\psi_\theta(S,A)(q^\pi(S,A)\Phi(S))\big] > 0$

Now lets look at variance of $\hat{\Delta}(\theta,\tilde{r})$ and $\hat{\Delta}(\theta,r_p)$, lets use $X_i = \psi_\theta(S_i,A_i)q^\pi(S_i,A_i)$, similarly for Y_i and Z_i. Hence we can write $\hat{\Delta}(\theta,\tilde{r}) = \sum_{i=0}^T Y_i$ and $\hat{\Delta}(\theta,r_p) = \sum_{i=0}^T X_i$. Hence, as we saw that it is possible that $\mathbb{V}[Y] > \mathbb{V}[X]$, let's look at the variance of $\mathbb{V}\Big[\sum_{i=0}^T Y_i\Big] > \mathbb{V}_{d^\pi}\Big[\sum_{i=0}^T X_i\Big]$, for simplicity lets assume only 2-time steps, or states at time i, j.

$$\mathbb{V}[Y_i + Y_j] = \mathbb{V}[Y_i] + \mathbb{V}[Y_j] + 2\texttt{Cov}(Y_i, Y_j)$$

and

$$\mathbb{V}[X_i + X_j] = \mathbb{V}[X_i] + \mathbb{V}[X_j] + 2\texttt{Cov}(X_i, X_j)$$

As we can see that $\mathbb{V}[Y] > \mathbb{V}[X]$, hence we need to see the relationship between $\texttt{Cov}(Y_i, Y_j)$ and $\texttt{Cov}(X_i, X_j)$ to understand if $\mathbb{V}[Y_i + Y_j] > \mathbb{V}[X_i + X_j]$.

We can write $Y_i = X_i - Z_i$, hence

$$
\begin{aligned}
\texttt{Cov}(Y_i, Y_j) &= \mathbb{E}[Y_iY_j] - \mathbb{E}[Y_i])\mathbb{E}[Y_j] \\
&= \mathbb{E}[(X_i - Z_i)(X_j - Z_j)] - \mathbb{E}[X_i]\,\mathbb{E}[X_j] \\
&= \mathbb{E}[X_iX_j - X_iZ_j - X_jZ_i + Z_iZ_j] - \mathbb{E}[X_i]\,\mathbb{E}[X_j] \\
&= \mathbb{E}[X_iX_j] - \mathbb{E}[X_iZ_j] - \mathbb{E}[X_jZ_i] + \mathbb{E}[Z_iZ_j] - \mathbb{E}[X_i]\,\mathbb{E}[X_j] \\
&= \texttt{Cov}(X_i, X_j) - \mathbb{E}[X_iZ_j] - \mathbb{E}[X_jZ_i] + \mathbb{E}[Z_iZ_j] \\
&= \texttt{Cov}(X_i, X_j) - \mathbb{E}[X_i\mathbb{E}[Z_j|(S_i,A_i)]] - \mathbb{E}[X_j\mathbb{E}[Z_i|(S_j,A_j)]] + \mathbb{E}[Z_i\mathbb{E}[Z_j|S_i,A_i]]
\end{aligned}
$$

And we have seen that $\mathbb{E}[Z_i] = 0\forall i$, hence

$$= \texttt{Cov}(X_i, X_j) + 0$$

Hence , $\texttt{Cov}(X_i, X_j) = \texttt{Cov}(Y_i, Y_j)$, hence if $\mathbb{V}[X] < \mathbb{V}[Y]$ then $\mathbb{V}[X_i + X_j] < \mathbb{V}[Y_i + Y_j]$, and hence $\mathbb{V}\Big[\hat{\Delta}(\theta, r_p)\Big] < \mathbb{V}\Big[\hat{\Delta}(\theta,\tilde{r})\Big]$.

□

Examples: Let's look at some example where the above condition can be true, first, lets consider that we have a parameterizing where we separately update the policies for each state, i.e. a tabular representation then, in that case, we can consider the variance of the update to the policy at each state separately i.e.

$$\mathbb{V}_\pi[Y|s] - \mathbb{V}_\pi[X|s] = \Phi(s)^2\mathbb{E}_\pi\Big[\psi_\theta(s,A)^T\psi_\theta(s,A)\Big] - 2\Phi(s)\mathbb{E}_\pi\Big[\psi_\theta(s,A)^T\psi_\theta(s,A)(q^\pi(s,A))\Big]$$

Hence let's see under what conditions we might get that the variance of the potential-based method might be more than the variance only on primary reward, i.e. for that to be true, the difference between the 2 terms should be positive, i.e.

$$\Phi(s)^2\mathbb{E}_\pi\Big[\psi_\theta(s,A)^T\psi_\theta(s,A)\Big] - 2\Phi(s)\mathbb{E}_\pi\Big[\psi_\theta(s,A)^T\psi_\theta(s,A)(q^\pi(s,A))\Big] > 0$$
$$\Phi(s)^2\mathbb{E}_\pi\Big[\psi_\theta(s,A)^T\psi_\theta(s,A)\Big] > 2\Phi(s)\mathbb{E}_\pi\Big[\psi_\theta(s,A)^T\psi_\theta(s,A)(q^\pi(s,A))\Big]$$

Further, let's consider the case where $\Phi(s) \neq 0$, cause otherwise for those states, the variance would be the same, and $\Phi(s) > 0$.

$$\cancel{\Phi(s)}^2\mathbb{E}_\pi\Big[\psi_\theta(s,A)^T\psi_\theta(s,A)\Big] > 2\cancel{\Phi(s)}\mathbb{E}_\pi\Big[\psi_\theta(s,A)^T\psi_\theta(s,A)(q^\pi(s,A))\Big]$$
$$\Phi(s)\mathbb{E}_\pi\Big[\psi_\theta(s,A)^T\psi_\theta(s,A)\Big] > 2\mathbb{E}_\pi\Big[\psi_\theta(s,A)^T\psi_\theta(s,A)(q^\pi(s,A))\Big]$$

We can see that the above condition can easily be satisfied by choosing a potential function that might be overly optimistic about the action-value of the state s, i.e. any $\Phi(s)$, s.t. $\Phi(s) > 2q^\pi(s,a)\forall a$ would lead to an increase in variance. A good example of this could be using an optimal value function (as hinted by [42]) as a baseline for a bad/mediocre policy initially.

Property 2. *There exists $r_\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and $\gamma_\varphi \in [0,1)$ such that $\Delta_{on}(\theta,\phi,\varphi) = \Delta_\gamma(\theta, r_p)$.*

Proof. Recall the definition of $\Delta_\gamma(\theta, r_p)$ from Section 3.1,

$$\Delta_\gamma(\theta, r_p) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T}\gamma^t\psi_\theta(S_t,A_t)\sum_{j=t}^{T}\gamma^{j-t}r_p(S_j,A_j)\right].$$

Using the law of total expectation,

$$\begin{aligned}
\Delta_\gamma(\theta, r_p) &= \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T}\gamma^t\psi_\theta(S_t,A_t)\mathbb{E}_{\pi_\theta}\left[\sum_{j=t}^{T}\gamma^{j-t}r_p(S_j,A_j)\middle| S_t,A_t\right]\right]\\
&= \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T}\gamma^t\psi_\theta(S_t,A_t)q^{\pi_\theta}(S_t,A_t)\right]\\
&= \sum_{s\in\mathcal{S},a\in\mathcal{A}}\sum_{t=0}^{T}\gamma^t\Pr(S_t=s,A_t=a;\pi_\theta)\psi_\theta(s,a)q^{\pi_\theta}(s,a)\\
&= \sum_{s\in\mathcal{S},a\in\mathcal{A}}\psi_\theta(s,a)q^{\pi_\theta}(s,a)\sum_{t=0}^{T}\gamma^t\Pr(S_t=s,A_t=a;\pi_\theta)\\
&= \sum_{s\in\mathcal{S},a\in\mathcal{A}}\psi_\theta(s,a)q^{\pi_\theta}(s,a)d_\gamma^{\pi_\theta}(s,a). \qquad (11)
\end{aligned}$$

Notice from (9) and (10) that for any (s,a) pair, if $d_\gamma^{\pi_\theta}(s,a) > 0$, then it has to be that $\bar{d}^{\pi_\theta}(s,a) > 0$ as well since $\gamma \geq 0$. Therefore, dividing and multiplying by $\bar{d}^{\pi_\theta}(s,a)$,

$$\begin{aligned}
\Delta_\gamma(\theta, r_p) &= \sum_{s\in\mathcal{S},a\in\mathcal{A}}\bar{d}^{\pi_\theta}(s,a)\psi_\theta(s,a)q^{\pi_\theta}(s,a)\frac{d_\gamma^{\pi_\theta}(s,a)}{\bar{d}^{\pi_\theta}(s,a)}\\
&= \sum_{s\in\mathcal{S},a\in\mathcal{A}}\sum_{t=0}^{T}\Pr(S_t=s,A_t=a;\pi_\theta)\psi_\theta(s,a)q^{\pi_\theta}(s,a)\frac{d_\gamma^{\pi_\theta}(s,a)}{\bar{d}^{\pi_\theta}(s,a)}\\
&= \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T}\psi_\theta(S_t,A_t)q^{\pi_\theta}(S_t,A_t)\frac{d_\gamma^{\pi_\theta}(S_t,A_t)}{\bar{d}^{\pi_\theta}(S_t,A_t)}\right].
\end{aligned}$$

Now, notice that if $\gamma_\varphi = 0$ and $r_\phi(s,a) = q^{\pi_\theta}(s,a)\frac{d_\gamma^{\pi_\theta}(s,a)}{\bar{d}^{\pi_\theta}(s,a)}$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$, then

$$\Delta_{\text{on}}(\theta,\phi,\varphi) = \Delta_\gamma(\theta, r_p).$$

$\square$

Property 3. *There exists $r_\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and $\gamma_\varphi \in [0,1)$ such that $\Delta_{off}(\theta,\phi,\varphi) = \Delta_{off}(\theta, r_p)$.*

Proof. This proof follows a similar technique as the proof for Property 2. Recall the definition of $\Delta_{\text{off}}(\theta, r_p)$,

$$\Delta_{\text{off}}(\theta, r_p) \coloneqq \mathbb{E}_\beta\left[\sum_{t=0}^{T} \gamma^t \psi_\theta(S_t, A_t) \sum_{j=t}^{T} \gamma^{j-t} \rho_j r_p(S_j, A_j)\right]$$
$$\coloneqq \mathbb{E}_\beta\left[\sum_{t=0}^{T} \gamma^t \rho_t \psi_\theta(S_t, A_t) \sum_{j=t}^{T} \gamma^{j-t} \rho_{j-t} r_p(S_j, A_j)\right]$$

Now using the law of total expectations,

$$\begin{aligned}
\Delta_{\text{off}}(\theta, r_p) &= \mathbb{E}_\beta\left[\sum_{t=0}^{T} \gamma^t \rho_t \psi_\theta(S_t, A_t) \mathbb{E}_\beta\left[\sum_{j=t}^{T} \gamma^{j-t} \rho_{j-t} r_p(S_j, A_j) \middle| S_t, A_t\right]\right] \\
&= \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t \psi_\theta(S_t, A_t) \mathbb{E}_{\pi_\theta}\left[\sum_{j=t}^{T} \gamma^{j-t} r_p(S_j, A_j) \middle| S_t, A_t\right]\right] \\
&= \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t \psi_\theta(S_t, A_t) q^{\pi_\theta}(S_j, A_j)\right] \\
&= \sum_{s \in \mathcal{S}, a \in \mathcal{A}} \psi_\theta(s, a) q^{\pi_\theta}(s, a) d_\gamma^{\pi_\theta}(s, a),
\end{aligned}$$

where the last line follows similar to (11). Now, notice that for any (s, a) pair, the assumption that $\pi_\theta(s, a)/\beta(s, a) < \infty$ for all $s \in \mathcal{S}, a \in \mathcal{A}$, implies $d_\gamma^{\pi_\theta}(s, a)/d_\gamma^{\beta}(s, a) < \infty$. Further, if $d_\gamma^{\beta}(s, a) > 0$ it has to be that $d^\beta(s, a) > 0$ as well. Therefore, $d_\gamma^{\pi_\theta}(s, a)/d^\beta(s, a) < \infty$ as well. Multiplying and dividing by $d^\beta(s, a)$,

$$\begin{aligned}
\Delta_{\text{off}}(\theta, r_p) &= \sum_{s \in \mathcal{S}, a \in \mathcal{A}} d^\beta(s, a) \psi_\theta(s, a) q^{\pi_\theta}(s, a) \frac{d_\gamma^{\pi_\theta}(s, a)}{d^\beta(s, a)} \\
&= \sum_{s \in \mathcal{S}, a \in \mathcal{A}} \sum_{t=0}^{T} \Pr(S_t = s, A_t = a; \beta) \psi_\theta(s, a) q^{\pi_\theta}(s, a) \frac{d_\gamma^{\pi_\theta}(s, a)}{d^\beta(s, a)} \\
&= \mathbb{E}_\beta\left[\sum_{t=0}^{T} \psi_\theta(S_t, A_t) q^{\pi_\theta}(S_t, A_t) \frac{d_\gamma^{\pi_\theta}(S_t, A_t)}{d^\beta(S_t, A_t)}\right].
\end{aligned}$$

Now, notice that if $\gamma_\varphi = 0$ and $r_\phi(s, a) = q^{\pi_\theta}(s, a) \frac{d_\gamma^{\pi_\theta}(s,a)}{d^\beta(s,a)}$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$,

$$\Delta_{\text{off}}(\theta, \phi, \varphi) = \Delta_{\text{off}}(\theta, r_p).$$

□

B Algorithm

In this section we discuss the algorithm for the proposed method. As the proposed method does behavior alignment reward function's implicit optimization, we name our method `Barfi`. Pseudo-code for `Barfi` is presented in Algorithm 5. We will first build on some preliminaries to understand the concepts

B.1 Vector Jacobian Product

Lets assume the following, $x \in \mathbb{R}^d, y \in \mathbb{R}^m, f(x, y) \in \mathbb{R}$, then we know that $\partial f(x, y)/\partial x \in \mathbb{R}^d, \partial f(x, y)/\partial x \in \mathbb{R}^m, \partial^2 f(x, y)/\partial y \partial x \in \mathbb{R}^{d \times m}$. Let us also assume that we have a vector $v \in \mathbb{R}^d$, and when we need to calculate the following

$$\underbrace{\underbrace{v}_{\mathbb{R}^d} \underbrace{\frac{\partial^2 f(x, y)}{\partial y \partial x}}_{\mathbb{R}^{d \times m}}}_{\mathbb{R}^m} = \underbrace{\frac{\partial}{\partial y} \underbrace{\left\langle \underbrace{v}_{\mathbb{R}^d}, \underbrace{\frac{\partial f(x,y)}{\partial x}}_{\mathbb{R}^d} \right\rangle}_{\mathbb{R}^1}}_{\mathbb{R}^m}$$

As we can see the vector jacobian product can actually be broken down into differentiating a vector product but shifting the place of multiplication, in which case we assume that the gradient simply passes through v w.r.t. y and hence we don't ever have to deal with large multiplications. Also note, that the outer partial w.r.t can easily be handled by `autodiff` packages. A simple pseudo-code is show in Algorithm 1.

Algorithm 1: Jacobian Vector Product

Input: $f(x, y) \in \mathbb{R}^1, x \in \mathbb{R}^d, y \in \mathbb{R}^m, v \in \mathbb{R}^d$
$f' \leftarrow \texttt{grad}(f(x, y), x)$
$jvp \leftarrow \texttt{grad}(f', y, \texttt{grad\_outputs} = v)$
Return: jvp

B.2 Neumann Series Approximation for Hessian Inverse

Recall, that for a given real number $\beta \in \mathbb{R}$, such that $0 \leq \beta < 1$, we know that the geomertric series of this has a closed form solution, i.e.

$$\begin{aligned} s &= 1 + \beta^1 + \beta^2 + \beta^3 + \cdots + \\ &= \frac{1}{1-\beta} \end{aligned}$$

Similarly given we have a value α, such that $\beta = 1 - \alpha$, we can write α^{-1} as follows

$$\begin{aligned} \frac{1}{1-\beta} &= 1 + \beta + \beta^2 + \beta^3 + \cdots + \\ \frac{1}{1-(1-\alpha)} &= 1 + (1-\alpha) + (1-\alpha)^2 + (1-\alpha)^3 + \cdots + \\ \alpha^{-1} &= 1 + (1-\alpha) + (1-\alpha)^2 + (1-\alpha)^3 + \cdots + \\ \alpha^{-1} &= \sum_{i=0}^{\infty} (1-\alpha)^i \end{aligned}$$

The same can be generalized for a matrix, i.e. given a matrix $\mathbf{A} \in \mathbb{R}^{d \times d}$, we can write $\mathbf{A}^{-1}$ as follows

$$\mathbf{A}^{-1} = \sum_{i=0}^{\infty} (\mathbf{I} - \mathbf{A})^i$$

Note for the above to hold, matrix a, where we represent $\texttt{eig}(\mathbf{A})$ as the eigen values of matrix $\mathbf{A}$, we should have the following condition to hold, $0 < \texttt{eig}(\mathbf{A}) < 1$. Note here we would regularize $\mathbf{A}$ to ensure that all eigen values are positive, and then we can always scale the matrix $\mathbf{A}$, by its biggest eigen value to ensure that the above condition holds. Lets say $\eta = 1/\max \texttt{eig}(\mathbf{A})$, then we can write the following

$$\begin{aligned} \mathbf{A}^{-1} &= \frac{\eta}{\eta} \mathbf{A}^{-1} \\ &= \eta (\eta \mathbf{A})^{-1} \\ &= \eta \sum_{i=0}^{\infty} (\mathbf{I} - \eta \mathbf{A})^i \end{aligned}$$

As $\eta \mathbf{A}$ would always satisfy the above condition.

B.3 Neumann Approximation for Hessian Vector Product

Given we have seen how we can approximate the Inverse of a matrix without relying $O(d^3)$ operations, through Neumann approximation, lets look what needs to be done for our updates. Recall that the update ϕ, φ (7) and (8) were,

$$\frac{\partial J(\theta(\phi,\varphi))}{\partial \phi} = -\underbrace{\frac{\partial J(\theta(\phi,\varphi))}{\partial \theta(\phi,\varphi)}}_{v} \Bigg(\underbrace{\frac{\partial \Delta(\theta(\phi,\varphi),\phi,\varphi)}{\partial \theta(\phi,\varphi)}}_{\mathbf{H}} \Bigg)^{-1} \underbrace{\frac{\partial \Delta(\theta(\phi,\varphi),\phi,\varphi)}{\partial \phi}}_{\mathbf{A}}$$

and

$$\frac{\partial \big(J(\theta(\phi,\varphi)) - \frac{1}{2}\|\gamma_\varphi\|^2 \big)}{\partial \varphi} = -\underbrace{\frac{\partial J(\theta(\phi,\varphi))}{\partial \theta(\phi,\varphi)}}_{v} \Bigg(\underbrace{\frac{\partial \Delta(\theta(\phi,\varphi),\phi,\varphi)}{\partial \theta(\phi,\varphi)}}_{\mathbf{H}} \Bigg)^{-1} \underbrace{\frac{\partial \Delta(\theta(\phi,\varphi),\phi,\varphi)}{\partial \varphi}}_{\mathbf{B}} - \frac{\partial \gamma_\varphi}{\partial \varphi}.$$

Lets look closely at the update for ϕ and we can generalize things easily for the case of φ.

$$\frac{\partial J(\theta(\phi,\varphi))}{\partial \phi} = -v\mathbf{H}^{-1}\mathbf{A}$$

We first look at how we can approximate the value of $v\mathbf{H}^{-1}$ efficiently as we can always make use of the Jacobian Vector product later to get $(v\mathbf{H}^{-1})\mathbf{A}$, as $v\mathbf{H}^{-1}$ becomes a vector. Lets assume we wish to run the Neumann approximation upto n steps, i.e. we want to approximate $\mathbf{H}^{-1}$ upto n order Neumann expansion,

$$\eta(\eta\mathbf{H}^{-1}) \approx \eta\sum_{i=0}^{n}(I-\eta\mathbf{H})^i \tag{12}$$

Over here we are assuming that the outer optimization for update (1) is for the function $J(\theta(\phi,\varphi))$ and the inner optimization which is represented by the update (4) is $f(\theta(\phi,\varphi),\phi,\varphi)$, i.e.

$$\Delta(\theta, r_p) = \frac{\partial J(\theta(\phi,\varphi))}{\partial\theta}$$
$$\Delta(\theta,\phi,\varphi) = \frac{\partial f(\theta(\phi,\varphi),\phi,\varphi)}{\partial\theta}$$

The most common form in which $f(;B)$ is usually defined is the following:

$$f(\theta,\phi,\varphi;B) \coloneqq \frac{1}{|B|}\sum_{\tau\in B}\left[\sum_{t=0}^{T}\log(\pi_\theta(S_t^\tau,A_t^\tau))\sum_{j=t}^{T}\gamma_\varphi^{j-t}r_\phi(S_j^\tau,A_j^\tau)\right].$$

Similalry this can be defined for J, except making use of r_p and the actual γ,

$$J(\theta;B) \coloneqq \frac{1}{|B|}\sum_{\tau\in B}\left[\sum_{t=0}^{T}\log(\pi_\theta(S_t^\tau,A_t^\tau))\sum_{j=t}^{T}\gamma^{j-t}r_p(S_j^\tau,A_j^\tau)\right].$$

Algorithm 2: Vector Hessian Inverse Product for (7) i.e. $v\mathbf{H}^{-1}$

```
Input: θ(φ,ϕ), φ, ϕ, J, f, n, η, D_off, D_on
v ← grad(J(θ(φ,ϕ); D_on), θ)
v' ← η × grad(f(θ(φ,ϕ), φ, ϕ; D_off), θ)
Let: v_0 ← v, p_0 ← v
for i ∈ [0, n) do
    v_{i+1} ← v_i − grad(v', θ, grad_outputs = v_i)
    p_{i+1} ← p_i + v_{i+1}
Return: η p_n ;                 // Approximation of vH^{-1} as in (12)
```

Finally once we have $v\mathbf{H}^{-1}$ we can use the Vector Jacobian Product to calculate $(v\mathbf{H}^{-1})\mathbf{A}$ as follows:

Algorithm 3: Update for ϕ, i.e. (7) i.e. $v\mathbf{H}^{-1}\mathbf{A}$

```
Input: θ(φ,ϕ), φ, ϕ, J, f, n, η, D_off, D_on
v ← Algorithm 2 (θ(φ,ϕ), φ, ϕ, J, f, n, η, D_off, D_on)
v' ← grad(f(θ(φ,ϕ), φ, ϕ; D_off), θ)
Δ_φ ← grad(v', φ, grad_outputs = v)
Return Δ_φ
```

We can similarly derive updates for φ. Note we are not including the different forms of regularizers over here to reduce clutter, but adding them is simple.

B.4 Pseduo Code

Line 8–10 and 21–23 represent the inner optimization process, and the outer optimization process if from lines 16-17. Lines 8–10 is the initial step of updates to converge to the current values of ϕ,φ, and from there onwards after each update of outer optimization, we consequently update the policy in (21–23). The flow of the algorithm is show in Figure 6.

Algorithm 4: Update for ϕ, i.e. (8) i.e. $v\mathbf{H}^{-1}\mathbf{B}$

1 **Input:** $\theta(\phi,\varphi), \phi, \varphi, J, f, n, \eta, \mathcal{D}_{\text{off}}, \mathcal{D}_{\text{on}}$
2 $v \leftarrow$ Algorithm 2 $(\theta(\phi,\varphi), \phi, \varphi, J, f, n, \eta, \mathcal{D}_{\text{off}}, \mathcal{D}_{\text{on}})$
3 $v' \leftarrow \texttt{grad}(f(\theta(\phi,\varphi),\phi,\varphi), \mathcal{D}_{\text{off}}), \theta)$
4 $\Delta_\varphi \leftarrow \texttt{grad}(v', \varphi, \texttt{grad\_outputs} = v)$
5 **Return** Δ_φ

Algorithm 5: `Barfi`: Behavior Alignment Reward Function's Implicit optimization

1 **Input:** $J, f, \alpha_\theta, \alpha_\phi, \alpha_\varphi, \eta, n, \delta, \texttt{optim}, E, N_i, N_0,$
2 **Initialize:** $\pi_\theta, r_\phi, \gamma_\varphi$
3 **Initialize:** $\texttt{optim}_\theta \leftarrow \texttt{optim}(\alpha_\theta), \texttt{optim}_\phi \leftarrow \texttt{optim}(\alpha_\phi), \texttt{optim}_\varphi \leftarrow \texttt{optim}(\alpha_\varphi)$
4 $\mathcal{D}_{\text{off}} \leftarrow \{\}$
5 **for** $e \in [1, N_0]$ **do**
6   Generate τ_e using π_θ
7   Save τ_e in $\mathcal{D}_{\text{off}}$
8 **for** $i \in [0, N_i + N_0]$ **do**
9   Sample a batch of trajectories B from $\mathcal{D}_{\text{off}}$
10     $\theta \leftarrow \theta + \texttt{optim}_\theta(\texttt{grad}(f(\theta,\phi,\varphi;B),\theta))$;   `// Make use of` B `to perform update rule` (4)
11 **for** $e \in [N_0, E]$ **do**
    # Collect a batch of on-policy data
12     $\mathcal{D}_{\text{on}} \leftarrow \{\}$
13   **for** $j \in [0, \delta)$ **do**
14   Generate trajectory τ_{e+j} using π_θ and save in $\mathcal{D}_{\text{on}}$
15     $e \leftarrow e + \delta$
    # Update r_ϕ and γ_φ
16     $\Delta_\phi \leftarrow$ Algorithm 3$(\theta(\phi,\varphi), \phi, \varphi, J, f, n, \eta, \mathcal{D}_{\text{off}}, \mathcal{D}_{\text{on}})$
17     $\Delta_\varphi \leftarrow$ Algorithm 4$(\theta(\phi,\varphi), \phi, \varphi, J, f, n, \eta, \mathcal{D}_{\text{off}}, \mathcal{D}_{\text{on}})$
18     $\phi \leftarrow \phi + \texttt{optim}_\phi(\Delta_\phi)$
19     $\varphi \leftarrow \varphi + \texttt{optim}_\varphi(\Delta_\varphi)$
20     $\mathcal{D}_{\text{off}} \leftarrow \mathcal{D}_{\text{off}} \cup \mathcal{D}_{\text{on}}$
21   **for** $i \in [0, N_i]$ **do**
22   Sample a batch of trajectories B from $\mathcal{D}_{\text{off}}$
23         $\theta \leftarrow \theta + \texttt{optim}_\theta(\texttt{grad}(f(\theta,\phi,\varphi;B),\theta))$;   `// Make use of` B `to perform update rule` (4)

As discussed in Section C, using regularizers in $\Delta(\theta,\phi,\varphi)$ smoothens the objective $J(\theta(\phi,\varphi))$ with respect to ϕ and φ. This is helpful as gradual changes in r_ϕ an γ_φ can result in gradually changes in the fixed point for the inner optimization. Therefore, for computational efficiency, we initialize the policy parameters from the fixed-point of the previous inner-optimization procedure such that the inner-optimization process may start close to the new fixed-point.

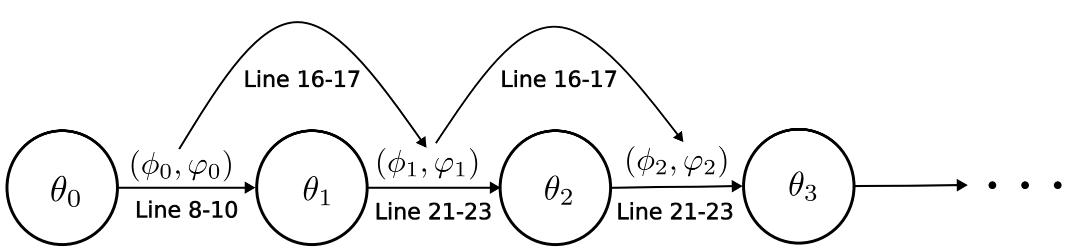

Figure 6: **Algorithm Flow:** The change in different parameters

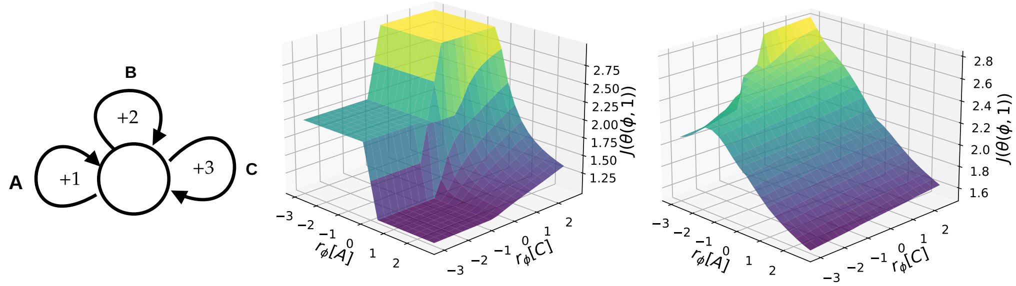

Figure 7: **(Left)** A bandit problem, where the data is collected from a policy β that samples action A mostly. **(Middle)** Each point on the 3D surface corresponds to the performance of $\theta(\phi, 1)$ returned by an `Alg` that uses the update rule $\Delta_{\text{off}}(\theta, \phi, 1)$ corresponding to the value of r_ϕ for actions A and C in the bottom axes; r_θ for action B is set to 0 to avoid another variable in a 3D plot. Notice that small perturbation in r_ϕ may lead to no or sudden changes in $J(\theta(\phi, 1))$. **(Right)** Performance of $\theta(\phi, 1)$ returned by an `Alg` that uses the update rule $\Delta_{\text{off}}(\theta, \phi, 1) - \theta$ that incorporates gradient of the L2 regularizer. Vector fields in Figure 1 were also obtained from this setup.

In lines 8–10, the inner optimization for the policy parameters θ are performed till (approximate) convergence. Note that only trajectories from past interactions are used and no new-trajectories are sampled for the inner optimization.

In Lines 13–14, a new batch $\mathcal{D}_{\text{on}}$ of data is sampled using the policy returned by the inner-optimization process. This data is used to compute $\partial J(\theta(\phi, \varphi))/\partial\theta(\phi, \varphi)$. Existing data $\mathcal{D}_{\text{off}}$ that was used in the inner-optimization process is then used to compute $\partial\theta(\phi, \varphi)/\partial\phi$ and $\partial\theta(\phi, \varphi)/\partial\varphi$. Using these in (7) and (8), the parameters for r_ϕ and γ_φ are updated in Lines 16 and 17, respectively.

Finally, the new data $\mathcal{D}_{\text{on}}$ is merged into the existing data $\mathcal{D}_{\text{off}}$ and the entire process continues.

C Smoothing the objective

To understand why $J(\theta(\phi, \varphi))$ might be ill-conditioned is to note that, often a small perturbation in the reward function doesn't necessarily lead to a change in the corresponding optimal policy. This can lead lack of gradient directions in the neighborhood of ϕ, φ for gradient methods to be effective. This issue can be addressed by employing common regularization techniques like L2 regularization of the policy parameters or entropy regularization for the policy. We discuss two ways to regularize the objective in the upcoming sections.

C.1 L2 Regularization

To understand how severely ill-conditioned $J(\theta(\phi, \varphi))$ can be, notice that a small perturbation in the reward function often does not change the corresponding optimal policies or the outcome of a policy optimization algorithm `Alg`. Therefore, if the parameters of the behavior alignment reward are perturbed from ϕ to ϕ', it may often be that $J(\theta(\phi, \varphi)) = J(\theta(\phi', \varphi))$ and this limits any gradient based optimization for ϕ as $\partial J(\theta(\phi, \varphi))/\partial\phi$ is 0. Similarly, minor perturbations in φ may result in no change in $J(\theta(\phi, \varphi))$ either.

Fortunately, there exists a remarkably simple solution: incorporate regularization for the *policy parameters* θ in objective for `Alg` in the inner-level optimization. For example, the optimal policy for the following regularized objective $\mathbb{E}_{\pi_\theta}[\sum_{t=0}^{T} \gamma_\varphi^t r_\phi(S_t, A_t)] - \frac{\lambda}{2}\|\theta\|^2$ varies smoothly to trade-off between the regularization value of θ and the magnitude of the performance characterized by (r_ϕ, γ_φ), which changes with the values of r_ϕ and γ_φ. See Figure 7 for an example with L2 regularization.

C.2 Entropy Regularized

In Section C.1, smoothing of $J(\theta(\phi, \varphi))$ was done by using L2 regularization on the policy parameters θ in the inner-optimization process. However, alternate regularization methods can also be used. For example, in the following we present an alternate update rule for θ based on entropy regularization,

$$\Delta(\theta, \phi, \varphi) \coloneqq \mathbb{E}_{\mathcal{D}}\left[\sum_{t=0}^{T} \psi_\theta(S_t, A_t) \sum_{j=t}^{T} \gamma_\varphi^{j-t} \left(r_\phi(S_j, A_j) - \lambda \ln \pi_\theta(S_j, A_j)\right)\right].$$

Notice that new update rule for ϕ and φ can be obtained from steps (5) to (8) with the following $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{H}$ instead, where for shorthand $\theta^* = \theta(\phi, \varphi)$,

$$\mathbf{A} = \mathbb{E}_{\mathcal{D}}\left[\sum_{t=0}^{T} \psi_{\theta^*}(S_t, A_t)\left(\sum_{j=t}^{T} \gamma_\varphi^{j-t}\frac{r_\phi(S_j, A_j)}{\partial \phi}\right)^\top\right],$$

$$\mathbf{B} = \mathbb{E}_{\mathcal{D}}\left[\sum_{t=0}^{T} \psi_{\theta^*}(S_t, A_t)\left(\sum_{j=t}^{T} \frac{\partial \gamma_\varphi^{j-t}}{\partial \varphi}\left(r_\phi(S_j, A_j) - \lambda \ln \pi_{\theta^*}(S_j, A_j)\right)\right)\right],$$

$$\mathbf{H} = \mathbb{E}_{\mathcal{D}}\left[\sum_{t=0}^{T} \frac{\partial \psi_{\theta^*}(S_t, A_t)}{\partial \theta^*}\left(\sum_{j=t}^{T} \gamma_\varphi^{j-t}\left(r_\phi(S_j, A_j) - \lambda \ln \pi_{\theta^*}(S_j, A_j)\right)\right) - \lambda \psi_{\theta^*}(S_t, A_t)\left(\sum_{j=t}^{T} \gamma_\varphi^{j-t}\psi_{\theta^*}(S_j, A_j)^\top\right)\right].$$

D Meta Learning via Implicit Gradient: Derivation

The general technique of implicit gradients [14, 34, 19] has been used in a vast range of applications, ranging from energy models [13, 36], differentiating through black-box solvers [61], few-shot learning [38, 49], model-based RL [50], differentiable convex optimization neural-networks layers [4, 2], to hyper-parameter optimization [37, 8, 12, 40]. In this work, we show how implicit gradients can also be useful to efficiently leverage auxiliary rewards r_a and overcome various sub-optimalities.

Taking the derivative (6) $\Delta(\theta(\phi, \varphi), \phi, \varphi) = 0$ of the above point of convergence w.r.t to ϕ and φ we get the following updates.

Therefore, taking total derivative in (6) with respect to ϕ,

$$\frac{d\Delta(\theta(\phi, \varphi), \phi, \varphi)}{d\phi} = \frac{\partial \Delta(\theta(\phi, \varphi), \phi, \varphi)}{\partial \phi} + \frac{\partial \Delta(\theta(\phi, \varphi), \phi, \varphi)}{\partial \theta(\phi, \varphi)}\frac{\partial \theta(\phi, \varphi)}{\partial \phi} = 0. \tag{13}$$

Lets try to understand why the above is true, considering the finite difference approach for this derivative,

$$\frac{d\Delta(\theta(\phi, \varphi), \phi, \varphi)}{d\phi} = \lim_{\|d\phi\| \to 0} \frac{\Delta(\theta(\phi + d\phi, \varphi), \phi + d\phi, \varphi) - \Delta(\theta(\phi, \varphi), \phi, \varphi)}{d\phi}$$

$\Delta(\theta(\phi + d\phi, \varphi), \phi + d\phi, \varphi) = \Delta(\theta(\phi, \varphi), \phi, \varphi) = 0$, as $\theta(\cdot, \cdot)$ defines convergence to fixed point

$$= \frac{0 - 0}{d\phi} = 0$$

By re-arranging terms in (13) we obtain the term (b) in (5),

$$\frac{\partial \theta(\phi, \varphi)}{\partial \phi} = -\left(\frac{\partial \Delta(\theta(\phi, \varphi), \phi, \varphi)}{\partial \theta(\phi, \varphi)}\right)^{-1}\frac{\partial \Delta(\theta(\phi, \varphi), \phi, \varphi)}{\partial \phi}. \tag{14}$$

On combining (14) with (5) we obtain the desired gradient expression for ϕ,

$$\frac{\partial J(\theta(\phi, \varphi))}{\partial \phi} = -\frac{\partial J(\theta(\phi, \varphi))}{\partial \theta(\phi, \varphi)}\Bigg(\underbrace{\frac{\partial \Delta(\theta(\phi, \varphi), \phi, \varphi)}{\partial \theta(\phi, \varphi)}}_{\mathbf{H}}\Bigg)^{-1}\underbrace{\frac{\partial \Delta(\theta(\phi, \varphi), \phi, \varphi)}{\partial \phi}}_{\mathbf{A}},$$

and following similar steps, it can be observed that the gradient expression for φ,

$$\frac{\partial\big(J(\theta(\phi, \varphi)) - \frac{1}{2}\|\gamma_\varphi\|^2\big)}{\partial \varphi} = -\frac{\partial J(\theta(\phi, \varphi))}{\partial \theta(\phi, \varphi)}\Bigg(\underbrace{\frac{\partial \Delta(\theta(\phi, \varphi), \phi, \varphi)}{\partial \theta(\phi, \varphi)}}_{\mathbf{H}}\Bigg)^{-1}\underbrace{\frac{\partial \Delta(\theta(\phi, \varphi), \phi, \varphi)}{\partial \varphi}}_{\mathbf{B}} - \frac{\partial \gamma_\varphi}{\partial \varphi},$$

where using θ^* as a shorthand for $\theta(\phi, \varphi)$ the terms $\mathbf{A}, \mathbf{B}$ and $\mathbf{H}$ can be expressed as,

$$\mathbf{A} = \mathbb{E}_{\mathcal{D}}\left[\sum_{t=0}^{T} \psi_{\theta^*}(S_t, A_t)\left(\sum_{j=t}^{T} \gamma_\varphi^{j-t}\frac{\partial r_\phi(S_j, A_j)}{\partial \phi}\right)^\top\right], \quad \mathbf{B} = \mathbb{E}_{\mathcal{D}}\left[\sum_{t=0}^{T} \psi_{\theta^*}(S_t, A_t)\left(\sum_{j=t}^{T} \frac{\partial \gamma_\varphi^{j-t}}{\partial \varphi} r_\phi(S_j, A_j)\right)\right],$$

$$\mathbf{H} = \mathbb{E}_{\mathcal{D}}\left[\sum_{t=0}^{T} \frac{\partial \psi_{\theta^*}(S_t, A_t)}{\partial \theta^*}\left(\sum_{j=t}^{T} \gamma_\varphi^{j-t} r_\phi(S_j, A_j)\right)\right] - \lambda.$$

These provide the necessary expressions for updating ϕ and φ in the outer loop. As $\mathbf{A}$ involves an outer product and $\mathbf{H}$ involves second derivatives, computing them *exactly* might not be practical when dealing with high-dimensions. Standard approximation techniques like conjugate-gradients or Neumann series can thus be used to make it more tractable [40]. In our experiments, we made use of the Neumann approximation to the Hessian Inverse vector product ($\mathbf{AH}^{-1}$, which requires the same magnitude of resources as the baseline policy gradient methods that we build on top off.

Algorithm: Being based on implicit gradients, we call our method `Barfi`, shorthand for *behavior alignment reward function's implicit* optimization. Overall, `Barfi` iteratively solves the bi-level optimization specified in (2) by alternating between using (4) till approximate converge of `Alg` to $\theta(\phi, \varphi)$ and then updating r_ϕ and γ_φ. Importantly, being based on (4) for sample efficiency, `Alg` leverages only the past samples and does *not* sample any new trajectories for the inner level optimization. Further, due to policy regularization which smoothens the objective as discussed in C, updates in r_ϕ and γ_φ changes the policy resulting from `Alg` gradually. Therefore, for compute efficiency, we start `Alg` from the policy obtained from the previous inner optimization, such that it is in proximity of the new fixed point. This allows `Barfi` to be both sample and compute efficient while solving the bi-level optimization iteratively online. Pseudo-code for `Barfi` and more details on the approximation techniques can be found in Appendix B.

E Details for the Empirical Results

E.1 Implementation Details

In this section we will briefly describe the implementation details around the different environments that were used.

GridWorld(GW): In the case of GridWorld we made use of the Fourier basis (of Order = 3) over the raw coordinates of agent position in the GridWorld. Details about this could be found in the `src/utils/Basis.py` file.

MountainCar(MC): For this environment, to reduce the limitation because of the function approximator we used TileCoding [58], which offers a suitable representation for the MountainCar problem. We used 4 Tiles and Tilings of 5.

CartPole(CP): For CartPole also make use of Fourier Basis of (Order = 3), with linear function approximator on top of that.

MuJoco(MJ): For this we made use of a neural network with 1 hidden layer of 32 nodes and ReLU activation as the function approximator over the raw observations. The output of the policy is continuous actions, hence we used a Gaussian representation, where the policy outputs the mean of the multivariate Gaussian and we used a fixed diagonal standard deviation, fixed to $\sigma = 0.1$.

General Details: All the outer returns are evaluated without any discounting, whereas all the inner optimizations were initialized with $\gamma_\varphi = 0.99$. Hence to do this we made φ a single bias unit, initialized to 4.6, and passed through a sigmoid (i.e., $\sigma(4.6) = 0.99$).

For GW, CP and MC r_ϕ is defined as below

$$r_\phi(s, a) = \phi_1(s) + \phi_2(s) r_p + \phi_3(s) r_a$$

Wherein ϕ_1, ϕ_2, ϕ_3 are scalar outputs of a 3-headed function, in this case simply a linear layer over the states inputs.

Whereas in the case of MJ, we have

$$r_\phi(s, a) = \phi_1 + r_p + \phi_3 r_a$$

Wherein ϕ_1 is initialized to zero and ϕ_3 is 1.0 act like bias units.

Gradient normalization was used for all the cases where Neural Nets were involved (i.e., MJ), and also for MJ we modified the Baseline (Reinforce) update to subtract the running average of the performance as a baseline to get acceptable performance for the baseline method.

E.2 Hyper-parameter Selection

As different make use of different function approximators hence the range of hyper-params can vary we talk about all the above over here. All the experiments were conducted on a personal computer with 32 GiB of memory and an Intel Core i7 CPU with 12 threads. Total runtime for all the experiments combined was less than a day.

Best-performing Parameters for different methods and environments are listed where

Hyper Parameter	Barfi Value	Reinforce Value	Actor Critic Value
α_θ	1×10^{-3}	1×10^{-3}	1×10^{-3}
α_ϕ	5×10^{-3}	–	–
α_φ	5×10^{-3}	–	–
`optim`	RMSprop	RMSprop	RMSprop
λ_θ	0.25	0.25	0.25
λ_ϕ	0.0625	–	–
λ_φ	4.0	–	–
Buffer	1000	–	–
Batch Size	1	1	1
η	0.0005	–	–
δ	3	–	–
n	5	–	–
N_0	150	–	–
N_i	15	–	–

Table 2: Hyper parameters for GridWorld

Hyper Parameter	Barfi Value	Reinforce Value	Actor Critic Value
α_θ	0.015625	0.125	0.03125
α_ϕ	0.0625	–	–
α_φ	0.0625	–	–
`optim`	RMSprop	RMSprop	RMSprop
λ_θ	0.0	0.0	0.25
λ_ϕ	0.0	–	–
λ_φ	0.25	–	–
Buffer	50	–	–
Batch Size	1	1	1
η	0.001	–	–
δ	3	–	–
n	5	–	–
N_0	50	–	–
N_i	15	–	–

Table 3: Hyper parameters for MountainCar

Hyper Parameter	Barfi Value	Reinforce Value	Actor Critic Value
α_θ	1×10^{-3}	1×10^{-3}	5×10^{-4}
α_ϕ	1×10^{-3}	–	–
α_φ	5×10^{-3}	–	–
`optim`	RMSprop	RMSprop	RMSprop
λ_θ	1.0	1.0	0.0
λ_ϕ	0.0	–	–
λ_φ	4.0	–	–
Buffer	10000	–	–
Batch Size	1	1	1
η	0.0005	–	–
δ	3	–	–
n	5	–	–
N_0	150	–	–
N_i	15	–	–

Table 4: Hyper parameters for CartPole

Hyperparameter Sweep : Here we list the details about how we swept the values for different hyper-params. We used PyTorch [47] for all our implementations. We usually used an optimizer between RMSProp or Adam

Hyper Parameter	Barfi Value	Reinforce Value	Actor Critic Value
α_θ	7.5×10^{-5}	5×10^{-4}	2.5×10^{-4}
α_ϕ	2.5×10^{-3}	–	–
α_φ	0.0	–	–
`optim`	Adam	Adam	Adam
λ_ϕ	0.0625	–	–
λ_φ	0.0	–	–
Buffer	50	–	–
Batch Size	1	1	1
η	0.0005	–	–
δ	3	–	–
n	5	–	–
N_0	30	–	–
N_i	15	–	–

Table 5: Hyper parameters for MuJoco

with default parameters as provided in Pytorch. For $\alpha_\theta \in \{5 \times 10^{-3}, 2.5 \times 10^{-3}, 1 \times 10^{-3}, 5 \times 10^{-4}, 2.5 \times 10^{-4}, 1 \times 10^{-4}, 7.5 \times 10^{-5}\}$, we use similar ranges for $\alpha_\phi, \alpha_\varphi$ (which tend to be larger). For λ_θ and λ_ϕ, we sweeped from $[0, 0.25, 0.5, 1.0]$ and for λ_γ we sweeped from $[0, 0.25, 1.0, 4.0, 16.0]$. We simply list ranges for different values and later we present sensitivity curves showing that these values are usually robust for `Barfi` across different methods as we can see from the tables above. $\delta \in [1, 3, 5]$, $n \in [1, 3, 5]$, $N_i \in [1, 3, 6, 9, 12, 15]$, $\eta \in [1 \times 10^{-3}, 5 \times 10^{-4}, 1 \times 10^{-4}]$, $N_0 \in [30, 50, 100, 150]$, buffer $\in [25, 50, 100, 1000]$. α for Tilecoding was adopted from [58] and hence similar ranges were sweeped in that case. Most sweeps were done with around 10 seeds, and later the parameter ranges were reduced and performed with more seeds.

E.3 Compute

The computer is used for a cluster where the CPU class is Intel Xeon Gold 6240 CPU @2.60GHz. The total compute required for GW was around 3 CPU years[2], CP also required around 3 CPU years, and MC required around 4 CPU years. For MJ we needed around 5-6 CPU years. In total we utilized around 15-16 CPU years, where we needed around 1 GB of memory per thread.

F Extra Results & Ablations

Figure 8 and Figure 9 summarize the return based on r_ϕ and the γ learned by the agent across different domains and reward specification. We observe that **Reinforce** often optimizes the naive combination of reward for sure, but that doesn't really lead to a good performance on r_p, whereas **Barfi** does achieve appropriate return on r_ϕ, but is also able to successively decay γ as the learning progress across different domains. Particularly notice Figure 8 (a) Bottom, where **Reinforce** does optimize aux return a lot, but actually fails to solve the problem, as it simply learns to loop around the center state.

Ablations: Figure 10 represents the ablation of **Barfi** on GridWorld with the misspecified reward for its different params. We can see that usually having $\eta = 0.001, 0.0005$, $n = 5$ works for the approximation.

[2] 1 CPU year := Compute equal to running a CPU thread for a year.

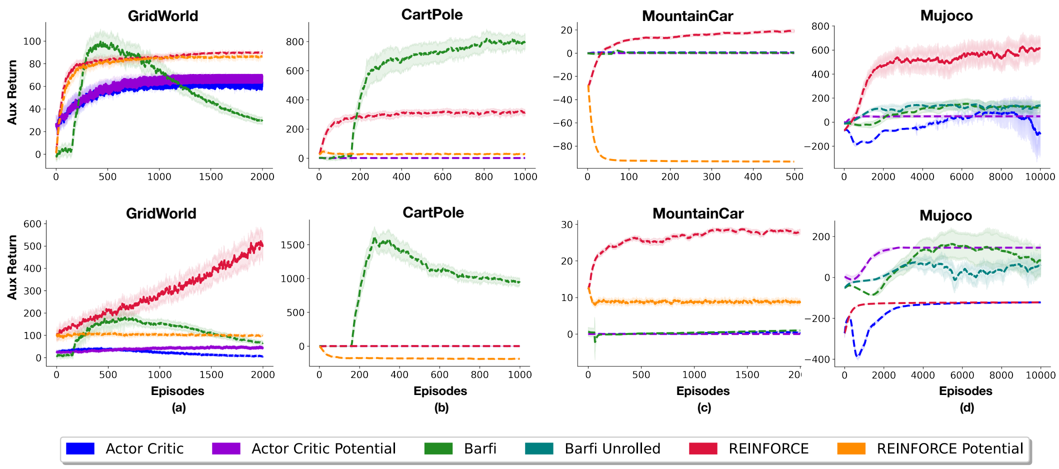

Figure 8: **Learned Reward Returns:** This figure illustrates the aux return collected by agent based on the learned r_ϕ, the curves are chosen based on best performing curves on r_p, and averaged over 20 runs (except 40 for GW). **(a) Top** – $r^1_{\texttt{aux,GW}}$, **Bottom** – $r^2_{\texttt{aux,GW}}$, **(b) Top** – $r^1_{\texttt{aux,CP}}$, **Bottom** – $r^2_{\texttt{aux,CP}}$, **(c) Top** – $r^2_{\texttt{aux,MC}}$, **Bottom** – $r^1_{\texttt{aux,MC}}$, **(d) Top** – $r^1_{\texttt{aux,MJ}}$, **Bottom** – $r^2_{\texttt{aux,MJ}}$.

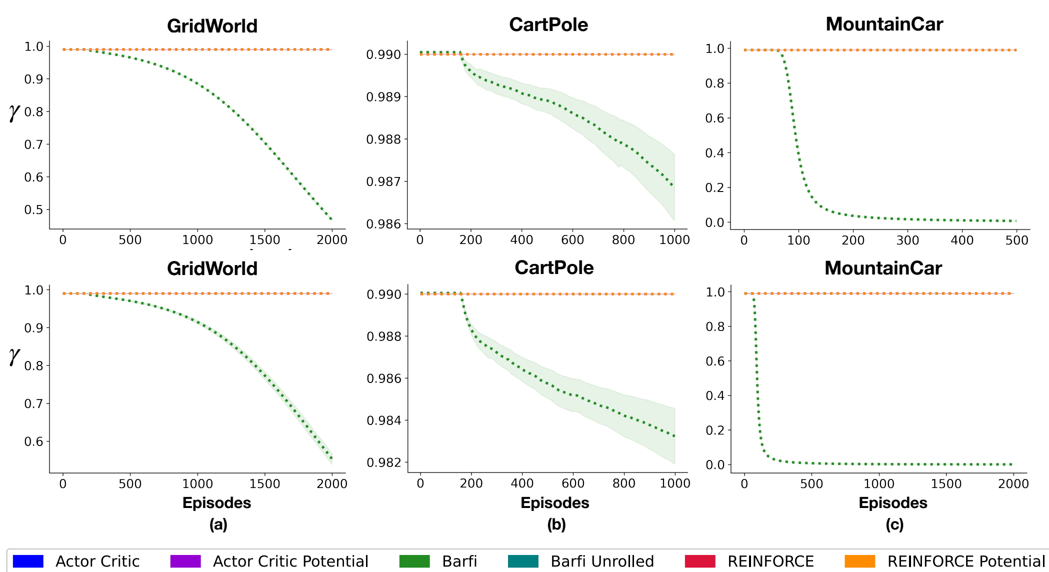

Figure 9: **Learned γ_φ:** This figure illustrates the learned γ_φ for `Barfi` and normal γ for other methods, the curves are chosen based on best-performing curves on r_p, and averaged over 20 runs (except 40 for GW). **(a) Top** – $r^1_{\texttt{aux,GW}}$, **Bottom** – $r^2_{\texttt{aux,GW}}$, **(b) Top** – $r^1_{\texttt{aux,CP}}$, **Bottom** – $r^2_{\texttt{aux,CP}}$, **(c) Top** – $r^2_{\texttt{aux,MC}}$, **Bottom** – $r^1_{\texttt{aux,MC}}$.. MJ is not included as the γ was not learned in that case. We can observe that the agents start to learn to decay γ at the appropriate pace.

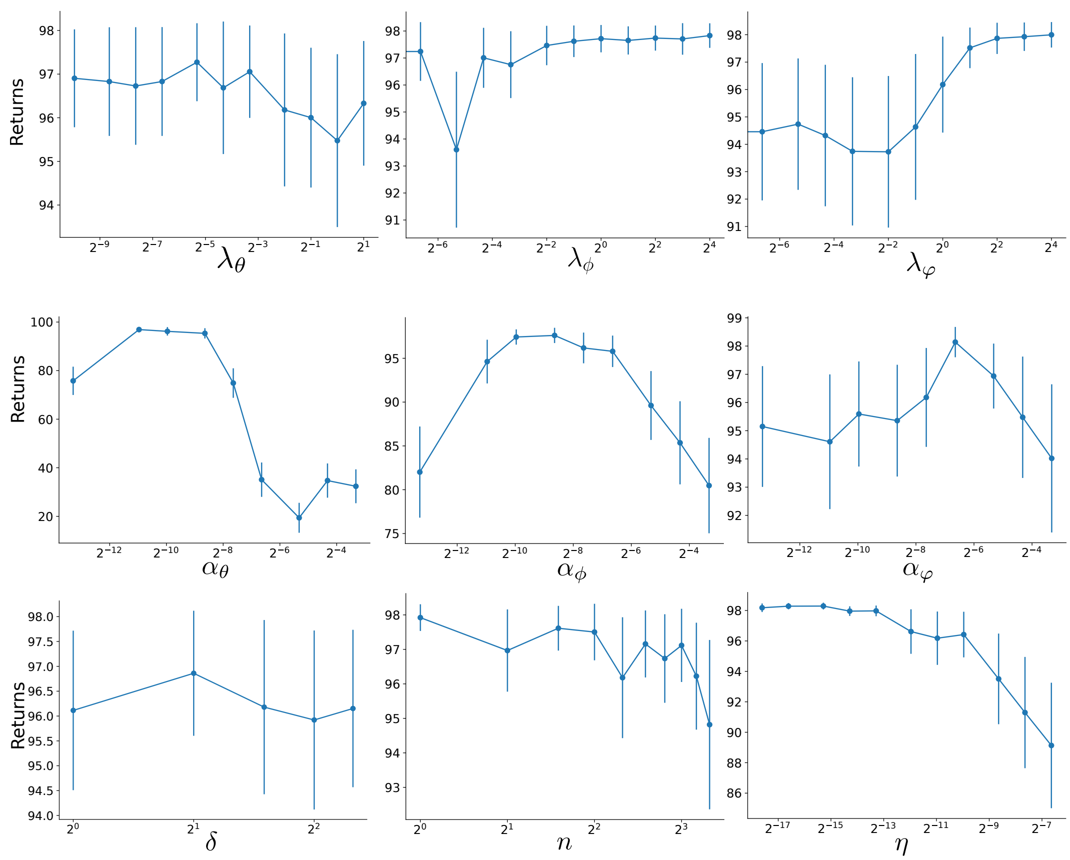

Figure 10: **Sensitivity Curves:** The set of graphs representing the sensitivity of different hyper-params keeping all the other params fixed. The sensitivity is for **`Barfi`** in GW with $r^2_{\texttt{aux,GW}}$, i.e., the misspecified reward. We choose the best-performing parameters and vary each parameter to see its influence. The curves are obtained for 50 runs (seeds) in each case, and error bars are standard errors. We can notice that α_θ and α_ϕ can have a large influence, and tend to stay around similar values. $\lambda_{\theta,\phi,\varphi}$ tends to help but doesn't really influence a lot in terms of its magnitude, except larger values of λ_φ seem to do better. Smaller values of η seems to work fine, hence something around $5 \times 10^{-4}, 1 \times 10^{-3}$ usually should suffice. n, δ can be chosen to around 5 and 3, and usually workout fine. We also defined $N_i = 5 \times \delta$ in this case.