A. Matrix Notation for Proofs

In this section, we introduce matrix-related notation for the proofs. Part of this notation is derived from Sutton (1988). We refer to state features using vectors indexed by the state. So features $\boldsymbol{x}(i)$ for state i is referred to as $\boldsymbol{x}_i$. Reward $r(j|i,a)$ (the reward for transitioning to state j from state i after taking action a) is referred to by r^a_{ij}. The policy $\pi(a|i)$ is π^a_i and the transition function $P(j|i,a)$ is accordingly the value p^a_{ij}. $\mathcal{N}$ and $\mathcal{T}$ are the set of non-terminal and terminal states respectively, and $\mathcal{A}$ is the set of possible actions. $\mathcal{D}$ is the batch of data used to train the value function. For any transition from a non-terminal state to a (non-) terminal state, $i \to j$, $1 \le i \le |\mathcal{N}|$ and $1 \le j \le |\mathcal{N} \cup \mathcal{T}|$. Finally, the maximum-likelihood estimate (MLE) of the above quantities according to $\mathcal{D}$, is given with a hat ($\hat{}$) on top of the quantity. Further notations that are used in the proofs are explained when introduced.

Notation	Description	Dimension
$\mathcal{S}$	set of states	$\lvert\mathcal{S}\rvert$
$\widehat{\mathcal{S}}$	set of states that appear in batch $\mathcal{D}$	$\lvert\widehat{\mathcal{S}}\rvert$
$\mathcal{A}$	set of actions	$\lvert\mathcal{A}\rvert$
$\widehat{\mathcal{A}}_i$	set of actions that appear in batch $\mathcal{D}$ when agent is in state i	$\lvert\widehat{\mathcal{A}}_i\rvert$
$\mathcal{N} \subset \mathcal{S}$	non-terminal states	$\lvert\mathcal{N}\rvert$
$\widehat{\mathcal{N}} \subset \widehat{\mathcal{S}}$	non-terminal states that appear in batch $\mathcal{D}$	$\lvert\widehat{\mathcal{N}}\rvert$
$\mathcal{T} \subset \mathcal{S}$	terminal states	$\lvert\mathcal{T}\rvert$
$\widehat{\mathcal{T}} \subset \widehat{\mathcal{S}}$	terminal states that appear in batch $\mathcal{D}$	$\lvert\widehat{\mathcal{T}}\rvert$
I	identity matrix	$\lvert\mathcal{N}\rvert \times \lvert\mathcal{N}\rvert$
p^π_{jk}	probability of transitioning from state j to state k under a policy, π	-
p^a_{jk}	probability of transitioning from state j to state k after taking action a	-
$\bar{r}_j$	mean reward when transitioning from state j	-
$\bar{r}^a_{ij}$	mean reward when transitioning from state i to state j after taking action a	-
Q	$Q_{jk} := p^\pi_{jk}$	$\lvert\mathcal{N}\rvert \times \lvert\mathcal{N}\rvert$
$[\mathbf{m}]_i$	expected reward on transitioning from state i to non-terminal state j i.e. $\sum_{j \in \mathcal{N}} p^\pi_{ij} r_{ij}$ or $\sum_{j \in \mathcal{N}} \sum_{a \in \mathcal{A}} \pi^a_i p^a_{ij} r^a_{ij}$	$\lvert\mathcal{N}\rvert$
$[\mathbf{h}]_i$	expected reward on transitioning from state i to non-terminal state j to terminal state i.e. $\sum_{j \in \mathcal{T}} p^\pi_{ij} r_{ij}$ or $\sum_{j \in \mathcal{T}} \sum_{a \in \mathcal{A}} \pi^a_i p^a_{ij} r^a_{ij}$	$\lvert\mathcal{N}\rvert$
$d_{\pi_e}(i)$	$\forall i \in \mathcal{S}$ weighted proportion of time spent in state i under policy π_e	-

B. Fixed-point for an MDP in the Per-step Reward and Discounted Case

In this section, we establish several fixed-points that we expect the value function to converge to in the discounted per-step reward case for an MRP and MDP. Note that Sutton (1988) derived these fixed-points for MRPs when rewards are only received on termination and there is no discounting. We first specify the fixed-points for an MRP in the discounted per-step reward case, and then extend this result for MDPs.

For both an MRP and MDP, we establish two types of fixed-points in the per-step reward and discounted case. The first fixed-point is the *true* fixed-point in that it is the value function computed assuming that we have access to the true policy and transition dynamics distributions. Ideally, we would like our value function learning algorithms to converge to this fixed-point. The second fixed-point is the *certainty-equivalence estimate* fixed-point, which is the value function computed using the maximum-likelihood estimates of the policy and transition dynamics from a batch of fixed data. We note that due to sampling error in the policy and transition dynamics, the certainty-equivalence estimate is an *inaccurate* estimate of the true value function. Finally, and only for MDPs, we specify the fixed-point that we expect the value function to learn after applying PSEC.

B.1. MRP True Fixed-Point

The true value function v for a state, $i \in \mathcal{N}$, induced by the policy-integrated transition dynamics p^π, reward function r, and policy, π, is given by:

$$\begin{aligned}
v(i) &= \sum_{j \in \mathcal{N} \cup \mathcal{T}} p^\pi_{ij} \left[r_{ij} + \gamma v(j) \right] && \text{Bellman equation} \\
&= \sum_{j \in \mathcal{N}} p^\pi_{ij} \left[r_{ij} + \gamma v(j) \right] + \sum_{j \in \mathcal{T}} p^\pi_{ij} r_{ij} && \text{expected return from } \mathcal{T}, v(\mathcal{T}) = 0 \\
&= \sum_{j \in \mathcal{T}} p^\pi_{ij} r_{ij} + \sum_{j \in \mathcal{N}} p^\pi_{ij} \left[r_{ij} + \gamma \left[\sum_{k \in \mathcal{N} \cup \mathcal{T}} p^\pi_{jk} \left[r_{jk} + \gamma v(k) \right] \right] \right] && \text{recursively apply } v(i)
\end{aligned}$$

$$\begin{aligned}
v(i) = &\sum_{j \in \mathcal{T}} p^\pi_{ij} r_{ij} + \sum_{j \in \mathcal{N}} p^\pi_{ij} r_{ij} + \gamma \sum_{j \in \mathcal{N}} p^\pi_{ij} \sum_{k \in \mathcal{N} \cup \mathcal{T}} p^\pi_{jk} r_{jk} \\
&+ \gamma^2 \sum_{j \in \mathcal{N}} p^\pi_{ij} \sum_{k \in \mathcal{N} \cup \mathcal{T}} p^\pi_{jk} v(k) \\
= &\sum_{j \in \mathcal{T}} p^\pi_{ij} r_{ij} + \sum_{j \in \mathcal{N}} p^\pi_{ij} r_{ij} \\
&+ \gamma \sum_{j \in \mathcal{N}} p^\pi_{ij} \sum_{k \in \mathcal{N}} p^\pi_{jk} r_{jk} + \gamma \sum_{j \in \mathcal{N}} p^\pi_{ij} \sum_{k \in \mathcal{T}} p^\pi_{jk} r_{jk} \\
&+ \gamma^2 \sum_{j \in \mathcal{N}} p^\pi_{ij} \sum_{k \in \mathcal{N}} p^\pi_{jk} v(k) && \text{splitting } \mathcal{N} \text{ and } \mathcal{T}
\end{aligned}$$

We define vectors, $\mathbf{h}$ and $\mathbf{m}$ with, $[\mathbf{h}]_i = \sum_{j \in \mathcal{T}} p^\pi_{ij} r_{ij}$, $[\mathbf{m}]_i = \sum_{j \in \mathcal{N}} p^\pi_{ij} r_{ij}$, and Q is the true transition matrix of the Markov reward process induced by π and P, i.e., $[Q]_{ij} = p^\pi_{ij}$. Then continuing from above, we have

$$v(i) = [\mathbf{h}]_i + [\mathbf{m}]_i + \gamma Q[\mathbf{h}]_i + \gamma Q[\mathbf{m}]_i + \gamma^2 Q^2[\mathbf{h}]_i + \gamma^2 Q^2[\mathbf{m}]_i + \dots \quad \text{unrolling } v(\mathcal{N} \cup \mathcal{T}) \tag{6}$$

$$= \left[\sum_{k=0}^{\infty} (\gamma Q)^k (\mathbf{m} + \mathbf{h}) \right]_i \tag{7}$$

$$v(i) = \left[(I - \gamma Q)^{-1} (\mathbf{m} + \mathbf{h}) \right]_i \tag{8}$$

The existence of the limit and inverse are assured by Theorem A.1 in Sutton (1988). The theorem is applicable here since $\lim_{k \to \infty} (\gamma Q)^k = 0$.

B.2. MRP Certainty-Equivalence Fixed -Point

For the certainty-equivalence fixed-point, we consider a batch of data, $\mathcal{D}$. We follow the same steps and similar notation from Equation (8), with the slight modification that the maximum-likelihood estimate (MLE) of the above quantities according to $\mathcal{D}$, is given with a hat ($\hat{}$) on top of the quantity. The observed sets of non-terminal and terminal states in the batch are given by $\widehat{\mathcal{N}}$ and $\widehat{\mathcal{T}}$ respectively.

Then similar to above, we can derive the certainty-equivalence estimate of the value function according to the MLE of the MRP transition dynamics from the batch for a particular state i, $\forall i \in \widehat{N}$ is:

$$\hat{v}(i) = \left[(I - \gamma \widehat{Q})^{-1} (\hat{\mathbf{m}} + \hat{\mathbf{h}}) \right]_i \tag{9}$$

B.3. MDP True Fixed-Point

The true value function, v^π, for a policy, π, for a state i, $\forall i \in \mathcal{N}$, induced by the transition dynamics and reward function, p and r is given by:

$$
\begin{aligned}
v^\pi(i) &= \sum_{a\in\mathcal{A}} \pi_i^a \sum_{j\in\mathcal{N}\cup\mathcal{T}} p_{ij}^a \left[r_{ij}^a + \gamma v^\pi(j)\right] && \text{Bellman equation} \\
&= \sum_{a\in\mathcal{A}} \pi_i^a \sum_{j\in\mathcal{N}} p_{ij}^a \left[r_{ij}^a + \gamma v^\pi(j)\right] + \sum_{a\in\mathcal{A}} \pi_i^a \sum_{j\in\mathcal{T}} p_{ij}^a r_{ij}^a && \text{expected return from } \mathcal{T}, v^\pi(\mathcal{T}) = 0
\end{aligned}
$$

$$
\begin{aligned}
v^\pi(i) &= \sum_{a\in\mathcal{A}} \pi_i^a \sum_{j\in\mathcal{T}} p_{ij}^a r_{ij}^a + \sum_{a\in\mathcal{A}} \pi_i^a \sum_{j\in\mathcal{N}} p_{ij}^a \left[r_{ij}^a + \gamma \left[\sum_{a'\in\mathcal{A}} \pi_j^{a'} \sum_{k\in\mathcal{N}\cup\mathcal{T}} p_{jk}^{a'} \left[r_{jk}^{a'} + \gamma v^\pi(k)\right]\right]\right] \\
&= \sum_{j\in\mathcal{T}} \sum_{a\in\mathcal{A}} \pi_i^a p_{ij}^a r_{ij}^a + \sum_{j\in\mathcal{N}} \sum_{a\in\mathcal{A}} \pi_i^a p_{ij}^a r_{ij}^a + \gamma \sum_{j\in\mathcal{N}} \sum_{a\in\mathcal{A}} \pi_i^a p_{ij}^a \sum_{k\in\mathcal{N}\cup\mathcal{T}} \sum_{a'\in\mathcal{A}} \pi_j^{a'} p_{jk}^{a'} r_{jk}^{a'} \\
&+ \gamma^2 \sum_{j\in\mathcal{N}} \sum_{a\in\mathcal{A}} \pi_i^a p_{ij}^a \sum_{k\in\mathcal{N}\cup\mathcal{T}} \sum_{a'\in\mathcal{A}} \pi_j^{a'} p_{jk}^{a'} v^\pi(k) \\
&= \sum_{j\in\mathcal{T}} \sum_{a\in\mathcal{A}} \pi_i^a p_{ij}^a r_{ij}^a + \sum_{j\in\mathcal{N}} \sum_{a\in\mathcal{A}} \pi_i^a p_{ij}^a r_{ij}^a \\
&+ \gamma \sum_{j\in\mathcal{N}} \sum_{a\in\mathcal{A}} \pi_i^a p_{ij}^a \sum_{k\in\mathcal{N}} \sum_{a'\in\mathcal{A}} \pi_j^{a'} p_{jk}^{a'} r_{jk}^{a'} + \gamma \sum_{j\in\mathcal{N}} \sum_{a\in\mathcal{A}} \pi_i^a p_{ij}^a \sum_{k\in\mathcal{T}} \sum_{a'\in\mathcal{A}} \pi_j^{a'} p_{jk}^{a'} r_{jk}^{a'} \\
&+ \gamma^2 \sum_{j\in\mathcal{N}} \sum_{a\in\mathcal{A}} \pi_i^a p_{ij}^a \sum_{k\in\mathcal{N}} \sum_{a'\in\mathcal{A}} \pi_j^{a'} p_{jk}^{a'} v^\pi(k) && \text{splitting } \mathcal{N} \text{ and } \mathcal{T}
\end{aligned}
$$

Similar to earlier, we have vectors, $\mathbf{h}$ and $\mathbf{m}$ with, $[\mathbf{h}]_i = \sum_{j\in\mathcal{T}} \sum_{a\in\mathcal{A}} \pi_i^a p_{ij}^a r_{ij}^a$, $[\mathbf{m}]_i = \sum_{j\in\mathcal{N}} \sum_{a\in\mathcal{A}} \pi_i^a p_{ij}^a r_{ij}^a$, and Q is the true transition matrix of the Markov reward process induced by π and P, i.e., $[Q]_{ij} = \sum_a \pi_i^a p_{ij}^a$. The terms are not overloaded since the expectation over the true policy yields the same values. Then continuing from above, we have

$$
\begin{aligned}
v^\pi(i) &= [\mathbf{h}]_i + [\mathbf{m}]_i + \gamma Q[\mathbf{h}]_i + \gamma Q[\mathbf{m}]_i + \gamma^2 Q^2[\mathbf{h}]_i + \gamma^2 Q^2[\mathbf{m}]_i + \dots && \text{unrolling } v^\pi(\mathcal{N}\cup\mathcal{T}) \quad (10) \\
&= \left[\sum_{k=0}^{\infty} (\gamma Q)^k (\mathbf{m} + \mathbf{h})\right]_i && (11) \\
v^\pi(i) &= \left[(I - \gamma Q)^{-1}(\mathbf{m} + \mathbf{h})\right]_i && (12)
\end{aligned}
$$

The existence of the limit and inverse are assured by Theorem A.1 in Sutton (1988). The theorem is applicable here since $\lim_{k\to\infty}(\gamma Q)^k = 0$.

B.4. MDP Certainty-Equivalence Fixed-Point

Similar to the above subsection, for certainty-equivalence fixed-point, we consider a batch of data, $\mathcal{D}$, with the maximum-likelihood estimate (MLE) of the above quantities according to $\mathcal{D}$ given with a hat ($\hat{}$) on top of the quantity. The observed sets of non-terminal and terminal states in the batch are given by $\widehat{\mathcal{N}}$ and $\widehat{\mathcal{T}}$ respectively.

Then similar to above, we can derive the certainty-equivalence estimate of the value function according to the MLE of the policy and transition dynamics from the batch for a particular state $i, \forall i \in \widehat{\mathcal{N}}$ is:

$$
v^{\hat{\pi}}(i) = \left[(I - \gamma \widehat{Q})^{-1}(\hat{\mathbf{m}} + \hat{\mathbf{h}})\right]_i \qquad (13)
$$

This fixed-point is called the certainty-equivalence estimate (CEE) (Sutton, 1988) for an MDP. We note that MLE of the policy and transition dynamics according to the batch may not be representative of the true policy and transition dynamics. In that case, MDP-CEE (Equation (13)) is inaccurate with respect to Equation (12) due to policy and transition dynamics *sampling error*.

B.5. Policy Sampling Error Corrected MDP Certainty-Equivalence Fixed-Point

We now derive a new fixed-point, the *policy sampling error corrected MDP certainty-equivalence fixed-point*. This fixed-point corrects the policy sampling error that occurs in the value function given by Equation (13), making the estimation more accurate with respect to the true value function given by Equation (12).

We introduce the PSEC weight, $\hat{\rho}_i^a = \frac{\pi_i^a}{\hat{\pi}_i^a}$, with π being the policy that we are interested in evaluating and $\hat{\pi}$ being the MLE of the policy according to batch $\mathcal{D}$. $\hat{\rho}$ is then applied to the above quantities to introduce a slightly modified notation. In particular, $\hat{\rho}$ applied to $\widehat{Q}$ results in $[\widehat{U}]_{ij} = \sum_{a \in \hat{\mathcal{A}}_i} \hat{\rho}_i^a \hat{\pi}_i^a \hat{p}_{ij}^a$, and applied to vectors $\hat{\mathbf{h}}$ and $\hat{\mathbf{m}}$ results in $[\hat{\mathbf{l}}]_i = \sum_{j \in \mathcal{T}} \sum_{a \in \hat{\mathcal{A}}_i} \hat{\rho}_i^a \hat{\pi}_i^a \hat{p}_{ij}^a \bar{r}_{ij}^a$ and $[\hat{\mathbf{o}}]_i = \sum_{j \in \mathcal{N}} \sum_{a \in \hat{\mathcal{A}}_i} \hat{\rho}_i^a \hat{\pi}_i^a \hat{p}_{ij}^a \bar{r}_{ij}^a$ respectively. After simplification, we have $[\widehat{U}]_{ij} = \sum_{a \in \hat{\mathcal{A}}_i} \pi_i^a \hat{p}_{ij}^a$, $[\hat{\mathbf{l}}]_i = \sum_{j \in \mathcal{T}} \sum_{a \in \hat{\mathcal{A}}_i} \pi_i^a \hat{p}_{ij}^a \bar{r}_{ij}^a$, and $[\hat{\mathbf{o}}]_i = \sum_{j \in \mathcal{N}} \sum_{a \in \hat{\mathcal{A}}_i} \pi_i^a \hat{p}_{ij}^a \bar{r}_{ij}^a$. Using these policy sampling error corrected quantities, we can derive the fixed-point for true policy, π, in a similar manner as earlier:

$$v^\pi(i) = \left[(I - \gamma \widehat{U})^{-1} (\hat{\mathbf{o}} + \hat{\mathbf{l}}) \right]_i \tag{14}$$

In computing this new fixed-point, we have corrected for the policy sampling error, resulting in a more accurate estimation of Equation (12) than Equation (13). Now, the value function is computed for the true policy that we are interested in evaluating, π.

C. Convergence of Batch Linear TD(0) to the MRP CE Fixed-Point

Theorem 1 (Batch Linear TD(0) Convergence)**.** *For any batch whose observation vectors $\{\boldsymbol{x}(s) | s \in \widehat{\mathcal{S}}\}$ are linearly independent, there exists an $\epsilon > 0$ such that, for all positive $\alpha < \epsilon$ and for any initial weight vector, the predictions for linear TD(0) converge under repeated presentations of the batch with weight updates after each complete presentation to the fixed-point (3).*

Proof. Batch linear TD(0) makes an update to the weight vector, $\boldsymbol{w}_n$ (of dimension, length of the feature vector), after each presentation of the batch:

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n + \sum_{\tau \in \mathcal{D}} \sum_{t=1}^{L_\tau} \alpha \left[(\bar{r}_t + \gamma \boldsymbol{w}_n^T \boldsymbol{x}_{t+1}) - \boldsymbol{w}_n^T \boldsymbol{x}_t \right] \boldsymbol{x}_t$$

where $\mathcal{D}$ is the batch of episodes, L_τ is the length of each episode τ, and α is the learning rate.

We can re-write the whole presentation of the batch of data in terms of the number of times there was a transition from state i to state j in the batch i.e. $\hat{c}_{ij} = \hat{d}_i \hat{p}_{ij}$, where $\hat{d}_i$ is the number of times state $i \in \widehat{\mathcal{N}}$ appears in the batch.

$$
\begin{aligned}
\boldsymbol{w}_{n+1} &= \boldsymbol{w}_n + \sum_{\tau \in \mathcal{D}} \sum_{t=1}^{L_\tau} \alpha \left[(\bar{r}_t + \gamma \boldsymbol{w}_n^T \boldsymbol{x}_{t+1} - \boldsymbol{w}_n^T \boldsymbol{x}_t) \right] \boldsymbol{x}_t \\
&= \boldsymbol{w}_n + \alpha \sum_{i \in \widehat{\mathcal{N}}} \sum_{j \in \widehat{\mathcal{N}} \cup \widehat{\mathcal{T}}} \hat{c}_{ij} \left[(\bar{r}_{ij} + \gamma \boldsymbol{w}_n^T \boldsymbol{x}_j - \boldsymbol{w}_n^T \boldsymbol{x}_i) \right] \boldsymbol{x}_i \\
&= \boldsymbol{w}_n + \alpha \sum_{i \in \widehat{\mathcal{N}}} \sum_{j \in \widehat{\mathcal{N}} \cup \widehat{\mathcal{T}}} \hat{d}_i \hat{p}_{ij} \left[(\bar{r}_{ij} + \gamma \boldsymbol{w}_n^T \boldsymbol{x}_j - \boldsymbol{w}_n^T \boldsymbol{x}_i) \right] \boldsymbol{x}_i \\
&= \boldsymbol{w}_n + \alpha \sum_{i \in \widehat{\mathcal{N}}} \sum_{j \in \widehat{\mathcal{N}} \cup \widehat{\mathcal{T}}} \hat{d}_i \hat{p}_{ij} (\bar{r}_{ij} + \gamma \boldsymbol{w}_n^T \boldsymbol{x}_j) \boldsymbol{x}_i - \alpha \sum_{i \in \widehat{\mathcal{N}}} \sum_{j \in \widehat{\mathcal{N}} \cup \widehat{\mathcal{T}}} \hat{d}_i \hat{p}_{ij} (\boldsymbol{w}_n^T \boldsymbol{x}_i) \boldsymbol{x}_i \\
&= \boldsymbol{w}_n + \alpha \sum_{i \in \widehat{\mathcal{N}}} \hat{d}_i \boldsymbol{x}_i \sum_{j \in \widehat{\mathcal{N}} \cup \widehat{\mathcal{T}}} \hat{p}_{ij} (\bar{r}_{ij} + \gamma \boldsymbol{w}_n^T \boldsymbol{x}_j) - \alpha \sum_{i \in \widehat{\mathcal{N}}} \hat{d}_i (\boldsymbol{w}_n^T \boldsymbol{x}_i) \boldsymbol{x}_i \sum_{j \in \widehat{\mathcal{N}} \cup \widehat{\mathcal{T}}} \hat{p}_{ij} \\
&= \boldsymbol{w}_n + \alpha \sum_{i \in \widehat{\mathcal{N}}} \hat{d}_i \boldsymbol{x}_i \left[\sum_{j \in \widehat{\mathcal{N}} \cup \widehat{\mathcal{T}}} \hat{p}_{ij} (\bar{r}_{ij} + \gamma \boldsymbol{w}_n^T \boldsymbol{x}_j) - \boldsymbol{w}_n^T \boldsymbol{x}_i \right] && \sum_{j \in \widehat{\mathcal{N}} \cup \widehat{\mathcal{T}}} \hat{p}_{ij} = 1 \\
&= \boldsymbol{w}_n + \alpha \sum_{i \in \widehat{\mathcal{N}}} \hat{d}_i \boldsymbol{x}_i \left[\left(\sum_{j \in \widehat{\mathcal{N}}} \hat{p}_{ij} (\bar{r}_{ij} + \gamma \boldsymbol{w}_n^T \boldsymbol{x}_j) \right) + \left(\sum_{j \in \widehat{\mathcal{T}}} \hat{p}_{ij} \bar{r}_{ij} \right) - \boldsymbol{w}_n^T \boldsymbol{x}_i \right] && \text{If } \boldsymbol{x}_j \in \widehat{\mathcal{T}}, \boldsymbol{w}_n^T \boldsymbol{x}_j = 0 \\
&= \boldsymbol{w}_n + \alpha \sum_{i \in \widehat{\mathcal{N}}} \hat{d}_i \boldsymbol{x}_i \left[\left(\sum_{j \in \widehat{\mathcal{N}}} \hat{p}_{ij} \bar{r}_{ij} \right) + \left(\gamma \sum_{j \in \widehat{\mathcal{N}}} \hat{p}_{ij} \boldsymbol{w}_n^T \boldsymbol{x}_j \right) + \left(\sum_{j \in \widehat{\mathcal{T}}} \hat{p}_{ij} \bar{r}_{ij} \right) - \boldsymbol{w}_n^T \boldsymbol{x}_i \right]
\end{aligned}
$$

$$
\boldsymbol{w}_{n+1} = \boldsymbol{w}_n + \alpha \widehat{X} \widehat{D} \left[\hat{\mathbf{m}} + \gamma \widehat{Q} \widehat{X}^T \boldsymbol{w}_n + \hat{\mathbf{h}} - \widehat{X}^T \boldsymbol{w}_n \right] \tag{15}
$$

where $\widehat{X}$ denotes the matrix (of dimensions, length of the feature vector by $|\widehat{\mathcal{S}}|$) with columns, $\boldsymbol{x}_i \in \widehat{\mathcal{S}}$ and $\widehat{D}$ is a diagonal matrix (of dimensions, $|\widehat{\mathcal{S}}|$ by $|\widehat{\mathcal{S}}|$) with $\widehat{D}_{ii} = \hat{d}_i$. Given the successive updates to the weight vector $\boldsymbol{w}_n$, we now consider the actual values predicted as the following by multiplying $\widehat{X}^T$ on both sides:

$$
\begin{aligned}
\widehat{X}^T \boldsymbol{w}_{n+1} &= \widehat{X}^T \boldsymbol{w}_n + \alpha \widehat{X}^T \widehat{X} \widehat{D} \left(\hat{\mathbf{m}} + \hat{\mathbf{h}} + \gamma \widehat{Q} \widehat{X}^T \boldsymbol{w}_n - \widehat{X}^T \boldsymbol{w}_n \right) \\
&= \widehat{X}^T \boldsymbol{w}_n + \alpha \widehat{X}^T \widehat{X} \widehat{D} \left(\hat{\mathbf{m}} + \hat{\mathbf{h}} \right) + \alpha \widehat{X}^T \widehat{X} \widehat{D} \left(\gamma \widehat{Q} \widehat{X}^T \boldsymbol{w}_n - \widehat{X}^T \boldsymbol{w}_n \right) \\
&= \alpha \widehat{X}^T \widehat{X} \widehat{D} \left(\hat{\mathbf{m}} + \hat{\mathbf{h}} \right) + \left(I - \alpha \widehat{X}^T \widehat{X} \widehat{D} \left(I - \gamma \widehat{Q} \right) \right) \widehat{X}^T \boldsymbol{w}_n
\end{aligned}
$$

We then unroll the above equation by recursively applying $\widehat{X}^T \boldsymbol{w}_n$ till $n = 0$.

$$
\begin{aligned}
\widehat{X}^T \boldsymbol{w}_{n+1} &= \alpha \widehat{X}^T \widehat{X} \widehat{D} \left(\hat{\mathbf{m}} + \hat{\mathbf{h}} \right) + \left(I - \alpha \widehat{X}^T \widehat{X} \widehat{D} \left(I - \gamma \widehat{Q} \right) \right) \alpha \widehat{X}^T \widehat{X} \widehat{D} \left(\hat{\mathbf{m}} + \hat{\mathbf{h}} \right) \\
&\quad + \left(I - \alpha \widehat{X}^T \widehat{X} \widehat{D} \left(I - \gamma \widehat{Q} \right) \right)^2 \widehat{X}^T \boldsymbol{w}_{n-1} \\
&\quad \vdots \\
&= \sum_{k=0}^{n-1} \left(I - \alpha \widehat{X}^T \widehat{X} \widehat{D} \left(I - \gamma \widehat{Q} \right) \right)^k \alpha \widehat{X}^T \widehat{X} \widehat{D} \left(\hat{\mathbf{m}} + \hat{\mathbf{h}} \right) \\
&\quad + \left(I - \alpha \widehat{X}^T \widehat{X} \widehat{D} \left(I - \gamma \widehat{Q} \right) \right)^n \widehat{X}^T \boldsymbol{w}_0
\end{aligned} \tag{16}
$$

Assuming that as $n \to \infty$, $(I - \alpha \widehat{X}^T \widehat{X} \widehat{D}(I - \gamma \widehat{Q}))^n \to 0$, we can drop the second term and the sequence $\{\widehat{X}^T \boldsymbol{w}_n\}$ converges to:

$$\begin{aligned}
\lim_{n\to\infty} \widehat{X}^T \boldsymbol{w}_n &= \left(I - (I - \alpha \widehat{X}^T \widehat{X} \widehat{D}(I - \gamma \widehat{Q}))\right)^{-1} (\alpha \widehat{X}^T \widehat{X} \widehat{D}(\hat{\mathbf{m}} + \hat{\mathbf{h}})) \\
&= (I - \gamma \widehat{Q})^{-1} \widehat{D}^{-1} (\widehat{X}^T \widehat{X})^{-1} \alpha^{-1} \alpha \widehat{X}^T \widehat{X} \widehat{D}(\hat{\mathbf{m}} + \hat{\mathbf{h}}) \\
&= (I - \gamma \widehat{Q})^{-1} (\hat{\mathbf{m}} + \hat{\mathbf{h}}) \\
\lim_{n\to\infty} \mathbb{E}\left[\boldsymbol{x}_i^T \boldsymbol{w}_n\right] &= \left[(I - \gamma \widehat{Q})^{-1} (\hat{\mathbf{m}} + \hat{\mathbf{h}})\right]_i, \forall i \in \widehat{\mathcal{N}}
\end{aligned}$$

What is left to show now is $n \to \infty$, $(I - \alpha \widehat{X}^T \widehat{X} \widehat{D}(I - \gamma \widehat{Q}))^n \to 0$. Following Sutton (1988), we first show that $\widehat{D}(I - \gamma \widehat{Q})$ is positive definite, and then that $\widehat{X}^T \widehat{X} \widehat{D}(I - \gamma \widehat{Q})$ has a full set of eigenvalues all of whose real parts are positive. This enables us to show that α can be chosen so that eigenvalues of $(I - \alpha \widehat{X}^T \widehat{X} \widehat{D}(I - \gamma \widehat{Q}))$ are less than 1 in modulus, which assures us that its powers converge to 0.

To show that $\widehat{D}(I - \gamma \widehat{Q})$ is positive definite, we refer to the Gershgorin Circle theorem (Gerschgorin, 1931), which states that if a matrix, A, is real, symmetric, and strictly diagonally dominant with positive diagonal entries, then A is positive definite. However, we cannot apply this theorem as is to $\widehat{D}(I - \gamma \widehat{Q})$ since the matrix is not necessarily symmetric. To use the theorem, we first apply another theorem (Theorem A.3 from Sutton (1988)) that states: a square matrix A is positive definite if and only if $A + A^T$ is positive definite. So it suffices to show that $\widehat{D}(I - \gamma \widehat{Q}) + (\widehat{D}(I - \gamma \widehat{Q}))^T$ is positive definite.

Consider the matrix $S = \widehat{D}(I - \gamma \widehat{Q}) + (\widehat{D}(I - \gamma \widehat{Q}))^T$. We know that S is real and symmetric. It remains to show that the diagonal entries are positive and that S is strictly diagonally dominant. First, we look at the diagonal entries, $S_{ii} = 2[\widehat{D}(I - \gamma \widehat{Q})]_{ii} = 2\hat{d}_i(1 - \gamma \hat{p}_{ii}) > 0, \forall i \in \widehat{\mathcal{N}}$, which are positive. Second, we have the non-diagonal entries for $i \neq j$ as $S_{ij} = [\widehat{D}(I - \gamma \widehat{Q})]_{ij} + [\widehat{D}(I - \gamma \widehat{Q})]_{ji} = -\gamma \hat{d}_i \hat{p}_{ij} - \gamma \hat{d}_j \hat{p}_{ji} \leq 0$, which are nonpositive. We want to show that $|S_{ii}| \geq \sum_{j \neq i} |S_{ij}|$, with strict inequality holding for at least one i; we know that the diagonal elements $S_{ii} > 0$ and non-diagonal elements $S_{ij} \leq 0$, $i \neq j$. Hence, to show that S is strictly diagonally dominant, it is enough to show that $S_{ii} > -\sum_{j \neq i} S_{ij}$, which means we can simply show that the sum of each entire row is greater than 0, i.e. $\sum_j S_{ij} > 0$.

Before we show that $\sum_j S_{ij} > 0$, we note that $\hat{d}^T = \hat{\mu}^T (I - \widehat{Q})^{-1}$ where $\hat{\mu}_i$ is the empirical state distribution of state i. Given the definitions of $\hat{d}$, $\hat{\mu}$, and $\widehat{Q}$, this fact follows from Kemeny et al. (1960) and is used by Sutton (1988). Using this fact, we show that $\sum_j S_{ij} \geq 0$:

$$\begin{aligned}
\sum_j S_{ij} &= \sum_j \left([\widehat{D}(I - \gamma \widehat{Q})]_{ij} + [\widehat{D}(I - \gamma \widehat{Q})]^T_{ij} \right) \\
&= \hat{d}_i \sum_j ([I - \gamma \widehat{Q}]_{ij} + \sum_j \hat{d}_j [I - \gamma \widehat{Q}]^T_{ij}) \\
&= \hat{d}_i \sum_j (1 - \gamma \hat{p}_{ij}) + \left[\hat{d}^T (I - \widehat{Q})\right]_i \\
&= \hat{d}_i \sum_j (1 - \gamma p_{ij}) + \left[\hat{\mu}^T (I - \widehat{Q})^{-1} (I - \widehat{Q})\right]_i \qquad \hat{d}^T = \hat{\mu}^T (I - \widehat{Q})^{-1} \\
&= \hat{d}_i (1 - \gamma \sum_j \hat{p}_{ij}) + \hat{\mu}_i \\
&\geq 0,
\end{aligned}$$

where the final inequality is strict since $\hat{\mu}$ is positive for at least one element. Given the above, we have shown that S is real, symmetric, and strictly diagonally dominant; hence, S is positive definite according to the Gershgorin Circle theorem (Gerschgorin, 1931). Since, $S = \widehat{D}(I - \gamma \widehat{Q}) + (\widehat{D}(I - \gamma \widehat{Q}))^T$ is positive definite, we have $\widehat{D}(I - \gamma \widehat{Q})$ to be positive definite.

Now we need to show that $\widehat{X}^T \widehat{X} \widehat{D}(I - \gamma \widehat{Q})$ has a full set of eigenvalues, all of whose real parts are positive. We know that $\widehat{X}^T \widehat{X} \widehat{D}(I - \gamma \widehat{Q})$ has a full set of eigenvalues for the same reason shown by Sutton (1988), i.e. $\widehat{X}^T \widehat{X} \widehat{D}(I - \gamma \widehat{Q})$ is a product of three non-singular matrices, which means $\widehat{X}^T\widehat{X}\widehat{D}(I-\gamma\widehat{Q})$ is nonsingular as well; hence, no eigenvalues are 0 i.e. its set of eigenvalues is full.

Consider λ and y to be an eigevalue and eigenvector pair of $\widehat{X}^T\widehat{X}\widehat{D}(I-\gamma\widehat{Q})$. First lets consider that y may be a complex number and is of the form $y = a + bi$, and let $z = (\widehat{X}^T\widehat{X})^{-1}y, y \neq 0$. Second, we consider $\widehat{D}(I-\gamma\widehat{Q})$ from earlier i.e. where $*$ is the conjugate-transpose:

$$\begin{aligned}
y^*\widehat{D}(I-\gamma\widehat{Q})y &= z^*\widehat{X}^T\widehat{X}\widehat{D}(I-\gamma\widehat{Q})y && \text{substituting } y^* \\
&= z^*\lambda y && \widehat{X}^T\widehat{X}\widehat{D}(I-\gamma\widehat{Q})y = \lambda y \\
&= \lambda z^*\widehat{X}^T\widehat{X}z && \text{substituting } y \\
&= \lambda(\widehat{X}z)^*\widehat{X}z \\
(a^T - b^Ti)(\widehat{D}(I-\gamma\widehat{Q}))(a^T + b^Ti) &= \lambda(\widehat{X}z)^*\widehat{X}z && \text{substituting } y^* \text{ and } y
\end{aligned}$$

From the above equality, we know that the real parts (Re) of the LHS and RHS are equal as well i.e.

$$\begin{aligned}
\text{Re}\left(y^*\widehat{D}(I-\gamma\widehat{Q})y\right) &= \text{Re}\left(\lambda(\widehat{X}z)^*\widehat{X}z\right) \\
a^T\widehat{D}(I-\gamma\widehat{Q})a + b^T\widehat{D}(I-\gamma\widehat{Q})b &= (\widehat{X}z)^*\widehat{X}z\text{Re}\,(\lambda)
\end{aligned}$$

LHS must be strictly positive since we already proved that $\widehat{D}(I-\gamma\widehat{Q})$ is positive definite and by definition, RHS, $(\widehat{X}z)^*\widehat{X}z$, is strictly positive as well. Thus, the Re(λ) must be positive. Finally, using this result we want to show that the eigenvalues of $(I - \alpha\widehat{X}^T\widehat{X}\widehat{D}(I-\gamma\widehat{Q}))$ are of modulus less than 1 for a suitable α.

First, we can see that y is also an eigenvector of $(I - \alpha\widehat{X}^T\widehat{X}\widehat{D}(I-\gamma\widehat{Q}))$, since $(I - \alpha\widehat{X}^T\widehat{X}\widehat{D}(I-\gamma\widehat{Q}))y = y - \alpha\lambda y = (1-\lambda\alpha)y$, where $\lambda' = (1-\alpha\lambda)$ is an eigenvalue of $(I - \alpha\widehat{X}^T\widehat{X}\widehat{D}(I-\gamma\widehat{Q}))$. Second, we want to find suitable α such that the modulus of λ' is less than 1. We have the modulus of λ':

$$\begin{aligned}
\|\lambda'\| &= \|1-\alpha\lambda\| \\
&= \sqrt{(1-\alpha a)^2 + (-\alpha b^2)} && \text{substituting } \lambda = a + bi \text{ of general complex form} \\
&= \sqrt{1 - 2\alpha a + \alpha^2 a^2 + \alpha^2 b^2} \\
&= \sqrt{1 - 2\alpha a + \alpha^2(a^2 + b^2)} \\
&< \sqrt{1 - 2\alpha a + \alpha\frac{2a}{(a^2+b^2)}(a^2+b^2)} && \text{using } \alpha = \frac{2a}{(a^2+b^2)} \\
&= \sqrt{1 - 2\alpha a + 2\alpha a} = 1
\end{aligned}$$

From above, we can see that if α is chosen such that $0 < \alpha < \frac{2a}{a^2+b^2}$, then λ' will have modulus less than 1. Then using the theorem that states: if a matrix A has n independent eigenvectors with eigenvalues λ_i, then $A^k \to 0$ as $k \to \infty$ if and only if all $\|\lambda_i\| < 1$, which implies that $\lim_{n\to\infty}\left(I - \alpha\widehat{X}\widehat{D}(I-\widehat{Q})\widehat{X}^T\right)^n = 0$, taking the trailing element in Equation (16) to 0 for a suitable α. We thus prove convergence to the fixed point in Equation (3) if a batch linear TD(0) update is used with an appropriate step size α. □

D. Convergence of Batch Linear TD(0) to the MDP CE Fixed-Point

Theorem 2 (Batch Linear TD(0) Convergence)**.** *For any batch whose observation vectors $\{\boldsymbol{x}(s)|s \in \widehat{\mathcal{S}}\}$ are linearly independent, there exists an $\epsilon > 0$ such that, for all positive $\alpha < \epsilon$ and for any initial weight vector, the predictions for linear TD(0) converge under repeated presentations of the batch with weight updates after each complete presentation to the fixed-point (4).*

Proof. Batch linear TD(0) makes an update to weight vector, $\boldsymbol{w}_n$ (of dimension, length of the feature vector), after each presentation of the batch:

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n + \sum_{\tau \in \mathcal{D}} \sum_{t=1}^{L_\tau} \alpha \left[(\bar{r}_t + \gamma \boldsymbol{w}_n^T \boldsymbol{x}_{t+1}) - \boldsymbol{w}_n^T \boldsymbol{x}_t \right] \boldsymbol{x}_t$$

where $\mathcal{D}$ is the batch of episodes, L_τ is the length of each episode τ, and α is the learning rate.

We can re-write the whole presentation of the batch of data in terms of the number of times there was a transition from state i to state j when taking action a in the batch i.e. $\hat{c}_{ij}^a = \hat{d}_i \hat{\pi}_i^a \hat{p}_{ij}^a$, where $\hat{d}_i$ is the number of times state $i \in \widehat{\mathcal{N}}$ appears in the batch.

$$\begin{aligned}
\boldsymbol{w}_{n+1} &= \boldsymbol{w}_n + \sum_{\tau \in \mathcal{D}} \sum_{t=1}^{L_\tau} \alpha \left[(\bar{r}_t + \gamma \boldsymbol{w}_n^T \boldsymbol{x}_{t+1} - \boldsymbol{w}_n^T \boldsymbol{x}_t) \right] \boldsymbol{x}_t \\
&= \boldsymbol{w}_n + \alpha \sum_{i \in \widehat{\mathcal{N}}} \sum_{j \in \widehat{\mathcal{N}} \cup \widehat{\mathcal{T}}} \sum_{a \in \hat{\mathcal{A}}_i} \hat{c}_{ij}^a \left[(\bar{r}_{ij}^a + \gamma \boldsymbol{w}_n^T \boldsymbol{x}_j - \boldsymbol{w}_n^T \boldsymbol{x}_i) \right] \boldsymbol{x}_i \\
&= \boldsymbol{w}_n + \alpha \sum_{i \in \widehat{\mathcal{N}}} \sum_{j \in \widehat{\mathcal{N}} \cup \widehat{\mathcal{T}}} \sum_{a \in \hat{\mathcal{A}}_i} \hat{d}_i \hat{\pi}_i^a \hat{p}_{ij}^a \left[(\bar{r}_{ij}^a + \gamma \boldsymbol{w}_n^T \boldsymbol{x}_j - \boldsymbol{w}_n^T \boldsymbol{x}_i) \right] \boldsymbol{x}_i \\
&= \boldsymbol{w}_n + \alpha \sum_{i \in \widehat{\mathcal{N}}} \sum_{j \in \widehat{\mathcal{N}} \cup \widehat{\mathcal{T}}} \sum_{a \in \hat{\mathcal{A}}_i} \hat{d}_i \hat{p}_{ij}^a \hat{\pi}_i^a (\bar{r}_{ij}^a + \gamma \boldsymbol{w}_n^T \boldsymbol{x}_j) \boldsymbol{x}_i - \alpha \sum_{i \in \widehat{\mathcal{N}}} \sum_{j \in \widehat{\mathcal{N}} \cup \widehat{\mathcal{T}}} \sum_{a \in \hat{\mathcal{A}}_i} \hat{d}_i \hat{\pi}_i^a \hat{p}_{ij}^a (\boldsymbol{w}_n^T \boldsymbol{x}_i) \boldsymbol{x}_i \\
&= \boldsymbol{w}_n + \alpha \sum_{i \in \widehat{\mathcal{N}}} \hat{d}_i \boldsymbol{x}_i \sum_{j \in \widehat{\mathcal{N}} \cup \widehat{\mathcal{T}}} \sum_{a \in \hat{\mathcal{A}}_i} \hat{p}_{ij}^a \hat{\pi}_i^a (\bar{r}_{ij}^a + \gamma \boldsymbol{w}_n^T \boldsymbol{x}_j) - \alpha \sum_{i \in \widehat{\mathcal{N}}} \hat{d}_i (\boldsymbol{w}_n^T \boldsymbol{x}_i) \boldsymbol{x}_i \sum_{j \in \widehat{\mathcal{N}} \cup \widehat{\mathcal{T}}} \sum_{a \in \hat{\mathcal{A}}_i} \hat{\pi}_i^a \hat{p}_{ij}^a
\end{aligned}$$

$$= \boldsymbol{w}_n + \alpha \sum_{i \in \widehat{\mathcal{N}}} \hat{d}_i \boldsymbol{x}_i \left[\sum_{j \in \widehat{\mathcal{N}} \cup \widehat{\mathcal{T}}} \sum_{a \in \hat{\mathcal{A}}_i} \hat{p}_{ij}^a \hat{\pi}_i^a (\bar{r}_{ij}^a + \gamma \boldsymbol{w}_n^T \boldsymbol{x}_j) - \boldsymbol{w}_n^T \boldsymbol{x}_i \right] \qquad \sum_{j \in \widehat{\mathcal{N}} \cup \widehat{\mathcal{T}}} \sum_{a \in \hat{\mathcal{A}}_i} \hat{\pi}_i^a \hat{p}_{ij}^a = 1$$

$$= \boldsymbol{w}_n + \alpha \sum_{i \in \widehat{\mathcal{N}}} \hat{d}_i \boldsymbol{x}_i \left[\left(\sum_{j \in \widehat{\mathcal{N}}} \sum_{a \in \hat{\mathcal{A}}_i} \hat{p}_{ij}^a \hat{\pi}_i^a (\bar{r}_{ij}^a + \gamma \boldsymbol{w}_n^T \boldsymbol{x}_j) \right) + \left(\sum_{j \in \widehat{\mathcal{T}}} \sum_{a \in \hat{\mathcal{A}}_i} \hat{p}_{ij}^a \hat{\pi}_i^a \bar{r}_{ij}^a \right) - \boldsymbol{w}_n^T \boldsymbol{x}_i \right] \quad \text{If } \boldsymbol{x}_j \in \widehat{\mathcal{T}}, \boldsymbol{w}_n^T \boldsymbol{x}_j = 0$$

$$= \boldsymbol{w}_n + \alpha \sum_{i \in \widehat{\mathcal{N}}} \hat{d}_i \boldsymbol{x}_i \left[\left(\sum_{j \in \widehat{\mathcal{N}}} \sum_{a \in \hat{\mathcal{A}}_i} \hat{p}_{ij}^a \hat{\pi}_i^a \bar{r}_{ij}^a \right) + \left(\gamma \sum_{j \in \widehat{\mathcal{N}}} \sum_{a \in \hat{\mathcal{A}}_i} \hat{p}_{ij}^a \hat{\pi}_i^a \boldsymbol{w}_n^T \boldsymbol{x}_j \right) + \left(\sum_{j \in \widehat{\mathcal{T}}} \sum_{a \in \hat{\mathcal{A}}_i} \hat{p}_{ij}^a \hat{\pi}_i^a \bar{r}_{ij}^a \right) - \boldsymbol{w}_n^T \boldsymbol{x}_i \right]$$

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n + \alpha \widehat{X} \widehat{D} \left[\hat{\mathbf{m}} + \gamma \widehat{Q} \widehat{X}^T \boldsymbol{w}_n + \hat{\mathbf{h}} - \widehat{X}^T \boldsymbol{w}_n \right] \tag{17}$$

where $\widehat{X}$ denotes the matrix (of dimensions, length of the feature vector by $|\widehat{\mathcal{S}}|$) with columns, $\boldsymbol{x}_i \in \widehat{\mathcal{S}}$ and $\widehat{D}$ is a diagonal matrix (of dimensions, $|\widehat{\mathcal{S}}|$ by $|\widehat{\mathcal{S}}|$) with $\widehat{D}_{ii} = \hat{d}_i$.

Notice that Equation (17) is the same as Equation (15) since the considered MRP and MDP settings are equivalent. Due to this similarity, we omit the proof from here below as it is identical to the Theorem 1 proof.

□

E. Convergence of Batch Linear PSEC-TD(0) to the PSEC-MDP-CE Fixed-Point

E.1. PSEC Correction Applied to the New Estimate

We now show that batch linear PSEC-TD(0) converges to the policy corrected MDP-CE established in Equation (14), which is equivalent to Equation (5)

Theorem 3 (Batch Linear PSEC-TD(0) Convergence)**.** *For any batch whose observation vectors $\{\boldsymbol{x}(s)|s \in \widehat{\mathcal{S}}\}$ are linearly independent, there exists an $\epsilon > 0$ such that, for all positive $\alpha < \epsilon$ and for any initial weight vector, the predictions for linear PSEC-TD(0) converge under repeated presentations of the batch with weight updates after each complete presentation to the fixed-point (5).*

Proof. The proof for PSEC-TD(0) follows in large part the structure of the proof for TD(0). Below we highlight the salient points in the proof.

Batch linear PSEC-TD(0) makes an update to the weight vector, $\boldsymbol{w}_n$ (of dimension, length of the feature vector), after each presentation of the batch:

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n + \sum_{\tau \in \mathcal{D}} \sum_{t=1}^{L_\tau} \alpha \left[\hat{\rho}_t(\bar{r}_t + \gamma \boldsymbol{w}_n^T \boldsymbol{x}_{t+1}) - \boldsymbol{w}_n^T \boldsymbol{x}_t\right] \boldsymbol{x}_t$$

where $\mathcal{D}$ is the batch of episodes, L_τ is the length of each episode τ, $\hat{\rho}_t$ is the PSEC correction weight at time t for a given episode τ, and α is the learning rate.

We can re-write the whole presentation of the batch of data in terms of the number of times there was a transition from state i to state j when taking action a in the batch i.e. $\hat{c}_{ij}^a = \hat{d}_i \hat{\pi}_i^a \hat{p}_{ij}^a$, where $\hat{d}_i$ is the number of times state $i \in \widehat{\mathcal{N}}$ appears in the batch.

$$\begin{aligned}
\boldsymbol{w}_{n+1} &= \boldsymbol{w}_n + \sum_{\tau \in \mathcal{D}} \sum_{t=1}^{L_\tau} \alpha \left[\hat{\rho}_t(\bar{r}_t + \gamma \boldsymbol{w}_n^T \boldsymbol{x}_{t+1}) - \boldsymbol{w}_n^T \boldsymbol{x}_t\right] \boldsymbol{x}_t \\
&= \boldsymbol{w}_n + \alpha \sum_{i \in \widehat{\mathcal{N}}} \sum_{j \in \widehat{\mathcal{N}} \cup \widehat{\mathcal{T}}} \sum_{a \in \hat{\mathcal{A}}_i} \hat{c}_{ij}^a \left[\hat{\rho}_i^a(\bar{r}_{ij}^a + \gamma \boldsymbol{w}_n^T \boldsymbol{x}_j) - \boldsymbol{w}_n^T \boldsymbol{x}_i\right] \boldsymbol{x}_i \\
&= \boldsymbol{w}_n + \alpha \sum_{i \in \widehat{\mathcal{N}}} \sum_{j \in \widehat{\mathcal{N}} \cup \widehat{\mathcal{T}}} \sum_{a \in \hat{\mathcal{A}}_i} \hat{d}_i \hat{\pi}_i^a \hat{p}_{ij}^a \left[\hat{\rho}_i^a(\bar{r}_{ij}^a + \gamma \boldsymbol{w}_n^T \boldsymbol{x}_j) - \boldsymbol{w}_n^T \boldsymbol{x}_i\right] \boldsymbol{x}_i \\
&= \boldsymbol{w}_n + \alpha \sum_{i \in \widehat{\mathcal{N}}} \sum_{j \in \widehat{\mathcal{N}} \cup \widehat{\mathcal{T}}} \sum_{a \in \hat{\mathcal{A}}_i} \hat{d}_i \hat{\pi}_i^a \hat{p}_{ij}^a \left[\frac{\pi_i^a}{\hat{\pi}_i^a}(\bar{r}_{ij}^a + \gamma \boldsymbol{w}_n^T \boldsymbol{x}_j) - \boldsymbol{w}_n^T \boldsymbol{x}_i\right] \boldsymbol{x}_i \\
&= \boldsymbol{w}_n + \alpha \sum_{i \in \widehat{\mathcal{N}}} \sum_{j \in \widehat{\mathcal{N}} \cup \widehat{\mathcal{T}}} \sum_{a \in \hat{\mathcal{A}}_i} \hat{d}_i \hat{p}_{ij}^a \pi_i^a (\bar{r}_{ij}^a + \gamma \boldsymbol{w}_n^T \boldsymbol{x}_j) \boldsymbol{x}_i - \alpha \sum_{i \in \widehat{\mathcal{N}}} \sum_{j \in \widehat{\mathcal{N}} \cup \widehat{\mathcal{T}}} \sum_{a \in \hat{\mathcal{A}}_i} \hat{d}_i \hat{\pi}_i^a \hat{p}_{ij}^a (\boldsymbol{w}_n^T \boldsymbol{x}_i) \boldsymbol{x}_i \\
&= \boldsymbol{w}_n + \alpha \sum_{i \in \widehat{\mathcal{N}}} \hat{d}_i \boldsymbol{x}_i \sum_{j \in \widehat{\mathcal{N}} \cup \widehat{\mathcal{T}}} \sum_{a \in \hat{\mathcal{A}}_i} \hat{p}_{ij}^a \pi_i^a (\bar{r}_{ij}^a + \gamma \boldsymbol{w}_n^T \boldsymbol{x}_j) - \alpha \sum_{i \in \widehat{\mathcal{N}}} \hat{d}_i (\boldsymbol{w}_n^T \boldsymbol{x}_i) \boldsymbol{x}_i \sum_{j \in \widehat{\mathcal{N}} \cup \widehat{\mathcal{T}}} \sum_{a \in \hat{\mathcal{A}}_i} \hat{\pi}_i^a \hat{p}_{ij}^a
\end{aligned}$$

$$= \boldsymbol{w}_n + \alpha \sum_{i \in \widehat{\mathcal{N}}} \hat{d}_i \boldsymbol{x}_i \left[\sum_{j \in \widehat{\mathcal{N}} \cup \widehat{\mathcal{T}}} \sum_{a \in \hat{\mathcal{A}}_i} \hat{p}_{ij}^a \pi_i^a (\bar{r}_{ij}^a + \gamma \boldsymbol{w}_n^T \boldsymbol{x}_j) - \boldsymbol{w}_n^T \boldsymbol{x}_i\right] \qquad \sum_{j \in \widehat{\mathcal{N}} \cup \widehat{\mathcal{T}}} \sum_{a \in \hat{\mathcal{A}}_i} \hat{\pi}_i^a \hat{p}_{ij}^a = 1$$

$$= \boldsymbol{w}_n + \alpha \sum_{i \in \widehat{\mathcal{N}}} \hat{d}_i \boldsymbol{x}_i \left[\left(\sum_{j \in \widehat{\mathcal{N}}} \sum_{a \in \hat{\mathcal{A}}_i} \hat{p}_{ij}^a \pi_i^a (\bar{r}_{ij}^a + \gamma \boldsymbol{w}_n^T \boldsymbol{x}_j)\right) + \left(\sum_{j \in \widehat{\mathcal{T}}} \sum_{a \in \hat{\mathcal{A}}_i} \hat{p}_{ij}^a \pi_i^a \bar{r}_{ij}^a\right) - \boldsymbol{w}_n^T \boldsymbol{x}_i\right] \qquad \text{If } \boldsymbol{x}_j \in \widehat{\mathcal{T}}, \boldsymbol{w}_n^T \boldsymbol{x}_j = 0$$

$$\begin{aligned}
&= \boldsymbol{w}_n + \alpha \sum_{i \in \widehat{\mathcal{N}}} \hat{d}_i \boldsymbol{x}_i \left[\left(\sum_{j \in \widehat{\mathcal{N}}} \sum_{a \in \hat{\mathcal{A}}_i} \hat{p}_{ij}^a \pi_i^a \bar{r}_{ij}^a\right) + \left(\gamma \sum_{j \in \widehat{\mathcal{N}}} \sum_{a \in \hat{\mathcal{A}}_i} \hat{p}_{ij}^a \pi_i^a \boldsymbol{w}_n^T \boldsymbol{x}_j\right) + \left(\sum_{j \in \widehat{\mathcal{T}}} \sum_{a \in \hat{\mathcal{A}}_i} \hat{p}_{ij}^a \pi_i^a \bar{r}_{ij}^a\right) - \boldsymbol{w}_n^T \boldsymbol{x}_i\right] \\
&= \boldsymbol{w}_n + \alpha \widehat{X} \widehat{D} \left[\hat{\mathbf{o}} + \gamma \widehat{U} \widehat{X}^T \boldsymbol{w}_n + \hat{\mathbf{l}} - \widehat{X}^T \boldsymbol{w}_n\right]
\end{aligned}$$

where $\widehat{X}$ denotes the matrix (of dimensions, length of the feature vector by $|\widehat{\mathcal{S}}|$) with columns, $\boldsymbol{x}_i \in \widehat{\mathcal{S}}$ and $\widehat{D}$ is a diagonal matrix (of dimensions, $|\widehat{\mathcal{S}}|$ by $|\widehat{\mathcal{S}}|$) with $\widehat{D}_{ii} = \hat{d}_i$.

Assuming that as $n \to \infty$, $(I - \alpha \widehat{X}^T \widehat{X} \widehat{D}(I - \gamma \widehat{U}))^n \to 0$, we can drop the second term and the sequence $\{\widehat{X}^T \boldsymbol{w}_n\}$ converges to:

$$\begin{aligned}
\lim_{n\to\infty} \widehat{X}^T \boldsymbol{w}_n &= \left(I - (I - \alpha \widehat{X}^T \widehat{X} \widehat{D}(I - \gamma \widehat{U}))\right)^{-1} (\alpha \widehat{X}^T \widehat{X} \widehat{D}(\hat{\mathbf{o}} + \hat{\mathbf{l}})) \\
&= (I - \gamma \widehat{U})^{-1} \widehat{D}^{-1} (\widehat{X}^T \widehat{X})^{-1} \alpha^{-1} \alpha \widehat{X}^T \widehat{X} \widehat{D}(\hat{\mathbf{o}} + \hat{\mathbf{l}}) \\
&= (I - \gamma \widehat{U})^{-1} (\hat{\mathbf{o}} + \hat{\mathbf{l}}) \\
\lim_{n\to\infty} \mathbb{E}\left[\boldsymbol{x}_i^T \boldsymbol{w}_n\right] &= \left[(I - \gamma \widehat{U})^{-1} (\hat{\mathbf{o}} + \hat{\mathbf{l}})\right]_i, \forall i \in \hat{\mathcal{N}}
\end{aligned}$$

What is left to show now is that as $n \to \infty$, $(I - \alpha \widehat{X}^T \widehat{X} \widehat{D}(I - \gamma \widehat{U}))^n \to 0$, which we can show by following the steps shown for Equation (16). Thus we prove convergence to the fixed-point (5).

$\square$

E.2. Convergence to the MDP True Fixed-Point with Infinite Data

With batch linear PSEC-TD(0) we have corrected for the policy sampling error in batch linear TD(0). The remaining inaccuracy of the policy sampling corrected certainty-equivalence fixed-point is due to the transition dynamics sampling error. In a model-free setting, however, we cannot correct for this error in the same way we corrected the policy sampling error.

We argue that as the batch size approaches infinite, the maximum-likelihood estimate of the transition dynamics will approach the true transition dynamics i.e. $\hat{p} \to p$. It then follows that in expectation, the true value function will be reached. Thus, the batch linear PSEC-TD(0) with an infinite batch size will correctly converge to the true value function fixed-point given by Equation (12).

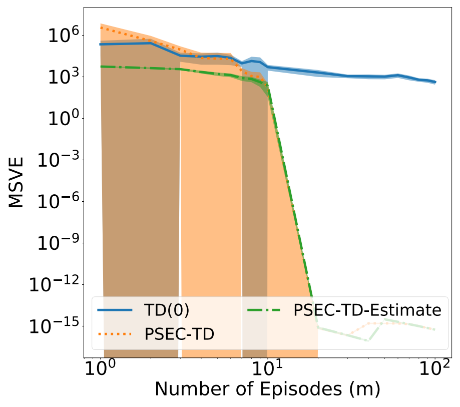
F. Additional Empirical Results

In this section, we include additional results that were omitted in the main text due to space constraints.

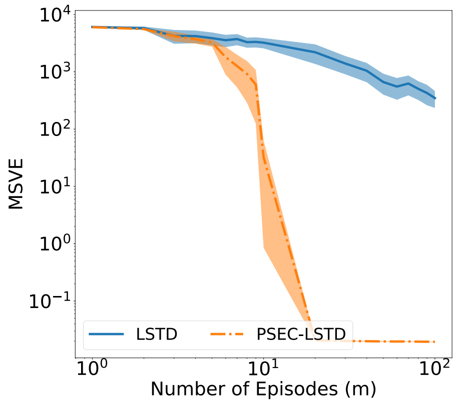
F.1. Tabular Setting: Discrete States and Actions

F.1.1. Off-policy Results

For off-policy TD(0), we always use the variant that applies the importance weight to the TD-error.



(a) Off-Policy TD



(b) Off-Policy LSTD

Figure 5. Deterministic Gridworld experiments. Both axes are log-scaled, leading to the asymmetric confidence intervals. Errors are computed over 50 trials with 95% confidence intervals. Figure 5(a) and Figure 5(b) compare the final errors achieved by variants of PSEC-TD(0) and TD(0), and PSEC-LSTD(0) and LSTD(0) respectively for varying batch size in the off-policy case.

For off-policy TD(0) we only consider the PSEC-TD variant as we found multiplying the new estimate by the weight was divergent. All methods show higher variance for the off-policy setting, however, PSEC-TD(0) variants still provide more accurate value function estimates.

F.1.2. Effect of Environment Stochasticity

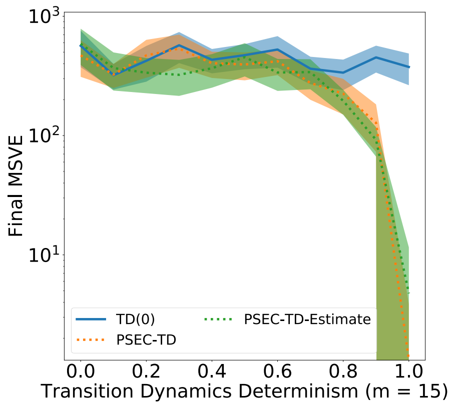


Figure 6. Additional Gridworld experiments. Errors are computed over 50 trials with 95% confidence intervals. Figure 6 is a y-axis log scaled graph that shows the final error (averaged over 100 trials) achieved by the two variants of PSEC-TD(0) and TD(0) for a given batch size (15 episodes) with varying levels of determinism of the transition dynamics.

According to Theorem 2, TD suffers from policy *and* transition dynamics sampling error. We study this observation through Figure 6, which illustrates how the performance of PSEC changes with different levels of transition dynamics determinism for a fixed batch size. In Gridworld, the determinism is varied according to a parameter, p, where the environment becomes purely deterministic or stochastic as $p \to 1$ or $p \to 0$ respectively. From Theorem 3 we expect PSEC to fully correct for the policy sampling error but not transition dynamics sampling error. Figure 6 confirms that PSEC is achieves a lower final MSVE than TD as $p \to 1$. As $p \to 0$, the transition dynamics become the dominant source of sampling error and PSEC-TD(0) and TD(0) perform similarly.

F.2. Function Approximation: Continuous States and Discrete Actions (CartPole)

In each experiment below, unless stated, the following components were fixed: a batch size of 10 episodes, the value function was represented with a neural network of single hidden layer of 512 neurons using tanh activation, the gradients were normalized to unit norm before the gradient descent step was performed, we used a learning rate of 1.0 and decayed the learning rate by 10% every 50 presentations of the batch to the algorithm. The true MSVE was computed by 200 Monte Carlo rollouts for 150 sampled states.

F.2.1. Data Efficiency

From Figure 3(a). Both algorithms used a learning rate of 1.0 and decayed the learning rate by 5% every 10 presentations of the batch. The PSEC model architecture was a neural network with 3 hidden layers with 16 neurons each. Batch sizes of 10, 50, and 500 episodes used a learning rate of 0.0125, batch size of 100 used 0.003125, and batch size of 1000 used 0.001563.

F.2.2. Effect of Value Function Model Architecture

From Figure 4(a). PSEC used a model architecture of 3 hidden layers with 16 neurons each and tanh activation, and learning rate of 0.025.

F.2.3. Effect of PSEC Learning Rate

Figure 7 compares the data efficiency of PSEC-TD vs TD for varying learning rates of the PSEC policy, while holding the value function, PSEC model, and behavior policy architectures fixed, on CartPole. Since TD does not use PSEC, its error

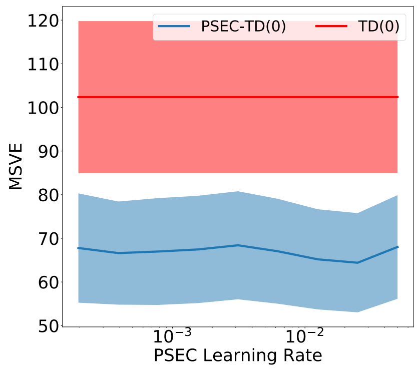


Figure 7. Compares data efficiency of PSEC with varying learning rates against TD respectively on CartPole. Results use a batch size of 10 episodes, and results are averaged over 300 trials with 95% confidence error bars.

for a given batch size is independent of the PSEC learning rate. From above, PSEC appears to be relatively stable in its improvement over TD regardless of the learning rate used. The PSEC policy used in this experiment was a neural network with: 3 hidden layers with 16 neurons each and tanh activation.

F.2.4. Effect of PSEC Model Architecture

From Figure 4(b). PSEC used a learning rate of 0.025. The chosen PSEC neural network models are with respect to the behavior policy described earlier, a 2 hidden layered with 16 neurons architecture.

F.2.5. Varying PSEC Training Style

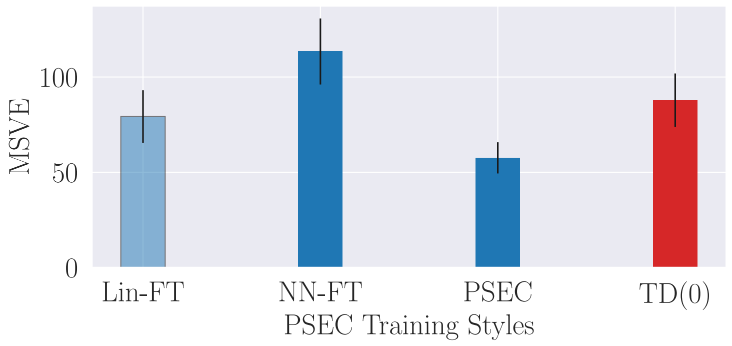


Figure 8. Comparing data efficiency of varying training styles of PSEC against TD for a fixed batch size of 10 episodes. Results shown are averaged over 300 trials and shaded region is 95% confidence. Darker shades represent statistically significant result.

Figure 8 illustrates the data efficiency of three variants of PSEC, while holding the value-function PSEC model, and behavior policy architectures fixed. All three variants use the same PSEC model architecture as that of the behavior policy, and each used a learning rate of 0.025 with tanh activation. The three variants are as follows: 1) Lin-FT is when PSEC initializes the PSEC model to the weights of the behavior policy and trains on the batch of data by finetuning only the last linear layer, 2) NN-FT is when PSEC initializes the PSEC model to the weights of the behavior policy but finetunes the all the weights of the network, and 3) PSEC uses the same training style in the previous experiments, where the model is initialized randomly and all the weights are tuned. We found that Lin-FT performed similarly to TD with a statistically insignificant improvement over TD; we believe this may be so since Lin-FT is initialized to the behavior policy and since there are only few weights to change in the linear layer, the newly learned Lin-FT is still similar to the behavior policy, which would produce PSEC corrections close to 1 (equivalent to TD). Interestingly, tuning all the weights of the neural network did better when the model was initialized randomly versus when it was initialized to the behavior policy.

F.2.6. Effect of Underfitting and Overfitting during PSEC Policy Training

This experiment attempts to give an understanding of how the MSVE achieved by each PSEC variant is dependent on the number of epochs the PSEC model was trained for, while holding the value-function, PSEC and behavior policy architectures fixed. We conduct the experiment as follows: the PSEC algorithm performs 10 gradient descent steps (epochs) on the full batch of data, after which the resulting training and validation mean cross-entropy losses are plotted along with the MSVE achieved by that trained PSEC policy. For example, after 10 epochs, the training and validation loss of the PSEC model was nearly 0.5 and the model achieved an MSVE of nearly 150.

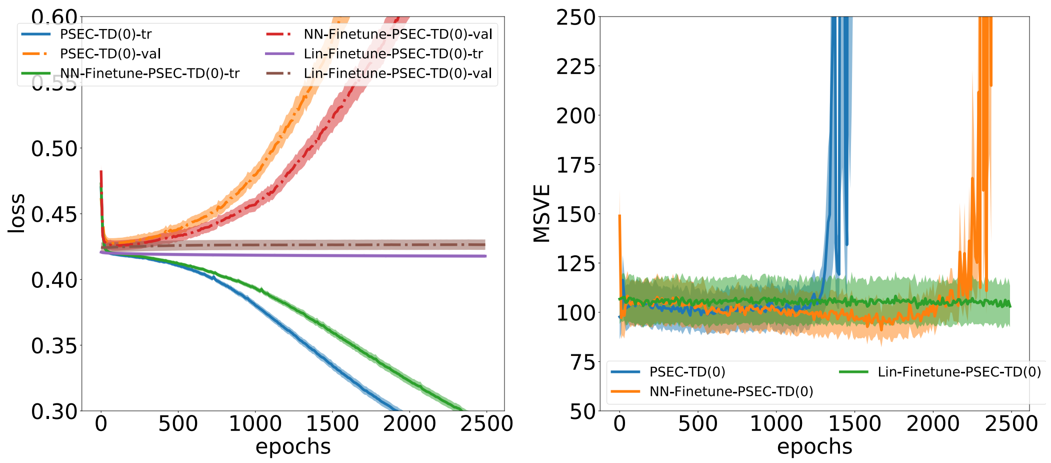


Figure 9. Comparing MSVE achieved and cross entropy loss (training (tr) and validation (val)) by variants of PSEC after each epoch of training for a fixed batch size of 10 episodes. Results shown are averaged over 50 trials and shaded region is 95% confidence.

Since computing the MSVE can be computationally expensive, as it requires processing the batch until the value function converges, we change the learning rate decay schedule to starting with a learning rate of 1.0 but decaying learning rate by 50% every 50 presentations of the full batch to the algorithm (this change is also the reason why these results may be different from the ones shown earlier). All PSEC variants used a learning rate of 0.025 and PSEC model architecture of 2 hidden layers with 16 neurons each and tanh activation.

Figure 9 suggests that performance of PSEC, regardless of the variant, depends on the number of epochs it was trained for. Naturally, we do want to fit sufficiently well to the data, and the graph suggests that some overfitting is tolerable. However, if overfitting becomes extreme, PSEC's performance suffers, resulting in MSVE nearly 1000 times larger than the minimum error achieved (not shown for clarity). From the graph, we can see that the PSEC variant, which is initialized randomly, starts to extremely overfit before the NN-finetune variant does, causing its MSVE to degrade before that of NN-finetune variant. We also see that the Lin-finetune variant is not able to overfit since the last linear layer may not be expressible enough to overfit, causing it to have a relatively stable MSVE across all epochs.

F.2.7. Effect of Behavior Policy Distribution

So far, PSEC has used a function approximator of the similar function class as that of the behavior policy i.e. both the PSEC models and the behavior policy were neural networks of similar architectures. In this experiment, we evaluate the performance of PSEC with a neural network policy when the behavior policy that models a discontinuous function generates a larger batch size of 100 episodes, while the value function and PSEC model architectures are fixed. In particular, we use a behavior policy in CartPole that does the following: if the sign of the pole angle is negative, move left with probability 0.75 and right with probability 0.25, and if the sign of the pole angle is positive, move right with probability 0.75 and left with probability 0.25. The PSEC policy is a neural network with 3 hidden layers with 16 neurons each with tanh activation.

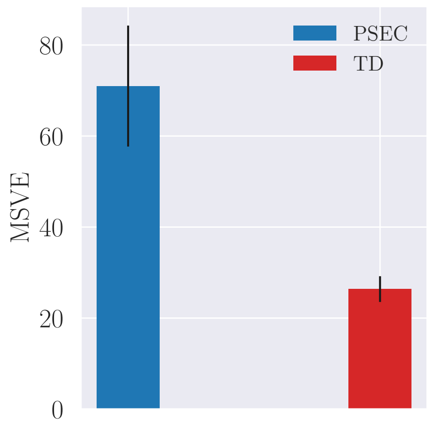


Figure 10. Comparing data efficiency of PSEC against TD for a fixed batch size of 100 episodes when the behavior policy models a discontinuous function. Results shown are averaged over 30 trials and shaded region is 95% confidence.

Figure 10 shows that PSEC performs much worse than TD when the behavior policy is the discontinuous type function described above. We reason that the neural network finds it difficult to compute the MLE of the data since this discontinuous distribution is “hard" to model; therefore, producing incorrect PSEC weights, which degrade its performance. While we can largely ignore the distribution of the behavior policy, this experiment shows that PSEC may suffer in situations like the one described.

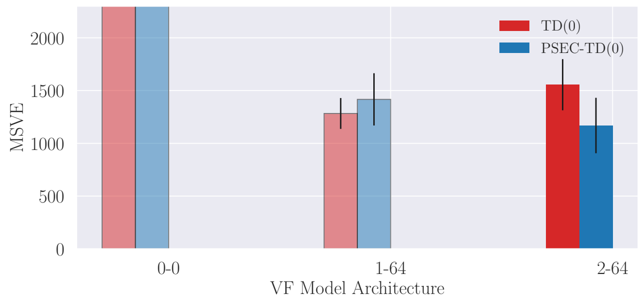
F.3. Function Approximation: Continuous States and Actions (InvertedPendulum)

In each experiment below, unless stated, the following components were fixed: a batch size of 20 episodes, the value function was represented with a neural network of 2 hidden layer with 64 neurons each using tanh activation, the gradients were normalized to unit norm before the gradient descent step was performed, we used a learning rate of 1.0 and decayed the learning rate by 5% every 10 presentations of the batch to the algorithm. The true MSVE was computed by 100 Monte Carlo rollouts for 100 sampled states.

F.3.1. Data Efficiency

From Figure 3(b). The PSEC model architecture was a neural network with 2 hidden layers with 64 neurons each. Batch sizes $10, 50, 100, 500, 1000$ used a learning rate of 0.000781. Batch size of 10 used an L2 weight penalization of 0.02. All used a value function model architecture of 3 hidden layers with 64 neurons each.

F.3.2. Effect of Value Function Model Architecture

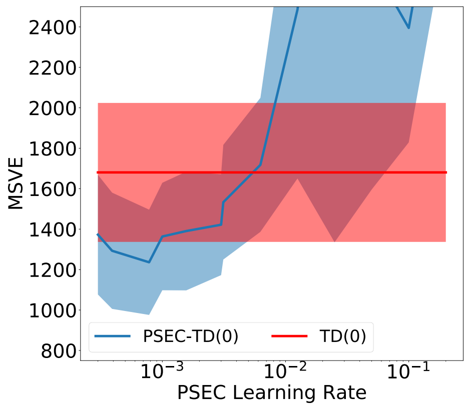


(a) InvertedPendulum

Figure 11. Comparing data efficiency of PSEC with varying VF model architectures against TD. Figure 11(a) use batch size of 20 episodes , and results shown are averaged over 350 trials respectively with error bars of 95% confidence. Darker shades represent statistically significant result. The label on the x axis shown is (# hidden layers - # neurons).

From Figure 11(a). The PSEC policy is 2 hidden layers with 64 neurons each and used a learning rate of 0.000781.

F.3.3. Effect of PSEC Learning Rate



(a)

Figure 12. Comparing data efficiency of PSEC with varying learning rates against TD for a fixed batch size of 20 episodes. Results shown are averaged over 200 trials and error bar is 95% confidence.

Figure 12 compares the data efficiency of PSEC-TD to TD with varying learning rates for PSEC, while holding the PSEC and value function architecture fixed. Unlike earlier, the PSEC learning rate heavily influences the learned value function in the continuous state and action setting. In general, PSEC performance heavily degraded when the learning rate increased (y-axis limited for clarity). Among the tested learning rates, 0.000781 was the optimal, giving a statistically significant result.

F.3.4. Effect of PSEC Model Architecture

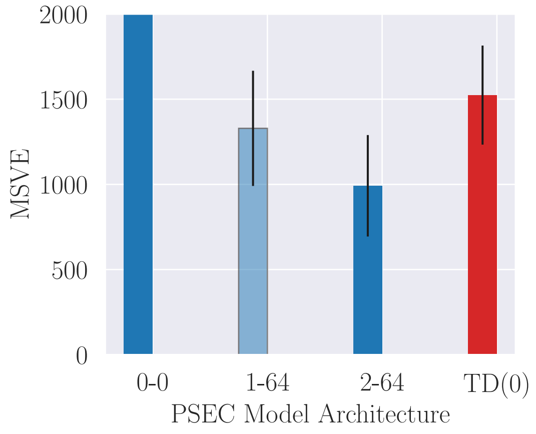


Figure 13. Comparing data efficiency of PSEC with varying model architectures against TD for a fixed batch size of 20 episodes. Results shown are averaged over 200 trials and error bar is 95% confidence. Darker shades represent statistically significant result. The label on the x axis shown is (# hidden layers - # neurons). The value function represented by 0-0 is a linear mapping with no activation function.

Figure 13 compares the data efficiency of PSEC-TD with TD with varying PSEC model architectures, while holding the value-function and behavior policy architectures fixed. All the shown PSEC architectures used a learning rate of learning rate 0.000781. Similar to our earlier findings, a more expressive network was able to better model the batch of data and produce a statistically significant improvement over TD. Less expressive PSEC models performed worse than TD, and any improvement was statistically insignificant. Note that the linear architecture used produced an MSVE of ~ 5800 (not shown for clarity) and its poor data efficiency with respect to TD(0) was statistically significant.

F.3.5. Effect of Underfitting and Overfitting during PSEC Policy Training

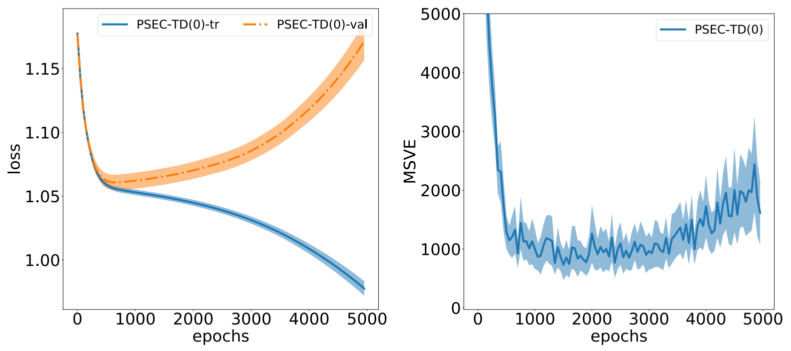


Figure 14. Comparing MSVE achieved, and training (tr) and validation (val) loss by PSEC after each epoch of training for a fixed batch size of 20 episodes on InvertedPendulum. Results shown are averaged over 100 trials and shaded region is 95% confidence.

This experiment attempts to give an understanding of how the MSVE achieved by each PSEC variant is dependent on the number of epochs the PSEC model was trained for, while holding the value-function, PSEC and behavior policy architectures fixed on InvertedPendulum. The experiment is conducted in a similar manner as before except we perform 50 gradient descent steps (epochs) before plotting the training and validation loss, and MSVE achieved by PSEC after the gradient steps. The loss shown here is a regression loss detailed in Appendix G.3. Figure 14 suggests that performance of PSEC depends on the number of epochs it was trained for. Naturally, we do want to fit sufficiently well to the data, and the graph suggests that some overfitting is tolerable. However, if overfitting becomes extreme, PSEC's performance suffers. If some overfitting is desirable, then early stopping is not the preferred principled approach to terminate PSEC model training. When computing MSVE, we used the learning rate schedule specified at the beginning of this section. PSEC used a learning rate of 0.000781 and model architecture of 2 hidden layers with 64 neurons each and tanh activation.

G. Extended Empirical Description

In this appendix we provide additional details for our empirical evaluation.

G.1. Gridworld

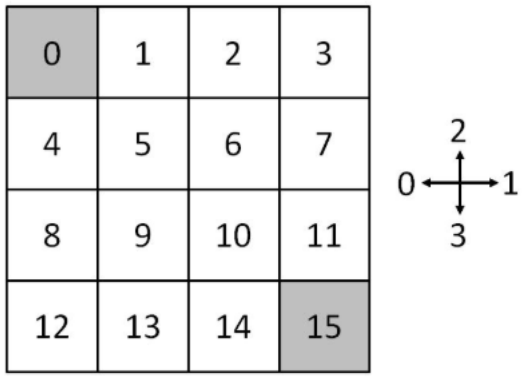


Figure 15. The Gridworld environment. Start at top left, bottom right is terminal state, discrete action space consists of the cardinal directions, and discrete state space is the location in the grid. This specific image was taken from this link.

This domain is a 4×4 grid, where an agent starts at $(0, 0)$ and tries to navigate to $(3, 3)$. The states are the discrete positions in the grid and actions are the 4 cardinal directions. The reward function is 100 for reaching $(3, 3)$, -10 for reaching $(1, 1)$, 1 for reaching $(1, 3)$, and -1 for reaching all other states. If an agent takes an action that hits a wall, the agent stays in the same location. The transition dynamics are controlled by a parameter, p, where with probability p, an agent takes the intended action, else it takes an adjacent action with probability $(1 - p)/2$. All policies use a softmax action selection distribution with value θ_{sa}, for each state, s, and action a. The probability of taking action a in state s is given by:

$$\pi(a|s) = \frac{e^{\theta_{sa}}}{\sum_{a' \in \mathcal{A}} e^{\theta_{sa'}}}$$

In the on-policy experiments, the evaluation and behavior policies were equiprobable policies in each cardinal direction. In the off-policy experiments, the evaluation policy was such that each θ was generated from a standard normal distribution and behavior policy was the equiprobable policy.

For the comparisons of batch linear PSEC-TD(0) and TD(0), we conducted a parameter sweep of the learning rates for the varying batch sizes. The parameter sweep was over: $\{5e^{-3}, 1e^{-3}, 5e^{-2}, 1e^{-2}, 5e^{-1}\}$. We used a value function convergence threshold of $1e^{-10}$. For PSEC-LSTD and LSTD, we stabilized the matrix, A, before inverting it by adding ϵI to the computed A. We conducted a parameter sweep over the following: $\epsilon \in \{1e^{-6}, 1e^{-5}, 1e^{-4}, 1e^{-3}, 1e^{-2}, 1e^{-1}\}$.

G.2. CartPole

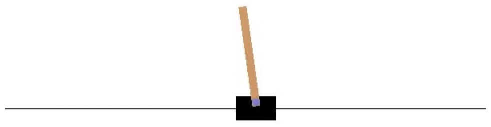


Figure 16. CartPole-v0 from OpenAI Gym (Brockman et al., 2016)

In this domain, the goal of the agent is to balance a pole for as long as possible. We trained our behavior policy using REINFORCE (Williams, 1992) with the Adam optimizer (Kingma & Ba, 2014) with learning rate $3e^{-4}$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. The behavior policy mapped raw state features to a softmax distribution over actions. The policy was a neural network with 2 hidden layers with 16 neurons each, and used the tanh activation function and was initialized with Xavier initialization (Glorot & Bengio, 2010).

The value function used by all algorithms was initialized by Xavier initialization and used the tanh activation function, and was trained using semi-gradient TD (Sutton & Barto, 2018). We used a convergence threshold of 0.1.

The PSEC policy was initialized by Xavier initialization and used the tanh activation function. PSEC-TD sweeped over the following learning rates $\alpha \in \{0.1 \times 2.0^j | j = -7, -6, ...1, 2\}$. It used a validation set of 10% the size of the batch size. It used an L2 regularization of $2e^{-2}$. More details can be found in Section 5 and Appendix F.2.

G.3. InvertedPendulum

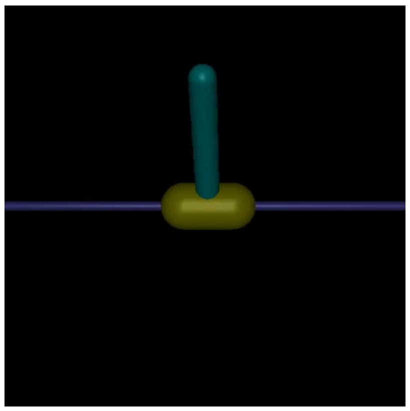


Figure 17. InvertedPendulum-v2 from OpenAI Gym and MuJoCo (Brockman et al., 2016; Todorov et al., 2012)

In this domain, the goal of the agent is to balance a pole for as long as possible. We trained our behavior policy using PPO (Schulman et al., 2017) with the default settings found on Gym (Brockman et al., 2016). The policy was a neural network with 2 hidden layers with 64 neurons each, and used the tanh activation function and was initialized with Xavier initialization (Glorot & Bengio, 2010). It mapped state features to an output vector that represented the mean vector of a Gaussian distribution. This mapping along with a separate parameter set representing the log standard deviation of each element in the output vector, make up the policy. The policy was trained by minimizing the following loss function:

$$\mathcal{L} = \sum_{i=1}^{m} 0.5((a_i - \mu(s_i))/e^{\sigma})^2 + \sigma$$

where m are the number of state-action training examples, a_i is the action vector of the i^{th} example, $\mu(s_i)$ is the mean vector outputted by the neural network of the Gaussian distribution for state s_i, and σ is the the seperate parameter representing the log standard deviation of each element in the output vector, $\mu(s_i)$.

The value function used by all algorithms was initialized by Xavier initialization and used the tanh activation function, and was trained using semi-gradient TD (Sutton & Barto, 2018). We used a convergence threshold of 0.1.

The PSEC policy was initialized by Xavier initialization and used the tanh activation function. PSEC-TD swept over the following learning rates $\alpha \in \{0.1 \times 2.0^j | j = -8, -6, ..., 1, 2\}$. It used a validation set of 20% the size of the batch size. More details can be found in Section 5 and Appendix F.2.