Learning MDPs from Features: Predict-Then-Optimize for Sequential Decision Problems by Reinforcement Learning

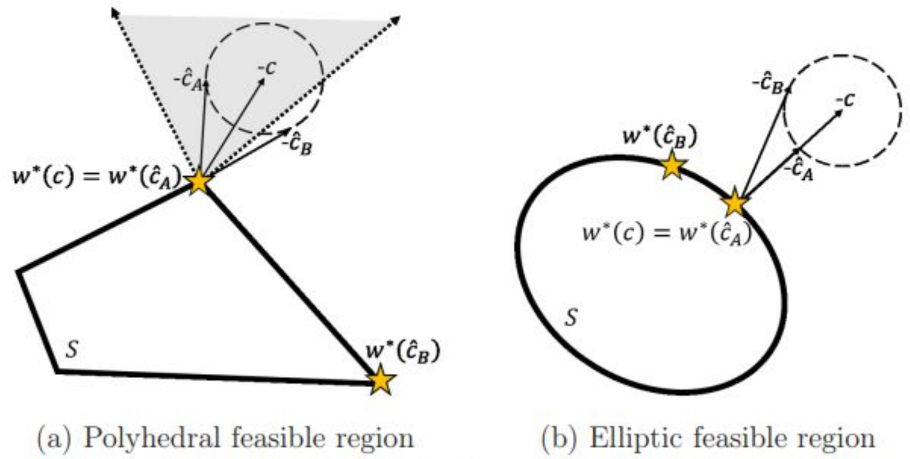
Summary

- Paper from Neurips 2021
- Want to obtain an ‘optimal’ policy in an environment with ‘unknown MDP parameters’
- Use the “Predict-then-Optimize” framework and reinforcement learning to solve sequential decision problem.
    - Achieve ‘Decision-Focused Learning’ by backpropagation of policy evaluation (‘through MDPs’) all the way back to the feature learning.

Summary

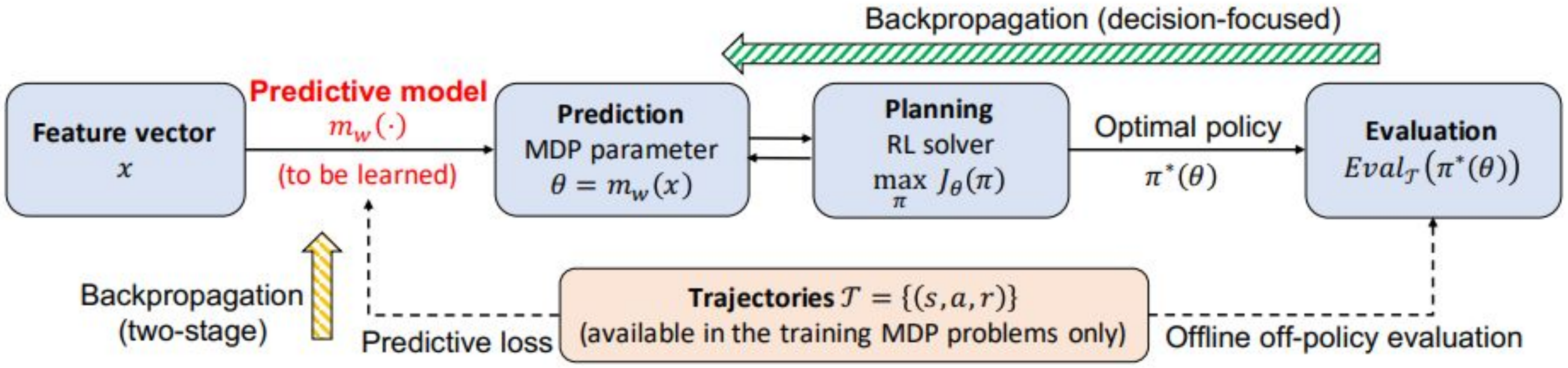
- Computational challenges
    - Large state and action spaces make it infeasible for existing techniques to differentiate through MDP problems
        - Sample Unbiased Derivatives
    - The high-dimensional policy space, as parameterized by a neural network, makes differentiating through a policy expensive
        - Use low-rank approximation (for Hessian matrix)

Figure 1 Geometric Illustration of SPO Loss



$-\hat{c}_A$
$-c$
$-\hat{c}_B$
$w^*(c) = w^*(\hat{c}_A)$
S
$w^*(\hat{c}_B)$
(a) Polyhedral feasible region
$-\hat{c}_B$
$-c$
$w^*(\hat{c}_B)$
$-\hat{c}_A$
$w^*(c) = w^*(\hat{c}_A)$
S
(b) Elliptic feasible region

Note. In these two figures, we consider a two-dimensional polyhedron and ellipse for the feasible region S. We plot the (negative) of the true cost vector c, as well as two candidate predictions $\hat{c}_A$ and $\hat{c}_B$ that are equidistant from c and thus have equivalent LS loss. One can see that the optimal decision for $\hat{c}_A$ coincides with that of c, since $w^*(\hat{c}_A) = w^*(c)$. In the polyhedron example, any predicted cost vector whose negative is not in the gray region will result in a wrong decision, where as in the ellipse example any predicted cost vector that is not exactly parallel with c results in a wrong decision.

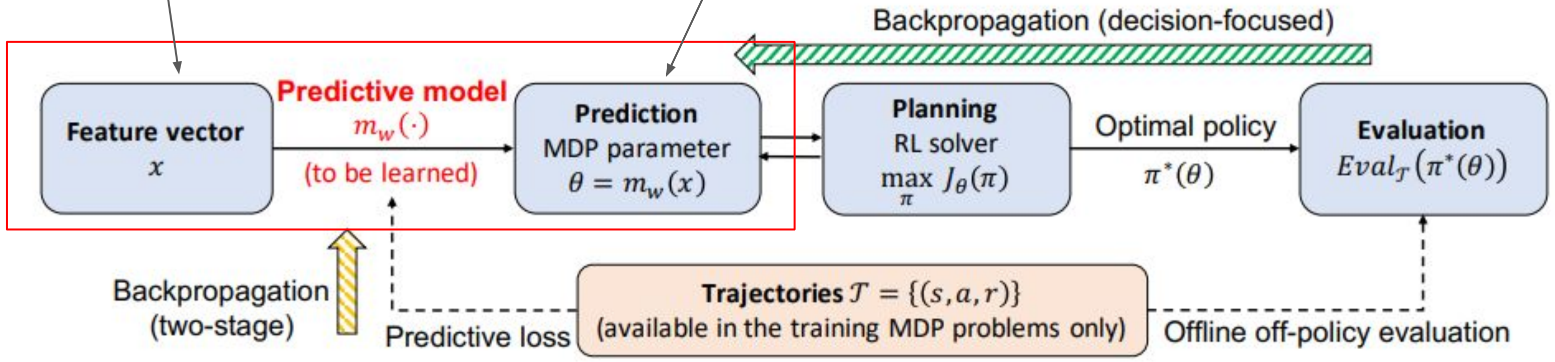

Backpropagation (decision-focused)
Feature vector x
Predictive model
$m_w(\cdot)$
(to be learned)
Prediction
MDP parameter
$\theta = m_w(x)$
Planning
RL solver
$\max_\pi J_\theta(\pi)$
Optimal policy
$\pi^*(\theta)$
Evaluation
$Eval_{\mathcal{T}}(\pi^*(\theta))$
Backpropagation
(two-stage)
Predictive loss
Trajectories $\mathcal{T} = \{(s,a,r)\}$
(available in the training MDP problems only)
Offline off-policy evaluation

Note this does not take into account the decision process

Feed features, training MDPs (set of trajectories under an unknown behavior policy)

Predict MDP unknown parameters using neural networks (supervised learning)

E.g. transition probabilities, rewards, etc


Backpropagation (decision-focused)
Feature vector
x
Predictive model
$m_w(\cdot)$
(to be learned)
Prediction
MDP parameter
$\theta = m_w(x)$
Planning
RL solver
$\max_{\pi} J_\theta(\pi)$
Optimal policy
$\pi^*(\theta)$
Evaluation
$Eval_{\mathcal{T}}(\pi^*(\theta))$
Backpropagation
(two-stage)
Predictive loss
Trajectories $\mathcal{T} = \{(s,a,r)\}$
(available in the training MDP problems only)
Offline off-policy evaluation

Note theta is the vector of missing MDP parameters

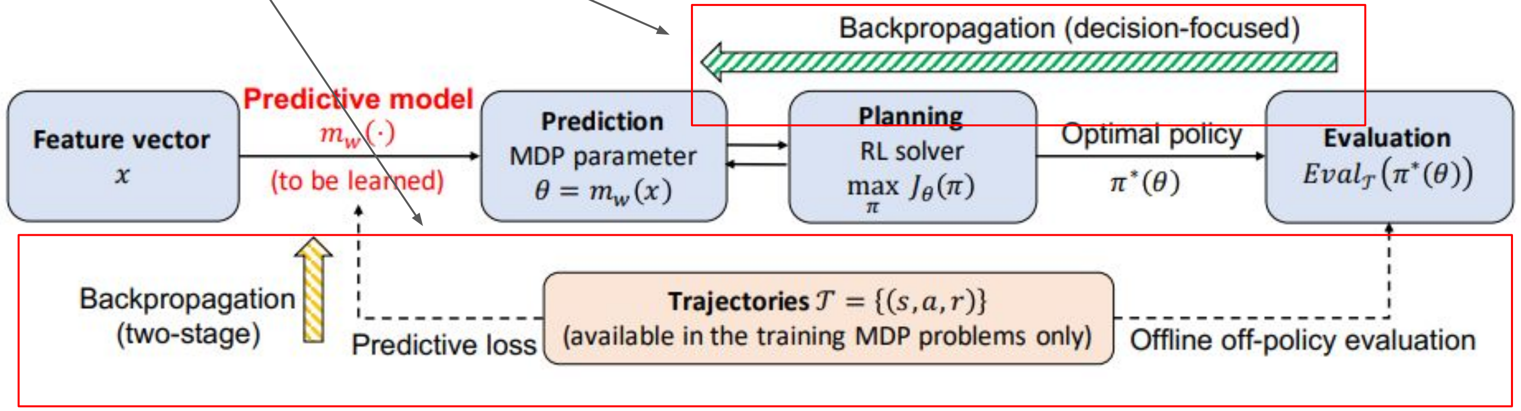
Two competing methods to optimize model

- Decision-focused
- Predictive-loss

Optimization formulation Given a set of training features and trajectories $D_{\text{train}} = \{(x_i, \mathcal{T}_i)\}_{i \in I_{\text{train}}}$, our goal is to learn a predictive model m_w to optimize the training performance:

$$\max_w \quad \mathbb{E}_{(x,\mathcal{T}) \in D_{\text{train}}} [\text{Eval}_{\mathcal{T}}(\pi^*(m_w(x)))] \tag{2}$$

The testing performance is evaluated on the unseen test set $D_{\text{test}} = \{(x_i, \mathcal{T}_i)\}_{i \in I_{\text{test}}}$ with trajectories hidden from training, and only used for evaluation: $\mathbb{E}_{(x,\mathcal{T}) \in D_{\text{test}}} [\text{Eval}_{\mathcal{T}}(\pi^*(m_w(x)))]$.



Backpropagation (decision-focused)
Feature vector x
Predictive model $m_w(\cdot)$ (to be learned)
Prediction MDP parameter $\theta = m_w(x)$
Planning RL solver $\max_\pi J_\theta(\pi)$
Optimal policy $\pi^*(\theta)$
Evaluation $Eval_\mathcal{T}(\pi^*(\theta))$
Backpropagation (two-stage)
Predictive loss
Trajectories $\mathcal{T} = \{(s,a,r)\}$ (available in the training MDP problems only)
Offline off-policy evaluation

Note: backpropagation = gradient descent

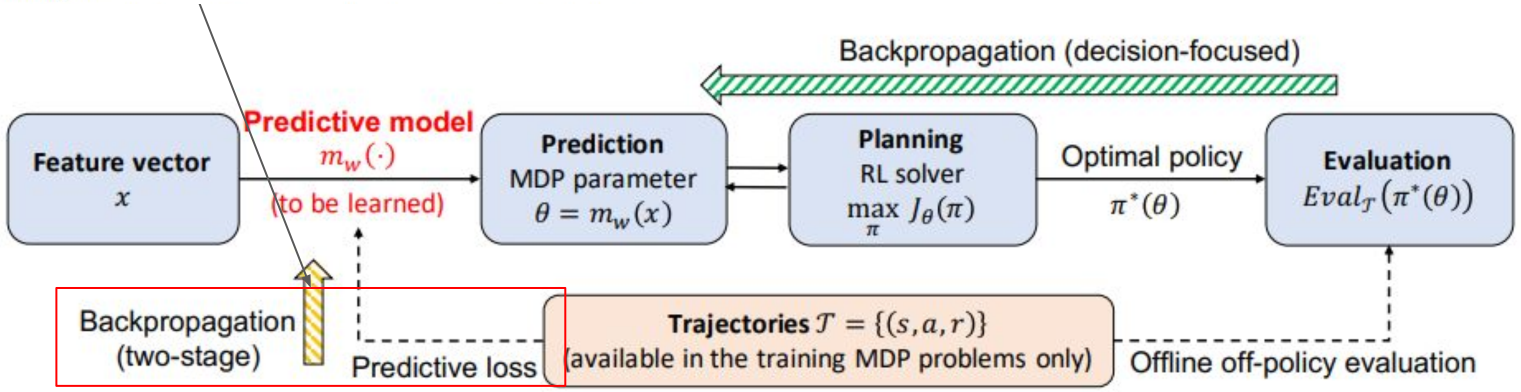
3 Two-stage Learning

To learn the predictive model m_w from trajectories, the standard approach is to minimize an expected predictive loss, which is defined by comparing the prediction $\theta = m_w(x)$ with the trajectories $\mathcal{T}$:

$$\min_w \quad \mathbb{E}_{(x,\mathcal{T})\sim\mathcal{D}_{\text{train}}} \mathcal{L}(\theta, \mathcal{T}) \qquad \text{where} \quad \mathcal{L}(\theta, \mathcal{T}) = \mathbb{E}_{\tau\sim\mathcal{T}}\, \ell_\theta(\tau), \quad \theta = m_w(x) \qquad (3)$$

For example, when the rewards are missing, the loss could be the squared error between the predicted rewards and the actual rewards we see in the trajectories for each individual step. When the transition probabilities are missing, the loss could be defined as the negative log-likelihood of seeing the trajectories in the training set.

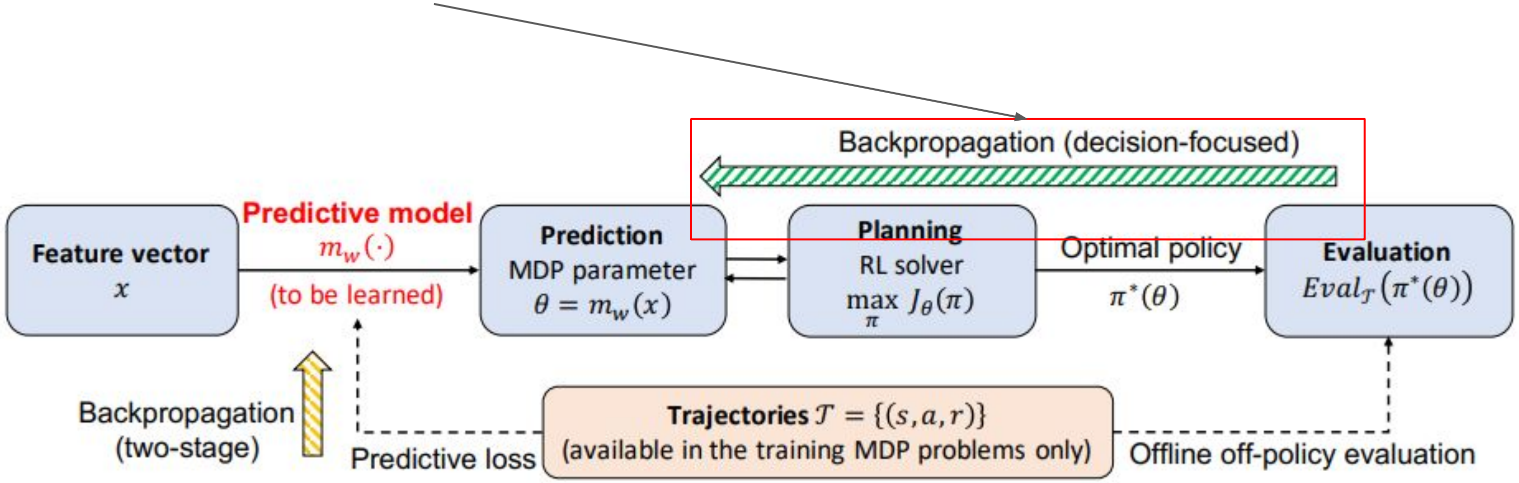
In the first stage, to train the predictive model, we run gradient descent to minimize the loss function defined in Equation (3) and make prediction on the model parameter $\theta = m_w(x)$ of each problem. In the second stage, we solve each MDP problem with the predicted parameter θ using an RL algorithm to generate the optimal policy $\pi^*(\theta)$. However, predictive loss and the final evaluation metric are commonly misaligned especially in deep learning problems with imbalanced data [6, 14, 15, 20]. This motivates us to learn the predictive model end-to-end and therefore avoid the misalignment.



4 Decision-focused Learning

In this section, we present our main contribution, decision-focused learning in sequential decision problems, as illustrated in Figure 1. Decision-focused learning integrates an MDP problem into the training pipeline to directly optimize the final performance. Instead of relying on a predictive loss to train the predictive model m_w, we can directly optimize the objective in Equation (2) by running end-to-end gradient descent to update the predictive model m_w:

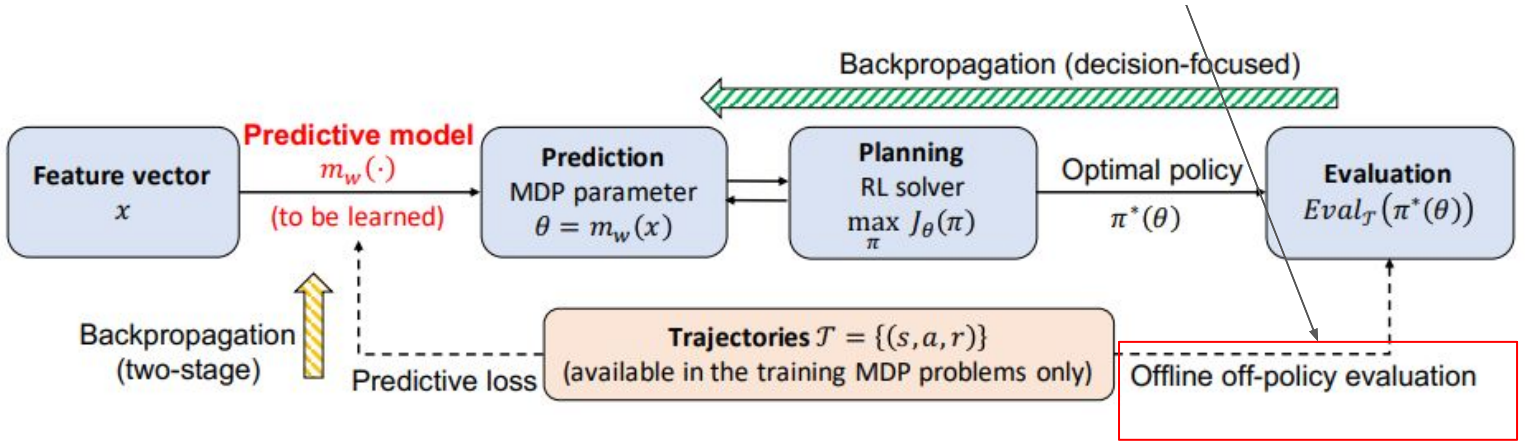
$$\frac{d\,\mathrm{Eval}(\pi^*)}{dw} = \frac{d\,\mathrm{Eval}(\pi^*)}{d\pi^*} \frac{d\pi^*}{d\theta} \frac{d\theta}{dw} \tag{4}$$



Offline off-policy evaluation We evaluate a policy π in a fully offline setting with trajectories $\mathcal{T} = \{\tau_i\}, \tau_i = (s_{i1}, a_{i1}, r_{i1}, \cdots, s_{ih}, a_{ih}, r_{ih})$ generated from the MDP using behavior policy π_{beh}. We use an OPE metric used by Futoma et al. [11] — we evaluate a policy π and trajectories $\mathcal{T}$ as:

$$\text{Eval}_{\mathcal{T}}(\pi) \coloneqq V^{\text{CWPDIS}}(\pi) - \lambda_{\text{ESS}} / \sqrt{\text{ESS}(\pi)} \tag{1}$$

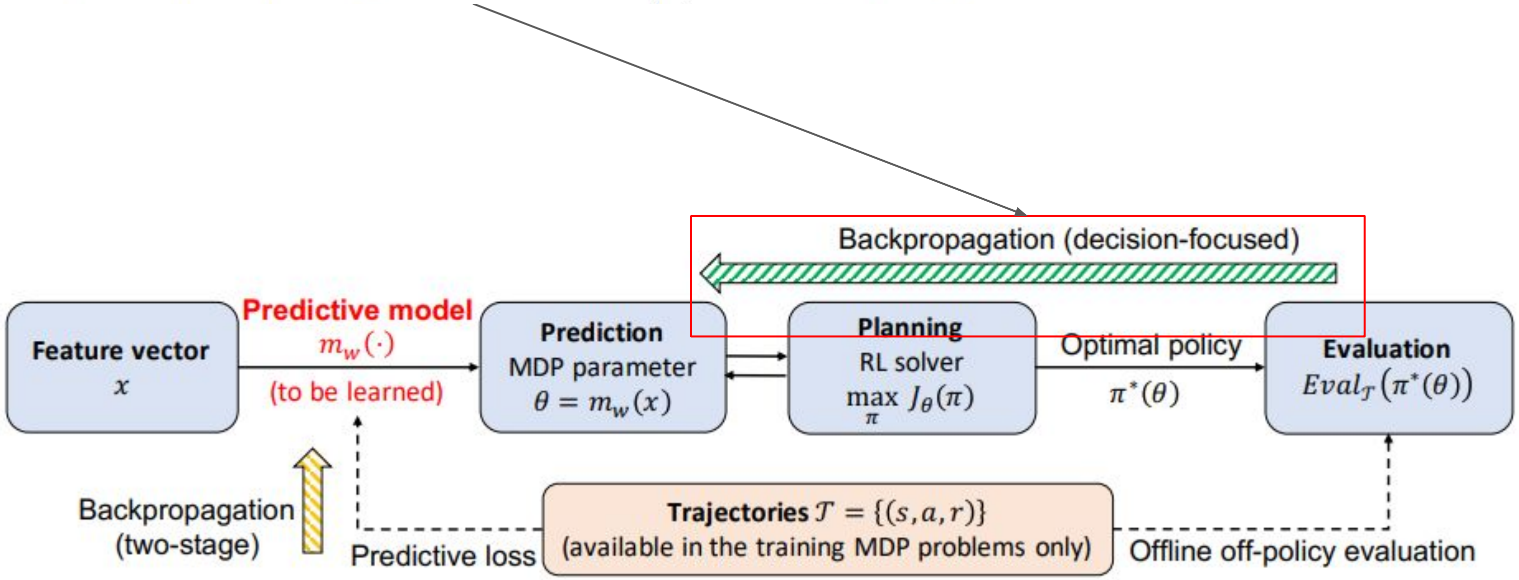
where $V^{\text{CWPDIS}}(\pi) \coloneqq \sum_{t=1}^{h} \gamma^t \frac{\sum_i r_{it} \rho_{it}(\pi)}{\sum_i \rho_{it}(\pi)}$ and $\text{ESS}(\pi) \coloneqq \sum_{t=1}^{h} \frac{(\sum_i \rho_{it})^2}{\sum_i \rho_{it}^2}$, and $\rho_{it}(\pi)$ is the ratio of the proposed policy and the behavior policy likelihoods up to time t: $\rho_{it}(\pi) \coloneqq \prod_{t'=1}^{t} \frac{\pi(a_{it'} | s_{it'})}{\pi_{\text{beh}}(a_{it'} | s_{it'})}$.


Backpropagation (decision-focused)
Feature vector
x
Predictive model
$m_w(\cdot)$
(to be learned)
Prediction
MDP parameter
$\theta = m_w(x)$
Planning
RL solver
$\max_{\pi} J_\theta(\pi)$
Optimal policy
$\pi^*(\theta)$
Evaluation
$Eval_{\mathcal{T}}(\pi^*(\theta))$
Backpropagation
(two-stage)
Predictive loss
Trajectories $\mathcal{T} = \{(s, a, r)\}$
(available in the training MDP problems only)
Offline off-policy evaluation

Definition 1 (Policy gradient-based optimality condition)**.** *Defining $J_\theta(\pi)$ to be the expected cumulative reward under policy π, the optimality condition of the optimal policy π^* is:*

$$\pi^* = \arg\max_\pi J_\theta(\pi) \qquad \text{where} \quad J_\theta(\pi) := \mathbb{E}_{\tau\sim\pi,\theta}\, G_\theta(\tau) \tag{5}$$

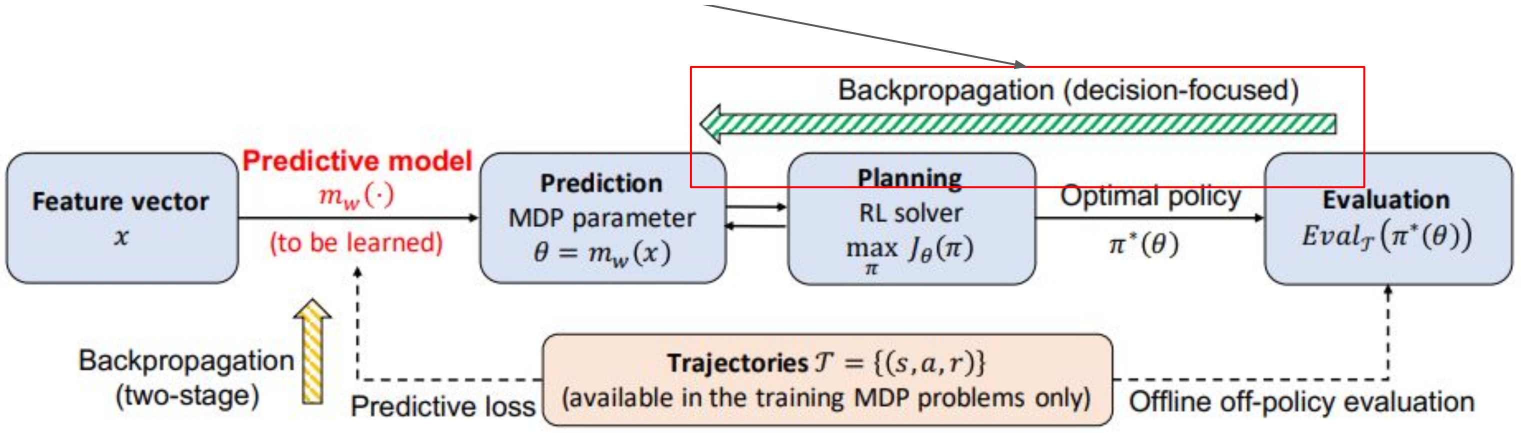
where $G_\theta(\tau)$ is the discounted value of trajectory τ given parameter θ, and the expectation is taken over the trajectories following the optimal policy and transition probability (as part of θ).


Backpropagation (decision-focused)
Feature vector x
Predictive model $m_w(\cdot)$ (to be learned)
Prediction MDP parameter $\theta = m_w(x)$
Planning RL solver $\max_\pi J_\theta(\pi)$
Optimal policy $\pi^*(\theta)$
Evaluation $Eval_{\mathcal{T}}(\pi^*(\theta))$
Backpropagation (two-stage)
Predictive loss
Trajectories $\mathcal{T} = \{(s,a,r)\}$ (available in the training MDP problems only)
Offline off-policy evaluation

Definition 2 (Bellman-based optimality condition). *Defining $J_\theta(\pi)$ to be the mean-squared Bellman error*[3] *under policy π, the optimality condition of the optimal policy π^* (valuation function) is:*

$$\pi^* = \arg\min_\pi J_\theta(\pi) \qquad \textit{where} \quad J_\theta(\pi) := \mathbb{E}_{\tau\sim\pi,\theta} \frac{1}{2}\delta_\theta^2(\tau,\pi) \tag{6}$$

where $\delta_\theta(\tau,\pi) = \sum_{(s,a,s')\in\tau} Q_\pi(s,a) - R_\theta(s,a) - \gamma\,\mathbb{E}_{a'\sim\pi}\, Q_\pi(s',a')$ is the total Bellman error of a trajectory τ, and each individual term $\delta_\theta(\tau,\pi)$ can depend on the parameter θ because the Bellman error depends on the immediate reward R_θ, which can be a part of the MDP parameter θ. The expectation in Equation (6) is taken over all the trajectories generated from policy π and transition probability (as part of θ).



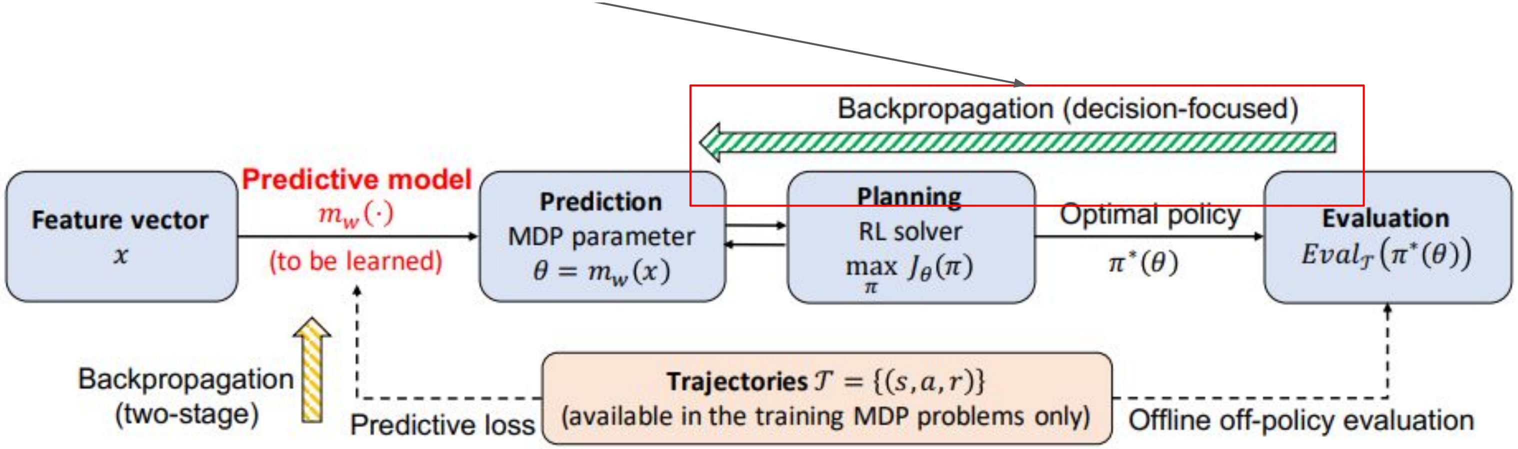
Definition 3 (KKT Conditions). *Given objective $J_\theta(\pi)$ in an MDP problem, since the policy parameters are unconstrained, the necessary KKT conditions can be written as:* $\nabla_\pi J_\theta(\pi^*) = 0$.

In particular, computing the total derivative of KKT conditions gives:

$$0 = \frac{d}{d\theta}\nabla_\pi J_\theta(\pi^*) = \frac{\partial}{\partial\theta}\nabla_\pi J_\theta(\pi^*) + \frac{\partial}{\partial\pi}\nabla_\pi J_\theta(\pi^*)\frac{d\pi^*}{d\theta} = \nabla^2_{\theta\pi} J_\theta(\pi^*) + \nabla^2_\pi J_\theta(\pi^*)\frac{d\pi^*}{d\theta}$$

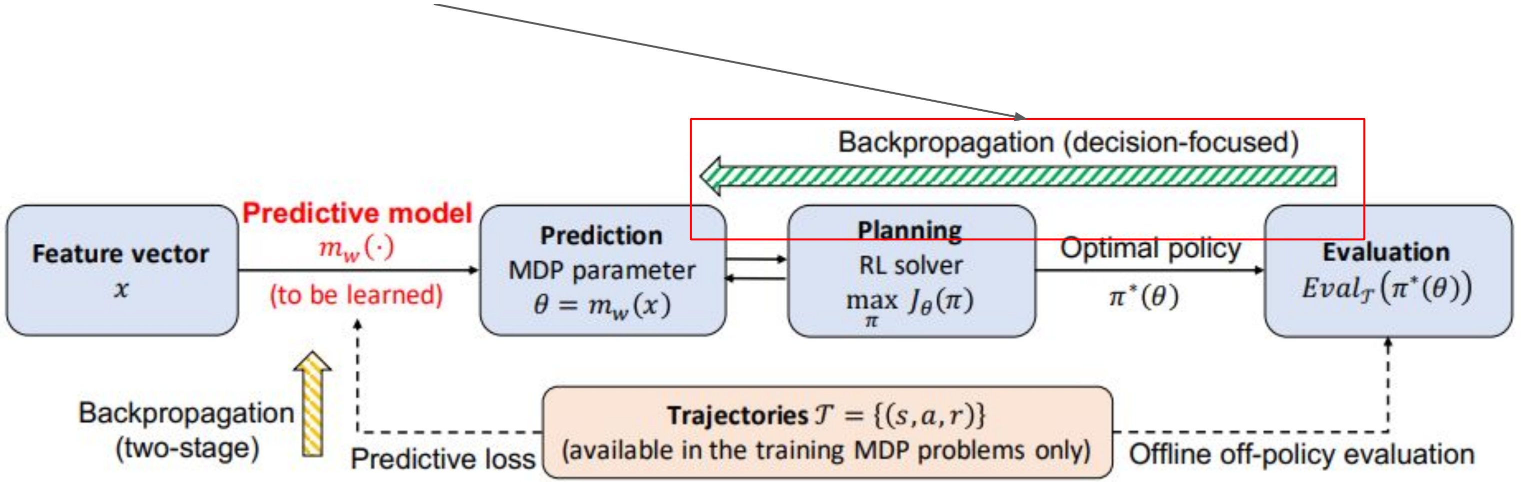
$$\implies \frac{d\pi^*}{d\theta} = -(\nabla^2_\pi J_\theta(\pi^*))^{-1}\nabla^2_{\theta\pi} J_\theta(\pi^*) \tag{7}$$

KKT requires convex objective function?



We can use Equation (7) to replace the term $\frac{d\pi^*}{d\theta}$ in Equation (4) to compute the full gradient to back-propagate from the final evaluation to the predictive model parameterized by w:

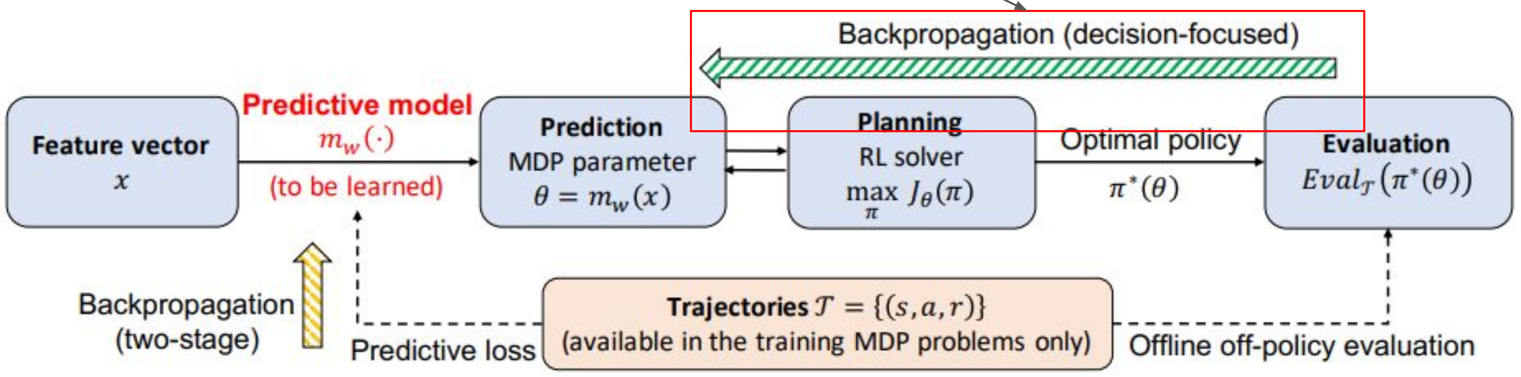
$$\frac{d\,\mathrm{Eval}(\pi^*)}{dw} = -\frac{d\,\mathrm{Eval}(\pi^*)}{d\pi^*}(\nabla^2_{\pi} J_\theta(\pi^*))^{-1}\nabla^2_{\theta\pi} J_\theta(\pi^*)\frac{d\theta}{dw} \tag{8}$$



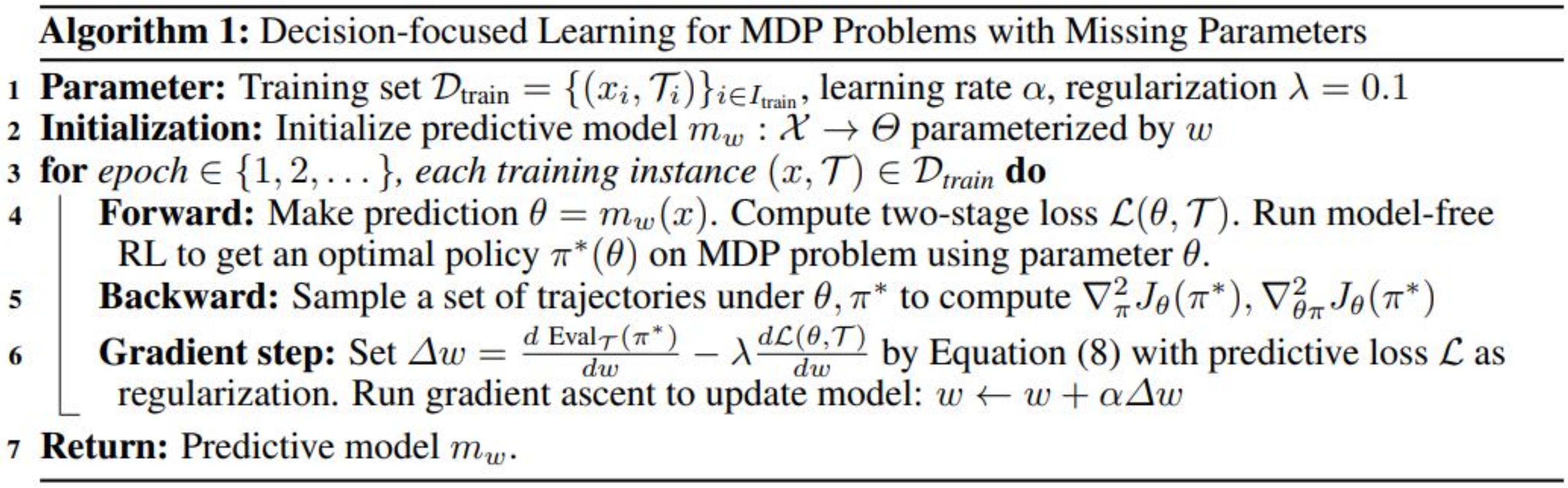
Backpropagation (decision-focused)
Feature vector x
Predictive model
$m_w(\cdot)$
(to be learned)
Prediction
MDP parameter
$\theta = m_w(x)$
Planning
RL solver
$\max_\pi J_\theta(\pi)$
Optimal policy
$\pi^*(\theta)$
Evaluation
$Eval_{\mathcal{T}}(\pi^*(\theta))$
Backpropagation (two-stage)
Predictive loss
Trajectories $\mathcal{T} = \{(s,a,r)\}$
(available in the training MDP problems only)
Offline off-policy evaluation

Computing full Hessian is infeasible

- Sample trajectories to obtain unbiased estimate of a low-rank approximation of hessian


Backpropagation (decision-focused)
Feature vector
x
Predictive model
$m_w(\cdot)$
(to be learned)
Prediction
MDP parameter
$\theta = m_w(x)$
Planning
RL solver
$\max_\pi J_\theta(\pi)$
Optimal policy
$\pi^*(\theta)$
Evaluation
$Eval_\mathcal{T}(\pi^*(\theta))$
Backpropagation (two-stage)
Predictive loss
Trajectories $\mathcal{T} = \{(s,a,r)\}$
(available in the training MDP problems only)
Offline off-policy evaluation

Full algorithm

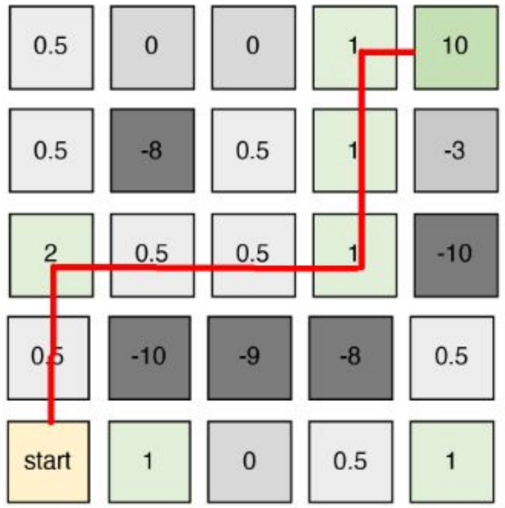


Algorithm 1: Decision-focused Learning for MDP Problems with Missing Parameters

```
1 Parameter: Training set $\mathcal{D}_{\text{train}} = \{(x_i, \mathcal{T}_i)\}_{i \in I_{\text{train}}}$, learning rate $\alpha$, regularization $\lambda = 0.1$
2 Initialization: Initialize predictive model $m_w : \mathcal{X} \to \Theta$ parameterized by $w$
3 for epoch $\in \{1, 2, \dots\}$, each training instance $(x, \mathcal{T}) \in \mathcal{D}_{train}$ do
4     Forward: Make prediction $\theta = m_w(x)$. Compute two-stage loss $\mathcal{L}(\theta, \mathcal{T})$. Run model-free
        RL to get an optimal policy $\pi^*(\theta)$ on MDP problem using parameter $\theta$.
5     Backward: Sample a set of trajectories under $\theta, \pi^*$ to compute $\nabla^2_\pi J_\theta(\pi^*), \nabla^2_{\theta\pi} J_\theta(\pi^*)$
6     Gradient step: Set $\Delta w = \frac{d\ \text{Eval}_{\mathcal{T}}(\pi^*)}{dw} - \lambda \frac{d\mathcal{L}(\theta, \mathcal{T})}{dw}$ by Equation (8) with predictive loss $\mathcal{L}$ as
        regularization. Run gradient ascent to update model: $w \leftarrow w + \alpha \Delta w$
7 Return: Predictive model $m_w$.
```

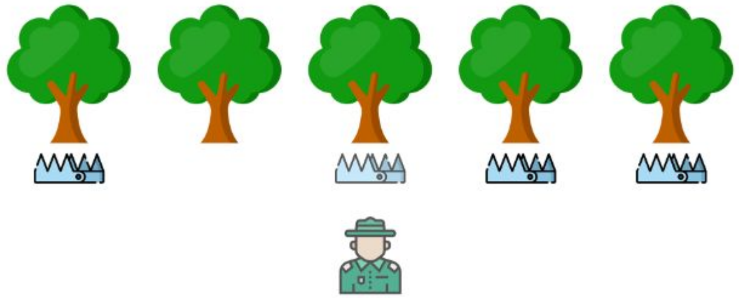
Experiments from the paper

- Unknown Rewards
  - Gridworld with unknown cliffs
    - Agent does not know goal state
    - Random reward at goal state
    - Intermediate states have reward of n(0,1)
    - 20% states have penalty of n(-10, 1)



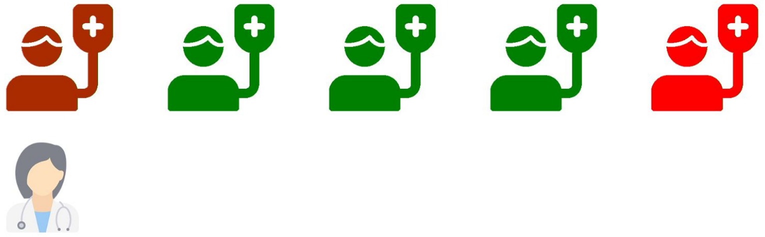
Experiments from the paper

- Unknown transition function
  - Partially Observable snare-finding problems with missing transition function
    - Park ranger (Agent) has to pick between 20 sites (states)
    - 4 sites have a high chance of a ‘snare’ - n(0.8, 0.1), other 16 sites have a chance of (0.1, 0.05) - unknown to agent.
    - Agent gets rewards if it finds a snare, if snare not removed or agent visits a state that does not have a snare then agent will receive a penalty
    - Not a Multi-armed bandit problem because states are continuous - i.e. snares stay if not found



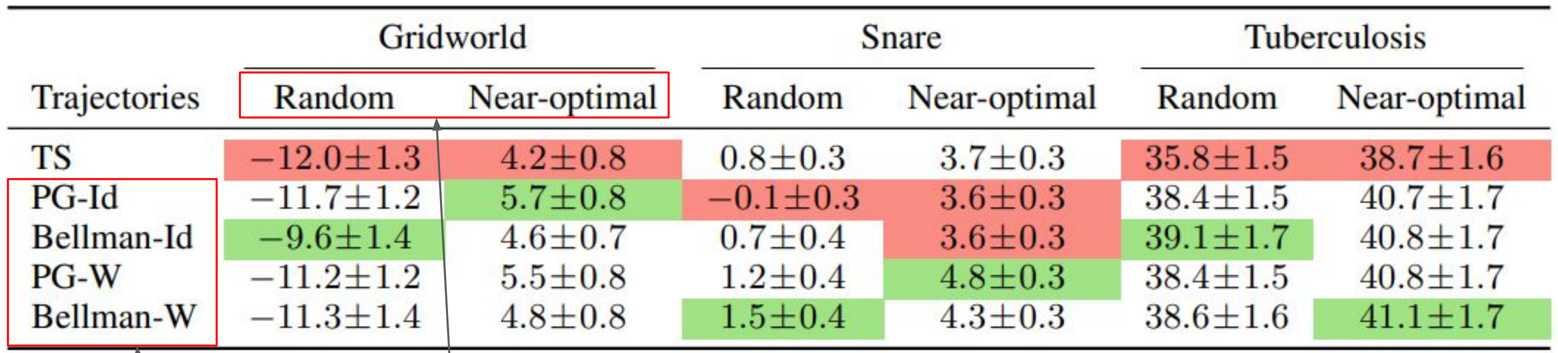
Example - Test from paper

- Unknown transition probability
    - Partially Observable patient treatment problems with missing transition probability
        - 5 Patients (states), each can have 2 possible states non-adhering or adhering.
        - Each day (step), if the nurse (agent) visits a patient that is not adhering then the patient will change state to adhering (with some probability).
        - Reward at each step is number of patients adhering.
        - Want to maximize reward over 30 days.



Results

Table 1: OPE performances of different methods on the test MDPs averaged over 30 independent runs. Decision-focused learning methods consistently outperform two-stage approach, with some exception using identity matrix based Hessian approximation which may lead to high gradient variance.

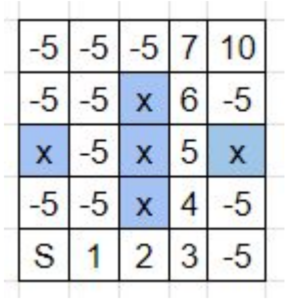


	Gridworld		Snare		Tuberculosis	
Trajectories	Random	Near-optimal	Random	Near-optimal	Random	Near-optimal
TS	-12.0 ± 1.3	4.2 ± 0.8	0.8 ± 0.3	3.7 ± 0.3	35.8 ± 1.5	38.7 ± 1.6
PG-Id	-11.7 ± 1.2	5.7 ± 0.8	-0.1 ± 0.3	3.6 ± 0.3	38.4 ± 1.5	40.7 ± 1.7
Bellman-Id	-9.6 ± 1.4	4.6 ± 0.7	0.7 ± 0.4	3.6 ± 0.3	39.1 ± 1.7	40.8 ± 1.7
PG-W	-11.2 ± 1.2	5.5 ± 0.8	1.2 ± 0.4	4.8 ± 0.3	38.4 ± 1.5	40.8 ± 1.7
Bellman-W	-11.3 ± 1.4	4.8 ± 0.8	1.5 ± 0.4	4.3 ± 0.3	38.6 ± 1.6	41.1 ± 1.7

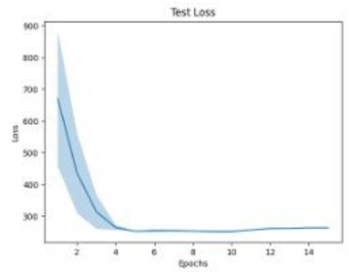
Refers to training trajectories

Id and W are different approximations of Hessian Matrix - Identity less accurate than Woodbury but cheaper

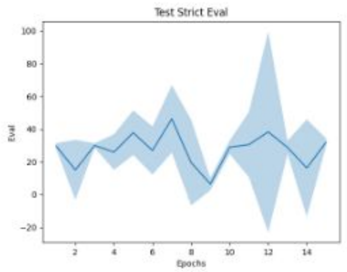
Experiments



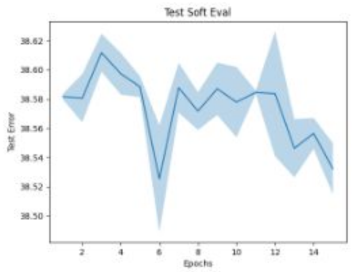
-5	-5	-5	7	10
-5	-5	x	6	-5
x	-5	x	5	x
-5	-5	x	4	-5
S	1	2	3	-5



(a) Test Loss

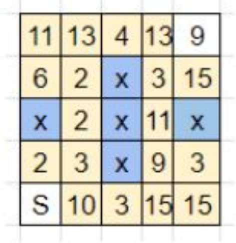


(b) Average Reward Per Episode



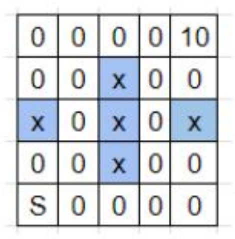
(c) OPE Evaluation

Experiments



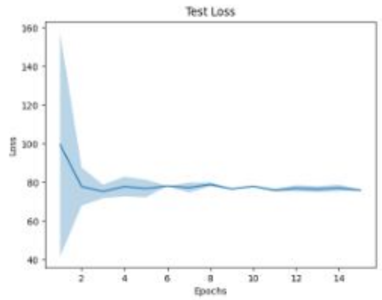
11	13	4	13	9
6	2	x	3	15
x	2	x	11	x
2	3	x	9	3
S	10	3	15	15

(a) Features

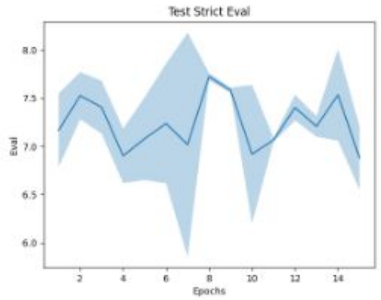


0	0	0	0	10
0	0	x	0	0
x	0	x	0	x
0	0	x	0	0
S	0	0	0	0

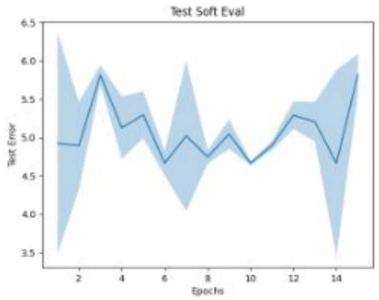
(b) Rewards



(a) Test Loss

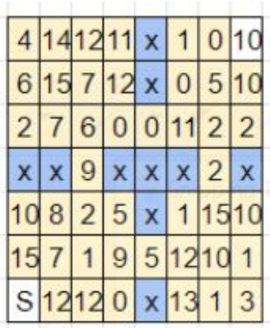


(b) Average Reward Per Episode



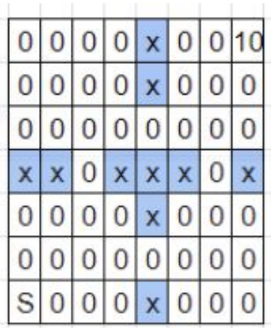
(c) OPE Evaluation

Experiments



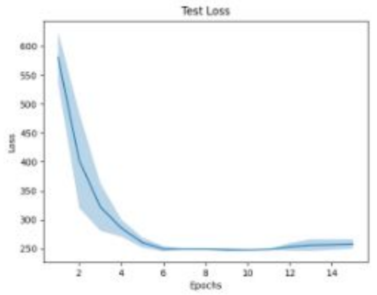
4	14	12	11	x	1	0	10
6	15	7	12	x	0	5	10
2	7	6	0	0	11	2	2
x	x	9	x	x	x	2	x
10	8	2	5	x	1	15	10
15	7	1	9	5	12	10	1
S	12	12	0	x	13	1	3

(a) Features

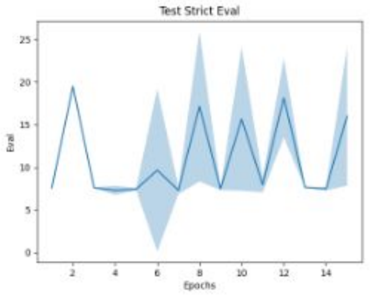


0	0	0	0	x	0	0	10
0	0	0	0	x	0	0	0
0	0	0	0	0	0	0	0
x	x	0	x	x	x	0	x
0	0	0	0	x	0	0	0
0	0	0	0	0	0	0	0
S	0	0	0	x	0	0	0

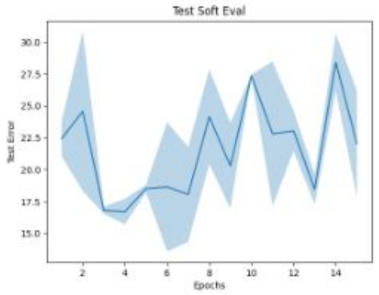
(b) Rewards



(a) Test Loss

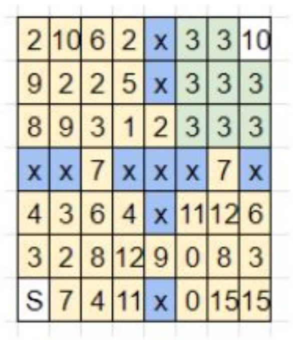


(b) Average Reward Per Episode



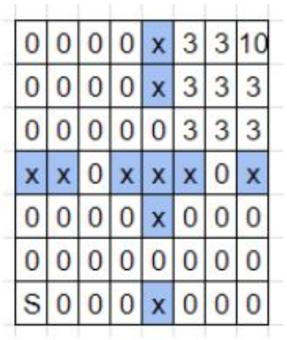
(c) OPE Evaluation

Experiments



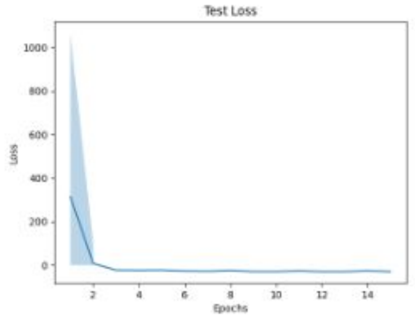
2	10	6	2	x	3	3	10
9	2	2	5	x	3	3	3
8	9	3	1	2	3	3	3
x	x	7	x	x	x	7	x
4	3	6	4	x	11	12	6
3	2	8	12	9	0	8	3
S	7	4	11	x	0	15	15

(a) Features

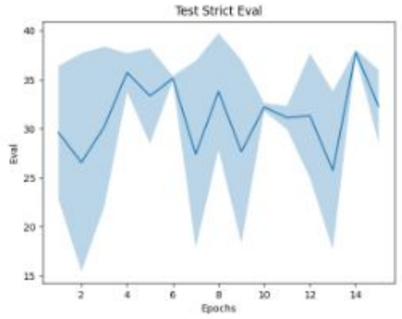


0	0	0	0	x	3	3	10
0	0	0	0	x	3	3	3
0	0	0	0	0	3	3	3
x	x	0	x	x	x	0	x
0	0	0	0	x	0	0	0
0	0	0	0	0	0	0	0
S	0	0	0	x	0	0	0

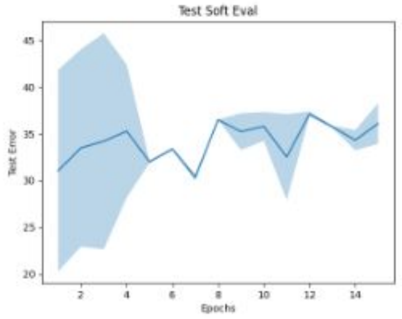
(b) Rewards



(a) Test Loss



(b) Average Reward Per Episode



(c) OPE Evaluation