Evaluation of Optimal Decision Making with Dead-ends in High-risk Environments

STAT 234 Final paper

Alex Kashi [1] **Marcel Torné Villasevil** [1]

Abstract

In this manuscript, we use a simulation of a tabular environment to investigate the identification of dead-ends in high-risk environments under multiple data collection policies. We run the experiments on the Life-Gate toy dataset by Fatemi et al. The data collection is performed under three policies: optimal policy, suboptimal policy (choosing the best action most of the time and with a low probability of taking a random action), and a random policy. The methods used to learn the q-value functions are Q-Learning, the dead-end discovery method by Fatemi et al., and, Fitted Q iteration. The results are that all three methods recover decently well the dead-ends using any of the three data collection policies. However, Q-learning seems to be the one that works best for this specific environment.

1. Introduction

Reinforcement learning is a commonly used tool in sequential decision making that often attempts to model an environment using states, actions, and rewards. Many online RL algorithms rely on taking sub-optimal actions to explore their environment to uncover new, more optimal policies. This is the well-known problem of exploration vs exploitation problem. Known algorithms that try to solve this are adding epsilon greedy exploration (MKS+13) and LinUCB (LCLS10). In high-risk situations like healthcare, where patients' lives and well-being depend on the actions we take concerning their treatment, sub-optimal exploration is not an option. Reinforcement learning in the medical world relies on previously collected datasets, where medical experts choose actions. Consequently, exploration of alternative actions is difficult, if not impossible, and algorithms are typically developed in an offline fashion.

In this paper, we will explore more deeply environments with "dead-ends" terminal states where the patient will eventually finish in a negative terminal state no matter which action is taken. We will evaluate previous methods for "dead-ends" discovery and observe what happens in different scenarios to make good use of these. The evaluation of these methods will be done through a toy simulation environment proposed by Fatemi et al -(FKSG21). We will perform experiments on collecting data using multiple policies, random, optimal, and biased, and analyze whether the dead-ends discovery method can always recover meaningful and right Q-value functions.

We aim to verify whether it is possible to identify dead ends from data and under what circumstances. We are curious about the consequences of having collected a poor dataset. For example, in medical settings, doctors might try more alternative and uncertain treatments for patients in worse conditions. As a result, we might see that these alternative treatments have a lower success rate because the patients were less likely to survive any treatment. Could this be a problem when learning our value functions? We are also curious to know how good this dead-end discovery method proposed by Fatemi et al. compared to learning Q-value functions using well-known offline RL algorithms as Fitted Q Iteration(EGW05). Finally, we are also curious to know how reliable the value functions we obtain are. Is there any way to know about the uncertainty of the value function for some patients at a given state? This is a necessary condition for such ML algorithms since if the uncertainty is high, the doctor can be informed and ignore the recommendation given by the algorithm.

2. Related Work

There has been some research already in this field. Fatemi et al. proposed the Dead-end Discovery method in their paper Medical Dead-ends and Learning to Identify High-risk States and Treatments. Their goal is to find Medical Dead-ends in septic patients in an intensive care unit (ICU) setting (FKSG21). They use Reinforcement Learning to inform the doctors about which treatments not to take. To solve this task, they train multiple networks. The first, they call SC-network (State construction), which is the network that transforms the information at a given point into an embedding to create the state. Then, they trained two separate networks for identification (D-Network and R-Network).

These two networks are used to compute Q_D and Q_R two value functions where $-Q_D(s,a)$ will correspond to the minimum probability of a negative outcome if the treatment a is administered at state s and $1 + Q_D(s,a)$ will be the maximum hope of a positive outcome. Furthermore, these two value functions are learned from two MDPs M_R, which gives a reward of -1 for any transition to a negative terminal state (and 0 for the rest of the transition) and M_R, which returns +1 for a positive terminal state (zero otherwise).

3. Methods

3.1. Environment

The environment proposed by Fatemi et al. will be adapted and used to investigate these methods. The environment is a grid where the agent can move around (no-op, up, down, left, right). Some states are dead ends, some are neutral, and the ones on the right and top edge are terminal states, where either the patient can survive or not. Furthermore, we added some stochasticity to this environment by adding a simulated wind. If an agent is in a black square, it can move in any direction, where actions into walls result in the agent staying in the same state. However, with a 20% chance, the user's desired action is ignored, and the agent is forced to the right. The motivation for this wind is to simulate the patient's aging and the consequences for the human body. Transitioning into the right-hand column results in a negative reward. If the agent is in a dead-end state, 70% of the time, the agent will be forced to the right. Otherwise, the agent stays in the same location, ignoring all actions. Actions are seen as treatments, and the observed state (the agent's coordinates in the grid) is the patient's current health. The transition function will be designed such that there are dead-ends, and these are represented in yellow. Furthermore, grey cells are simply walls (meaning that the agent cannot traverse them), and black cells are neutral. The reward function will be a sparse reward in this case, where if the agent finishes in a red cell, the reward will be -1. If the agent finishes in a blue cell, the reward will be +1.

This environment will be then used to collect datasets for offline training and to roll out the learned policies at test time.

3.2. Data Collection

We aim to investigate how the data collection policy will affect our results when we run the offline algorithms with their corresponding collected datasets. We explored three different data collection methods: optimal, sub-optimal, and random. Each method of data collection represents a type of clinical scenario. Optimal data is representative of a dataset collected by observing an excellent doctor, whereas the suboptimal dataset represents a dataset collected by observing a less experienced doctor or a doctor acting in a lesser known environment. Finally, the random dataset can be compared to a scenario where there may be a new, poorly understood disease where best practices have yet to be established yet.

Figure 1. Life-gate toy simulated environment.

Algorithm 12 Value Iteration

procedure VALUEITERATION(π, ϵ)
  Initialize $V_1(s) = 0\ \forall s \in \mathcal{S}$
  for $i = 1 \cdots$ **do**
    $V_{i+1}(s) \leftarrow \max_a \sum_{s' \in \mathcal{S}} \mathcal{T}(s'|s,a)\left[r(s,a) + \gamma V_i(s')\right]\ \forall s \in \mathcal{S}$
    if $\|V_{i+1} - V_i\|_\infty \leq \epsilon$ **then**
      break
    end if
  end for
  Set policy $\pi(s) = \arg\max_a \mathbb{E}_\pi\left[r(s_t,a_t) + \gamma V^\pi(s_{t+1}) | s_t = s, a_t = a\right]\ \forall s \in \mathcal{S}$
end procedure

Figure 2. Value iteration algorithm as proposed in (ZWH22)

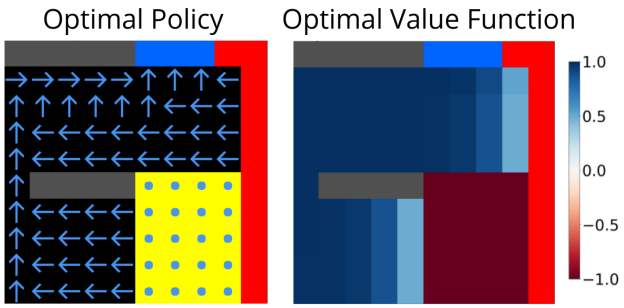

Figure 3. (Left) optimal Policy calculated using value iteration, (right) Optimal value function under the optimal policy

Since our environment is tabular with known transitions and rewards, we calculated can calculate the optimal policy $\pi^*(s)$ and optimal value function $V^{\pi^*}(s)$ by using value

iteration and policy extraction delineated in the algorithm 12. The results of the optimal policy and value function are displayed in Figure 3. The suboptimal policy acts optimally 75% of the time and randomly 25% of the time.

Heatmaps of the states of the trajectories of 2500 rollouts of each policy are shown in figure 4. Under the optimal and suboptimal policies, the agent spends most of its time on the left-hand side of the field, since it tries to avoid both dead-ends and terminal negative states. The random trajectories cover much more of the environment and will provide more context to the agent. We will use these datasets to fit the following offline RL algorithms to the data, determine an optimal policy for treating patients, and identify possible dead-ends in the environment.

3.3. Algorithm 1: Dead-end discovery by Fatemi et al.

The first method we will analyze will be the dead-end discovery method proposed by Fatemi et al. (FKSG21). This one separates the environment's Markov Decision Process (MDP) into two different MDPs, M_D and M_R. M_D returns -1 for all transitions to a negative terminal state and 0 for the rest. This MDP will be used to identify dead-ends. The second MDP, M_R returns +1 for transitions to recovery states and 0 for the rest first. This second MDP will be used to identify recovery states. Given that we know the rewards for these MDPs and using $\gamma = 1$, we know that $Q_D*(s,a) \in [-1,0]$ and $Q_R*(s,a) \in [0,1]$ for all states and transitions. In their paper, Fatemi et al. prove that $-Q_D*(s,a)$ corresponds to the minimum probability of a negative outcome, and $1 + Q_D*(s,a)$ as the maximum hope of a positive outcome if action a is taken in state s.

We will only use $Q_D(s,a)$ to extract the policies and value function to identify dead-ends in our experiments. This q-table is obtained by performing offline Q-learning in the M_D MDP using the collected off-policy dataset.

3.4. Algorithm 2: Q-learning

The second method we will analyze and compare is standard Q-learning (SB18). Since we are doing offline RL, learning the q-table will involve iterating through the replay buffer and updating the q-table using Bellman's equation. The replay buffer is initialized with the collected dataset. The algorithm is presented under algorithm 1.

3.5. Algorithm 3: Fitted Q Iteration

Fitted Q Iteration (EGW05) is an off-policy offline method for determining a Q function. The algorithm is straightforward and generalizable. As long as we have trajectories containing (s,a,r,s') tuples, we can use Fitted Q Iteration. Fitted Q iteration works by fitting a model to the bellman target as defined in Figure 5. Usually, this model is a deep neural network. However, this is not necessary. The main advantage of Fitted Q iteration over classical Q learning is its ability to aggregate states, relieving the need to store a $|S|X|A|$ sized Q-table. Additionally, Fitted Q iteration can be adapted to continuous action spaces easily.

Algorithm 1 Offline Q-learning

```
Buffer ← collected dataset
Q(s, a) ← −1
epoch ← 0
for epoch < total_epochs do
  epoch ← epoch + 1
  for (s, s', a, r) in < Buffer do
    Q(s,a) ← Q(s,a) + α[r + γ max'_a Q(s',a') − Q(s,a)]
  end for
end for
return Q(s, a)
```

4. Experiments

This project analyzes how each method performs in the given environment to identify the dead-ends. Furthermore, it is crucial to know when we should trust these methods. More concretely, we will experiment using different types of data collection policies and observe if these methods are robust to these changes.

The main experiment of the paper will consist in running each of the policies in the same environment presented earlier varying the data collection policy. As mentioned before, we will collect data three times: with an optimal policy, a suboptimal policy, and a random policy. In this way, we simulate different doctors' skills and observe how robust the method is to this variation.

The data collection consisted of 2500 different trajectories with a maximum of 100 steps rolling out each of the data collection agents described earlier. Moreover, for each of these trajectories, the initial position of the agent was randomly selected in the grid (among the valid cells). This is to mimic the situation of patients who might go to the doctor in different stages of their disease, from very early on to too late in the process.

5. Results

5.1. Dead-end discovery by Fatemi et al.

In Figure 6, we observe that the value function extracted is relatively good in the three cases of data collection methods. The best is when data is collected randomly since we see the clearly defined boundary for dead ends. We would explain this behavior by stating that when the policy that collected

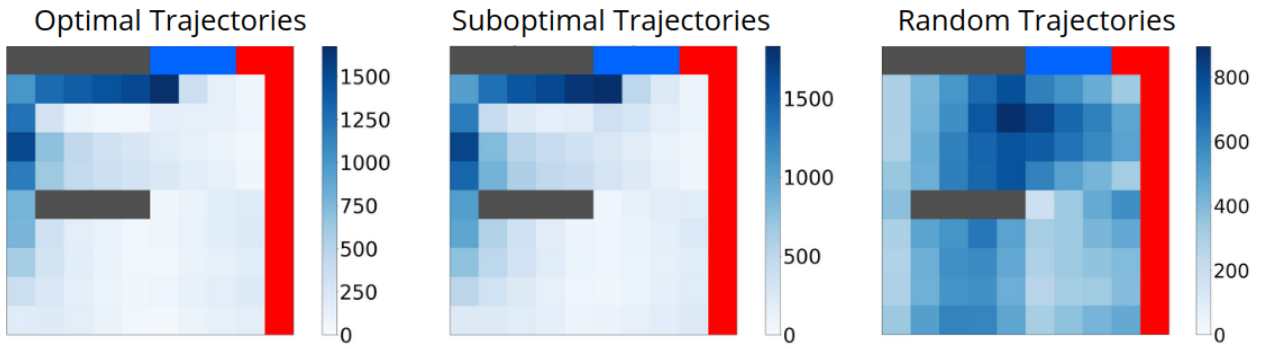

Figure 4. Heatmap of trajectories rolled out under the: optimal policy (left), suboptimal policy (middle), random policy (center)

Algorithm 14 Fitted Q iteration

procedure FQI
  Initialize the neural network weights ϕ randomly
  while not terminate **do**
    Collect dataset $\mathcal{D}$ using an arbitrary policy to interact with the environment
    for $i = 1 \cdots$ **do**
      Compute the target $y \leftarrow r(s,a) + \gamma \max_{a'} Q_\phi(s',a') \; \forall (s,a,r,s') \in \mathcal{D}$
      Update the network $\phi \leftarrow \arg\min_{\phi'} \frac{1}{|\mathcal{D}|} \sum_{(s,a,r,s') \in \mathcal{D}} \|y - Q_{\phi'}(s,a)\|^2$
    end for
  end while
end procedure

Figure 5. Fitted Q iteration pseudocode (ZWH22)

the data is random, then the exploration is much broader (at least in this small environment). Hence, it will visit most of the states and will perform better. This might be due to the small size of the environment, a large number of episodes, and a large maximum steps per episode. We obtain many trajectories reaching positive/negative goals, which might not be the case in more complex or bigger environments if we do not increase the size of the data collected.

Furthermore, we observe that all of the values in this value function are smaller or equal to zero. The reason being that for, Fatemi et al.'s method, the Q table is learned from the MDP M_D where positive rewards are mapped to 0.

This same reason is why we observe that the policies learned using the suboptimal and random data collection policies are entirely useless. These will not lead to a positive goal. Nevertheless, Fatemi et al.'s method aims to detect dead-ends and not necessarily identify the best policy. However, there is an extension to this method presented in their paper were they use the Q-table learned for the recovery states to obtain some probabilities about each action to lead the agent to a dead-end or a recovery state, in which case the policy might be better, but this is not the goal of our project so we will not look further into it.

Even though for the suboptimal and random data collection policies, the optimal policy was not found, it is interesting to see that in the optimal case, the optimal policy was indeed found. You might ask yourself, how is this the case? The answer is that we initialized the value function at -1. Since the optimal policy dataset mainly consists of optimal actions, these are the only ones that are accepted in the Q table, and hence their value becomes 0. For this reason, the most frequent actions in the dataset will be mapped to the policy extracted, which is also the optimal policy. It is a fascinating phenomenon, but we want to point out that it is not due to the algorithm learning the reward, but it does more of a behavior cloning of the data collection policy.

5.2. Q-learning

In Figure 7, we observe the results obtained in the experiments when using the Q-learning method. The results are surprisingly good. The dead-ends boundary is apparent. All of the states inside the dead-end zone have close to -1 value, and all of the ones outside the dead-end zone have positive values, except for a single cell in the random data collection experiment. This makes this method especially good since we are able to clearly differentiate dead-ends from non-dead-ends. One question arises, how is it possible that Q-learning performs better than Fatemi's dead-end discovery method, which was designed explicitly for the purpose we are studying here? Especially that the difference between our implementation of Fatemi's approach is that we are applying Q-learning on a modified MDP. One of our insights is that the information about positive terminal states (used in Q-learning but omitted in Fatemi's method) is helpful for learning more about dead-ends. Second, Fatemi's method brings more information about the probability of an action leading us to a dead-end, which is not the case in Q-learning and can be preferred depending on the setting. Finally, we believe that in those environments where the boundaries between dead-ends and recovery states is not as

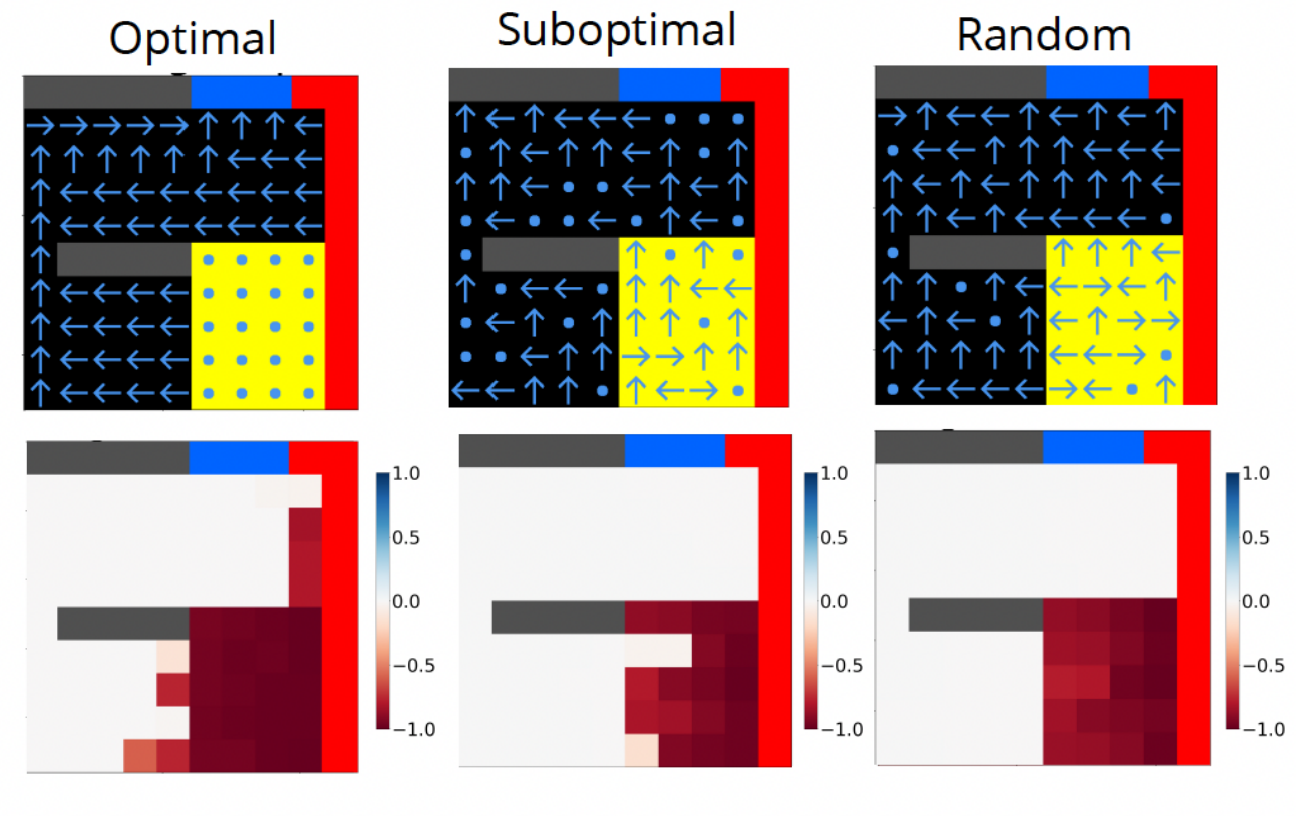

Figure 6. Policy (top) and value function (bottom) using Fatemi et al's method for each one of the data collection policies: optimal, suboptimal and random.

clear, then omitting the information about recovery states can help q-learning learn a better representation on where dead ends are, so Fatemi might work best. We leave this line of work as future work.

5.3. Fitted Q iteration

In Figure 8, we present the results of running fitted Q iteration on the three datasets. Our model is a 6-layer neural network with 32 nodes in each hidden layer. In the optimal case, the algorithm generally finds where the dead-ends are and, finds a policy that aligns with the data. However, there is a critical flaw in the output policy. The top right corner calls for an action that leads to certain death. This is most likely a result of the neural network aggregating between states. Since the optimal policy will never take action upward from that state, the classifier does not know to avoid the action since it is so close to a reward state. This raises profound implications for the validity of fitted Q iteration in high-risk environments, where small changes in the state space could result in significant shifts in optimal action. The suboptimal case also exhibits this erroneous action for similar reasons. Figure 4 demonstrates the data sparsity of this region, and it is unlikely that the sub-optimal policy has performed an up action in the state of interest. We predict that a significant increase in data would resolve this issue in the suboptimal case, however, in the optimal case, it may never be resolved. The results on the random dataset bolster this prediction, demonstrating that enough data in the top right corner will converge to the optimal action.

5.4. Overall Comparison

We devised two metrics for quantitatively evaluating a policy's ability to identify dead ends and the policy's clinical utility. When $\gamma = 1$, dead-ends by definition, have a -1 reward. However, in a clinical setting, the location of the dead-ends is unknown and would need to be extracted from the value function under a trained policy. One such way of extracting dead-ends is by setting a value threshold α such that any state with a value less than α will be considered a dead end. Setting the alpha too low will result in many false-positive dead-end states, while an alpha that is too high may exclude dead-ends for algorithms that attempt to aggregate states, like Fitted Q iteration. To quantitatively measure how good an algorithm is at determining

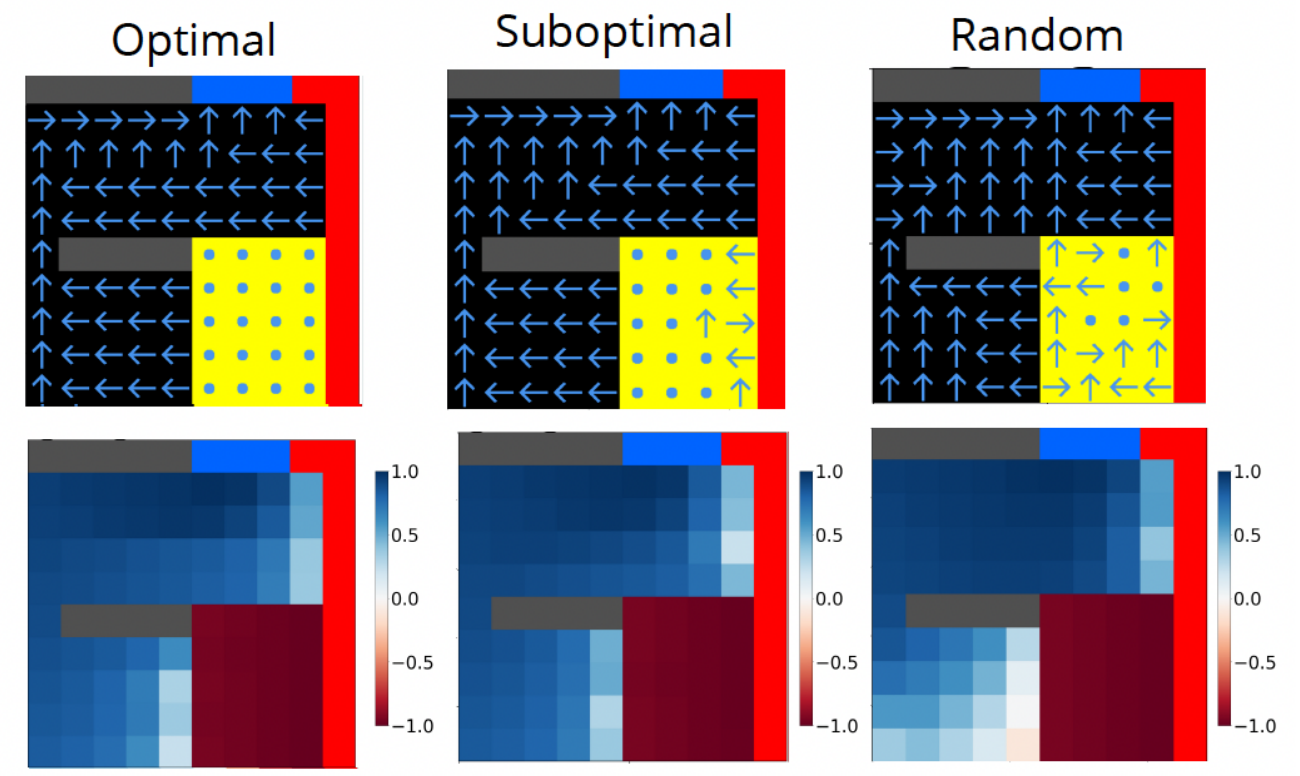

Figure 7. Policy (top) and value function (bottom) using Q-learning method for each one of the data collection policies: optimal, suboptimal and random.

dead-ends and can tolerate a higher α, we decided to take the MSE between the optimal value function in each of the known dead-end end states. The results can be found in 9

We rolled out 2500 agent trajectories following the fitted policy in each scenario and calculated the average reward for each policy to determine if the policy can detect dead-ends and be used to drive clinical decisions. We observed that Q-learning was 9 the most optimal in all instances. However, it performed the best with optimal data. Fitted Q iteration was also able to find viable policies in each instance. On the other side, we observe that the dead-end discovery method by Fatemi et al. is not able to extract a good policy in either the suboptimal or random data collection policies. As we mentioned earlier, this is because this method is not constructed to identify the best policies but just to identify dead-ends without being biased by recovery states. Nevertheless, in the paper, Fatemi et al. propose some ways to identify the transition probability to finish at a dead-end which might be useful in some scenarios but again does not help you obtain the optimal policy.

6. Conclusion

To conclude, we want to reiterate the importance of identifying dead-ends. This could be used in many different scenarios. As for the example, the area that has been the focus of our project, dead-end detection could be used in medical settings by doctors to identify which treatments should not be applied to a patient. Going further, this can be used for Safe-RL, to avoid systems reaching some very undesirable outcomes that might be negative for the human user for example. This did not seem like a trivial problem since rewards are sparse in most real-life scenarios, and hence dead-ends do not have a negative reward provided. However, we have observed that both methods, the dead-end discovery by Fatemi et al. and Q-learning, accurately identified dead-ends. Particularly, Q-learning did the best job in this environment.

Nevertheless, we believe that this might not hold if the boundaries between the recovery states and the dead-ends are not as clearly separated as in this environment. We hypothesize that Fatemi's method might work better. We leave this line of work as future work. However, as Fatemi et al mentioned in the paper, their network learned gives

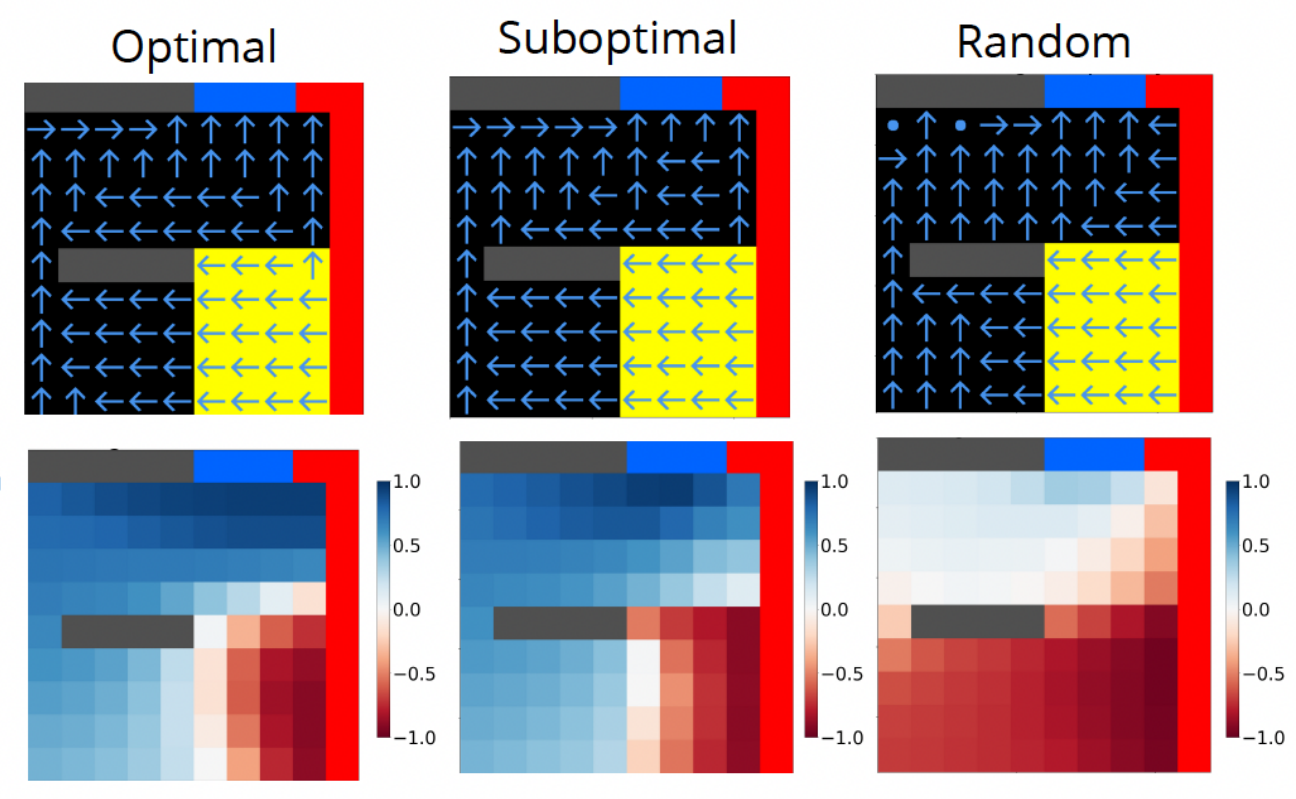

Figure 8. Policy (top) and value function (bottom) using Fitted-Q iteration method for each one of the data collection policies: optimal, suboptimal and random.

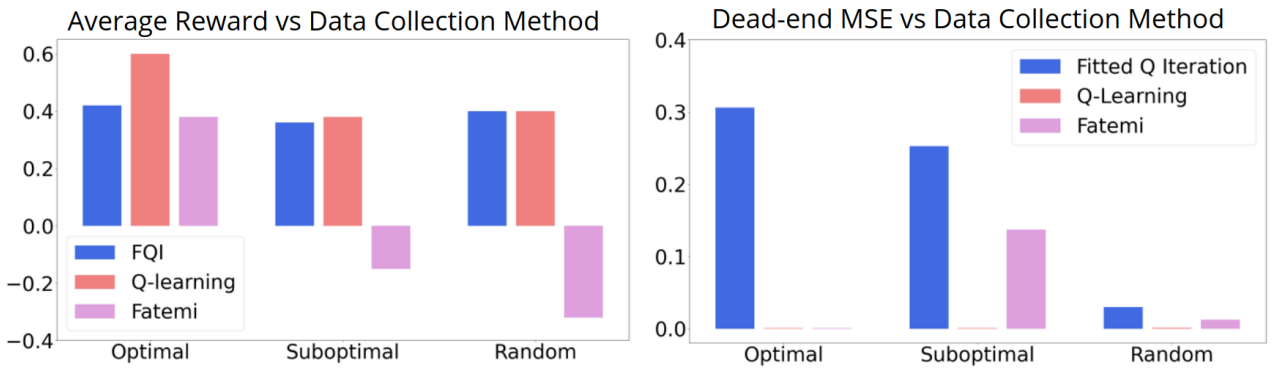

Figure 9. (left) Average reward for the final policy extracted from the each data collection method. (right) The MSE between the optimal value function for dead-ends and the value function extracted by

us probability distributions on the probability of arriving at a dead end after a transition. This is not the case in Q-learning, or we have not been able to prove it yet. For this reason, we can see that having access to these transition probabilities to get to dead-ends could be helpful in some scenarios. Q-learning will blindly inform us that some state is a dead end, but we will not be able to assess the quality of intermediary transitions. Finally, we observed that Fitted Q learning struggles to learn the optimal policy because the models tend to aggregate states. The model predicts that taking up actions when the agent is in the top right corner is optimal 8. This is because there is limited data for actions

taken in the top right corner, as seen in 4. This poses an interesting question if Fitted Q iteration should be used in high-risk environments, especially environments that are very non-smooth. Our recommendation is to only use it as a tool to aid clinicians and not as the only factor.

References

[EGW05] Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 04 2005.

[FKSG21] Mehdi Fatemi, Taylor W Killian, Jayakumar Subramanian, and Marzyeh Ghassemi. Medical dead-ends and learning to identify high-risk states and treatments. *Advances in Neural Information Processing Systems*, 34, 2021.

[LCLS10] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. *CoRR*, abs/1003.0146, 2010.

[MKS+13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[SB18] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[ZWH22] Pulkit Agrawal Zhang-Wei Hong. Lecture notes of 6.484 computational sensorimotor learning. *MIT 6.484*, 1, 2022.