Policy evaluation in offline reinforcement learning with low-rank structure

by

Komal Krishna Mogilipalepu

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Electrical Engineering

Program of Study Committee:
Namrata Vaswani, Major Professor
Shana Moothedath
Sung Yell Song

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this thesis. The Graduate College will ensure this thesis is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2025

Copyright © Komal Krishna Mogilipalepu, 2025. All rights reserved.

DEDICATION

I would like to dedicate this thesis to my grand mother Saraswathi Thorlikonda (late).

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

First, I am thankful to my advisor Professor Namrata Vaswani and Doctor Shana Moothedath for giving me an opportunity to work on this project. Prof. Namrata's advice to work on the implementation part stands as a novelty for this work, and her approach to research shaped me as an independent researcher and helped me to complete this thesis. Dr. Shana's suggestion to understand the proofs of the work helped me in the implementation of this work. Prof. Namrata and Dr. Shana's questions on the work during the discussions helped me to gain much clearer understanding of the topic. Prof. Namrata's course on high dimensional probability and linear algebra taught me to derive and understand the rigorous probably approximately correct algorithms that helped me to understand the theoretical literature of this work. Dr. Shana's course taught me the basics of game theory and reinforcement learning. Apart from the work, one quality that I learned from my advisor Prof. Namrata is being simple even though how much we achieve and one quality that I learned from Dr. Shana is working consistently without give up. Once again, thank you Prof. Namrata and Dr. Shana for guiding me professionally and personally.

Secondly, I would like to thank the faculty Dr. Sung Yell Song and Dr. Aditya Ramamoorthy, whose courses helped me in this work. Dr. Sung Yell Song course on applied linear algebra helped me to code the linear algebra concepts and that is the base for this outcome. Dr. Aditya Ramamoorthy's teaching has given me a good understanding on the basics of convex optimization and helped me to derive update equations for the optimization problem of this work. Further, I would like to thank the instructors Dr. pavan Kumar Aduri, Dr. Hongyang Gao, Dr. Jenna Bertilson, Dr. Cheng Huang, Dr. Ben Godard, and Dr. Febriana Lestari, whose courses taught me the required basics towards my MS degree.

On the other hand, I would like to thank my department Electrical and Computer Engineering for supporting me financially by providing the teaching assistant ship. I am thankful to all my friends and well wishers, who helped me in my difficult times either directly or indirectly. Finally, I would like to express regards to my family for their consistent support.

ABSTRACT

Evaluating the target policy, which is learned from an offline data that is generated from an unknown behavior policy, offline reinforcement learning, is crucial before deploying in the environment to mitigate the risk. Previous works provided the policy evaluation guarantees when all the state-action pairs visited by the target policy is observed in the offline data generated by the behavior policy. Recently, Xi et al. consider the practical scenario of the target policy to be evaluated visits new state-action pairs that are not observed by the behavior policy and provided the policy evaluation guarantees by assuming the low-rank structure on Markov decision process (MDP). Towards this, to bridge the theory and practice, this work provides a practically feasible target policy evaluation algorithm by using the fast and robust low-rank matrix completion algorithm Alternating Gradient Descent minimization (AltGDmin). First, we discuss the simple and effective AltGDmin algorithm, and show its performance by comparing with Altmin in the noise-free and noisy setting. Secondly, we simulate an uniform low-rank MDP with a disjoint support of target and behavior data, and implement the policy evaluation algorithm using AltGDmin. The simulation results show that the proposed policy evaluation algorithm has finite sample error bound. We compare the results of policy evaluation using AltGDmin with the Alternating minimization (Altmin) approach, the plots show that AltGDmin based policy evaluation outperform the Altmin one in every scenario.

0.1 References

Xi, X., Yu, C., and Chen, Y. Matrix estimation for offline evaluation in reinforcement learning with low-rank structure. In *3rd Offline RL Workshop: Offline RL as a"Launchpad"*.

CHAPTER 1. GENERAL INTRODUCTION

1.1 Notation

We denote $[S]$ as $[1, 2, 3, \ldots, S]$. The term $\mathbf{s} \times \mathbf{a}$ is the cartesian product of all possible state-action pairs. Similarly, $\mathbf{s} \times \mathbf{a} \times \mathbf{s}$ denotes the all possible state-action-state triplets. We consider $\mathbf{1}_{S\times A\times S}$ is an array of all ones of size $S \times A \times S$, similarly, $\mathbf{0}_{S\times A\times S}$ is the array of all zeros of size $S \times A \times S$. We denote $\mathbf{m}_k$ as the $k-$th column of matrix $\mathbf{M}$ and $\mathbf{U}_{\mathbf{m}_k,:}$ select the rows of matrix $\mathbf{U}$ based on the nonzero entries of the column vector $\mathbf{m}_k$. $\|\mathbf{A}\|_2$ denote the maximum singular value of $\mathbf{A}$. $\langle \mathbf{A}, \mathbf{B} \rangle$ denotes sum of all the elements of the entry wise product of $\mathbf{A}$, $\mathbf{B}$.

1.2 Introduction

In online reinforcement learning, the agent updates a policy through continuous interaction with the environment, whereas in offline reinforcement learning, the agent learns a target policy from an offline dataset that is generated by an unknown behavior policy interacting with the environment. Usually, the offline data contains the all possible state-action pairs of a dynamical system, however, in the real world there is a high possibility for target policy to encounter new state-action pairs that are not observed in the offline data. Therefore, the learned target policy needs to be evaluated on the outside of offline data distribution before deploying in the environment. This raises the problem of distribution shift in offline reinforcement learning (Levine et al., 2020). Towards this, previous works assumed full (Munos and Szepesvári, 2008) and partial (Uehara and Sun, 2021; Rashidinejad et al., 2021; Liu et al., 2020) coverage of the dataset to evaluate the learned target policy. The coverage of dataset is measured by concentrability coefficient (Uehara and Sun, 2021) $\mathcal{C}^{\mathbf{\Pi}} = \max_{s,a} \frac{\mathbf{D}^{\mathbf{\Pi}}(s,a)}{\widehat{\mathbf{D}}(s,a)}$, where $\mathbf{D}^{\mathbf{\Pi}}(s, a)$ is a state-action occupancy measure by a policy $\mathbf{\Pi}$ and $\widehat{\mathbf{D}}(s, a)$ is the empirical measure of a state-action pair observed in the offline data. The full and partial coverage means that $\mathcal{C}^{\mathbf{\Pi}} < \infty$

for all policies $\mathbf{\Pi}$ and $\mathcal{C}^{\mathbf{\Pi}^*} < \infty$ for an optimal policy $\mathbf{\Pi}^*$, respectively. Here, the finite concentrability coefficient states that the state-action pair visited by the target policy must be observed in the offline data. However, Xi et al. considers the scenario of infinite concentrability coefficient and evaluates the learned target policy $\mathbf{\Pi}$ with the assumption of low-rank Markov decision process (MDP). In particular, Xi et al.'s work assumed the low-rank state-action ($\mathbf{Q}$) value matrix, proposed a convex approach to estimate the $\mathbf{Q}$ values off the support and provided the finite sample total expected reward error bound. However, for large size matrix completion, convex methods takes more time to get the desired results and this is addressed using the alternating minimization based approach with provable guarantees by Jain et al. (2013). Towards this, Nayer and Vaswani (2022) introduced Alternating gradient descent based minimization (AltGDmin) algorithm for the low rank column wise compressive sensing (LRcCS) problem and shown the provable guarantees. Recently, Abbasi et al. (2023) applied AltGDmin to noise-free low-rank matrix completion problem and shown the results in a federated setting. However, in the policy evaluation the entries of a low-rank $\mathbf{Q}$ matrix are the non-uniform noisy observations where the noise is inherited from estimating the transition kernel. The state-action pair that is sampled more frequently leads to a better estimate of the transition probability and less sampled one has a poor estimate of the transition probability. It means that the noise is inherited in the sampling process that leads to non-uniform noisy matrix completion problem. To address this, in this work, first we show that, AltGDmin is robust and efficient for low-rank matrix completion in noise-free and noisy (uniform) cases, and further demonstrate that the target policy evaluation algorithm using AltGDmin has finite sample error bound on a simulated dataset that is sampled from low-rank MDP with uniform transition. The experimental results convey that AltGDmin based policy evaluation algorithm is able to effectively estimate the off support entries of a $\mathbf{Q}$ matrix from the non-uniform noisy observations.

1.3 References

Abbasi, A. A., Moothedath, S., and Vaswani, N. (2023). Fast federated low rank matrix completion. In *2023 59th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1–6.

Jain, P., Netrapalli, P., and Sanghavi, S. (2013). Low-rank matrix completion using alternating minimization. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 665–674.

Levine, S., Kumar, A., Tucker, G., and Fu, J. (2020). Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*.

Liu, Y., Swaminathan, A., Agarwal, A., and Brunskill, E. (2020). Provably good batch off-policy reinforcement learning without great exploration. *Advances in neural information processing systems*, 33:1264–1274.

Munos, R. and Szepesvári, C. (2008). Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, 9(5).

Nayer, S. S. and Vaswani, N. (2022). Fast low rank column-wise compressive sensing. In *2022 IEEE International Symposium on Information Theory (ISIT)*, pages 3285–3290. IEEE.

Rashidinejad, P., Zhu, B., Ma, C., Jiao, J., and Russell, S. (2021). Bridging offline reinforcement learning and imitation learning: A tale of pessimism. *Advances in Neural Information Processing Systems*, 34:11702–11716.

Uehara, M. and Sun, W. (2021). Pessimistic model-based offline reinforcement learning under partial coverage. *arXiv preprint arXiv:2107.06226*.

Xi, X., Yu, C., and Chen, Y. Matrix estimation for offline evaluation in reinforcement learning with low-rank structure. In *3rd Offline RL Workshop: Offline RL as a"Launchpad"*.

CHAPTER 2. LOW-RANK OFFLINE REINFORCEMENT LEARNING

Komal Krishna Mogilipalepu

Department of Electrical Engineering, Iowa State University, Ames, Iowa 50021, USA

Algorithm 1: AltGDmin

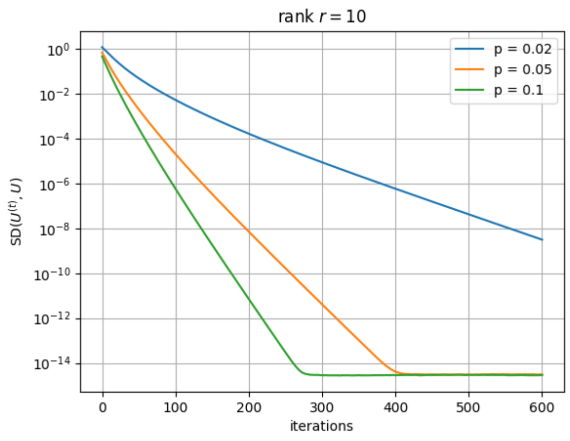
```
Input: observed matrix $\mathbf{Y}$, mask $\mathbf{M}$, rank $r$, stepsize $\eta$, and iterations $I$.
Output: $\mathbf{U}^{(I)}$, $\mathbf{B}^{(I)}$
1 Initialize $\mathbf{U}^{(0)}$ by first $r$ left singular vectors of $\mathbf{Y}$
2 for $i \leftarrow 1$ to $I$ do
3     $\mathbf{B}^{(i)}_{:,k} = (\mathbf{U}^{(i-1)^{\mathrm{T}}}_{\mathbf{m}_k,:}\mathbf{U}^{(i-1)}_{\mathbf{m}_k,:})^{-1}\mathbf{U}^{(i-1)^{\mathrm{T}}}_{\mathbf{m}_k,:}\mathbf{Y}_{\mathbf{m}_k,k}\ \forall\ k \in [q]$
4     $\mathbf{U} \leftarrow \mathbf{U}^{(i-1)} - \eta * 2(\mathbf{M} \circ (\mathbf{U}^{(i-1)}\mathbf{B}^{(i)}) - \mathbf{Y})\mathbf{B}^{(i)^{\mathrm{T}}}$
5     $\mathbf{U}^{(i)} \leftarrow \mathrm{QR}(\mathbf{U})$
6 end
```

2.1 Low-Rank Matrix Completion

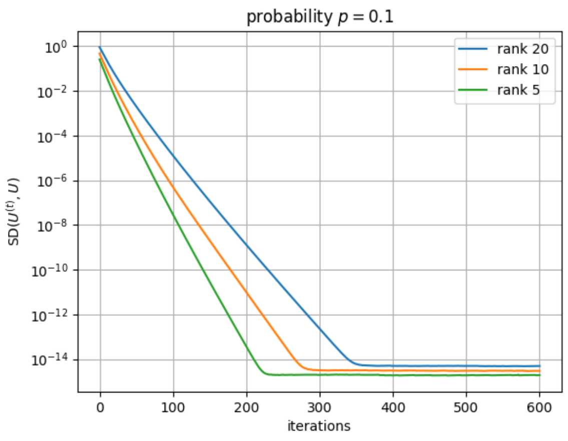
We compose a low-rank matrix $\mathbf{X} = \mathbf{U}\mathbf{B}$, where $\mathbf{U} \in \mathbb{R}^{n\times r}$ is generated by orthonormalizing the columns of a $n \times r$ matrix sampled from standard Gaussian $\mathcal{N}(0_r, \mathbf{I}_{r\times r})$ and $\mathbf{B} \in \mathbb{R}^{r\times q}$ directly sampled from standard Gaussian $\mathcal{N}(\mathbf{0}_r, \mathbf{I}_{r\times r})$. We generate a noisy matrix $\mathbf{N} \in \mathbb{R}^{n\times q}$ with each entry sampled from univariate Gaussian $\sigma\mathcal{N}(0, 1)$ with standard deviation σ. We obtain $\mathbf{Y}$ by observing the each element of $\mathbf{X} + \mathbf{N}$ with a probability p and the task is to retrieve $\mathbf{X}$ from $\mathbf{Y}$ by solving the following optimization problem

$$\min_{\mathbf{B},\mathbf{U}^{\mathrm{T}}\mathbf{U}=\mathbf{I}} \|\mathbf{Y} - \mathbf{M} \circ \mathbf{U}\mathbf{B}\|_F^2 \tag{2.1}$$

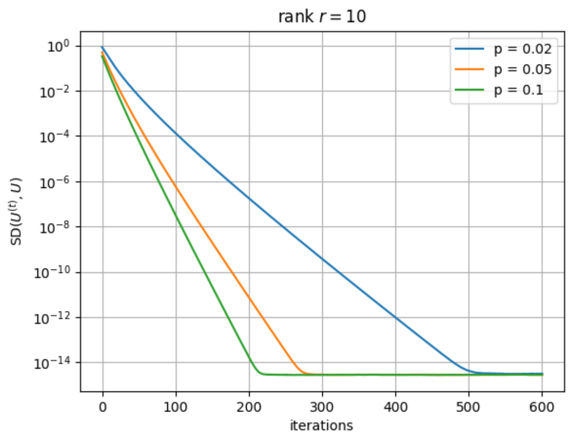
where $\mathbf{M} \in \{0, 1\}^{n\times q}$ is a Bernoulli matrix used for obtaining $\mathbf{Y} := \mathbf{M} \circ (\mathbf{X} + \mathbf{N})$. We solve (2.1) by alternating gradient descent minimization (AltGDmin) (Abbasi et al., 2023) with stepsize $\eta = \frac{cp}{\|\mathbf{Y}\|_2^2}$ as given in Algorithm 1. We compare the performance of AltGDmin with alternating minimization (Altmin) (Jain et al., 2013), which implement the least squares instead of gradient descent for updating $\mathbf{U}$ in Algorithm 1.



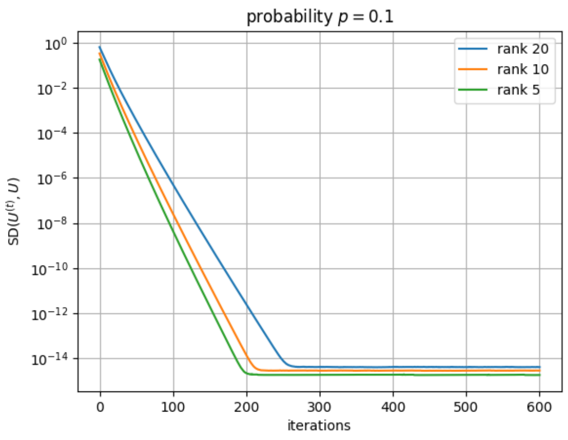
(a) rank $r = 10$, and $p = 0.02$, 0.05, and 0.1



(b) rank $r = 5$, 10, and 20, and $p = 0.1$

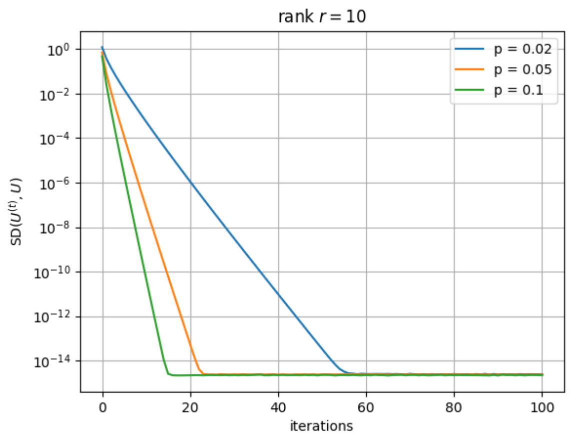


(c) rank $r = 10$, and $p = 0.02$, 0.05, and 0.1

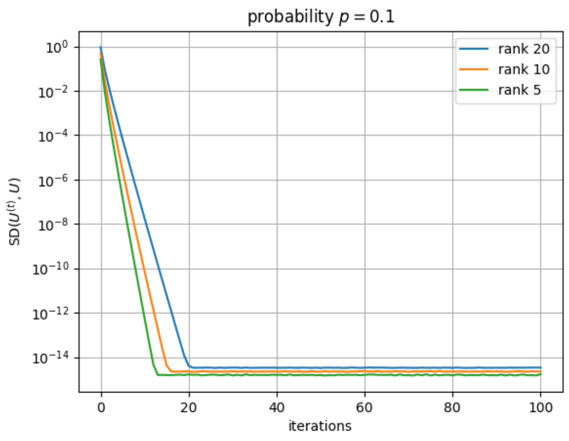


(d) rank $r = 5$, 10, and 20, and $p = 0.1$

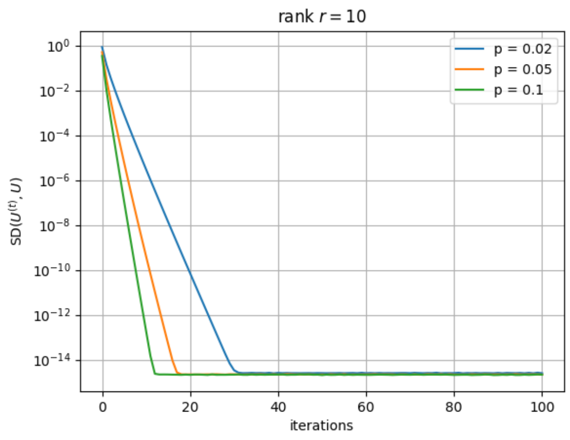
Figure 2.1: $(3000, 5000)$ and $(5000, 10000)$ low-rank matrix recovery using AltGDmin



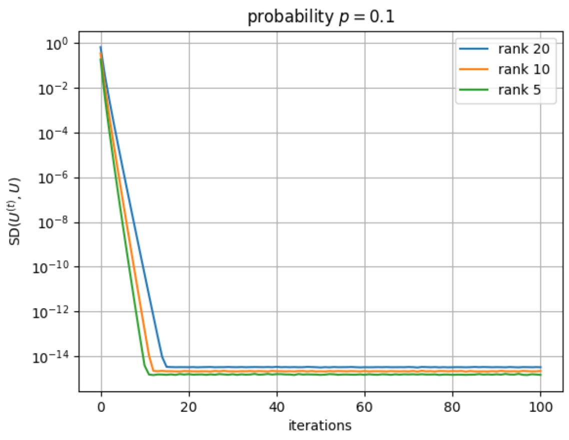
(a) rank $r = 10$, and $p = 0.02$, 0.05, and 0.1



(b) rank $r = 5$, 10, and 20, and $p = 0.1$

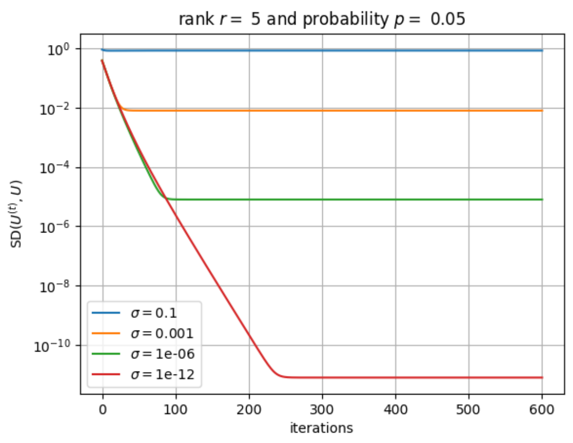


(c) rank $r = 10$, and $p = 0.02$, 0.05, and 0.1

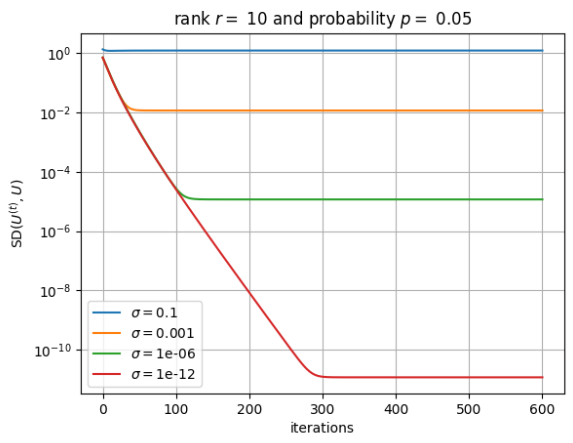


(d) rank $r = 5$, 10, and 20, and $p = 0.1$

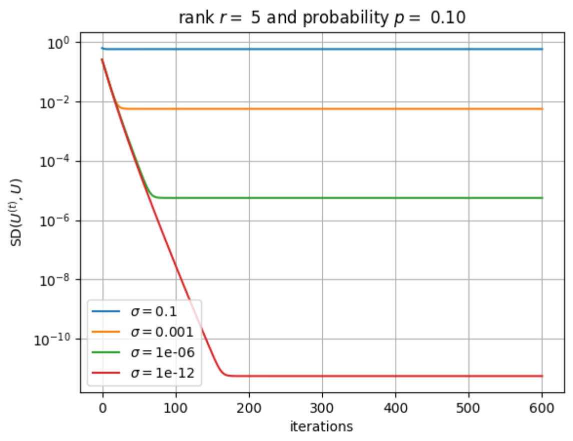
Figure 2.2: $(3000, 5000)$ and $(5000, 10000)$ low-rank matrix recovery using Altmin



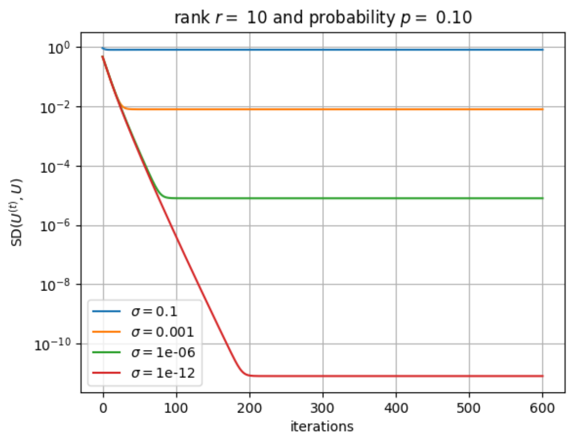
(a) rank $r = 5$ and $p = 0.05$



(b) rank $r = 10$ and $p = 0.05$,

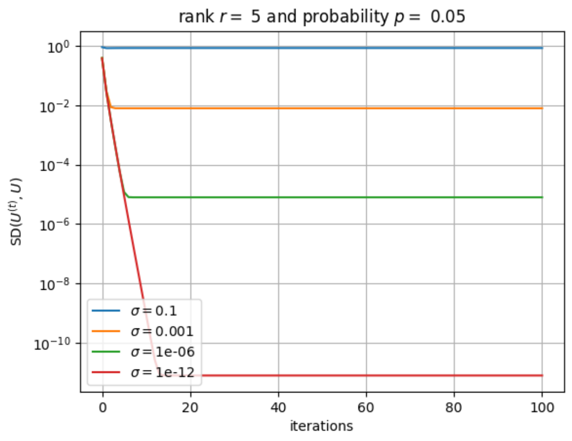


(c) rank $r = 5$ and $p = 0.1$

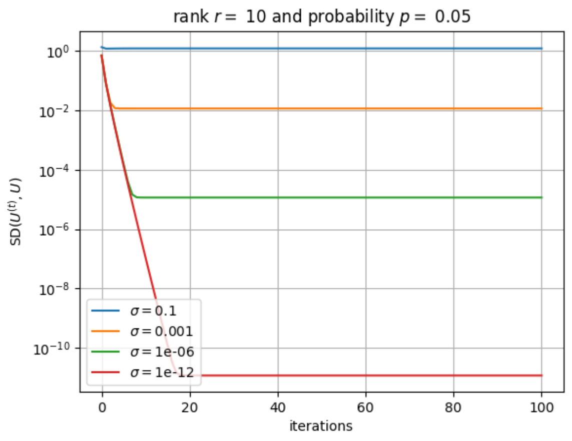


(d) rank $r = 10$ and $p = 0.1$,

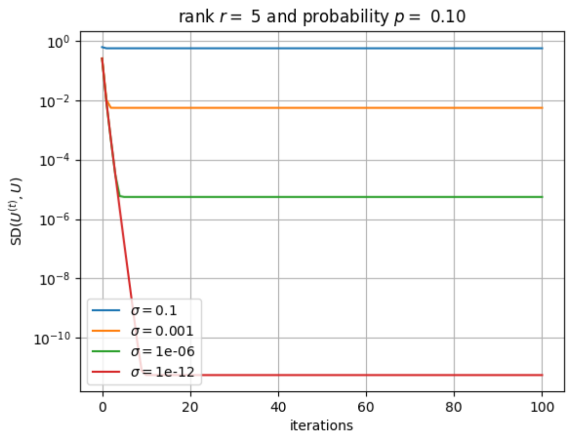
Figure 2.3: $(3000, 5000)$ low-rank matrix recovery from noisy observations using AltGDmin



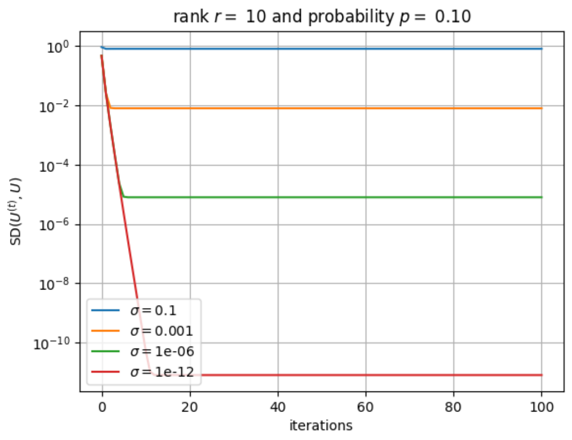
(a) rank $r = 5$ and $p = 0.05$



(b) rank $r = 10$ and $p = 0.05$,

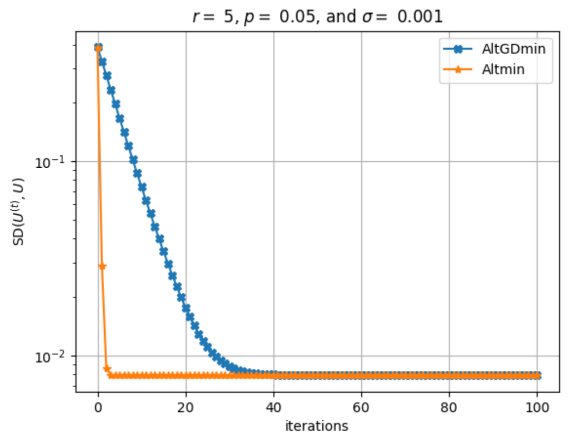


(c) rank $r = 5$ and $p = 0.1$

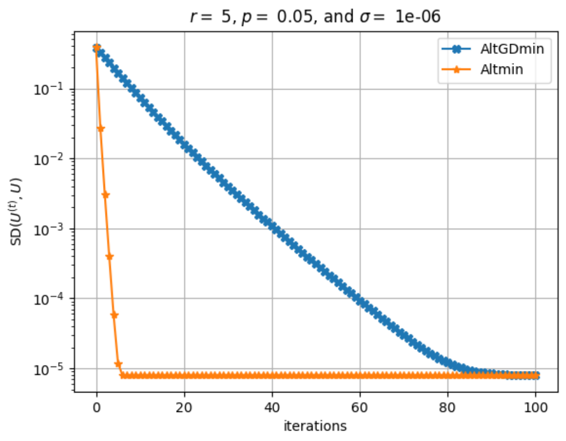


(d) rank $r = 10$ and $p = 0.1$,

Figure 2.4: $(3000, 5000)$ low-rank matrix recovery from noisy observations using Altmin



(a) $r = 5$, $p = 0.05$, and $\sigma = 0.001$



(b) $r = 5$, $p = 0.05$, and $\sigma = 1e - 06$

Figure 2.5: Comparing AltGDmin and Altmin for (3000, 5000) noisy LRMC

2.1.1 Performance of AltGDmin and Altmin

First, we show the performance of AltGDmin by considering $n = 3000$ and 5000, $q = 5000$ and 10000, and $\eta = \frac{cp}{\|\mathbf{Y}\|_{\text{op}}^2}$, where $\|\mathbf{Y}\|_{\text{op}}$ is the maximum singular value of $\mathbf{Y}$ and we consider $c = 0.1$, with variables rank r, probability p, and standard deviation σ. We use the subspace distance between the column spans of $\mathbf{U}^{(I)}$ and $\mathbf{U}$ defined as $\mathrm{SD}(\mathbf{U}^{(I)}, \mathbf{U}) = \|(\mathbf{I} - \mathbf{U}^{(I)}\mathbf{U}^{(I)^{\mathrm{T}}})\mathbf{U}\|_F$ as a metric to measure the AltGDmin performance in the noise-free ($\sigma = 0$) and noisy ($\sigma \neq 0$) settings. In the noise-free case, Figure 2.1 and 2.2 convey that both AltGDmin and Altmin almost have the same subspace distance measure as 10^{-15} in every setting, however, Altmin takes around 15-20 times lesser iterations than AltGDmin. For instance, Figure 2.1a, 2.1c and 2.2a, 2.2c shows that for a fixed rank, both AltGDmin and Altmin converge to almost 10^{-15} subspace distance with fewer iterations as they see more observations that is increasing p from 0.02 to 0.1, similarly Figure 2.1b, 2.1d and 2.2b, 2.2d shows that for a fixed p, recovering a matrix with lower rank has lowest error and faster convergence than the higher rank one for both AltGDmin and Altmin. Further, in the noisy scenario, Figure 2.3 and 2.4 tells that, in every setting, both AltGDmin and Altmin shows better recovery of the low-rank matrix as the noise level σ reduces. For example, Figure 2.3a, 2.3c, 2.4a, 2.4c and Figure 2.3b, 2.3d, 2.4b, 2.4d align with the point of converging with fewer iterations as the algorithm see more observations and lowest subspace distance measure for lower rank for both AltGDmin and Altmin. Further, we compare AltGDmin and Altmin in the noisy scenario as given in Figure 2.5 and it tells that Altmin takes fewer iterations to converge than AltGDmin for $c = 0.1$.

Algorithm 2: Markov decision process

```
1 MDP($S, A, H$)
2 $\mathbf{s} = [S]$, $\mathbf{a} = [A]$
3 $\boldsymbol{\mu} = [S]/S$
4 reward $\mathbf{R} = (\text{uniform}(0,1)_{S\times 1}\mathbf{1}^T_{1\times A})$
5 Initialize state transition kernel $\mathbf{P} = [\ ]$
6 for $t \leftarrow 1$ to $H$ do
7     Append $\mathbf{1}_{S\times A\times S} * \frac{1}{S}$ to $\mathbf{P}$
8 end
9 return $(\mathbf{s}, \mathbf{a}, \mathbf{P}, \mathbf{R}, \boldsymbol{\mu})$
```

2.2 Markov Decision Process with Low-Rank Structure

Consider a finite discrete Markov decision process containing S states $s \in \mathbf{s}$, A actions $a \in \mathbf{a}$, state transition kernel $\mathbf{P}$, reward function $\mathbf{R}$, initial state distribution $\boldsymbol{\mu}$, and horizon H as a tuple $(\mathbf{s}, \mathbf{a}, \mathbf{P}, \mathbf{R}, \boldsymbol{\mu}, H)$. The dynamics of an environment, described by MDP, is characterized by its state transition kernel $\mathbf{P} = \{\mathbf{P}_t\}_{t\in[H]}$ and its immediate reward $\mathbf{R} = \{\mathbf{R}_t\}_{t\in[H]}$, where $\mathbf{P}_t : \mathbf{s}\times\mathbf{a}\times\mathbf{s} \to [0,1]$ and $\mathbf{R}_t : \mathbf{s}\times\mathbf{a} \to [0,1]$ are the state transition probability and reward matrices in the t-th step, respectively. The policy $\boldsymbol{\Pi} : \mathbf{s}\times\mathbf{a} \to [0,1]$, a row stochastic matrix, is the probability of choosing an action a in state s. The state-action value in the t-th step according to policy $\boldsymbol{\Pi}$ is given by $\mathbf{Q}^{\boldsymbol{\Pi}}_t(s,a) = \mathbb{E}_{\boldsymbol{\Pi}}[\sum_{i=t}^{H} \mathbf{R}_i(s_i,a_i)|s_t = s, a_t = a]$, and the total expected reward $J^{\boldsymbol{\Pi}} = \mathbb{E}_{\boldsymbol{\Pi}}[\sum_{t=1}^{H} \mathbf{R}_t(s_t,a_t)|s_1 \sim \boldsymbol{\mu}]$. The low-rank assumption on MDP states that the transition kernel has a decomposition across the current state s and action a as $\mathbf{P}_t(s,a,s') = \mathbf{u}_t(s',s)^{\mathrm{T}}\mathbf{w}_t(a)$ or $\mathbf{P}_t(s,a,s') = \mathbf{u}_t(s)^{\mathrm{T}}\mathbf{w}_t(s',a)$ $\forall\ \ t, s', s, a$, where $\mathbf{u}_t(\cdot)$ and $\mathbf{w}_t(\cdot)$ are unknown functions, which maps the input to a $d/2$ length vector. We also assume that reward at t-th step $\mathbf{R}_t \in [0,1]_{S\times A}$ has rank at most $d/2$. The low-rank decomposition of state transition kernel $\mathbf{P}_t$ across current state s and action a lead to a low-rank $\mathbf{Q}$ matrix with rank at most d. The assumption of $\mathbf{Q}$ matrix being low-rank alleviates finding the unknown functions $\mathbf{u}_t(\cdot)$ and $\mathbf{w}_t(\cdot)$. With the assumption of low-rank $\mathbf{Q}$ matrix, given a target policy $\mathbf{\Pi}^\theta$ and an offline dataset $\mathcal{D}$, our goal is to evaluate the target policy on the offline data and estimate the total expected reward $J^{\mathbf{\Pi}^\theta}$.

Algorithm 3: dataset generation

```
Input: number of states $S$ and actions $A$, number of trajectories $K$, horizon $H$, action
       subset size $m$.
Output: dataset $\mathcal{D}$
1  Initialize dataset $\mathcal{D} = [\,]$
2  $(\mathbf{s}, \mathbf{a}, \mathbf{P}, \mathbf{R}, \boldsymbol{\mu}) = \texttt{MDP}(S, A, H)$
3  for $k \leftarrow 1$ to $K$ do
4      Initialize trajectory $\tau = [\,]$
5      $s_0 \sim \boldsymbol{\mu}(\mathbf{s})$
6      for $t \leftarrow 1$ to $H$ do
7          $a \sim \text{uniform}(\mathbf{a}_t^\theta),\ \text{where } |\mathbf{a}_t^\theta| = m,\ \text{and } \mathbf{a}_t^\theta \subseteq \mathbf{a}$
8          $r = \mathbf{R}_t(s_0, a)$
9          $s_1 \sim \mathbf{P}_t(\mathbf{s}|s_0, a)$
10         Append $(s_0, a, r, s_1)$ to $\tau$
11         $s_0 \leftarrow s_1$
12     end
13     Append trajectory $\tau$ to dataset $\mathcal{D}$
14 end
```

2.3 Policy Evaluation

2.3.1 Offline Dataset Generation

Consider an uniform transition model with $|\mathbf{s}| = |\mathbf{a}| = n$ that is at every step and under any policy $\mathbf{\Pi}$ the distribution on each state $\boldsymbol{\mu}_t^{\mathbf{\Pi}}(s) = 1/n\ \forall\ s \in \mathbf{s}$, also for every t and s, target policy $\mathbf{\Pi}_t^\theta(\cdot|s)$ selects an action uniformly at random from the subset $\mathbf{a}_t^\theta \subseteq \mathbf{a}$, where $\mathbf{a}_t^\theta$ is itself sampled uniformly from all the subsets of size m. Similarly, the behavior policy $\mathbf{\Pi}_t^\beta$ also follows the same policy model to select an action. The randomization involved in uniformly selecting an action creates an infinite concentrability coefficient because with high probability the state-action pairs generated by a target policy $\mathbf{\Pi}_t^\theta$ are different from that of the state-action pairs observed by a

Algorithm 4: policy evaluation through AltGDmin

```
Input: dataset $\mathcal{D}$, target policy $\mathbf{\Pi}^{\theta}$, initial state distribution $\boldsymbol{\mu}_1$, horizon $H$, number of
       states $S$ and actions $A$, rank $d$, stepsize $\eta$, and iterations $I$.
Output: estimated total expected reward $\widehat{J}$
1  Initialize $\widehat{\mathbf{Q}}^{\mathbf{\Pi}^{\theta}}_{H+1} \leftarrow \mathbf{0}_{S\times A}$
2  $K = \mathrm{len}(\mathcal{D})$
3  for $t \leftarrow H$ to 1 do
4      Initialize $\mathbf{N}_t \leftarrow \mathbf{0}_{S\times A}$, $\widehat{\mathbf{P}}_t \leftarrow \mathbf{0}_{S\times A\times S}$, and $\mathbf{R}_t \leftarrow \mathbf{0}_{S\times A}$
5      for $k \leftarrow 1$ to $K$ do
6          $(s,a,r,s') \quad = \mathcal{D}(k,t)$
7          $\mathbf{N}_t(s,a) \quad += 1$
8          $\widehat{\mathbf{P}}_t(s,a,s') += 1$
9          $\mathbf{R}_t(s,a) \quad = r$
10     end
11     $\widehat{\mathbf{P}}_t(s'|s,a) \leftarrow \widehat{\mathbf{P}}_t(s,a,s')/\mathbf{N}_t(s,a)\ \forall\ (s,a) \in \mathit{supp}(\mathbf{N}_t)$
12     $\mathbf{Z}_t(s,a) \quad = \mathbf{R}_t(s,a) + \sum_{s',a'} \widehat{\mathbf{P}}_t(s'|s,a)\mathbf{\Pi}^{\theta}_{t+1}(a'|s')\widehat{\mathbf{Q}}^{\mathbf{\Pi}^{\theta}}_{t+1}(s',a'),\ \forall\,(s,a) \in \mathit{supp}(\mathbf{N}_t)$
13     $\mathbf{M}_t \quad = \mathit{supp}(\mathbf{N}_t)$
14     $\mathbf{U}^{(I)}, \mathbf{B}^{(I)} = \texttt{AltGDmin}(\mathbf{Z}_t,\ \mathbf{M}_t,\ d,\ \eta,\ I)$
15     $\widehat{\mathbf{Q}}^{\mathbf{\Pi}^{\theta}}_{t} \leftarrow \mathbf{U}^{(I)}\mathbf{B}^{(I)}$
16 end
17 $\widehat{J} \leftarrow \sum_{s,a} \boldsymbol{\mu}_1(s)\mathbf{\Pi}^{\theta}_1(a|s)\widehat{\mathbf{Q}}^{\mathbf{\Pi}^{\theta}}_1(s,a)$
```

behavior policy $\mathbf{\Pi}^{\beta}_t$. Since for every state the policy has a uniform distribution on m actions, the state-action occupancy measure, $\mathbf{D}^{\mathbf{\Pi}^{\theta}}_t(s,a) = \boldsymbol{\mu}^{\mathbf{\Pi}^{\theta}}_t(s)\mathbf{\Pi}^{\theta}_t(a|s)$, has mn nonzero entries with each entry being $1/mn$. We simulate an MDP containing $S = A = n$ states and actions, uniform initial state distribution $\boldsymbol{\mu}_1$, rank-1 reward matrix $\mathbf{R}_t\ \forall t$, and rank-1 uniform probability transition tensor $\mathbf{P}_t\ \forall t$ as given in Algorithm 2. We sample a dataset of K trajectories each of length H as $\mathcal{D} = \{(s^k_t, a^k_t, r^k_t, s'^k_t)\}\ \forall\ t \in H, k \in K$ from the defined MDP and policy model as mentioned in Algorithm 3.

2.3.2 Algorithm

Given a dataset $\mathcal{D} = \{(s^k_t, a^k_t, r^k_t, s'^k_t)\}_{t\in[H],k\in[K]}$, target policy $\mathbf{\Pi}^{\theta}$, initial state distribution $\boldsymbol{\mu}_1$, rank of state-action value ($\mathbf{Q}^{\mathbf{\Pi}^{\theta}}_t$) matrix d, and stepsize η, Algorithm 4 estimate the total expected reward $\widehat{J}$ by finding the state-action values ($\mathbf{Q}$ matrix) backward from step $t = H$ to

$t = 1$. In each step, first the algorithm estimates the state-action occupancy count $\mathbf{N}_t$, state transition kernel $\widehat{\mathbf{P}}_t$, and immediate reward $\mathbf{R}_t$ from the t-th horizon of offline data over all trajectories K. After that on the support of $\mathbf{N}_t$, the state-action values $\mathbf{Z}_t$ are calculated using $\widehat{\mathbf{P}}_t$ based on the Bellman update. In the next step, to infer the $\mathbf{Q}$ value off the support, the algorithm solve the following optimization problem

$$\min_{\mathbf{B},\mathbf{U}^{\mathrm{T}}\mathbf{U}=\mathbf{I}} \|\mathbf{M}_t \circ (\mathbf{Z}_t - \mathbf{U}\mathbf{B})\|_F^2 \tag{2.2}$$

where $\mathbf{M}_t \in \{0,1\}_{S\times A}$ is a matrix with 1 on the support of $\mathbf{N}_t$ and 0 elsewhere. We solve (2.2) using AltGDmin given in Algorithm 1. After H steps, the resultant state-action value $\widehat{\mathbf{Q}}_1^{\mathbf{\Pi}^\theta}$ is used to calculate the total expected reward $\widehat{J}$. To show the correctness of estimated $\widehat{J}$, we bound the absolute error term $|\widehat{J} - J^{\mathbf{\Pi}^\theta}|$. Where,

$$\widehat{J} = \sum_{(s,a)\in\mathbf{s}\times\mathbf{a}} \mathbf{D}_1^{\mathbf{\Pi}^\theta}(s,a)\widehat{\mathbf{Q}}_1^{\mathbf{\Pi}^\theta}(s,a) \tag{2.3}$$

$$J^{\mathbf{\Pi}^\theta} = \sum_{(s,a)\in\mathbf{s}\times\mathbf{a}} \mathbf{D}_1^{\mathbf{\Pi}^\theta}(s,a)\mathbf{Q}_1^{\mathbf{\Pi}^\theta}(s,a) \tag{2.4}$$

$$\implies |\widehat{J} - J^{\mathbf{\Pi}^\theta}| = |\langle \mathbf{D}_1^{\mathbf{\Pi}^\theta}, \widehat{\mathbf{Q}}_1^{\mathbf{\Pi}^\theta} - \mathbf{Q}_1^{\mathbf{\Pi}^\theta}\rangle| \tag{2.5}$$

The term $\langle \mathbf{D}_1^{\mathbf{\Pi}^\theta}, \widehat{\mathbf{Q}}_1^{\mathbf{\Pi}^\theta} - \mathbf{Q}_1^{\mathbf{\Pi}^\theta}\rangle$ can be expressed as follows:

$$\langle \mathbf{D}_1^{\mathbf{\Pi}^\theta}, \widehat{\mathbf{Q}}_1^{\mathbf{\Pi}^\theta} - \mathbf{Q}_1^{\mathbf{\Pi}^\theta}\rangle = \sum_{t=1}^{H} \langle \mathbf{D}_t^{\mathbf{\Pi}^\theta}, \widehat{\mathbf{Q}}_t^{\mathbf{\Pi}^\theta} - \mathbf{Y}_t\rangle \tag{2.6}$$

$$\implies |\widehat{J} - J^{\mathbf{\Pi}^\theta}| = |\sum_{t=1}^{H} \langle \mathbf{D}_t^{\mathbf{\Pi}^\theta}, \widehat{\mathbf{Q}}_t^{\mathbf{\Pi}^\theta} - \mathbf{Y}_t\rangle| \tag{2.7}$$

Proof for (2.6) is given in the appendix of Xi et al..

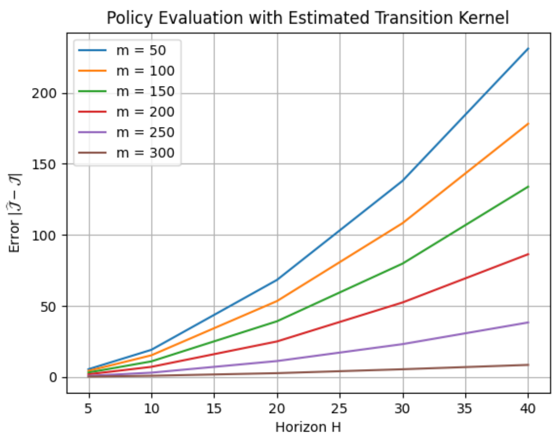
We compare the simulation results of AltGDmin based policy evaluation with the theoretical results of convex matrix completion based policy evaluation, in a finite sample setting (Xi et al.), which is given as follows:

$$|\widehat{J} - J^{\mathbf{\Pi}^\theta}| \le CH^2 \left(\sqrt{\frac{d \log(nH)}{m}} + \sqrt{\frac{dn \log(nH)}{K}} \right) \tag{2.8}$$

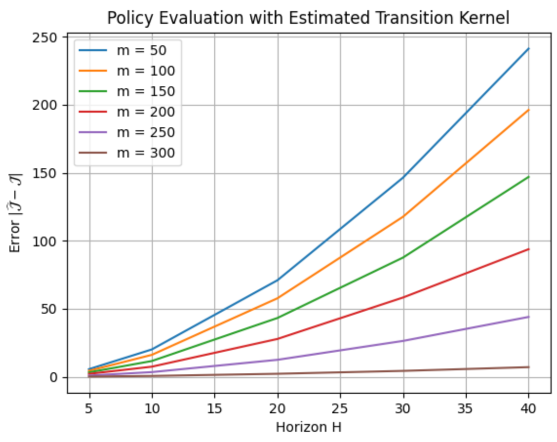
Where, C is a positive constant such that for the number of states $n \geq C$, (2.8) holds with probability of atleast $1 - \frac{1}{n}$. For $K >> dn\log(nH)$ the left term in the error equation of (2.8) dominates and the error can be written as:

$$|\widehat{J} - J^{\mathbf{\Pi}^{\theta}}| \leq CH^2\sqrt{\frac{d\log(nH)}{m}} \tag{2.9}$$

In this case, for a fixed rank d and action subset size m the error $|\widehat{J} - J^{\mathbf{\Pi}^{\theta}}| \propto H^2\sqrt{\log(nH)}$, which is the error varies approximately quadratic with horizon H. If the action subset size $m \geq \frac{H^2 d\log(nH)}{\epsilon^2}$ for some $\epsilon > 0$, then we have $|\widehat{J} - J^{\mathbf{\Pi}^{\theta}}| \leq \epsilon H$ that is the error varies linearly with horizon H. In addition, for a fixed horizon H, rank d, and number of states and actions n, the error $|\widehat{J} - J^{\mathbf{\Pi}^{\theta}}| \propto \frac{1}{\sqrt{m}}$, which means the error varies inversely with $\sqrt{m}$. We show that our simulation results corroborate the above three properties and infact the simulations tells that there exist a better upper bound than (2.9).

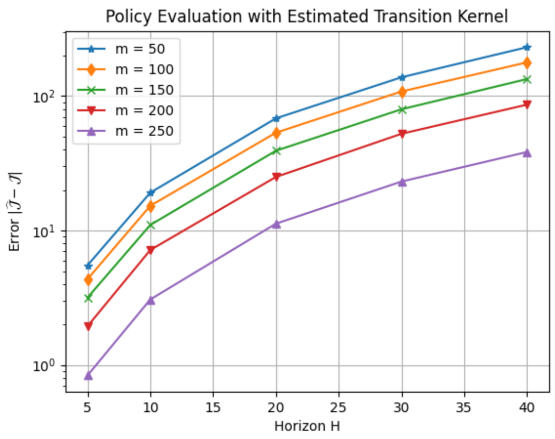


(a) states and actions $n = 300$ and $K = 5000$

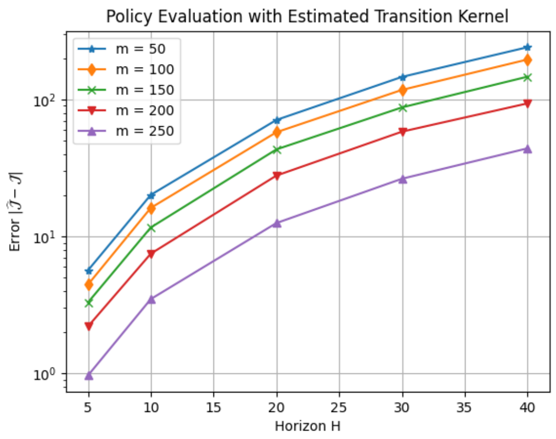


(b) states and actions $n = 300$ and $K = 10000$

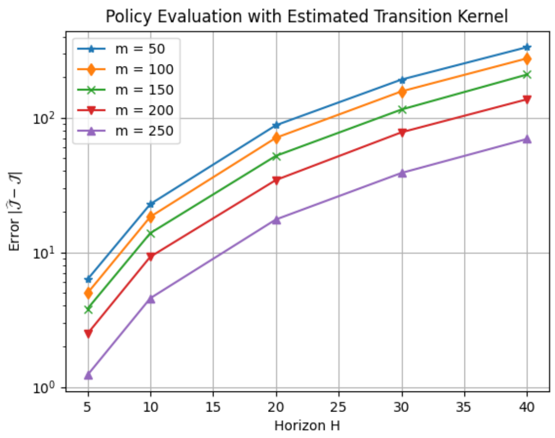
Figure 2.6: Target Policy Evaluation



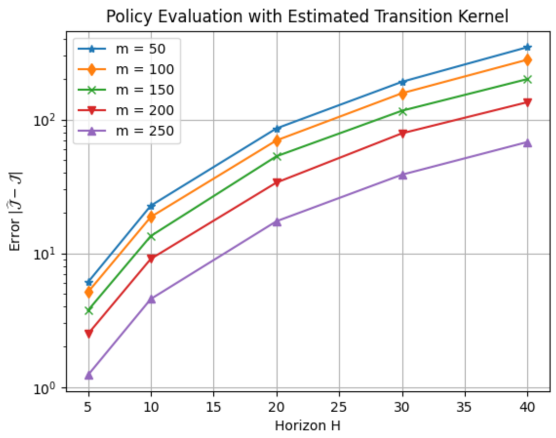
(a) states and actions $n = 300$ and $K = 5000$



(b) states and actions $n = 300$ and $K = 10000$

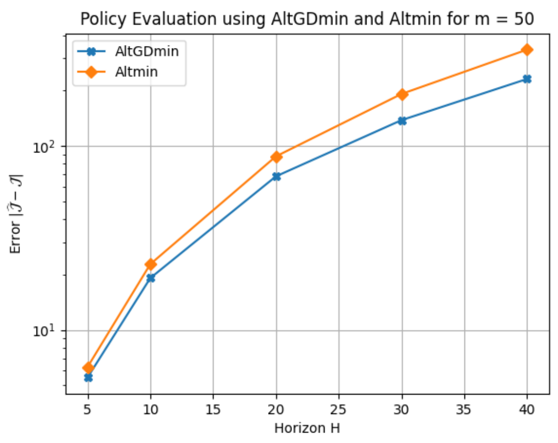


(c) states and actions $n = 300$ and $K = 5000$

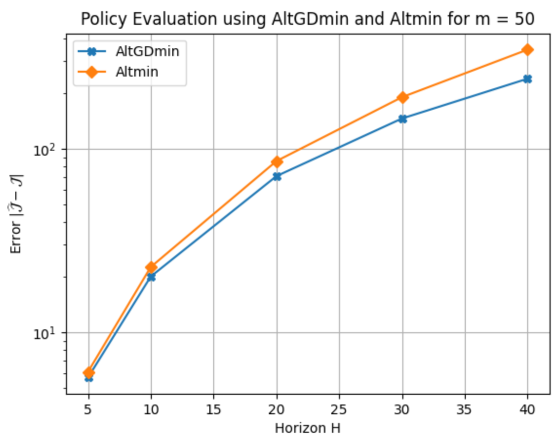


(d) states and actions $n = 300$ and $K = 10000$

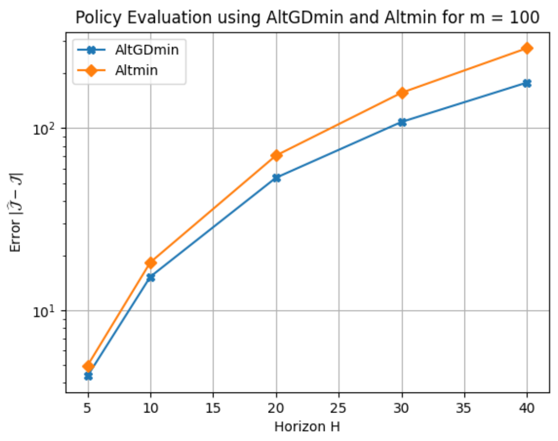
Figure 2.7: Target Policy Evaluation using AltGDmin and Altmin



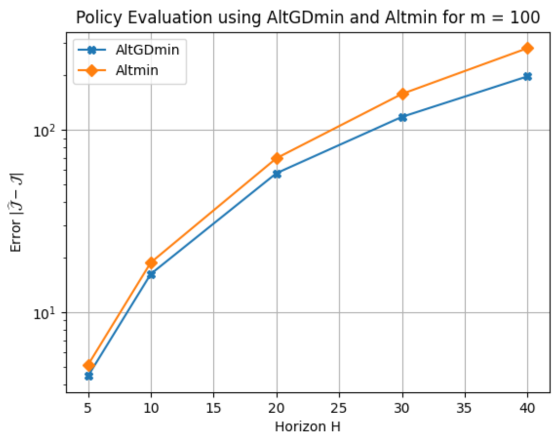
(a) states and actions $n = 300$ and $K = 5000$



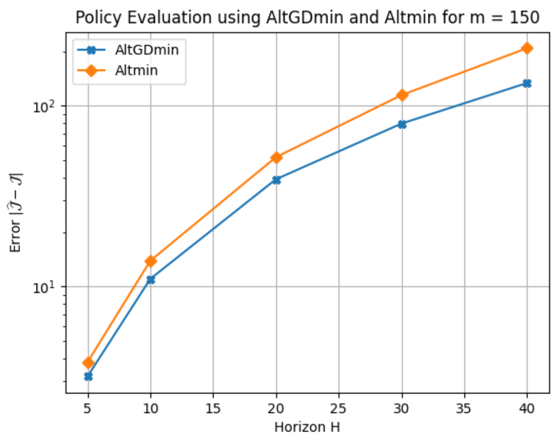
(b) states and actions $n = 300$ and $K = 10000$



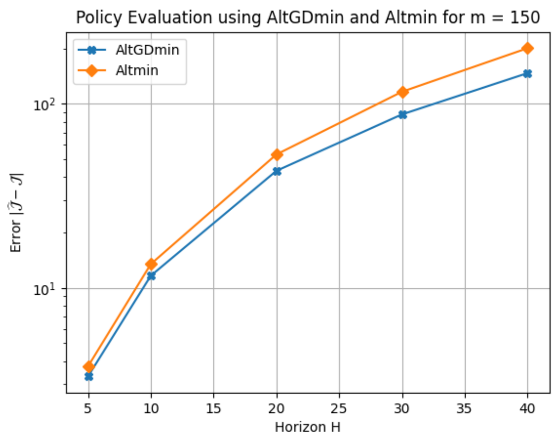
(c) states and actions $n = 300$ and $K = 5000$



(d) states and actions $n = 300$ and $K = 10000$

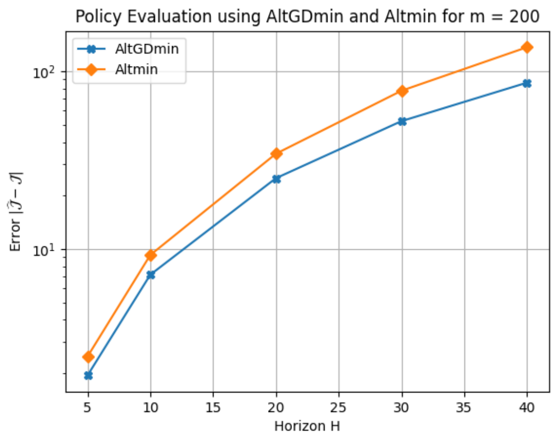


(e) states and actions $n = 300$ and $K = 5000$

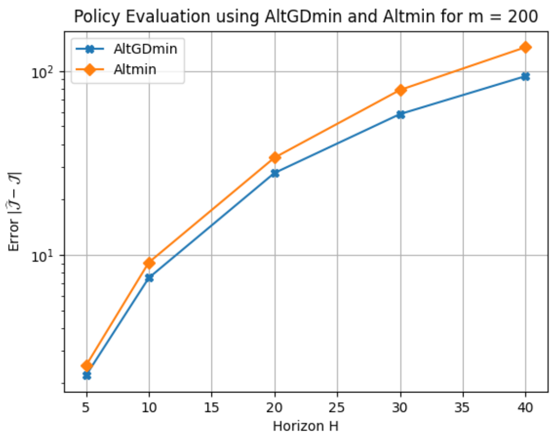


(f) states and actions $n = 300$ and $K = 10000$

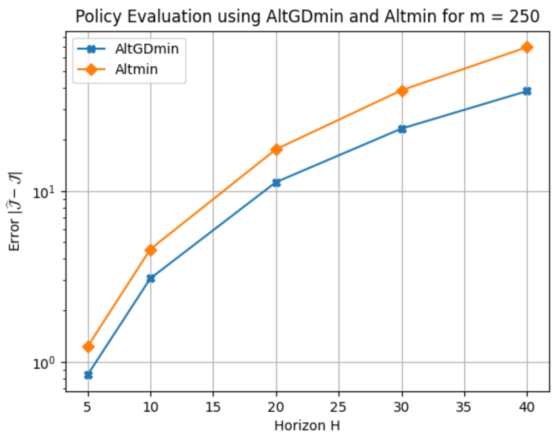
Figure 2.8: Target Policy Evaluation Error Comparison between AltGDmin and Altmin



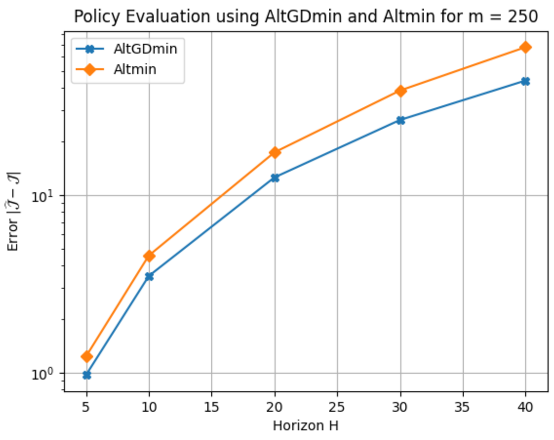
(a) states and actions $n = 300$ and $K = 5000$



(b) states and actions $n = 300$ and $K = 10000$



(c) states and actions $n = 300$ and $K = 5000$



(d) states and actions $n = 300$ and $K = 10000$

Figure 2.9: Target Policy Evaluation Error Comparison between AltGDmin and Altmin

2.3.3 Numerical Results on Policy Evaluation

Given a dataset $\mathcal{D}$, target policy $\mathbf{\Pi}^{\theta}$, initial state distribution $\boldsymbol{\mu}_{\mathbf{1}}$, horizon H, number of states and actions $S = A = n$, rank of the $\mathbf{Q}$ matrix d, stepsize η, and iterations I. We evaluate the target policy $\mathbf{\Pi}^{\theta}$ on the offline dataset $\mathcal{D}$ by finding the total expected reward $\widehat{J}$ as given in Algorithm 4 by considering $n = 300$, different action subset sizes $m = 50 \, \text{to} \, n$ with a stepsize of 50, $K = 5000 \, \text{and} \, 10000$, and horizon $H = 5, 10, 20, 30, \text{ and } 40$. Figure 2.6 convey that the error $|\widehat{J} - J|$ has inverse and direct relation with action subset size m and horizon H for $K = 5000$ and $K = 10000$, respectively. The error increases with horizon H for all values of action subset size m and as m increases there is more overlap between target and behavior data therefore error with respect to horizon decreases. For $m = 250$ the error increases linearly with horizon H and for $m = 300$ there is no unseen state-action pair in the target data that is not observed in the behavior data, hence the error increases by a small amount with horizon H. Figure 2.7 show that AltGDmin and Altmin based policy evaluation have finite sample error bound on a log scale. In addition, we compare AltGDmin and Altmin based policy evaluation for different values of action subset size m and trajectories $K = 5000$ and 10000. Figure 2.8 and 2.9 show that, in every setting, AltGDmin outperform the Altmin based policy evaluation.

2.4 References

Abbasi, A. A., Moothedath, S., and Vaswani, N. (2023). Fast federated low rank matrix completion. In *2023 59th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1–6.

Jain, P., Netrapalli, P., and Sanghavi, S. (2013). Low-rank matrix completion using alternating minimization. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 665–674.

Xi, X., Yu, C., and Chen, Y. Matrix estimation for offline evaluation in reinforcement learning with low-rank structure. In *3rd Offline RL Workshop: Offline RL as a"Launchpad"*.

CHAPTER 3. GENERAL CONCLUSION

We evaluate the target policy, in offline reinforcement learning, on the input data generated from an unknown behavior policy with infinite concentrability coefficient, which means the state-action pair visited by the target policy is not observed in the input data, by the assumption of low rank structure on Markov decision process. Particularly, with the assumption of low-rank $\mathbf{Q}$ matrix, we evaluate the target policy using a robust matrix completion algorithm AltGDmin by estimating the total expected reward $\widehat{J}$. To show that the proposed policy evaluation algorithm is practically feasible, we simulate an uniform low-rank MDP with infinite concentrability coefficient and compare the estimated $\widehat{J}$ with true total expected reward J. Numerical results show that, our algorithm has the finite sample error bound and corroborate with the theoretical guarantees provided in Xi et al.. In addition, we compare the AltGDmin based policy evaluation with Altmin one and the plots show that AltGDmin outperform the Altmin in every setting.

3.1 References

Xi, X., Yu, C., and Chen, Y. Matrix estimation for offline evaluation in reinforcement learning with low-rank structure. In *3rd Offline RL Workshop: Offline RL as a"Launchpad"*.