ONLINE LEARNING OF MARKOV JUMP DYNAMIC SYSTEMS

A Thesis

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Master of Science

by

Ritabrata Ray

August 2021

ABSTRACT

We study the online learning of Markovian Jump Linear Dynamical Systems. These are linear dynamical systems with several modes and which has different state dynamics matrices corresponding to each mode and the modes switch with time according to an underlying Markov Chain. In particular, we study the problem of LQR control for such Markovian Jump Linear Dynamical Systems when the matrices governing the state transitions corresponding to each mode (the mode parameters) are unknown and the Markov Chain's state transition matrix is also unknown but the number of modes of the system is known. In this setting, we propose an online learning algorithm for the system which is expected to suffer a sublinear (on the time horizon length) regret when compared against the optimal control of the system in expectation where all the system parameters and the Markov Chain is known. We also did numerical simulation of the algorithm learning a two-dimensional state-space and two-dimensional control-space with two modes and a simple Markov Chain between the two modes and compared it against the known optimal solution to the system and observed a sublinear regret.

BIOGRAPHICAL SKETCH

Ritabrata Ray was born in Ambika Kalna, a small town in the eastern state of West Bengal, India. A keen interest in Physics and Mathematics puzzles and theorem proving as well as participating in several Mathematical Olympiads during his high school made him choose the path of engineering. He did his undergradute studies at the Indian Institute of Technology, Kharagpur where he developed an interest in theoretical computer science. During his time at Kharagpur, he carried out research in the area of extremal combinatorics where he used linear algebraic techniques to prove theorems on families of sets satisfying certain intersection properties and also to families of finite-dimensional vector spaces defined over finite fields. During this period, he also did several research internships all of them in CS theory including working on a problem of software verification (MCMC sampling of traces) at the Max Planck Institute for Software Systems, Kaiserslautern, Germany and working in the area of theoretical machine learning (on finding the optimal regularizer for Matrix Factorization and general Tensor Factorisation problem) at EPFL, Switzerland. Towards the end of his undergraduate studies, he got interested in Machine Learning theory and to further pursue a research career in this area, he joined Cornell University as a graduate student in the Electrical and Computer Engineering department. Here he worked on the problem of Online Learning of Markovian Jump Dynamical Systems which is the subject of this thesis.

"Imagination is more important than knowledge." - Albert Einstein

"The first principle is that you must not fool yourself and you are the easiest person to fool." - Richard Feynman

ACKNOWLEDGEMENTS

I would like to thank my advisor Prof. Lang Tong for his continuous inputs and support throughout my thesis work. His advice and suggestions helped me develop my academic and interpersonal skills. I would also like to thank Prof. Qing Zhao for her advice and inputs, not only related to my project but also on what to do ahead in my career.

I would like to extend my gratitude to Prof. Jayadev Acharya and Prof. Siddhartha Banerjee for their invaluable advice related to my academic career. I would also like to thank Anirban Bhattacharjee, Soumyamouli Pal, Varinderjit Mann, Sounak Maji, Hrishikesh Danawe, Arkadev Roy, Ayush Sekhari, Devesh Khilwani, Sudeep Salgia, Kunal Pattanayak and Shubham Jadhav for their friendship and support.

I would finally like to thank my parents for their continuous support during my ups and downs and for encouraging me to continue following my dreams.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Markovian Jump Linear Dynamical Systems-Significance

Several real life systems could be modelled as systems with multiple modes of operation where in each mode they could be further modelled as linear dynamical systems. Although within each mode the system is linear, the mode switch introduces non-linearity. One such interesting model is the Markovian Jump Linear Dynamical Systems where the mode switches are Markovian that is the probability distribution for the next mode depends only on the previous mode. The underlying Markov Chain over the modes is assumed to be stationary.

These models capture several interesting real-life systems. For example, consider a failure prone production system as discussed in Example 4.1 of [1], where we have a constant demand of manufacturing d-items in every time instance and the goal is to meet this demand at all times. But the system being failure-prone is modelled as having two modes: the normal functioning mode and the failure mode. In the failure mode: no-production is possible whereas in the functioning mode, we could apply a rate of production u units per instant which is say at most a constant $r > d > 0$. Now for this system, we could have a one-dimensional state-space x which is equal to the net production till now - net demand till now and the control space is also the one-dimensional action u applied to it in the functional mode, then this could be modelled as two linear dynamical systems operating on the same

state and action spaces each corresponding to a mode of the system with its own parameters as follows:

$$x_{t+1} = \begin{cases} x_t + u_t - d, \text{if } \ r_t = 1 \ \text{ (functional mode)} \\ x_t - d, \qquad \text{if } \ r_t = 2 \ \text{ (failure mode)} \end{cases} \tag{1.1}$$

where r_t is the mode at time t which switch in a stationary Markovian sense that is if the probability of the machine transitioning to another mode depends only on its mode in the last time instance and these probabilities do not change with time . Now a large positive x would imply overproduction and wastage be seen as well as could be seen as causing stress on the environment whereas a highly negative x indicates inadequate production and likely customer dissatisfaction.Thus, it makes sense to penalize the square of x. Moreover large values of u could be seen as high production costs. As a result, we could model the cost at each time instance as a suitably weighted quadratic function of x and u, as follows:

$$c_t = ax_t^2 + bu_t^2, \tag{1.2}$$

for some suitably chosen constants $a, b > 0$.

Now the above example is a simple 1-dimensional instance of the standard LQR problem with Markovian Jump where the goal is to minimize the cumulative cost. A generalization of this to arbitrary finite dimensional state and action spaces is the setting considered in this thesis.

Here we consider the problem of learning to control such unknown systems. That is if we know that a given dynamical system has m (known to the learner) modes, but we do not know how the state space changes with the control and previous state nor do we know how the mode switch occurs, that is with unknown dynamics matrices for each mode and an unknown underlying Markov Chain, the problem is to learn the gradually learn the system parameters ensuring that the total cost incurred is close to the optimal value of the cost for the system when the dynamics matrices for each mode and the underlying Markov Chain is completely known. The learning is done in an online setting, where at each instance we sequentially choose an input (control) to the system and gradually learn the system to minimize the regret. The regret is defined as the difference between the cumulative cost suffered by the online learning algorithm and the optimal cumulative loss when the system is completely known. We seek to incur sub-linear regret, i.e., if the algorithm is run for T rounds, the total regret must grow strictly slower than any linear function of T, which would imply that when the online learning algorithm is run for very long duration, the average regret per instance goes to zero.

1.2 Related Work

Markov Jump models are widely used in communications, control and signal processing. [2] and [3] study some applications of Markov jump models in signal processing. Details about the theory of Markov jump linear systems could be found

in [4]. Markov Jump Linear Dynamical systems are studied in [1] where necessary and sufficient conditions for stability of such systems are characterized and the problem of optimal LQG control policy for such systems is also derived for a completely known system (known parameters and known Markov Chain Transition Matrix). We use this optimal LQG controller's performance as the benchmark to compare the performance of our online learning algorithm.

[5] considers the problem of online learning of Markov jump linear systems. Here, the system does not have any state space but the output is an affine function of the input (control) and the cost is square of the l_2-norm of the difference in the output and the constant target output. [5] obtains $\Theta(\sqrt{T})$ regret and show that it is the best possible regret. Our problem is a generalization of this problem where we also have a state space whose dynamics are governed both by the current state and the control in a linear way and the cost is a general quadratic function of both the state and the control (input).

The problem of online learning for LQR control has seen a huge interest in the recent years and has been the focus of several recent papers. [6] first considered the online LQR problem and obtained $O(\sqrt{T})$ regret but their guarantee was exponential in the dimension. Then, [7] reduced the dependence on the dimension to a polynomial factor. However, these algorithms were not computationally equivalent and [8] obtained the first computationally efficient algorithm which had $O(T^{2/3})$

regret and asked whether $O(\sqrt{T})$ regret is possible to achieve in a computationally efficient way. [9] found a positive answer to this with an SDP based algorithm which had an $\tilde{O}(\sqrt{T})$ regret where the $\sim$ sign hides logarithmic factors. More recently, [10] developed a simple exploration algorithm which obtained $\tilde{O}(\sqrt{T})$ regret and showed that the bound is tight by proving a matching lower bound.

The above algorithms address our problem when there is just one mode of the system, i.e., a totally linear dynamical system. More recently, there have been work (see [11]) on time-varying systems where the system parameters vary with time in an arbitrary manner whereas in our settings it changes according to a Markov Chain based switch on the mode. In the adversarial-time varying parameters setting, the regret is obtained against the best policy in any interval of time and this adaptive-regret bound is derived in terms of the net variation in the parameters over the interval.

1.3 Contributions

We propose a new SDP-based algorithm for online learning of the Markov Jump Dynamical System where we do LQR control with known cost matrices. The number of modes of the system is known but the parameters governing the state transitions for each mode as well as the underlying Markov Chain governing the mode switches are not known. Our algorithm is essentially an extension of [9]'s

algorithm. We run this algorithm for a system which has both two-dimensional system and action spaces and two-modes with an irreducible and aperiodic Markov chain governing the switches over these modes and observe a sublinear regret. The problem is of LQG control of this system where the regret is obtained by comparing against the optimal policy for the system when all the parameters for every mode as well as the transition probabilities of the Markov chain are known. We believe this algorithm also suffers a $\tilde{O}(\sqrt{T})$ regret and are currently attempting to prove this upper bound. We have a $\Omega(\sqrt{T})$ lower bound on the regret for this setting which can be seen directly from the regret lower bound for the simpler Markov Jump Linear System as shown in [5] which is a special case of the Markov Jump Linear Dynamical System setting that we consider here.

CHAPTER 2

BACKGROUND

2.1 Analytical Setting and preliminary definitions

We consider a system that has m modes of operation and at each time instant the modes switch according to an irreducible and aperiodic Markov chain with states corresponding to each mode of the system. The system's modes are a discrete time discrete space Markov process governed by this Markov chain where $r_t \in \{1, 2, \ldots, m\}$ is the system's mode at time t and time runs from $1, \ldots, T$ where T is the horizon length. The Markov chain's transition probability matrix is denoted as $P[r_{t+1} = j | r_t = i] = q_{ij}$. Since the Markov chain is ergodic, let π be its steady state distribution and π_t be its distribution at time instance t where $\pi_{i,t} = P[r_t = i]$. We know that π_t converges to π with time. The system has a d-dimensional state space where the state at time instance t is denoted by $x_t \in \mathbb{R}^d$ and a k-dimensional action/control space where the control/input at time instance t is denoted by $u_t \in \mathbb{R}^k$. The state space evolves with time as:

$$x_{t+1} = A_{r_t} x_t + B_{r_t} u_t + w_{r_t}, \tag{2.1}$$

where A_{r_t} is a $d \times d$ matrix corresponding to the mode at r_t, B_{r_t} is a $d \times k$ matrix corresponding to the same mode r_t and w_{r_t} is a d-dimensional zero-mean noise vector with distribution $w_{r_t} \sim \mathcal{N}(0, \sigma_{r_t}^2 I)$ where I is the $d \times d$ identity matrix. The noise is a zero mean normally distributed with each coordinate independent and

whose variance depends on the mode. $A_{r_t}, B_{r_t}, \sigma_{r_t}$ are the system parameters corresponding to mode r_t. We consider a setting in which the learner knows only the number of modes m but all the system parameters and the transition probabilities are unknown to it. The system incurs a cost c_t at each time instance given by:

$$c_t = x_t^\intercal Q x_t + u_t^\intercal R u_t, \tag{2.2}$$

where Q and R are fixed positive definite matrices known to the learner.

The problem of LQG control of the above system (2.1),(2.2) is to choose u_t at each time instance so that the expected cumulative cost $J_T = \sum_{t=1}^{T} \mathbb{E}[c_t]$ is minimized. At each time instance, we know which mode the system is in and we also observe the current value of the state x_t to choose u_t. A policy is a function $\Pi : \mathbb{R}^d \to \mathbb{R}^k$ such that $\Pi(x_t) = u_t$ which gives us the control on observing current state x_t. The cost of the policy Π is denoted by $J_T(\Pi) = \sum_{t=1}^{T} \mathbb{E}[c_t]$, when at each time instance $u_t = \Pi(x_t)$, that is the policy Π is played. We drop the subscript T to write the expected average cost for policy Π, i.e.,$J(\Pi) = \lim_{T\to\infty} \frac{1}{T} \mathbb{E}[J_T(\Pi)]$ when the horizon length is clear from the context. The optimal cost given by the best chosen policy is denoted by $J^* = J(\Pi^*)$. For the setting of a single mode, that is a simple LQR system, the optimal policy is stationary and linear that is the optimal control u_t is always given by a static policy $u_t = \Pi(x_t)$, where $\Pi^*(x) = K^* x$ and K^* is a constant $k \times d$ matrix. The optimal policy K^* is given by

$$K^* = -(B^\intercal P^* B + R)^{-1}(B^\intercal P^* A), \tag{2.3}$$

where P^* is the solution to the algebraic Ricatti equation:

$$P^* = A^\intercal P^* A + Q - A^\intercal P^* B(B^\intercal P^* B + R)^{-1} B^\intercal P^* A, \tag{2.4}$$

and

$$J^* = \sigma^2 \operatorname{Tr}(P^*). \tag{2.5}$$

When we have a finite horizon, we have non-stationary but linear optimal policy which can be calculated using the Dynamic Programming based recursion (algebraic Ricatti recursion) as follows:

$$K_t = -(B^\intercal P_{t+1} B + R)^{-1}(B^\intercal P_{t+1} A), \tag{2.6}$$

where $P_{T+1} = Q$ and P_t is computed backwards in time using the recursion:

$$P_t = A^\intercal P_{t+1} A + Q - A^\intercal P_{t+1} B(B^\intercal P_{t+1} B + R)^{-1} B^\intercal P_{t+1} A, \tag{2.7}$$

and the optimal total cost is given by

$$J_T^* = x_1^\intercal P_1 x_1 + \sigma^2 \sum_{t=1}^{T} \operatorname{Tr}(P_{t+1}), \tag{2.8}$$

where x_1 is the initial state and σ^2 is the variance of the Gaussian noise.

Further, a linear policy $\Pi(x) = Kx$ is defined as stable for the one-mode linear dynamical system if:

$$\rho(A + BK) < 1, \tag{2.9}$$

where $\rho(M)$ denotes the spectral radius of any matrix M , i.e., the largest absolute value of its singular values. All of these can be found in any standard text on optimal control such as [12]. A necessary and sufficient condition for the sability of the Markovian jump dynamic system when no control is applied can be found in [1].

In the next section, we generalize (2.6), (2.7) and (2.8) for the case of Markov Jump Dynamical System which would set the benchmark for comparison of our online learning algorithm. Once the optimal expected average cost per round J^* is computed for the Markov Jump Dynamic System (2.1),(2.2), we define the aggregate regret of the online learning algorithm as:

$$R_T = \sum_{t=1}^{T} (x_t^\intercal Q x_t + u_t^\intercal R u_t - J^*). \tag{2.10}$$

This aggregate regret is used as the performance metric of the online learning algorithm for the system.

2.2 Optimal policy for known Markovian jump linear dynamic system

In this section we state the optimal policy and cost for Markov jump linear dynamical systems for a finite horizon when the system parameters for every mode and the mode transition probabilities are all known.

Theorem 2.2.1. *Consider the Markovian jump linear dynamic system given in* (2.1),(2.2). *At time t, if the system is in mode $r_t = i$, then the optimal policy is given by:*

$$u_t = -(R + B_i^\intercal P_{i,t+1} B_i)^{-1} B_i^\intercal P_{i,t+1} A_i x_t \tag{2.11}$$

for any mode $i \in \{1, \ldots, m\}$ and for all $t \in \{1, \ldots, T\}$ and the matrices $P_{i,t}$ are given by the

recursion:

$$P_{i,t} = \sum_{j\in[m]} q_{ij}(Q + A_j^\intercal P_{j,t+1} A_j - A_j^\intercal P_{j,t+1}^\intercal B_j (R + B_j^\intercal P_{j,t+1} B_j)^{-1} B_j^\intercal P_{j,t+1} A_j), \tag{2.12}$$

where q_{ij} is the Markov chain transition probability from state i to state j, $\pi_{i,t}$ denotes the probability of the mode being i in time instant t, σ_i^2 is the variance of the i.i.d. Gaussian zero-mean noise in mode i, and the base case of the recursion is given by:

$$P_{i,T+1} = P_{T+1} = Q, \quad \forall i \in [m]. \tag{2.13}$$

When this optimal policy is followed, the optimal expected total loss is given by:

$$\sum_{j\in[m]} \mathbb{E}[x_1^\intercal P_{j,1} x_1 | r_1 = j]\pi_{j,1} \; + \; \sum_{t=1}^{T} \sum_{i\in[m]} \pi_{i,t}\sigma_i^2 \operatorname{Tr}(P_{i,t+1}), \tag{2.14}$$

where the expectation is over the initial state x_1 of the system at the beginning of time.

Now, in the limiting case for a large horizon T, if the initial state is bounded with probability 1, and if the matrices $P_{i,t}$ for each mode i, converge to P_i^*, and if the steady state probability of mode i is π_i^*, then the optimal expected average loss per round is approximately,

$$J^* = \sum_{i\in[m]} \pi_i^* \sigma_i^2 \operatorname{Tr}(P_i^*). \tag{2.15}$$

We use this J^* in our definition of regret (2.10), which we expect to grow sub-linearly with time for a no-regret online learning algorithm. The above theorem is an extension of theorem 4.3 in [1], where they derived the recursion for the system without the presence of noise.

2.3 An online learning algorithm for single mode system

In this section, we present [9]'s online learning algorithm for the LQR control of a single mode linear dynamical system. This was the first computationally efficient $\tilde{O}(\sqrt{T})$ regret algorithm. Later, [10] found a simpler naive exploration algorithm which achieves the same regret guarantee. In this thesis, however we focus on [9]'s algorithm which is a semidefinite programming based algorithm and it is this algorithm, which we have extended for the Markovian jump linear dynamic system, which will be the focus of the next chapter.

Before we state the algorithm, we assume a few conditions on the linear dynamical system, none of these assumptions are very restrictive as explained in [9]. We will have similar assumptions for the Markov jump model as well as explained in the next chapter.

Assumptions:

1. There are known positive constants $\alpha_0, \alpha_1, \sigma, \vartheta, \nu > 0$ such that, $\alpha_0 I \preccurlyeq Q \preccurlyeq \alpha_1 I$, $\alpha_0 I \preccurlyeq R \preccurlyeq \alpha_1 I$, $\|AB\| \leq \vartheta$, $J^* \leq \nu$, and we have i.i.d. zero mean Gaussian noise with variance σ^2 in each of the d-directions and the noise coordinates are uncorrelated,i.e., we haves $w_t \sim \mathcal{N}(0, \sigma^2 I)$, $\forall t$. Here, $\|AB\|$ denotes the maximum singular value norm of the augmented matrix AB (The two parameter matrices augmented).
2. There exists a stable policy, $K_0 \in \mathbb{R}^{k \times d}$ known to the learner.

Also, we now have $J^* = \sigma^2 \operatorname{Tr}(P^*)$, which is to be used in the same regret definition (2.10) for this system. In order to motivate the algorithm we also present an SDP formulation of the single mode LQR control problem details of which can be found in [13] and [9]. There they also introduce the concept of strong stability and sequential strong stability of policies and moreover, [13] shows how solving the SDP formulation of the LQR problem could help us extract strongly stable policy for the system.

The LQR problem is equivalent to solving the following semidefinite program:

$$\begin{aligned} &\text{minimize } \begin{pmatrix} Q & 0 \\ 0 & R \end{pmatrix} \bullet \Sigma \\ &\text{subject to} \\ &\Sigma_{xx} = (A \;\; B)\Sigma(A \;\; B)^\intercal + W, \\ &\Sigma \succcurlyeq 0. \end{aligned} \tag{2.16}$$

where $A \bullet B$ denotes $\operatorname{Tr}(A^\intercal B)$ for any two matrices A and B, and W is the noise covariance matrix which is $\sigma^2 I$ in our case. We take the state vector and put the control vector below it to get a vector of length $n = d + k$. Σ is a $n \times n$ matrix which is a proxy for the correlation matrix of this n-length vector. Thus Σ can be partitioned into four submatrices:

$$\Sigma = \begin{pmatrix} \Sigma_{xx} & \Sigma_{xu} \\ \Sigma_{ux} & \Sigma_{uu} \end{pmatrix} \tag{2.17}$$

where $\Sigma_{xx} \in \mathbb{R}^{d \times d}$, $\Sigma_{xu} \in \mathbb{R}^{d \times k}$, $\Sigma_{ux} \in \mathbb{R}^{k \times d}$, and $\Sigma_{uu} \in \mathbb{R}^{k \times k}$.

For the algorithm, we assume that we have good initial estimates A_0, B_0 of the matrices A, B such that $\left\|(AB) - (A_0 B_0)\right\|_F^2 \leq \epsilon$. From the second assumption of the existence of a known stable policy, a simple warmup exploration algorithm with

the control predicted by this algorithm added with a suitably scaled Gaussian noise for exploration would give us sufficient data within polynomial time such that if we perform ridge regression on this data, we can have such initial estimates.
Before discussing the warm-up policy, we define the concept of strongly-stable policy for an LQR system as follows:

Definition 1. *A policy K is (κ, γ) strongly-stable if we could write*

$$A + BK = HLH^{-1},$$
$$\textit{where } \|H\| \leq \alpha, \left\|H^{-1}\right\| \leq \tfrac{1}{\beta} \textit{ , with } \tfrac{\alpha}{\beta} \leq \kappa,$$
$$\|K\| \leq \kappa \textit{ and } \|L\| \leq 1 - \gamma.$$

It is shown in [13], that any strongly stable policy is (κ_0, γ_0) strongly-stable for some $\kappa_0 > 0, \gamma_0 \in (0, 1)$.
Now we describe the warm-up algorithm from section 6 of [9] below:

Algorithm 1: Warm-up using known stable policy

Input: (κ_0, γ_0) strongly-stable policy K_0, and horizon T_0.

for $t = 1, \ldots, T_0$ **do**
  observe state x_t
  play: $u_t \sim \mathcal{N}(K_0 x_t, 2\sigma^2 \kappa_0^2 I)$
end

It is shown in [9], that a polynomial overhead of T_0 for the warm-up procedure generates enough-data so that performing ridge-regression on them as in (3.1) gets us system parameter estimates (A_0, B_0) in the desired range of ϵ as chosen in (2.20).

Now we are ready to describe the OSLO algorithm from [9].

Algorithm 2: OSLO: Optimistic Semi-definite programming for Lq cOntrol

Input: parameters $\alpha_0, \alpha_1, \sigma^2, \vartheta, \nu > 0$; confidence $\delta \in (0,1)$; and an initial estimate $(A_0 B_0)$ such that $\left\|(AB) - (A_0 B_0)\right\|_F^2 \le \epsilon$.

Initialize: $\mu = 5\vartheta\sqrt{T}$, $V_1 = \lambda I$, where $\lambda = \frac{2^{11}\nu^5\vartheta\sqrt{T}}{\alpha_0^5\sigma^{10}}$ and $\beta = \frac{2^{18}\nu^4 n^2}{\alpha_0^4\sigma^6}\log(\frac{T}{\delta})$

for $t = 1, \ldots, T$ **do**

receive state x_t

if $det(V_t) > 2det(V_\tau)$ *or* $t = 1$ **then**

start new episode: $\tau = t$.

estimate system parameters: Let $(A_t B_t)$ be a minimizer of

$$\frac{1}{\beta}\sum_{s=1}^{t-1}\left\|(AB)z_s - x_{s+1}\right\|^2 + \lambda\left\|(AB) - (A_0 B_0)\right\|_F^2 \tag{2.18}$$

over all matrices $(AB) \in \mathbb{R}^{d\times n}$.

compute policy: Let Σ_t be an optimal solution to the following semi-definite program:

$$\begin{aligned} &\text{minimize } \begin{pmatrix} Q & 0 \\ 0 & R \end{pmatrix} \bullet \Sigma \\ &\quad \text{subject to} \\ &\Sigma_{xx} = (A_t \;\; B_t)\Sigma(A_t \;\; B_t)^\intercal + W - \mu\Sigma \bullet V_t^{-1} I, \\ &\qquad \Sigma \succcurlyeq 0. \end{aligned} \tag{2.19}$$

set $K_t = (\Sigma_t)_{ux}(\Sigma_t)_{xx}^{-1}$

end

else

set $K_t = K_{t-1}, A_t = A_{t-1}, B_t = B_{t-1}$.

end

play: $u_t = K_t x_t$

update: $z_t = \begin{pmatrix} x_t \\ u_t \end{pmatrix}$ and $V_{t+1} = V_t + \frac{1}{\beta} z_t z_t^\intercal$

end

We have the following regret guarantee for the above algorithm as proved in [9].

Theorem 2.3.1. *If the initial estimate* (A_0B_0) *satisfies* $\left\|(AB) - (A_0B_0)\right\|_F^2 \leq \epsilon$ *where*

$$\epsilon \leq \frac{1}{4\lambda} = \frac{\alpha_0^5 \sigma^{10}}{2^{13} \nu^5 \vartheta \sqrt{T}} \tag{2.20}$$

which is achievable using the warmup algorithm for some $T = poly(n, \nu, \vartheta, \alpha_0^{-1}, \sigma^{-1}, \|x_1\|)$*, then the above algorithm OSLO has total regret:*

$$R_T = O\left(\frac{\nu^5 n^3 \vartheta}{\alpha_0^4 \sigma^8} \sqrt{T \log^4 \frac{T}{\delta}} + \nu \sqrt{T \log^3 \frac{T}{\delta}}\right) \tag{2.21}$$

with probability at least $1 - \delta$*.*

CHAPTER 3

LEARNING ALGORITHM

3.1 OSLO for Jump dynamic system

In this chapter, we present our online learning algorithm for learning the Markovian jump linear dynamic system. Our algorithm is a generalization of the OSLO algorithm from 2.3. Here we build separate confidence ellipsoids for every mode, and keep track of separate learning epochs for each mode. At the start of every mode, since we know the index of the mode in which the system is, it is possible to take control step according to that mode's learning history. As a result, we do independent exploration and exploitation for each of the modes. Like in 2.3, again we need certain assumptions on the system which are not very restrictive. We stste the assumptions below. For the rest of this thesis we will use i, j to index the modes of the system and the system has m- modes denoted by the set $\{1, \ldots, m\} = [m]$ and this number m is known to the learner.

Assumptions:

1. There are known positive constants $\alpha_0, \alpha_1, \sigma, \vartheta, \nu > 0$ such that,
   $\alpha_0 I \preccurlyeq Q \preccurlyeq \alpha_1 I$, $\alpha_0 I \preccurlyeq R \preccurlyeq \alpha_1 I, ||A_i B_i|| \leq \vartheta,\ \forall i \in [m]$, $J^* \leq \nu$, and we have i.i.d. zero mean Gaussian noise with variance σ_i^2 in each of the d-directions and the noise coordinates are uncorrelated,i.e., we haves $w_i \sim \mathcal{N}(0, \sigma_i^2 I)$ for the mode $i \in [m]$. Here, again$||AB||$ denotes the maximum singular value norm of

the augmented matrix AB (The two parameter matrices augmented).

2. There exists a stable policy, $K_{0,i} \in \mathbb{R}^{k\times d}$ for each mode $i \in [m]$ known to the learner.

We remark that this time, J^* is defined by (2.15) and a policy $\Pi(x) = Kx$ is called stable for mode i if $\rho(A_i + B_iK) < 1$ where ρ represents the spectral radius of a matrix.

The first assumption above just asks for well-bounded system parameters for each mode which is a reasonable expectation for many real life instances of Markovian jump linear dynamic systems. For the second assumption, while the existence of stable policies for each mode is a reasonable assumption, but the assumption that the learner knows one such policy in advance for each of the modes need some justification. To this end, [14] provides an efficient algorithm for finding a stabilizing policy for any LQR system. applying this algorithm to each of the modes, we can find a stable policy $K_{0,i}$ for each mode $i \in [m]$. Moreover, one could always reset the "state" to prevent it from reaching unbounded states for every realistic mode of the system. This assumption of known stable policy has already appeared in [8] in the context of online learning of a single mode LQR system.

Now on the basis of this assumption, we run a warm-up policy first, so as to get system parameter estimates $(A_{i,0}B_{i,0})$ for each mode $i \in [m]$, such that $\left\|(A_iB_i) - (A_{0,i}B_{0,i})\right\|_F^2 \leq \epsilon \ \ \forall i \in [m]$. We obtain these estimates by performing ridge regression (3.1) on the data collected for each mode. More specifically, we run the warm-up procedure for a polynomial amount of time T_0 and record the state-space

evolution data while classifying them into data for each mode. Then for each mode, we run ridge regression to get these estimates $(A_{0,i}B_{0,i})$ for each mode $i \in [m]$. Next we describe the warm-up procedure for Markovian jump dynamic systems.

Algorithm 3: Warm-up using stable policies for jump dynamic system

Input: $(\kappa_{0,i}, \gamma_{0,i})$ strongly-stable policy $K_{0,i}$ for each mode $i \in [m]$, and horizon T_0.

for $t = 1, \ldots, T_0$ **do**

observe state x_t and mode i.

play: $u_t \sim \mathcal{N}(K_{0,i}x_t, 2\sigma_i^2\kappa_{0,i}^2 I)$

record: x_t in Z_i matrix and x_{t+1} in X_i matrix which would be used for ridge-regression for the mode i to learn the system parameter estimates $(A_{0,i}B_{0,i})$ for each mode $i \in [m]$.

end

We remark that although the system parameters could be unknown but the warm-up procedure requires the learner to know the noise variance of each mode. It is natural for the system noise to be the same for each mode and an estimate of the noise variance could be done as a short pre-processing step. Although our algorithm could deal with different noise levels for each mode if it knows the variances in advance. Alternately, one could assume that the learner does have some initial estimates of the system parameters to a degree of precision of the order $\tilde{O}(1/\sqrt{T})$. If this holds the warm-up procedure is no longer required and the learner could simply proceed with the learning algorithm. We are now ready to

describe the learning algorithm for Markovian jump linear dynamic system.

Algorithm 4: Optimistic Semi-definite programming for jump dynamic system.

Input: parameters $\alpha_0, \alpha_1, \sigma^2, \vartheta, \nu > 0$; confidence $\delta \in (0,1)$; and initial estimates $(A_{0,i}B_{0,i})$ such that $\left\|(A_iB_i) - (A_{0,i}B_{0,i})\right\|_F^2 \leq \epsilon$ for each mode $i \in [m]$.

Initialize: $\mu = 5\vartheta\sqrt{T}$, $V_{1,i} = \lambda I,\ t_i = 1\ \ \forall i \in [m]$, where $\lambda = \frac{2^{11}\nu^5\vartheta\sqrt{T}}{\alpha_0^5\sigma^{10}}$ and $\beta = \frac{2^{18}\nu^4 n^2}{\alpha_0^4\sigma^6}\log(\frac{T}{\delta})$

for $t = 1, \ldots, T$ **do**

receive state x_t and mode i and record $x_{t_i,i} = x_t$.

if $det(V_{t_i,i}) > 2det(V_{\tau_i,i})$ *or* $t_i = 1$ **then**

start new episode: $\tau_i = t_i$.

estimate system parameters: Let (A_tB_t) be a minimizer of

$$\frac{1}{\beta}\sum_{s=1}^{t_i-1}\left\|(AB)z_{s,i} - x_{s+1,i}\right\|^2 + \lambda\left\|(AB) - (A_{0,i}B_{0,i})\right\|_F^2 \tag{3.1}$$

over all matrices $(AB) \in \mathbb{R}^{d\times n}$.

compute policy: Let $\Sigma_{t,i}$ be an optimal solution to the following semi-definite program:

$$\begin{aligned} &\text{minimize } \begin{pmatrix} Q & 0 \\ 0 & R \end{pmatrix} \bullet \Sigma \\ &\quad \text{subject to} \\ &\Sigma_{xx} = (A_{t,i}\ \ B_{t,i})\Sigma(A_{t,i}\ \ B_{t,i})^\intercal + \sigma_i^2 I - \mu\Sigma \bullet V_{t,i}^{-1} I, \\ &\qquad \Sigma \succcurlyeq 0. \end{aligned} \tag{3.2}$$

set $K_{t,i} = (\Sigma_{t,i})_{ux}(\Sigma_{t,i})_{xx}^{-1}$

end

else

set $K_{t,i} = K_{t-1,i}, A_{t,i} = A_{t-1,i}, B_{t,i} = B_{t-1,i}$.

end

play: $u_t = K_{t,i}x_t$

update: $z_{t,i} = \begin{pmatrix} x_t \\ u_t \end{pmatrix}$, $t_i = t_i + 1$ and $V_{t+1,i} = V_{t,i} + \frac{1}{\beta}z_{t,i}z_{t,i}^\intercal$

end

The algorithm above maintains confidence ellipsoids around the parameter estimates $(A_{t,i}B_{t,i})$ for each mode. The matrix $V_{t,i}$ is an estimate of the empirical covariance of the data and is used as a measure of the size of the confidence ellipsoid. Whenever, the volume of the last marked ellipsoid becomes half or less than the current value of the current ellipsoid for that mode we switch to a new episode of learning for that mode. Within this new episode, the algorithm makes a new estimate of the mode parameters using ridge regression on all the data collected for this mode so far. Using this new estimate as the center it sets the next confidence ellipsoid using the current value of the covariance matrix for that mode. This ellipsoid in then used throughout for that episode. Using the new estimate of the parameters, the algorithm solves a relaxed version of the SDP formulation of the LQR problem for that mode. The SDP is relaxed so that the policy extracted from its solution would be stable and would have a cost which is an underestimate of the optimal cost for that mode. Since the true SDP formulation of the LQR system would require us to know the actual system parameters, so using the parameter estimates introduces an uncertainty into the SDP and the relaxation is added to this SDP to account for this uncertainty. Thus, this algorithm is "optimistic in the face of uncertainty".

The algorithm after retrieving its policy for a mode at the start of any of its episodes continues using this policy for that mode while simultaneously updating the confidence matrix for that mode. This way, the policy does both *exploration* and *exploitation*. Exploitation - by using the policy which is the best guess of the estimates available at the start of an episode, by solving the SDP and extracting

the best guess policy out of its solution. Exploration - by continuing to use the best guess at the start of the episode throughout out the episode and meanwhile collecting newer data in order to get better estimates for the next episode.

3.2 Implementation details

We implemented the optimistic semi-definite programming algorithm for jump dynamic system as described in the last section. The implementation was done for a 2-dimensional state space and 2-dimensional action space system with two modes of operation. The underlying Markov Chain has two states corresponding to the two modes "1" and "2", with transition probabilities $q_{11} = p, q_{12} = 1-p, q_{21} = 1-q, q_{22} = q$, where p and q are small probabilities less than $\frac{1}{2}$. The Markov chain is thus ergodic with steady state distribution: $\pi_1 = \frac{1-q}{2-p-q}$ and $\pi_2 = \frac{1-p}{2-p-q}$. The horizon length is kept at 3000000, which is sufficiently large to see the limit of an infinite horizon while solving the Ricatti equation based recursion for the P matrices of each mode (2.12),(2.13), so we get the P matrices to converge rapidly for each mode. Thus, $P_{1,1}$ was chosen as P_1^* and $P_{1,2}$ was chosen as P_2^* for calculating the optimal expected average cost per round J^* using (2.15). Now this J^* is used as a benchmark for measuring the regret of the algorithm using (2.10). The implementation can be found in the following Github repository: `https://github.com/ritabrata-ray/Markov-Jump-Linear-Dynamic-Systems`.

CHAPTER 4

PERFORMANCE ANALYSIS

4.1 Numerical results on performance

As discussed in 3.2, we performed numerical simulation of the *Optimistic Semi-Definite Programming for Jump Dynamic System* algorithm on randomly generated positive definite Q and R matrices and randomly generated system parameters for a two-dimensional state and two-dimensional action space system with two modes of operation.

On performing the simulation for a large enough time horizon of $3,000,000$, we found that the algorithm has a sub-linear regret. We computed the running average regret as a function of time:

$$Reg(t) = \frac{1}{t} \sum_{\tau=1}^{t} \left(x_t^\intercal Q x_t + u_t^\intercal R u_t - J^* \right). \tag{4.1}$$

We get the following plot for this running average regret against time. We see a monotonically decreasing plot which converges to zero with time, which implies a sub-linear regret for the algorithm.

We further plot the error in parameter estimates for each mode with time. Specifically, we plot the Frobenius norm parameter estimate erroralong the y-axis

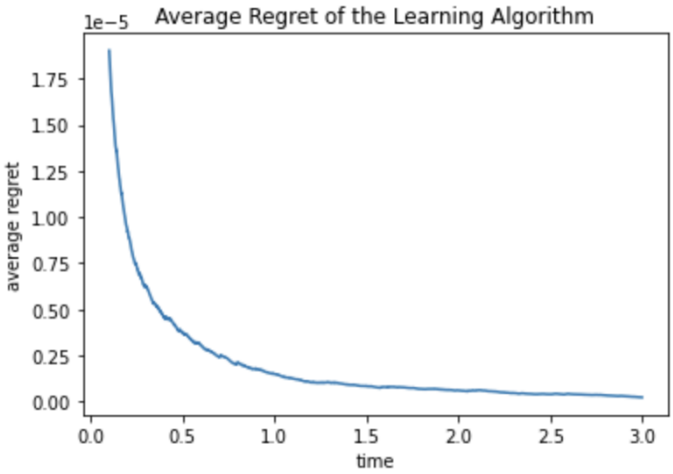

Figure 4.1: Running Average Regret of the algorithm against time

and time along the x-axis. The plots for both the modes are as follows:

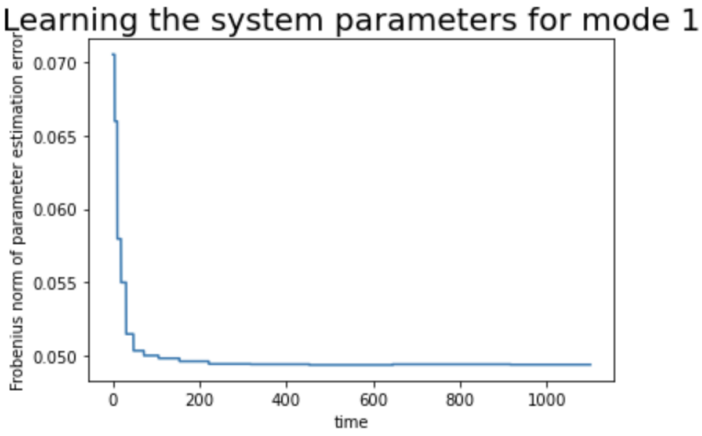

Figure 4.2: Parameter Estimation Error for Mode 1

We can see a staircase like reduction in the error for the parameter estimation plots. This is due to the fact that, the parameter estimation occurs only when the episode changes for that mode. Initially, the steps are narrow but they grow wider with time. This is due to the fact that the episodes are small initially, where we

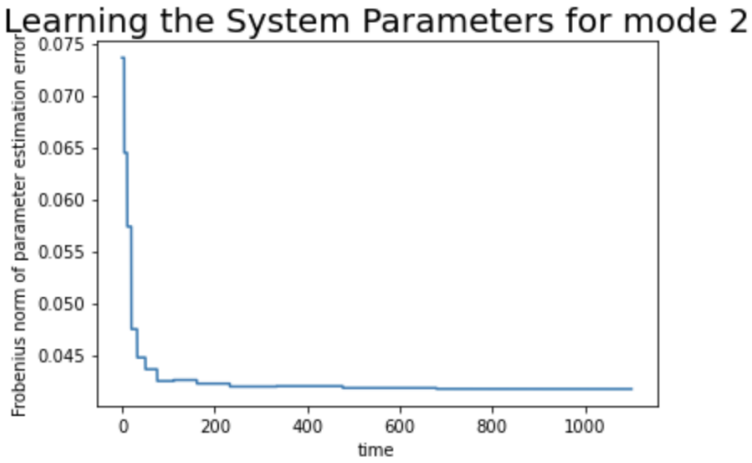

Figure 4.3: Parameter Estimation Error for Mode 2

have coarse estimates of the parameters and the covariance matrix builds up fast enough for the episode to switch. As time increases, the estimates get better and better with time, which means that the covariance matrix builds up slowly and the episodes are much longer than before. so, the algorithm does more exploration in the initial period by quickly changing its episodes but as it learns the system better, it exploits its current knowledge of the system more with longer episodes and sticking to the best known policy of that episode for longer until it gathers enough data so that changing the ellipsoid would cause a significant reduction in costs.

4.2 Regret Analysis

We expect a $\tilde{O}(\sqrt{T})$ regret for this algorithm similar to [9]'s OSLO algorithm for the single mode LQR control. However, the hurdle in extending [9]'s analysis lies in proving that the sequence of policies used by the algorithm is *sequentially strongly stable* [see [9] for details about sequential strong stability]. The non-triviality arises from the fact that the mode may switch with time and so even the sequence of policies learnt by the algorithm for each mode is sequentially-stongly-stable but the intermixing of modes with time may result in the overall sequence not being sequentially-stongly-stable. The absence of this guarantee makes it difficult to prove that the state of the system will be bounded in time. However, with repeated numerical simulations we always found the state variable x_t to not only remain bounded but go down to zero with time, that too exponentially fast. It is exactly this kind of decay of $||x_t||$ which is desirable to extend the proof and regret analysis to our *Optimistic Semi-definite Programming for Jump Dynamic System* algorithm, which gives us hopes of a similar regret analysis for our algorithm.

CHAPTER 5

CONCLUSION

If we could establish an $\tilde{O}(\sqrt{T})$ regret guarantee for the *Optimistic Semi-definite Programming for Jump Dynamic System* algorithm, then this will be an asymptotically optimal regret algorithm for online learning of Jump dynamical systems. This is due to the fact that [5] already established a $\Omega(\sqrt{T})$ regret for the special case of online learning of Markov Jump Linear Systems. Thus, an immediate open problem is to obtain a regret analysis for the present algorithm. We believe that a tight analysis of our algorithm would show the desired $\tilde{O}(\sqrt{T})$ regret and therefore that our algorithm is asymptotically optimal in terms of regret.

Overall, Markov Jump Linear Dynamical Systems are a very important class of systems which model several real world systems. As a result, having online learning algorithms for such systems is of considerable significance. Our algorithm is an instance of "optimism in the face of uncertainty" paradigm in online learning algorithm. an interesting open question is can other exploration techniques like robust synthesis [8] lead to simpler and more computationally efficient online learning algorithms for jump dynamic systems? Also, is there a much simpler (with also a simple regret analysis) naive-exploration algorithm for jump dynamical systems like the one in [10] for single mode LQR control? Also, it is worth pondering about the largest possible class of control problems for which such no-regret online learning algorithms exist.

APPENDIX A

CHAPTER 1 OF APPENDIX

Appendix chapter 1 text goes here

APPENDIX B

BIBLIOGRAPHY

[1] V. Gupta, R. M. Murray, L. Shi, and B. Sinopoli, "Networked sensing, estimation and control systems,"

[2] A. Logothetis and V. Krishnamurthy, "Expectation maximization algorithms for map estimation of jump markov linear systems," *IEEE Transactions on Signal Processing*, vol. 47, no. 8, pp. 2139–2156, 1999.

[3] A. Doucet, N. Gordon, and V. Krishnamurthy, "Particle filters for state estimation of jump markov linear systems," *IEEE Transactions on Signal Processing*, vol. 49, no. 3, pp. 613–624, 2001.

[4] O. L. V. Costa, M. D. Fragoso, and R. P. Marques, *Discrete-time Markov jump linear systems*. Springer Science & Business Media, 2006.

[5] S. Baltaoglu, L. Tong, and Q. Zhao, "Online learning and optimization of markov jump linear models," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2289–2293, 2016.

[6] Y. Abbasi-Yadkori and C. Szepesvári, "Regret bounds for the adaptive control of linear quadratic systems," in *Proceedings of the 24th Annual Conference on Learning Theory* (S. M. Kakade and U. von Luxburg, eds.), vol. 19 of *Proceedings of Machine Learning Research*, (Budapest, Hungary), pp. 1–26, PMLR, 09–11 Jun 2011.

[7] M. Ibrahimi, A. Javanmard, and B. V. Roy, "Efficient reinforcement learning for high dimensional linear quadratic systems," in *Proceedings of the 25th*

International Conference on Neural Information Processing Systems - Volume 2, NIPS'12, (Red Hook, NY, USA), p. 2636–2644, Curran Associates Inc., 2012.

[8] S. Dean, H. Mania, N. Matni, B. Recht, and S. Tu, "Regret bounds for robust adaptive control of the linear quadratic regulator," *arXiv preprint arXiv:1805.09388*, 2018.

[9] A. Cohen, T. Koren, and Y. Mansour, "Learning linear-quadratic regulators efficiently with only $\sqrt{T}$ regret," in *Proceedings of the 36th International Conference on Machine Learning* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 1300–1309, PMLR, 09–15 Jun 2019.

[10] M. Simchowitz and D. J. Foster, "Naive exploration is optimal for online LQR," *CoRR*, vol. abs/2001.09576, 2020.

[11] P. Gradu, E. Hazan, and E. Minasyan, "Adaptive regret for control of time-varying dynamics," *CoRR*, vol. abs/2007.04393, 2020.

[12] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, vol. I. Belmont, MA, USA: Athena Scientific, 3rd ed., 2005.

[13] A. Cohen, A. Hasidim, T. Koren, N. Lazic, Y. Mansour, and K. Talwar, "Online linear quadratic control," in *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, pp. 1029–1038, PMLR, 10–15 Jul 2018.

[14] S. Dean, H. Mania, N. Matni, B. Recht, and S. Tu, "On the sample complexity of the linear quadratic regulator," 2018.