MARKOV DECISION PROCESSES WITH LARGE STATE SPACE

CHIRAG MAHESHWARI AND OJAS DESHPANDE

Abstract. Markov Decision Processes are probabilistic models where outcomes are partly random and partly based on the actions of the agent on the environment. The aim of the agent is to take actions in such a way so as to maximize his expected reward. When the state space of such problems grows too large, traditional methods such as dynamic programming, linear programming are no longer efficient. Here, we present a few approaches to solving the problem for large state spaces using convex approximations.

Motivation

A large number of phenomena are modeled as a Markov Decision Process such as resource allocation, queue control, robotics, games. In most of the problems state space grows exponentially requiring efficient methods to solve these models. For e.g., in the case discrete-time queuing network problem, the state space is exponential with the number of queues which makes it almost impossible to solve it computationally with traditional methods and requires us to resort to approximation techniques.

1. Introduction

We consider average loss as well as total cost Markov Decision problems. Given the transition matrix and the loss function (reward function), aim is to devise a policy that will minimize the expected loss (maximize expected reward). The same problem with small state space has been well researched and can be solved using methods such as dynamic programming (value iteration, policy iteration), linear programming. With a large state spaces, these methods are not computationally feasible.

Here we analyze two closely related methods to solve this problem (with slight variations) for large state space. The methods presented in this represent this large state space with a low-dimensional set of features. The methods presented here to solve an Approximate Linear Programming problem which don't make use of any *distribution based on the optimal policy as was required in some prior works* [1].

1.1. **Notation.** Here, we introduce common the notations used throughout this report. Given a set $\mathcal{S}$, $\Delta_{\mathcal{S}}$ denotes the probability distributions over the set $\mathcal{S}$. With a matrix M of size $m \times n$, then, $M_{i,:}$ and $M_{:,j}$ denote the ith row and jth column of the matrix respectively. Taking a constant vector c and another vector v, we denote $\|v\|_{p,c} = \|c \odot |v|\|_p$ where $\odot$ symbolizes element wise multiplication. Also comparison of two vectors is done element-wise, i.e., $v \leq c$ means $v_i \leq c_i$ $\forall$

Date: May 10, 2017.

Key words and phrases. Reinforcement Learning, Markov Decision Process, Large State Space.

i. Here, we study the problem of minimizing the expected cost and problems with a reward function can easily be formulated in the same way by taking *cost as the negative of the given reward*.

1.2. Markov Decision Process.

Markov Decision Process. Markov decision processes is a stochastic system consisting of an environment and an agent wherein at each time-step agent takes an action on the environment based on the current state and the environment responds with a loss. Markov decision process is a 5-tuple $< \mathcal{X}, \mathcal{A}, \mathcal{P}, R, \gamma >$ where,

- $\mathcal{X}$ is a countable set of states (state space).
- $\mathcal{A}$ is a countable set of actions (action space), which will be fixed for all states.
- $P \in \mathcal{P}$, $P : \mathcal{X} \times \mathcal{A} \to \Delta_{\mathcal{X}}$ is the probability transition matrix between states given an action.
- $R : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is an immediate cost/loss function.
- $\gamma \in [0, 1]$ is a fixed discount factor.

1.3. **Control Policy.** At each time-step the agent takes an action from the set of actions available from the set $\mathcal{A}$. This action is based on the current state of the environment (or the agent's representation of the environment) which is a countable set $\mathcal{X}$ in our case. A function $\pi(x) : \mathcal{X} \to \Delta_{\mathcal{A}}$ specifies the action or probability distribution over the actions taken by the agent when presented with state x and is called the *policy*. When we say we want to solve a MDP, that requires us to find an ***optimal policy*** so as to minimize cost and is denoted by π^*.

Every policy π induces a probability transition matrix denoted by $P^\pi \in \mathcal{P}$. Furthermore, the fraction of time spent by the agent in state x under the policy π is denoted by $v_\pi(x)$. Its called the stationary distribution of states. State-action stationary distribution under policy π is denoted by $\mu_\pi(x, a)$. These supplementary constructs can be calculated as,

$$\begin{aligned} P^\pi(x, x') &= \sum_{a \in \mathcal{A}} \Pr(x'|x, a)\pi(a|x). \\ \mu_\pi(x, a) &= v_\pi(x)\pi(x, a) \end{aligned}$$

One can easily get back the policy function given the state-action stationary distribution by a simple calculation given in eq. (1.1).

$$\pi(a|x) = \frac{\mu_\pi(x, a)}{\sum_{a' \in \mathcal{A}} \mu_\pi(x, a)} \tag{1.1}$$

1.4. **Cost.** Every action taken by the agent results in a cost from the environment given by the cost function R. The agent tries to minimize its expected cost during its lifetime. These expected costs can take the form of average cost or total cost and our defined the subsequent subsections. The *value function* $h(x) : \mathcal{X} \to \mathbb{R}$ of the MDP is defined as the expected cost when the agent starts from state x.

Note: For our study, we assume that $\gamma = 1$ in cases.

1.4.1. *Total Cost.* Starting from state x_1, the value function when we model the total cost is given by eq. (1.2)

$$h(x_1) = \lim_{T\to\infty} \mathbb{E}\left[\sum_{t=1}^{T} \gamma^{t-1} l(x_t, P^\pi)\right] \tag{1.2}$$

Here, $l(x, P^\pi)$ is the immediate cost function when we follow the policy π which implicitly induces the state transition matrix P^π giving us the action and state at each time step.

In the case when $\gamma = 1$, we assume that there exists a set of states $S \subset \mathcal{X}$ called the *absorbing states*. This set of states has the property that $l(x, a) = 0 \ \forall \ a \in \mathcal{A}$, $x \in S$ and $P(x, x') = 1 \ \forall \ x, x' \in S$. This keeps the value function well defined in cases where $\gamma = 1$.

1.4.2. *Average Cost.* Similarly, the value function in case of average cost is defined as when starting from state x_1 is given by eq. (1.3).

$$h(x_1) = \lim_{T\to\infty} \frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T} \gamma^{t-1} l(x_t, P^\pi)\right] \tag{1.3}$$

If $\gamma = 1$, this value function is very well defined. Under any policy, there exists a stationary distribution over states which is known as the property of *ergodicity*. Named after this property, average cost is also known as ergodic cost.

In case of average cost, the value function is independent of the starting state, i.e., $h(x) = \lambda \ \forall x \in \mathcal{X}$. The immediate difference between value function and the average cost is called the differential value function again denoted by $h(x)$.

1.4.3. *Kullback-Leibler Loss.* We also study an interesting class of MDPs [2] in section 5 where the cost function is given by Kullback-Leibler divergence. Given some function $q : \mathcal{X} \to [0, Q]$ called the base policy and some fixed transition matrix $P_0 \in \mathcal{P}$, the KL loss function is given by eq. (1.4).

$$l(x, P) = q(x) + \sum_{x' \in \mathcal{X}} P(x, x') \log \frac{P(x, x')}{P_0(x, x')} \tag{1.4}$$

Here, if $P(x, x') > 0$ and $P_0(x, x') = 0$, then $l(x, P) = \infty$ which forces us to play with distributions absolutely "continuous" with P_0. One caveat with this cost is that we tend to explore policies only around the base policy P_0. It can be useful if we already have a good enough policy induced transition matrix and we need to optimize further.

2. Problem Definition

2.1. **Bellman Optimality Equation.** The bellman operator defined by eq. (2.1) which gives us the bellman optimality equation $\lambda^* + h*(x) = (Lh^*)(x)$.

$$(Lh)(x) = \min_{a\in\mathcal{A}} \left(l(x, a) + \sum_{x' \in \mathcal{X}} P_{(x,a),x'} h \right) \tag{2.1}$$

Under certain assumptions, there exist a scalar λ_* and a vector $h \in \mathcal{R}^x$ that satisfy the Bellman optimality equation. The scalar λ_* is called the optimal average loss, while the vector h_* is called a differential value function. The action that minimizes the right-hand side of the above equation is the optimal action in state x and is denoted by $a(x)$. The optimal policy is defined by $\pi(a(x)|x) = 1$. Given l and P, the objective of the planner is to compute the optimal action in all states, or equivalently, to find the optimal policy. For the case of total loss, the term λ_* becomes 0.

Bellman optimality equation is generally solved via Dynamic Programming methods such as Policy Iteration, Value Iteration.

2.2. **Linear Programming.**

2.2.1. *Primal LP.*

The same bellman equation for the MDP can be reformulated as follows :

$$\max_{\lambda,h} \lambda$$

$$\text{s.t. } B(\lambda 1 + h) \leq l + Ph$$

Here, $B \in \{0,1\}^{XA \times X}$ is just the blown up identity matrix. Each row of a $X \times X$ identity matrix is copied into A consecutive rows. The above formulation just writes down the bellman inequality for all state action pairs.

2.2.2. *Dual LP.*

Given the stationary distributions over the state action pair under an implicit policy π,We can write

$$\begin{aligned}\pi_* &= \arg\min_{\pi} \sum_{x \in \mathcal{X}} v_\pi(x) \sum_{a \in \mathcal{A}} \pi(a|x) l(x,a) \\ &= \arg\min_{\pi} \sum_{(x,a) \in \mathcal{X} \times \mathcal{A}} \mu_\pi(x,a) l(x,a) \\ &= \arg\min_{\pi} \mu_\pi^T l\end{aligned}$$

Thus, we get our dual formulation :

$$\min_{\mu \in \mathcal{R}^{\mathcal{X}\mathcal{A}}} \mu^T l,$$

$$\text{s.t. } \mu^T 1 = 1, \mu \geq 0, \mu^T (P - B) = 0$$

While the first two constraints ensure that μ is a probability distribution, the last one ensures that it is a stationary distribution.

3. Previous Work

MDP are easily solved when the state-space is small using methods of Linear Programming Dynamic Programming like policy iteration, value iteration [3]. In these methods computational complexity scales with the state and action space. In large state spaces, exact Dynamic Programming is not feasible and we resort to Approximate Dynamic Programming (ADP) and Approximate Linear Programming (ALP) methods. Using ADP methods like Temporal Difference-learning and Monte-Carlo learning suffers from the problem of exploration and exploitation. Also, noted in [1] prior work on ALP either requires access to samples from a distribution that depends on the optimal policy or assumes the ability to solve an LP with as many constraints as states.

4. Approximate Stationary Distribution

4.1. **Approximation for large state spaces.** To bring down the dimension of the state space, each state is represented instead by a set of d features (where d is ¡¡ $|\mathcal{X}|$). Denoting by Φ a $\mathcal{X}\mathcal{A} \times d$ matrix with columns for features, we can write μ as $\Phi\theta$. Thus, our dual problem becomes :

$$\begin{aligned} &\min_{\theta} \theta^T \Phi^T l \\ &\theta^T \Phi^T 1 = 1, \Phi\theta \geq 0, \theta^T \Phi^T (P - B) = 0 \end{aligned} \tag{4.1}$$

It can also be noted that adding a known stationary distribution μ_0 to $\Phi\theta$ doesn't change the optimal value of the above LP. Though just adding a stationry distribution doesn't make the new distribution stationary, it still defines the policy :

$$\pi_\theta(a|x) = \frac{[\mu_0(x,a) + \phi_{(x,a),:}\theta]_+}{\sum_{a' \in A}[\mu_0(x,a') + \phi_{(x,a'),:}\theta]_+}$$

4.2. **ALP Reformulation.** The efficient large-scale dual ALP problem is to produce parameters $\hat{\theta}$ such that

$$\mu_{\hat{\theta}}^T l \leq \min \left\{ \mu_\theta^T l : \theta \text{ feasible for eq. (4.1)} \right\}$$

in time polynomial in d and $1/\epsilon$. We have constant time access to arbitrary entries of $\Phi, l, P, \mu_0, P^T\Phi$, and $I^T\Phi$.

We further convert this optimization problem to an unconstrained convex optimization problem by adding violation functions of the constraints. These functions would serve as a penalty if θ doesn't abide by the constraints. On the other hand, the value of the violation function is 0 if θ is in the feasible set.

$$\mu_{\hat{\theta}}^T l \leq \mu_\theta^T l + O\left(\frac{1}{\epsilon}\|[\mu_0 + \Phi\theta]_-\|_1\right) + O\left(\frac{1}{\epsilon}\|(P-B)^T(\mu_0 + \Phi\theta)\|_1\right) + O\left(\epsilon \log \frac{1}{\delta}\right) \tag{4.2}$$

4.3. **Reduction to Stochastic Convex Optimization.** For a constant H, we use the following convex cost function :

$$\begin{aligned} c(\theta) &= l^T(\mu_0 + \Phi\theta) + H\|[\mu_0 + \Phi\theta]_-\|_1 + H\|(P-B)^T(\mu_0 + \Phi\theta)\|_1 \\ &= l^T(\mu_0 + \Phi\theta) + H\|[\mu_0 + \Phi\theta]_-\|_1 + H\|(P-B)^T\Phi\theta]_-\|_1 \\ &= l^T(\mu_0 + \Phi\theta) + H\sum_{(x,a)} \left|[\mu_0(x,a) + \Phi_{(x,a):}\theta]_-\right| + H\sum_{x'} \left|(P-B)^T_{:,x'}\Phi\theta\right| \end{aligned}$$

We will eventually show that if we can find a solution for this cost function with atmost $O(\epsilon)$ error, i.e., if we find $\hat{\theta}$ such that $c(\hat{\theta}) \leq c(\theta) + O(\epsilon)$, then eq. (4.2) holds true with a high probability.

4.4. **Gradient Calculation.** Unfortunately, computing the gradient of $c(\theta)$ also takes $O(XA)$ time. Instead, we use unbiased estimators and use stochastic gradient descent. Let T be the number of iterations of our algorithm. Let q_1 and q_2 be distributions over the state-action and state space, respectively. Let $((x_t, a_t)), t =$

1...T be i.i.d. samples from q_1 and $(x'_t)t = 1...T$ be i.i.d. samples from q_2. At round t, the algorithm estimates subgradient $\nabla c(\theta)$ by

(4.3)
$$g_t(\theta) = l^T\Phi - H\frac{\Phi_{(x_t,a_t)}}{q1(x_t,a_t)}.1_{\{\mu_0(x_t,a_t)+\Phi_{(x_t,a_t),:}\theta<0\}} + H\frac{(P-B)^T_{:,x'_t}\Phi}{q_2(x'_t)}s((P-B)^T_{:,x'_t}\Phi\theta).$$

As after this update, θ might not remain in the feasible set anymore, so we also need to take the projection of it. (Projected Subgradient descent). This estimate is fed to the projected subgradient method, which in turn generates a vector θ_t. After T rounds, we average vectors $(\theta_t), t = 1...T$ and obtain the final solution $\hat{\theta}_T = \sum_{t=1}^{T}\theta_t/T$. Vector $\mu_0 + \Phi\hat{\theta}_T$ defines a policy, which in turn defines a stationary distribution $\mu_{\hat{\theta}_T}$.

4.5. **Analysis.** In this section all the proofs related to this method would be provided. If you instead just want a summary of what's going to happen, we show that by using the violation functions of the constraints, if we find the solution to this new modified ALP admitting an error of atmost ϵ, then after $1/\epsilon^4$, we can get a solution to our original solution with error of the order of $O(\epsilon + 1/\delta)$ with probability at least $1 - \delta$.

Fast Mixing Assumption: Let M^π be a $X \times XA$ matrix which encodes the policy π. For any policy π, there exists a $\tau(\pi) > 0$ such that for all distributions d and d' over the state-action space, $\|dPM^\pi - d'PM^\pi\|_1 \le e^{-1/\tau(\pi)}\|d - d'\|_1$

We also assume that the columns of the feature matrix are positive and sum to 1. Define

$$C_1 = \max_{(x,a)\in\mathcal{X}\times\mathcal{A}} \frac{\|\Phi_{(x,a),:}\|}{q_1(x,a)},$$
$$C_2 = \max_{x\in\mathcal{X}} \frac{\|(P-B)^T_{:,x}\Phi\|}{q_2(x)}$$

These constants are going to appear in our performance bounds, thus it would be better to choose distributions which make these bounds small.

Theorem 4.1. *Consider an expanded efficient large-scale dual ALP, with violation function $V = O(V_1 + V_2)$, defined by*

$$V_1(\theta) = \|[\mu_0 + \Phi\theta]_-\|_1$$
$$V_2(\theta) = \|(P-B)^T(\mu_0 + \Phi\theta)\|$$

Assume $\tau = \sup\{\tau(\mu_\theta) : \theta \in \Theta\} < \infty$ is finite. Suppose we apply the stochastic subgradient method to the problem. Let $\epsilon \in (0,1)$, and $T = 1/\epsilon^4$ be the number of rounds for which we run SGD. Let $H = 1/\epsilon$, and $\hat{\theta}_T$ be the output of SGD after T rounds and let the learning rate be $\eta_t = S/(G'\sqrt{T})$, where $G' = \sqrt{d} + H(C_1 + C_2)$. Then, for any $\delta \in (0,1)$, with probability at least $1 - \delta$,

$$\mu^T_{\hat{\theta}_T} l \le \min_{\theta\in\Theta}\left(\mu^T_\theta l + O\left(\frac{1}{\epsilon}(V_1(\theta) + V_2(\theta))\right) + O(\epsilon)\right) \tag{4.4}$$

where the constants in the big-O notation are polynomials in $S, d, C_1, C_2,$ and $\log(1/\delta)$.

First of all, because $V_1(\theta)$ and $V_2(\theta)$ can be bounded.

$$\begin{aligned}
V_1(\theta) &\leq \|\mu_0\|_1 + \|\Phi\theta\|_1 \leq 1 + S\sqrt{d} \\
V_2(\theta) &\leq \sum_{x'} |P^T_{:,x'}(\mu_0 + \Phi\theta)| + \sum_{x'} |B^T_{:,x'}(\mu_0 + \Phi\theta)| \\
&\leq \sum_{x'} P^T_{:,x'}|(\mu_0 + \Phi\theta)| + \sum_{x'} B^T_{:,x'}|(\mu_0 + \Phi\theta)| \\
&= 2 \cdot 1^T|\mu_0 + \Phi\theta| \leq 2 \cdot 1^T(|\mu_0| + |\Phi\theta|) \\
&\leq 2(1 + S\sqrt{d})
\end{aligned}$$

Choosing the set of features optimally would make this bound small. The optimal choice of ϵ is $\sqrt{V_1(\theta_*) + V_2(\theta_*)}$, where θ_* is the minimizing value of eq. (4.4). As this value is not known, we choose the value of ϵ based on the current value of θ after every iteration.

Below lemma shows the relation between how much θ differs from a stationary distribution and the value of the violation functions.

Lemma 4.2. *Let $u \in \mathcal{R}^{XA}$ be a vector. Assume,*

$$\sum_{(x,a)} u(x,a) = 1, \|[u]_-\|_1 \leq \epsilon', \|u^T(P - B)\|_1 \leq \epsilon''$$

The vector $[u]_+/\|[u]_+\|_1$ defines a policy, which defines a stationary distribution μ_u. We have that

$$\|\mu_u u\|_1 \leq (\tau(\mu_u)\log(1/(2\epsilon' + \epsilon'')) + 2)(2\epsilon' + \epsilon'')$$

Proof. Let $h = [u]_+/\|[u]_+\|_1$. Then h is almost a stationary distribution, i.e.

$$\|h^T(P - B)\|_1 \leq 2\epsilon' + \epsilon''$$

Note that the first assumption above simply means that $\|[u]_+\|_1 - \|[u]_-\| = 1$, and therefore,

$$\begin{aligned}
\|h^T(P - B)\|_1 &= \|\frac{[u]_+^T}{\|[u]_+\|_1}(P - B)\|_1 \\
&= \frac{\|(u - [u]_-)^T(P - B)\|_1}{1 + \|[u]_-\|_1} \\
&\leq \|u^T(P - B)\|_1 + \|[u]_-^T(P - B)\|_1 \\
&\leq \epsilon'' + \|[u]_-\|_1\|(P - B)^T\|_1 \\
&\leq \epsilon'' + 2\epsilon'
\end{aligned}$$

(because the linear maps defined by P and B have operator norms (corresponding to the 1-norm) bounded above by 1). Also,

$$\|h - u\|_1 \leq \|h - [u]_+\|_1 + \|[u]_+ - u\|_1 = \|[u]_-\|_1 + \|[u]_-\|_1 \leq 2\epsilon'$$

Let $v_0 = h$ be the initial state-action distribution. We will show that as we run policy h, the state-action distribution converges to μ_h and this vector is also close to h itself. We already have shown that $v_0^T P = h^T B + v_0$, where norm 1 of v_0 is bounded by $2\epsilon' + \epsilon''$. Let M^h be the $X \times XA$ matrix that encodes the policy h. Then,

$$v_1^T = h^T P M^h = (h^T B + v_0) M^h = h^T B M^h + v_0 M^h = h^T + v_0 M^h = h^T + v_0 M^h$$

Let $v_1 = v_0 M^h P = v_0 P^h$ and note that $\|v_1\|_1 = \|P^{ht} v_0^T\|_1 \leq \|v_0\|_1 \leq 2\epsilon' + \epsilon''$. And so,

$$v_2^T = v_1^T P M^h = h^T + (v_0 + v_1) M^h$$

Generalizing this argument to k rounds,

$$v_k^T = h^T + (v_0 + v_1 + ... + v_{k-1}) M^h$$

And as the operator norm of M^h is bounded by 1, the norm 1 of v_k is also bounded by $k(2\epsilon' + \epsilon'')$. And so, by the mixing assumption we made previously, $\|v_k - \mu_h\|_1 \leq 2e^{-k/\tau(h)}$. The best choice of k in such a case would be $\tau(h)\log(1/(2\epsilon' + \epsilon''))$.

Theorem 4.3. *Let Z be a positive constant and $\mathcal{Z}$ by a bounded convex subset of $\mathbb{R}^d$ such that for any $z \in \mathcal{Z}$, $\|z\| \leq Z$. Let $(f_t)_{t=1,2,...,T}$ be a sequcne of real-valued convex cost functions defined over $\mathcal{Z}$. Let $z_1, z_2, ..., z_T \in \mathcal{Z}$ be defined by $z_1 = 0$ and $z_{t+1} = \Pi_{\mathcal{Z}}(z_t - \eta f_t')$, where $\Pi_{\mathcal{Z}}$ is the Euclidean projection onto $\mathcal{Z}$, $\eta > 0$ is a learning rate, and $f_1', f_2', ..$ are unbiased subgradient estimates such that $E[f_t'|z_t] = \nabla f(z_t)$ and $\|f_t'\| \leq F$ for some $F > 0$. Then, for $\eta = Z/(F\sqrt{T})$, for any $\delta \in (0,1)$, with probability at least $1 - \delta$,*

$$\sum_{t=1}^{T} f_t(z_t) - \min_{z \in \mathcal{Z}} \sum_{t=1}^{T} f_t(z) \leq ZF\sqrt{T} + \sqrt{(1 + 4Z^2T)\left(2\log\frac{1}{\delta} + d\log\left(1 + \frac{Z^2T}{d}\right)\right)} \tag{4.5}$$

The proof of the above theorem can be found in Flaxman, et al. [4].

Lemma 4.4. *Under the same conditions as in the very first theorem, we now have for any $\delta \in (0,1)$, with probability $1 - \delta$*

$$c(\hat{\theta}_T) - \min_{\theta \in \Theta} c(\theta) \leq \frac{SG'}{\sqrt{T}} + \sqrt{\frac{1 + 4S^2T}{T^2}\left(2\log\frac{1}{\delta} + d\log\left(1 + \frac{1 + S^2T}{d}\right)\right)} \tag{4.6}$$

Using the above proved lemma's and theorems, we can finally get back to proving the main theorem. *Proof.* Let b_T be the RHS of eq. (4.6). Then, with high probability for $\theta \in \Theta$,

$$l^T(\mu_0 + \Phi\hat{\theta}_T) + HV_1(\hat{\theta}_T) + HV_2(\hat{\theta}_T) \leq l^T(\mu_0 + \phi\theta) + HV_1(\theta) + HV_2(\theta) + b_T. \tag{4.7}$$

From eq. (4.7) we have :

$$V_1(\hat{\theta}_T) \leq \frac{1}{H}\left(2(1 + S\sqrt{d}) + HV_1(\theta) + HV_2(\theta) + b_T\right) = O(\epsilon')$$
$$V_2(\hat{\theta}_T) \leq \frac{1}{H}\left(2(1 + S\sqrt{d}) + HV_1(\theta) + HV_2(\theta) + b_T\right) = O(\epsilon'')$$

From the above inequalities, we have :

$$\left|\mu_{\hat{\theta}_T}^T l - (\mu_0 + \Phi\hat{\theta})^T l\right| \leq (\tau(\mu_{\hat{\theta}_T})\log(1/(2\epsilon' + \epsilon'')) + 2)(2\epsilon' + \epsilon'') \tag{4.8}$$

From eq. (4.7) we can also say that :

$$l^T(\mu_0 + \Phi\hat{\theta}_T) \le l^T(\mu_0 + \Phi\theta) + HV_1(\theta) + HV_2(\theta) + b_T$$

Therefore, finally

$$\begin{aligned}\mu_{\hat{\theta}_T}^T l &\le l^T(\mu_0 + \Phi\theta) + HV_1(\theta) + HV_2(\theta) + b_T \\ &+ (\tau(\mu_{\hat{\theta}_T})\log(1/(2\epsilon' + \epsilon'')) + 2)(2\epsilon' + \epsilon'') \\ &\le +\mu_\theta^T l + HV_1(\theta) + HV_2(\theta) + b_T + (\tau(\mu_{\hat{\theta}_T})\log(1/(2\epsilon' + \epsilon'')) + 2)(2\epsilon' + \epsilon'') \\ &+ (\tau(\mu_\theta)\log(1/(2V_1(\theta) + V_2(\theta)))) \times (2V_1(\theta) + V_2(\theta)).\end{aligned}$$

As we have $b_T = O(H/\sqrt{T})$ and $H = 1/\epsilon$ and $T = 1/\epsilon^4$, we get that with high probability of $1-\delta$, for any $\theta \in \Theta$, $\mu_{\hat{\theta}_T}^T l \le \mu_\theta^T l + O(\frac{1}{\epsilon}(V_1(\theta) + V_2(\theta))) + O(\epsilon)$.

The important point to note is that this bound is similar to a previous bound by de Farias and Van Roy (2006), although their algorithm wasn't as computationaly feasible as this one.

5. Approximate Value Function

Here we consider the class of MDPs given by KL loss function defined in eq. (1.4) with *total cost*. With KL Loss, the calculation of greedy policy becomes a linear computation. This way, the bellman operator given in eq. (2.1) also reduces to a much simpler version of itself which is given in eq. (5.1).

$$\begin{aligned}(Lh)(x) &= \min_{P\in\mathcal{P}}\left\{ l(x,P) + \sum_{x'\in\mathcal{X}} P(x,x')h(x') \right\} \\ &= \min_{P\in\mathcal{P}}\left\{ q(x) + \sum_{x'\in\mathcal{X}} P(x,x')\left[h(x') + \log\frac{P(x,x')}{P_0(x,x')}\right] \right\} \\ &= q(x) + \min_{P\in\mathcal{P}}\left\{ \sum_{x'\in\mathcal{X}} P(x,x')\log\frac{P(x,x')}{P_0(x,x')e^{-h(x')}} \right\}\end{aligned}$$

$$(Lh)(x) = q(x) - \log Z(x) \tag{5.1}$$

$$\text{where, } Z(x) = \sum_{x'\in\mathcal{X}} P_0(x,x')e^{-h(x')}$$

Here, $P(x,x') = \frac{P_0(x,x')e^{-h(x')}}{Z(x)}$ minimizes the above summation and can easily be derived given that the KL divergence of two equal distributions is 0. When we exponentiate the bellman optimality equation with the above bellman operator, it becomes linear opening a gate to plethora of methods to solve the MDP. This formulization is given in eq. (5.2).

$$e^{-h(x)} = e^{-q(x)} \sum_{x'\in\mathcal{X}} P_0(x,x')e^{-h(x')} \tag{5.2}$$

The above bellman optimality equation is easier to solve but still computationally inefficient when the state space $\mathcal{X}$ becomes too large and requires one to solve an eigenvalue decomposition in $|\mathcal{X}|$ dimensions. Thus, we parameterize the value function to smaller dimension and optimize giving us a best policy in the *restricted* class.

5.1. **Family of Value Functions.** We consider the class of value functions parameterized via $w \in \mathbb{R}^d$ where $d \ll |\mathcal{X}|$. This family of value function is given by $\mathcal{H}$ and defined in eq. (5.3).

$$\mathcal{H} = \left\{x \mapsto h_w(x) := -\log(\Psi(x,:)w) : w \in \mathcal{W} \text{ and } \mathcal{W} \subset \mathbb{R}^d\right\} \tag{5.3}$$

Here $\Psi \in \mathbb{R}^{|\mathcal{X}|\times d}$ is feature matrix which makes the parametrization different from ALP and ADP using a linear combination of basis functions. Also, we make a positivity assumption that there exists a $g > 0$ such that $\forall\ x \in \mathcal{X}$, $\Psi(x,:)w \geq g$.

5.2. **Problem Reformulation.** From the bellman optimality equation, we know that we need to minimize $h^*(x)$ such that $h(x) - (Lh)(x) = 0\ \forall\ x \in \mathcal{X}$. This difference $|h - Lh|$ is known as the *bellman gap or error.* To find a good value function, one tries to reduce the bellman gap as much as possible. From methods like policy iteration and value iteration, we know that keeping the bellman gap small gives a good approximation to the value function which is exploited in this method [5]. Also, minimizing the exponentiated bellman gap reduces the bellman gap as well as,

$$e^{-\max\{a,a'\}}|a - a'| \leq |e^{-a} - e^{-a'}| \leq e^{-\min\{a,a'\}}|a - a'|$$

Thus, we search for parameter $\hat{w} \in \mathcal{W}$ such that $|e^{-Lh_{\hat{w}}} - e^{-h_{\hat{w}}}| < \epsilon$ for some small $\epsilon > 0$.

5.3. **Convex Cost.** To recast our problem into a convex optimization problem, we devise a convex cost as given in eq. (5.4) which is an estimate of the total bellman gap and a regularization term. To sum over all the bellman errors, we take $\mathcal{T}$ to be the space of trajectories starting at some state x_1 and ending at an absorbing state $z \in S$. We also take a distribution $s(T)$ a probability distribution over all the trajectories $T \in \mathcal{T}$. For a positive constant H, we try to minimize the cost given in eq. (5.4) using *stochastic sub-gradient descent.*

$$\begin{aligned} c(w) = &-\log(\Psi(x_1,:)w) \\ &+ H\sum_{T\in\mathcal{T}} s(T) \sum_{x\in T} \left|\Psi(x,:)w - e^{-q(x)}P_0(x,:)\Psi w\right| \end{aligned} \tag{5.4}$$

5.4. **Efficient Subgradient Computation.** We can easily calculate the gradient of the above cost function and is given in eq. (5.5). But, summing over the all the trajectories of a MDP is computationally intractable for very large MDPs. To get an unbiased estimate of the gradient we sample an episode of the MDP (or a part of the full episode) at every iteration and sum over only the states seen in that one trajectory $T_{\text{sampled}} \sim s$. This unbiased estimate is given in eq. (5.6).

$$\begin{aligned}\nabla c(w) = &-\left(\frac{1}{(\Psi(x_1,:)w)}\right)\Psi(x_1,:) \\ &+ H\sum_{T\in\mathcal{T}} s(T)\sum_{x\in T}\Big[\text{sign}\Big(\Psi(x,:)w - e^{-q(x)}P_0(z,:)\Psi w\Big) \\ &\Big(\Psi(x,:) - e^{-q(x)}P_0(z,:)\Psi\Big)\Big]\end{aligned} \tag{5.5}$$

$$\begin{aligned}\nabla c(w) = &-\left(\frac{1}{(\Psi(x_1,:)w)}\right)\Psi(x_1,:) \\ &+ H\sum_{x\in T_{\text{sampled}}}\Big[\text{sign}\Big(\Psi(x,:)w - e^{-q(x)}P_0(z,:)\Psi w\Big) \\ &\Big(\Psi(x,:) - e^{-q(x)}P_0(z,:)\Psi\Big)\Big]\end{aligned} \tag{5.6}$$

With an efficient way to calculate the subgradient, we can use methods like stochastic subgradient descent methods which give us an ϵ-optimal solution in only $O(\frac{1}{\epsilon^2})$ iterations with high-probability. Thus, the method only scales with the number of dimensions $d \ll |\mathcal{X}|$.

5.5. **Analysis.** Here we prove that finding an ϵ-optimal solution to the convex optimization cost in eq. (5.4) produces a sub-optimal policy. i.e., getting an ϵ-optimal $\hat{w}$ such that $c(\hat{w}) \le c(w) + O(\epsilon)$ for any $w \in \mathcal{W}$ bounds the gap between $h_{\hat{w}}$ and h_w. This is given in theorem 5.1.

Theorem 5.1. *Assume that $\hat{w}$ is ϵ-optimal and choose any $H \ge e^{Q-\log g}$ where $\Psi(x,:)w \ge g \forall\ x \in \mathcal{X}$. Then, for any $w \in \mathcal{W}$ with $l_w = \min(h_w, Lh_w)$, we have,*

$$\begin{aligned}h_{P_{h_{\hat{w}}}}(x_1) - h_{P_{h_w}}(x_1) \le{}& \epsilon \\ &+ \|P_{h_{\hat{w}}} - s\|_1 \max_{T\in\tau}\sum_{xinT}|h_{\hat{w}}(x) - Lh_{\hat{w}}(x)| \\ &+ \sum_{T\in\tau} P_{h_w}(T)\sum_{x\in T}|h_w(x) - Lh_w(x)| \\ &+ H\sum_{T\in\tau} s(T)\sum_{x\in T} e^{l_w(x)}|h_w(x) - Lh_w(x)|\end{aligned}$$

where, with an abuse of notation, $P_h(T)$ denotes the probability of trajectory T under transition dynamics P_h.

Proof. We see that

$$h_{P_h}(x_1) - h(x_1) = \sum_{T\in\mathcal{T}} P_h(T)\sum_{x\in T}(Lh - h)(x) \tag{5.7}$$

Now,

$$\begin{aligned}(Lh_{\hat{w}})(x) &= q(x) - \log Z(x) \\ &\le Q - \log g \qquad\qquad (\Psi(x,:)w \le g) \\ \Rightarrow \max\{h_{\hat{w}}(x), (Lh_{\hat{w}})(x)\} &\le Q - \log g\end{aligned} \tag{5.8}$$

We know that $\hat{w}$ is an ϵ-optimal solution, thus,

$$\begin{aligned} & h_{\hat{w}}(x_1) + H\sum_{T\in\mathcal{T}} s(T)\sum_{x\in T}\left|\Psi(x,:)w - e^{-q(x)}P_0(x,:)\Psi w\right| \leq \\ & \qquad h_w(x_1) + H\sum_{T\in\mathcal{T}} s(T)\sum_{x\in T}\left|\Psi(x,:)w - e^{-q(x)}P_0(x,:)\Psi w\right| + \epsilon \\ \Rightarrow\ & h_{\hat{w}}(x_1) + He^{-Q+\log g}\sum_{T\in\mathcal{T}} s(T)\sum_{x\in T}\left|h_{\hat{w}}(x) - Lh_{\hat{w}}(x)\right| \leq \\ & \qquad h_w(x_1) + H\sum_{T\in\mathcal{T}} s(T)\sum_{x\in T} e^{-l_w(x)}\left|h_w(x) - Lh_w(x)\right| + \epsilon \end{aligned}$$

By eq. (5.8) and above,

$$\begin{aligned} h_{P_{h_{\hat{w}}}}(x_1) - h_{P_{h_w}}(x_1) \leq\ & \epsilon + \sum_{T\in\mathcal{T}}(P_{h_{\hat{w}}}(T) - s(T))\sum_{x\in T}|h_{\hat{w}}(x) - Lh_{\hat{w}}(x)| \\ & + H\sum_{T\in\mathcal{T}} s(T)\sum_{x\in T} e^{-l_w(x)}|h_w(x) - Lh_w(x)| \\ & + \sum_{T\in\mathcal{T}} P_{h_w}(T)\sum_{x\in T}|h_w(x) - Lh_w(x)| \\ \leq\ & \epsilon + \|P_{h_{\hat{w}}} - s\|_1 \max_{T\in\tau}\sum_{xinT}|h_{\hat{w}}(x) - Lh_{\hat{w}}(x)| \\ & + \sum_{T\in\tau} P_{h_w}(T)\sum_{x\in T}|h_w(x) - Lh_w(x)| \\ & + H\sum_{T\in\tau} s(T)\sum_{x\in T} e^{l_w(x)}|h_w(x) - Lh_w(x)| \end{aligned}$$

□

Other than the approximation error ϵ, the second term is the difference between sampling distribution s and transition dynamics P_{h_w}. The other terms are the difference between the expected bellman error and bellman error under transition dynamics P_{h_w}. Thus, all terms except ϵ are small if $h_w \in \mathcal{H}$ gives a small bellman error.

6. Conclusion

In this report we presented a way to solve MDPs with ergodic cost and a large state space by approximating the stationary distribution over state-action pairs and converting it into a convex optimization problem. This method proves to give a competitive optimal policy in the lower-dimensional class of policies considered. Also, we studied a special kind of MDP with KL total cost which produces a linear bellman optimality equation and solved it in the class of lower-dimensional value functions. This report outlines a basic framework of solving MDPs with approximation methods where complexity does not scale with the size of state-space. Essentially by taking a lower-dimensional class of policies and recasting the bellman optimality problem into a convex optimization problem gives a competitive policy in the subset of policies considered.

7. Future Work

If we look at the problem of minimizing total cost with absorbing states, we can rephrase this problem as follows: Consider a graph with 1 node for each state $x_i \in \mathcal{X}$. Additionally, consider $|A|$ nodes adjacent to each state node. The edge between the state node and this 'action node'$(a(x_i)$ where $a \in \mathcal{A})$ should have a cost of $l(x, a)$ and have a probability of 1. From this action node, add transitions to all state nodes with transition probability $p(x_i, a)$ and a cost of 0. The problem now is to find the expected shortest path from the starting node to any of the absorbing nodes in this probabilitic graph. Such problems have some work done on them in the past [6]. If we just reverse all edges in this graph and find the shortest path from the absorbing state to all states, we will implicitly also have the policy for each state to get to the absorbing state quickly. It would be interesting to see whether such an approach is actually efficient in practice and if we can provide a efficient theoretical bound on the time and error terms here.

References

[1] Vijay V Desai, Vivek F Farias, and Ciamac C Moallemi. Approximate dynamic programming via a smoothed linear program. *Operations Research*, 60(3):655–674, 2012.

[2] Emanuel Todorov et al. Linearly-solvable markov decision problems. In *NIPS*, pages 1369–1376, 2006.

[3] Richard E Bellman and Stuart E Dreyfus. *Applied dynamic programming*. Princeton university press, 2015.

[4] Abraham Flaxman, Adam Kalai, and Brendan McMahan. Online convex optimization in the bandit setting: Gradient descent without a gradient. In *Flaxman, et al. (2005)*, pages 385–394, January 2005.

[5] Yasin Abbasi-Yadkori, Peter Bartlett, Xi Chen, and Alan Malek. Large-scale markov decision problems with kl control cost and its application to crowdsourcing. In *International Conference on Machine Learning*, pages 1053–1062, 2015.

[6] Dimitri P Bertsekas, John N Tsitsiklis, et al. An analysis of stochastic shortest path problems. 1988.

[7] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction, 2011.

[8] Yasin Abbasi-Yadkori, Peter Bartlett, and Alan Malek. Linear programming for large-scale markov decision problems. In *Proceedings of the International Conference on Machine Learning*, 2014.

[9] Alan S Manne. Linear programming and sequential decisions. *Management Science*, 6(3):259–267, 1960.

Courant Institute of Mathematical Science, New York University, New York
E-mail address: `chirag.m@nyu.edu`

Courant Institute of Mathematical Science, New York University, New York
E-mail address: `oad230@nyu.edu`