

Stochastic Control

Neil Walton

January 29, 2021

Contents

0	Optimal Control	4
0.1	Dynamic Programming	5
0.2	Markov Chains with Rewards	17
0.3	Markov Decision Processes	29
0.4	Infinite Time Horizon	37
0.5	Algorithms for MDPs	55
0.6	Optimal Stopping	73
0.7	Inventory Control.	79
0.8	Partially Observable MDPs	85
0.9	LQR and the Kalman Filter	90
1	Continuous Time Control	102
1.1	Continuous Time Dynamic Programming	103
1.2	Calculus of Variations	108
1.3	Pontyagin's Maximum Principle	113
1.4	Stochastic Integration	115
1.5	Diffusion Control Problems	117
1.6	Merton Portfolio Optimization	121
2	Stochastic Approximation	134
2.1	Lyapunov Functions	135
2.2	Robbins-Monro.	148
2.3	Stochastic Gradient Decent	157
2.4	Asynchronous Update	164
2.5	ODE method for Stochastic Approximation	167
2.6	Online Convex Optimization	173
3	Bandits and Experts	177
3.1	Stochastic Multi-Armed Bandit	178
3.2	Upper Confidence Bound.	183
3.3	Lai and Robbins Lower Bound.	186

3.4	Gittins' Index Theorem*	190
3.5	Non-Stochastic Multi-armed Bandits*	195
3.6	Stochastic Regression	200
4	Tabular Reinforcement Learning	207
4.1	Principles of Reinforcement Learning	208
4.2	Policy Evaluation: MC and TD methods	212
4.3	Q-learning	224
4.4	SARSA	230
5	Reinforcement Learning with Function Approximation	233
5.1	Temporal Differences: Linear Approximation	234
5.2	Policy Gradients	239
5.3	Linear Approximation and TD Learning	247
5.4	Linear Approximation and Stopping	253
5.5	Cross Entropy Method	260
6	Reinforcement Learning with Neural Networks	263
6.1	Deep Q-Network (DQN)	264
A	Appendix	267
A.1	Probability	267
A.2	Stochastic Integration	272
A.3	Gronwall's Lemma	273
A.4	Projections	275
A.5	Misc Results	275
A.6	Utility Theory	276
A.7	Lagrangian Optimization and Duality	278
A.8	Linear Algebra	282
A.9	Random Matrices	288
B	Function Approximation	292
B.1	Overview of Statistical Learning	293
B.2	Linear Regression	293
B.3	Training, Development and Test sets	296
B.4	Bias and Variance	299
B.5	Neural Networks	304

Chapter 0

Optimal Control

- Dynamic Programs; Markov Decision Processes; Bellman's Equation; Complexity aspects.
 - Markov chains; Markov property; Summing Markov chain Rewards; Convergence to Equilibrium.
 - Infinite Time Horizon Control: Positive, Discounted and Negative Programming; Occupancy Measure.
 - Algorithms: Policy Improvement & Policy evaluation; Value Iteration; Policy Iteration; Linear Program formulation; Asynchronous Value Iteration.
 - Optimal Stopping; One-Step-Look-Ahead; Excessive Majorant.
 - Inventory Control; (s, S) policy.
 - POMDP; Belief states.
 - Linear-Quadratic Regularization (LQR); Riccati Equation; LQG; Certainty Equivalence; Kalman Filter.
-

0.1 Dynamic Programming

- Dynamic Program (Definition).
 - Bellman's Equation.
 - Principle of Optimality.
 - Curse of Dimensionality.
-

The Basic Idea.

Let's discuss the basic form of the problems that we want to solve. See Figure 1. Here there is a controller (in this case for a computer

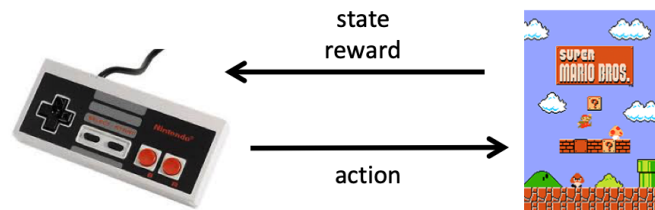


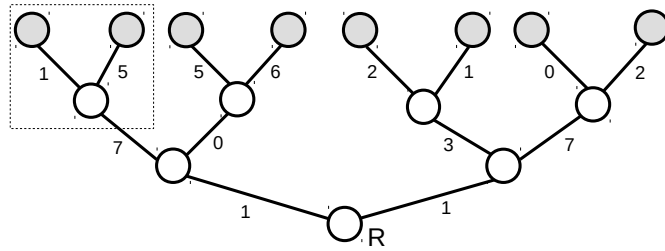
Figure 1: A control loop.

game). It sends actions to an environment (in this case the computer) which then returns its current state and a reward. Based on this, the controller selects a new action; the environment then returns its next state and reward and so on it goes. We want to find a sequence of actions that maximizes the sum of these rewards.

This interaction between states, actions and rewards are the key building blocks of dynamic programming, Markov decision processes, and much of the topics in rest of these notes. In each of these, our task is to optimize the sequence of rewards that we receive over time.

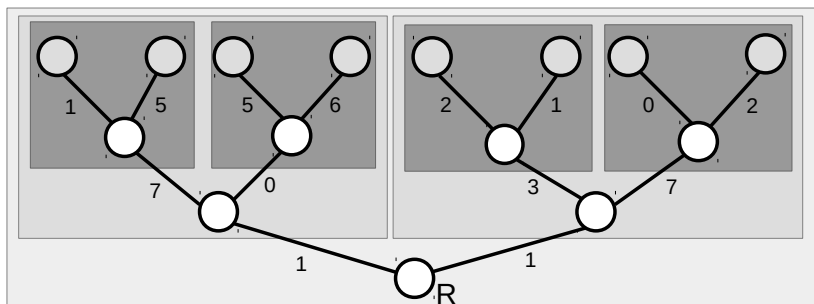
An Introductory Example

Let's solve a simple dynamic program. In the figure below there is a tree consisting of a root node labelled *R* and two leaf nodes colored grey. For each edge, there is a cost. Your task is to find the lowest cost path from the root node to a leaf.



There are a number of ways to solve this, such as enumerating all paths. However, we are interested in one approach where the problem is solved backwards, through a sequence of smaller sub-problems. Specifically, once we reach the penultimate node on the left (in the dashed box) then it is clearly optimal to go left with a cost of 1. This solves an easier sub problem and, after solving each sub problem, we can then attack a slightly bigger problem. If we solve for each leaf in this way we can solve the problem for the antepenultimate nodes (the node before the penultimate node).

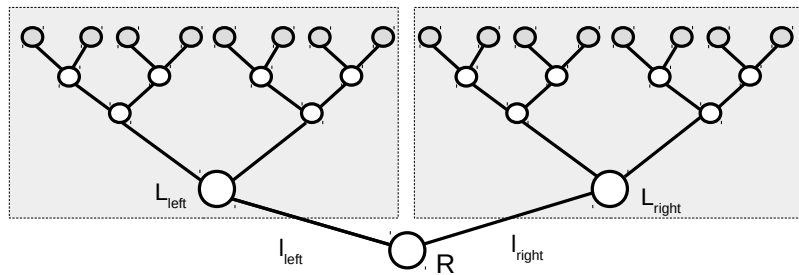
Thus the problem of optimizing the cost of the original tree can be broken down to a sequence of much simpler optimizations given by the shaded boxed below.



From this we see the optimal path has a cost of 5 and consists of going right, then left, then right.

Let's consider the problem a little more generally in the next figure. The tree on the righthand-side has a lowest cost path of value L_{rhs} and the lefthand-side tree has lowest cost L_{lhs} and the edges leading to each, respective tree, have costs l_{rhs} and l_{lhs} . Once the decision to go left or right is made (at cost l_{rhs} or l_{lhs}) it is optimal to follow the lowest cost path (at cost L_{rhs} or L_{lhs}). So L , the minimal cost path from the root to a leaf node satisfies

$$L = \min_{a \in \{lhs, rhs\}} \{l_a + L_a\}.$$



Similarly, convince yourself that the same argument applies from any node x in the tree network that is

$$L_x = \min_{a \in \{lhs, rhs\}} \{l_a + L_{x(a)}\}.$$

where L_x is the minimum cost from x to a leaf node and where for $a \in \{lhs, rhs\}$ $x(a)$ is the node to the lefthand-side or righthand-side of x . The equation above is an example of the *Bellman equation* for this problem, and in our example, we solved this equation recursively starting from leaf nodes and working our way back to the root node.

The idea of solving a problem from back to front and the idea of iterating on the above equation to solve an optimisation problem lies at the heart of dynamic programming.

Definitions for Dynamic Programming

We now give a general definition of a *dynamic programming*.

States, actions, rewards and next states. We consider a discrete, finite set of times $t = 0, 1, \dots, T$. We let x and \mathcal{X} denote the state and set of states. We let $x_t \in \mathcal{X}$ be the state of our dynamic program at time t . We let a and \mathcal{A} denote an action and set of actions. The (instantaneous) reward for taking action a in state x is $r(a, x)$. Further, $r(x)$ is the reward for terminating in state x at time T .¹ If action a is taken when in state x then the next state in \mathcal{X} , which we denote by \hat{x} , is given by

$$\hat{x} = f(x, a), \quad (\text{Plant eq})$$

for a function $f : \mathcal{X} \times \mathcal{A} \rightarrow \mathcal{X}$. This is sometimes called the plant equation of the dynamic program.

Policy, cumulative reward and value function. A policy $\pi = (\pi_t : t = 0, \dots, T-1)$ chooses an action π_t at each time $t = 0, 1, \dots, T-1$. Starting from an initial state x_0 , a the policy gives a sequence of states

$$x_{t+1} = f(x_t, \pi_t). \quad (1)$$

We evaluate how a good each policy is by the sum of its rewards:

$$R(x_0, \pi) := r(x_0, \pi_0) + r(x_1, \pi_1) + \dots + r(x_{T-1}, \pi_{T-1}) + r(x_T)$$

Given the cumulative reward function $R(x, \pi)$, we define the value function to be the maximum reward:

$$V(x_0) = \max_{\pi} R(x_0, \pi).$$

The main objective of dynamic programming is to solve this optimization. To do this, it helps to consider the future rewards and value after each time t , which we respectively define by

$$R_t(x_t, \pi) := \sum_{s=T-t}^{T-1} r(x_s, \pi_s) + r(x_T), \quad V_t(x_t) := \max_{\pi} R_t(x_t, \pi).$$

Dynamic Program Definition. We summarize the discussion above with the following definition.

¹Since we do not allow further actions from time T onwards, we can ignore the dependence on a .

Def 1 (Dynamic Program). *Given initial state x_0 , a dynamic program is the optimization*

$$\begin{aligned} V(x_0) := \text{Maximize} \quad & R(x_0, \pi) := \sum_{t=0}^{T-1} r(x_t, \pi_t) + r_T(x_T) && \text{(DP)} \\ \text{subject to} \quad & x_{t+1} = f(x_t, \pi_t), && t = 0, \dots, T-1 \\ \text{over} \quad & \pi_t \in \mathcal{A}, && t = 0, \dots, T-1 \end{aligned}$$

Further, let $R_\tau(x_\tau, \pi)$ (Resp. $V_\tau(x_\tau)$) be the objective (Resp. optimal objective) for (DP) when the summation is started from $t = T - \tau$, rather than $t = 0$.

The Bellman Equation

In our introductory example, we saw we could solve a dynamic program by a sequence of much simpler optimizations. The resulting sequence of equations is called the Bellman Equation. The Bellman equation is central to the study of control problems. The following result gives shows the optimality of the Bellman Equation for dynamic programming.

Thrm 2 (Bellman's Equation). $V_0(x) = r_T(x)$ and for $t = T - 1, \dots, 0$

$$V_t(x) = \max_{a \in \mathcal{A}} \{r(x, a) + V_{t-1}(\hat{x})\}, \quad \text{(Bell eq)}$$

where $x \in \mathcal{X}$ and $\hat{x} = f(x, a)$.

Proof. Let $\pi_t := (\pi_{T-t}, \dots, \pi_{T-1})$. Note that $R_t(x, \pi_t) = r(x, \pi_{T-t}) + R_{t-1}(\hat{x}, \pi_{t-1})$.

$$\begin{aligned} V_t(x) &= \max_{\pi_t} \{R_t(x, \pi_t)\} = \max_a \max_{\pi_{t-1}} \{r(x, a) + R_{t-1}(\hat{x}, \pi_{t-1})\} \\ &= \max_a \left\{ r(x, a) + \max_{\pi_{t-1}} R_{t-1}(\hat{x}, \pi_{t-1}) \right\} = \max_a \{r(x, a) + V_{t-1}(\hat{x})\}. \end{aligned}$$

□

Some Code

Below is some code (in Python) to solve a dynamic program. Notice we can solve the dynamic program by repeatedly calling the function DP() until the solution is trivial at time=0.

```

def DP(time , state , f , r , A) :
    ...
    Solves a dynamic program
    ...
    if time > 0 :
        Q = [ r[state][action] + DP(time-1, f[state][action]) for
              action in A ]
        V = max(Q)
    else :
        Q = r[state]
        V = max(Q)
    return V

```

The Principle of Optimality

Dynamic programming and the Bellman equation was invented by Richard Bellman. Bellman concisely summarizes why and when we expect the Bellman equation to hold for an optimization problem:

“Principle of Optimality: An optimal policy has the property that whatever the initial state and initial decisions are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decisions.” – Richard Bellman [4]

A good example is to see what this means is to consider shortest paths and longest paths in an undirected graph. See Figure 2. The shortest path from S to D is colored grey. Notice that this contains the shortest path from B to D , i.e. once we get from S to B what remains of the optimal solution is to take the shortest path from B to D . Here we see that shortest-path problems satisfy the principle of optimality. So, we can apply the dynamic programming and the Bellman equation to solve shortest path problems.

Notice, however, the longest path (without loops) from D to S contains B , but this does not take the longest path from B to S . So we cannot apply dynamic programming to solve longest path problems on an (undirected) graph.

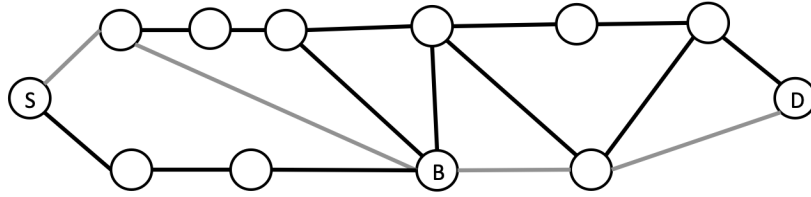


Figure 2: Shortest path from S to D.

The Curse of Dimensionality

Another term coined by Bellman is *The Curse of Dimensionality*. Although this used as a generic term applicable to many algorithms, it is particularly true of dynamic programming. Essentially the point is that as the size of a dynamic programming problem grows – in terms of the number of states and its time horizon – then the computation required to solve the optimization grows in unreasonably rapidly and usually exponentially. E.g. for our introductory example of a tree, if we let there be n nodes that can be reached after each action and let the depth of the tree be T , then the number of Bellman equations that we need to solve to find the minimum cost path from the root node is, roughly, of the order of n^T .

Thus to apply the dynamic programming, we need to either consider optimization problems that have additional structure that makes them easier to solve, or we need to find ways to approximating the solution to these problems. We will consider both of these approaches in these notes.

Other Observations

Let's list a few more properties and smaller observations about dynamic programming.

Minimization. Notice we can change the definition of a dynamic program to minimize costs $c(x, a)$ rather than maximizing rewards $r(x, a)$. You can check that, in this case, the Bellman equation becomes

$$L_t(x) = \min_{a \in \mathcal{A}} \{c(x, a) + L_{t-1}(\hat{x})\}.$$

General functions. So far we have assumed that the transition and reward function depends only on the current state x and the

action taken a . Notice that the Bellman equation holds equally when we consider rewards that depend on time t and the next state \hat{x} also. Also we can let the transition function f and the set of actions depend on time t . This gives the Bellman equation

$$V_t(x) = \max_{a \in \mathcal{A}_t} \{r_t(x, a, \hat{x}) + V_{t+1}(\hat{x})\}$$

where $\hat{x} = f_t(x, a)$. (You can check that the above “more general” formulation and definition given previously are equivalent by an appropriate choice of state space, action space and rewards.)

Dynamic Programming Examples

Ex 3. An investor has a fund. It has x pounds at time zero. Money can't be withdrawn. It pays $r \times 100\%$ interest per-year for T years. The investor consumes proportion a_t of the interest and reinvests the rest. What should the investor do to maximize consumption?

Ex 4. You invest in properties. The total value of these properties is x_t in year $t = 1, \dots, T$. Each year t , you gain rent of rx_t and you choose to consume a proportion $a_t \in [0, 1]$ of this rent. The remaining proportion is reinvested in buying new property. Further you pay mortgage payments of mx_t which are deducted from your consumed wealth. Here $m < r$. Your objective is to maximize the wealth consumed over T years. Prove that if $W_{T-s}(x) = x\rho_s$ for some constant ρ_s then

$$\rho_s = \max\{r - m + \rho_{s-1}, (1 + r)\rho_{s-1} - m\}.$$

Ex 5 (Shortest Paths). Consider a directed graph $G = (V, E)$ each edge has a cost, c_{ij} for each $(i, j) \in E$. Take a vertex d . Let L_i be the length of the shorest path from i to d and let $L_i(t)$ be the shortest path from i to d that uses t steps.

i) Argue that $L_i(t)$ satisfies, $L_d(t) = 0$ and

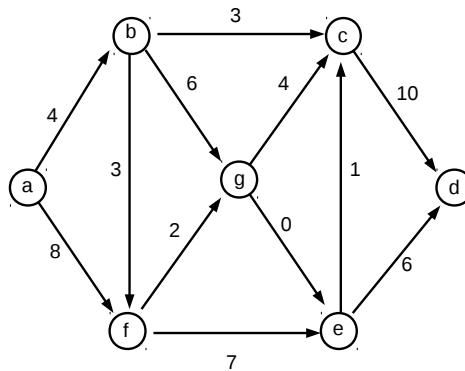
$$L_i(t + 1) = \min_{j:(i,j) \in E} \{c_{ij} + L_j(t)\}, \quad \text{for } i \neq d.$$

(Here you may assume that $L_d(t) = 0$ for all $t \geq 0$ and $L_i(t) = \infty$ unless assigned a value in the above set of equalities.)

ii) In addition to satisfying $L_d = 0$, argue that L_i satisfies the equations

$$L_i = \min_{j:(i,j) \in E} \{c_{ij} + L_j\}, \quad \text{for } i \neq d.$$

iii) Your answer to part ii) describes a algorithm called the Bellman-Ford algorithm. Use it to find the shortest path from node a to node d in the following graph



Ex 6 (Scheduling). There are N appointments that need to be successively scheduled over time. Each appointment $i = 1, \dots, N$ requires t_i units of time and when completed has reward r_i . Given discount factor $\beta \in (0, 1)$, the total reward arraigning the appointments in order $1, 2, \dots, N$ is

$$R(1, \dots, N) = r_1\beta^{t_1} + r_2\beta^{t_1+t_2} + \dots + r_N\beta^{t_1+\dots+t_N},$$

i) Write down a dynamic program for the optimal discounted reward. (Here let $W(S)$ be the optimal reward when $S \subset \{1, \dots, N\}$ is the remaining set of unassigned appointments.)

ii) Argue that it is optimal to order appointments so that the indices

$$G_i = \frac{r_i\beta^{t_i}(1 - \beta)}{1 - \beta^{t_i}}$$

indexed from highest to lowest.

Ex 7 (Discrete time LQ-regularization). We consider discrete time LQ minimization, here you minimize the objective

$$\begin{aligned} \min_{\mathbf{a}_0, \dots, \mathbf{a}_{T-1}} \quad & \mathbf{x}_T^\top R \mathbf{x}_T + \sum_{t=0}^{T-1} [\mathbf{x}_t^\top R \mathbf{x}_t + \mathbf{a}_t^\top Q \mathbf{a}_t] \\ \text{subject to} \quad & \mathbf{x}_{t+1} = A \mathbf{x}_t + B_t \mathbf{a}_t, \quad t = 0, \dots, T-1 \end{aligned}$$

Here R and Q are positive semi-definite matrices.

i) Show that the Bellman equation for this dynamic program is

$$L_t(\mathbf{x}) = \min_{\mathbf{a}} \{ \mathbf{x}^\top R \mathbf{x} + \mathbf{a}^\top Q \mathbf{a} + L_{t+1}(A \mathbf{x} + B \mathbf{a}) \}$$

ii) Assuming the solution is of the form $L_t(\mathbf{x}) = \mathbf{x}^\top \Lambda_t \mathbf{x}$ find the action that \mathbf{a} minimizes the above Bellman equation is given by

$$\mathbf{a} = (Q + B^\top \Lambda_t B)^{-1} B^\top \Lambda_t A \mathbf{x}$$

iii) Using your answer to Part ii), show that

$$\Lambda_{t-1} = R + A'^\top \Lambda_t A - (A^\top \Lambda_t B)(Q + B' \Lambda_t B)^{-1} B' \Lambda_t A.$$

This is the Riccati Recursion (the discrete time analogue of the Riccati equation).

Ex 8 (Critical Path Analysis / Longest Paths). A project consists of a number of tasks that must be completed in a specified order. This is represented by a directed acyclic graph $G = (V, E)$. Each task $j \in V$ takes an amount of time $c(j)$ to complete and the task cannot be started until all its parent tasks $\text{par}(j) = \{i : (i, j) \in E\}$ have been completed. We also let $\text{ch}(j) = \{k : (j, k) \in E\}$ be the children of task j .

We let $\text{EST}(j)$ and $\text{EFT}(j)$ represent the earliest start time and earliest finish time for task j .

i) Show that

$$\text{EST}(j) = \max_{i \in \text{par}(j)} \{c(i) + \text{EST}(i)\},$$

$$\text{EFT}(j) = c(j) + \max_{i \in \text{par}(j)} \text{EFT}(i),$$

with $\text{EST}(j) = 0$ and $\text{EFT}(j) = c(j)$ if $\text{par}(j)$ is empty.

ii) Let $L = \max_j EFT(j)$, the time that the project is completed. We let $LST(j)$ and $LFT(j)$ be the latest start time and latest finish time for task j (where the project is completed at time L). Give the equivalent expressions to those found in part i), for the latests start and finish times.

The critical path is said to be the set of tasks such that

$$EST(i) = LST(i), \quad EFT(i) = LFT(i)$$

iii) Find the earliest start and finish times, and then find the latest start and finish time for the following example:

Task	Time	Parents
<i>a</i>	1	-
<i>b</i>	3	<i>a</i>
<i>c</i>	1	<i>a</i>
<i>d</i>	2	<i>c</i>
<i>e</i>	1	<i>c</i>
<i>f</i>	3	<i>b, c</i>
<i>g</i>	2	<i>d, e</i>
<i>h</i>	1	<i>f, g</i>

iv) Find the critical path for the example from part iii).

Ex 9 (Forward Dynamic Programming). Notice that in the Bellman equation we consider

$$V_t(x_t) = \max_{a_t, \dots, a_{T-1}} r(x_t, a_t) + \dots + r(x_{T-1}, a_{T-1}) + r(x_T)$$

and we can recursively work out V_t backwards from $t = T - 1, \dots, 1$. However, suppose that we start with

$$U_t(x_t) = \max_{a_0, \dots, a_{t-1}} r(x_0, a_0) + r(x_1, a_1) + \dots + r(x_{t-1}, a_{t-1})$$

where x_0 is fixed and it is assume that x_t is the next state after taking action a_{t-1} from state x_{t-1} . (If no such solution – from x_0 to x_t in t steps – exists then we set $U_t(x_t) = -\infty$)

Show that

$$U_t(x_t) = \max_{x_{t-1}, a_{t-1}: f(x_{t-1}, a_{t-1})=x_t} \{U_{t-1}(x_{t-1}) + r(x_{t-1}, a_{t-1})\}$$

and $U_0(x_0) = 0$, and notice that

$$V_T(x_0) = \max_{a_{T-1}: f(x_{T-1}, a_{T-1}) = x_T} \{U_{T-1}(x_{T-1}) + r_T(x_T)\}$$

This approach to solving a dynamic program is sometimes referred to as *Forward Dynamic Program*, because the iteration proceed forward from their initial state x_0 .

Show (after reading the section on Markov decision processes), that we cannot apply this forward dynamic programming approach to MDPs.

Remark 10. Notice that the above approach has advantages over the Backward approach taken in the Bellman equation above that is because we can start from an initial state x_0 and then work out future states iteratively through taking different actions. This is different from the backward version where in principle we have to know all the states that the dynamic program will go to even if we do not know that they will actually be visited from state x_0 (which may not be feasible in the case of infinite state-spaces).

References and Further Reading.

Much of the theory of dynamic programming and Markov decision processes was laid out in the 1950's by Richard Bellman. An excellent early account of the field by Bellman is [4]. The textbooks of Whittle [56] and Bertsekas [6] are noteworthy modern treatments of the field.

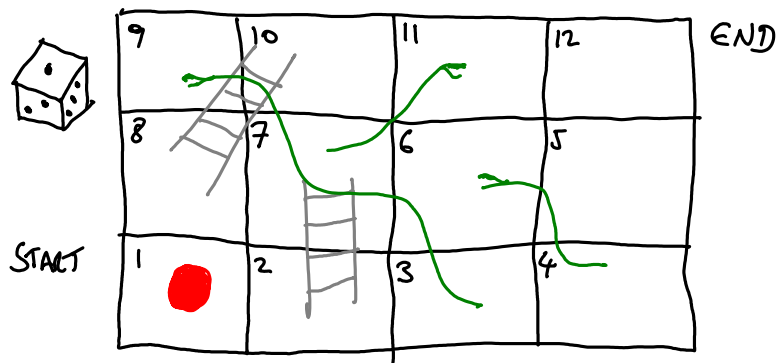
0.2 Markov Chains with Rewards

- Discrete-time Markov chains; The Markov property.
- Jump-chain construction; Rewards and Potential Theory; Martingale Problems.
- Convergence to Equilibrium.

We highlight some of the key properties of Markov chains: how to calculate transitions, how the past effects the current movement of the processes, how to construct a chain, what the long run behavior of the process may (or may not) look like and most importantly what happens when we sum up rewards as the process evolves in time.

Introductory example: snakes and ladders

First, we give an initial example to better position our intuition. Below in Figure 0.2, we are given a game of snakes and ladders (or shoots and ladders in the USA). Here a counter (colored red) is placed on the board at the start. You roll a dice. You move along the numbered squares by an amount given by the dice. The objective is to get to the finish. If the counter lands on a square with a snake's head, you must go back to the square at the snake's tail and, if you land on a square at the bottom of a ladder, go to the top of the ladder.



We let X_t be the position of the counter on the board after the dice has been thrown t times. The processes $X = (X_t : t \in \mathbb{Z})$ is a discrete time Markov chain. Two things to note: First, note that given the counter is currently at a state, e.g. on square 5, the next square reached by the counter – or indeed the sequence of states visited by the counter after being on square 5 – is not effected by the path that was used to reach the square. I.e. This is called the Markov Property. Second, notice the movement of the counter is a function of two pieces of information the current state and the independent random roll of the dice. In this way we can construct (or simulate) the random process.

Definitions

We now give a definition for a Markov chain (in discrete time and with a countable state space).

Let \mathcal{X} be a countable set. An initial distribution

$$\lambda = (\lambda_x : x \in \mathcal{X})$$

is a positive vector whose components sums to one. A transition matrix $P = (P_{xy} : x, y \in \mathcal{X})$ is a positive matrix whose rows sum to one, that is, for $x \in \mathcal{X}$

$$\sum_{y \in \mathcal{X}} P_{xy} = 1.$$

With an initial distribution λ and a transition matrix P , you can define a Markov chain. Basically λ_x determines the probability the process starts in state x and P_{xy} gives the probability of going to y if you are currently in state x .

Def 11 (Discrete Time Markov Chain). *We say that a sequence of random variables $X = (X_t : t \in \mathbb{Z}_+)$ is a discrete time Markov chain, with initial distribution λ and transition matrix P if for $x_0, \dots, x_{t+1} \in \mathcal{X}$,*

$$\mathbb{P}(X_0 = x_0) = \lambda_0$$

and

$$\begin{aligned} \mathbb{P}(X_{t+1} = x_{t+1} | X_t = x_t, \dots, X_0 = x_0) &= \mathbb{P}(X_{t+1} = x_{t+1} | X_t = x_t) && \text{(Markov)} \\ &= P_{x_t x_{t+1}} \end{aligned}$$

The condition (Markov) is often called the Markov property and is the key defining feature of a Markov chain or, more generally, Markov process. It states that the past (X_1, \dots, X_{t-1}) and future X_{t+1} are conditionally independent of the present X_t . Otherwise stated, it says that, when we know the past and present states $(X_1, \dots, X_t) = (x_0, \dots, x_t)$, the distribution of the future states X_{t+1}, X_{t+2}, \dots is only determined by the present state $X_t = x_t$. Think of a board game like snakes and ladders, where you go in the future is only determined by where you are now and not how you got there; this is the Markov property.

Remark 12 (Further notation).

- We will often use the notation \hat{x} to denote the next state reached by the Markov chain after being in state x . I.e.

$$\hat{x} \sim P_x = (P_{xy} : y \in \mathcal{X})$$

- We will sometimes use the notation

$$P(x_1|x_0) := P_{x_0,x_1}$$

to denote transition probabilities.

- Interpreting a function $R = (R(x) : x \in \mathcal{X})$ as a vector. We define PR as the vector achieved by right multiplication to the matrix P , i.e.

$$PR(x) := \sum_{y \in \mathcal{X}} P_{xy} R(y).$$

Calculating Probabilities. It is worth noting that calculating various probabilities and expectations for a Markov chain corresponds to vector and matrix multiplication.

Prop 13. For $X = (X_t : t \in \mathbb{Z}_+)$, a Markov chain with initial distribution λ and transition Matrix P , it holds that

a)

$$\mathbb{P}(X_0 = x_0, X_1 = x_1, \dots, X_t = x_t) = \lambda_{x_0} P_{x_0 x_1} \dots P_{x_{t-1} x_t}$$

b)

$$\mathbb{P}(X_t = x) = [\lambda P^t]_x$$

where $[\lambda P^t]_x$ denotes the x -th component of the matrix λP^t .

c)

$$\mathbb{E}_x[r(X_t)] = P^t r(x)$$

where $P^t r(x)$ is the x -th component of the vector $P^t r$.

Constructing Markov chains. The following proposition shows that the evolution of a Markov chain can be constructed from its the current state and an independent “dice throw”. This is more intuitive than the definition above and is useful when simulating Markov chains.

Prop 14 (Constructing Markov Chains). *Take a function $f : \mathcal{X} \times [0, 1] \rightarrow \mathcal{X}$, X_0 a random variable on \mathcal{X} , and $(U_t)_{t \geq 0}$, independent uniform $[0, 1]$ random variables. The sequence $(X_t)_{t \geq 0}$ constructed with the recursion*

$$X_{t+1} = f(X_t, U_t) \quad \text{for } t = 0, 1, 2, \dots$$

is a discrete time Markov chain. Moreover all discrete time Markov chains can be constructed in this way.

Summing Rewards. In dynamic programming we were interested in summing up rewards. Here is a brief result on that which we can compare with the Bellman equation from Theorem 2.

Lemma 1. *If*

$$R_T(x) = \mathbb{E}_x \left[\sum_{t=0}^T r(X_t) \right]$$

then

$$R_t(x) = r(x) + \mathbb{E}_x [R_{t-1}(X_1)]$$

Proof. We can take the first reward out of the expectation and pro-

ceed as follows

$$\begin{aligned}
R_t(x) &= r(x) + \mathbb{E}_x \left[\sum_{s=1}^t r(X_s) \right] \\
&= r(x) + \mathbb{E}_x \left[\mathbb{E} \left[\sum_{s=1}^t r(X_s) \mid X_1, X_0 = x \right] \right] && \text{(Tower property)} \\
&= r(x) + \mathbb{E}_x \left[\mathbb{E} \left[\sum_{s=1}^t r(X_s) \mid X_1 \right] \right] && \text{(Markov Property)} \\
&= r(x) + \mathbb{E}_x \left[R_{t-1}(X_1) \right] && \text{(Definition)}
\end{aligned}$$

as required. \square

The following proposition will be useful when we want to sum up a long sequence of rewards.

Prop 15 (Markov Chains and Potential Functions). *For $r : \mathcal{X} \rightarrow \mathbb{R}$ a bounded function and for $\beta \in (0, 1)$,*

$$R(x) = \mathbb{E}_x \left[\sum_{t=0}^{\infty} \beta^t r(X_t) \right] \quad (2)$$

is the unique solution to the equation

$$R(x) = \beta(PR)(x) + r(x), \quad x \in \mathcal{X}. \quad (3)$$

Moreover, if function $\tilde{R} : \mathcal{X} \rightarrow \mathbb{R}_+$ satisfies

$$\tilde{R}(x) \geq \beta(P\tilde{R})(x) + r(x), \quad x \in \mathcal{X}. \quad (4)$$

then $\tilde{R}(x) \geq R(x)$, $x \in \mathcal{X}$.

Before we embark on the proof a couple of quick remarks.

Remark 16. • Notice the expression (3) can equivalently be written as

$$R(x) = r(x) + \beta \mathbb{E}_x[R(\hat{x})], \quad \forall x \in \mathcal{X}.$$

• For the “moreover” part above we can also switch the inequality. You can check in the proof that if \tilde{R} is bounded² and such that

$$\tilde{R}(x) \leq \beta(P\tilde{R})(x) + r(x), \quad x \in \mathcal{X}.$$

²We know if r is bounded then R in (2) is bounded.

then $\tilde{R}(x) \leq R(x)$, $x \in \mathcal{X}$.

• The reward function can be generalized to the form $r(x, \hat{x})$, so the next state is included in the reward. Equation (3) now becomes

$$R(x) = \mathbb{E}[r(x, \hat{x}) + \beta R(\hat{x})], \quad x \in \mathcal{X}.$$

Proof. We first show that (3) holds. We give two short proofs: one probabilistic and one algebraic. For that probabilistic proof, note that for $R(x)$ in (2)

$$\begin{aligned} R(x) &= r(x) + \mathbb{E}_x \left[\sum_{t=1}^{\infty} \beta^t r(X_t) \right] \\ &= r(x) + \mathbb{E}_x \left[\mathbb{E} \left[\sum_{t=1}^{\infty} \beta^t r(X_t) \mid X_1, X_0 = x \right] \right] && \text{(Tower property)} \\ &= r(x) + \mathbb{E}_x \left[\beta \mathbb{E} \left[\sum_{t=1}^{\infty} \beta^{t-1} r(X_t) \mid X_1 \right] \right] && \text{(Markov Property)} \\ &= r(x) + \mathbb{E}_x \left[\beta R(X_1) \right] && \text{(Time Homogenous)} \\ &= r(x) + \beta(PR)(x). \end{aligned}$$

So $R(x)$ is a solution to (3).

For the algebraic proof, by Prop 13c) note that

$$R(x) = \sum_{t=0}^{\infty} \beta^t \mathbb{E}_x [r(X_t)] = r(x) + \beta Pr(x) + \beta^2 P^2 r(x) + \dots .$$

The above sequence is a geometric series so we can use our usual tricks for summing geometric series. Notice that multiplying the above expression by βP gives

$$\beta PR(x) = \beta Pr(x) + \beta^2 P^2 r(x) + \beta^3 P^3 r(x) + \dots .$$

Thus

$$R(x) - \beta PR(x) = r(x)$$

which, again, gives the required result.

Now take any solution \hat{R} then $\hat{R} - R = \beta P(\hat{R} - R)$. So

$$\|\hat{R} - R\|_{\infty} \leq \max_{x \in \mathcal{X}} \beta \sum_y P_{xy} |R(y) - \hat{R}(y)| \leq \beta \|\hat{R} - R\|_{\infty} .$$

which, since $\beta < 1$, only holds if $\hat{R} = R$. So the solution is unique.

Finally, suppose that \tilde{R} is a positive function such that $\tilde{R}(x) \geq r(x) + \beta P\tilde{R}(x)$. Repeated substitution gives

$$\tilde{R}(x) \geq r(x) + \mathbb{E}_x[\beta\tilde{R}(X_1)] \geq \dots \geq \mathbb{E}_x\left[\sum_{t=0}^T \beta^t r(X_t)\right] + \beta^{T+1}\mathbb{E}_x[\tilde{R}(X_{T+1})] \quad (5)$$

$$\geq \mathbb{E}_x\left[\sum_{t=0}^T \beta^t r(X_t)\right] \xrightarrow{T \rightarrow \infty} R(x). \quad (6)$$

□

Remark 17 (Rollout.). *The idea of repeated substitution that we applied in (5-6) the final part of the proof is called "rollout". It is a common trick that we will often use.*

Here repeated substitution for $\tilde{R}(x)$ showed that

$$\tilde{R}(x) \geq r(x) + \beta\mathbb{E}_x[\tilde{R}(\hat{x})]$$

implies

$$\tilde{R}(x) \geq \mathbb{E}_x\left[\sum_{t=0}^T \beta^t r(X_t)\right] + \beta^{T+1}\mathbb{E}_x[\tilde{R}(X_{T+1})] \geq R(x) + o(1)$$

and, thus letting $T \rightarrow \infty$, we have

$$\tilde{R}(x) \geq R(x)$$

for all x . (To have $o(1)$ error as $T \rightarrow \infty$ above, we require either that \tilde{R} is bounded & $|\beta| < 1$, or that \tilde{R} is positive.)

Markov Chains and Martingales. There is a close link between Markov chains and martingales, which is often useful for analyzing cumulative rewards.

Prop 18. *Given a bounded function $R : \mathcal{X} \rightarrow \mathbb{R}$, we define*

$$M_t = R(X_0) - \beta^t R(X_t) - \sum_{s=0}^{t-1} \beta^s r(X_s).$$

If M_t , $t \in \mathbb{Z}_+$, is a martingale then

$$R(x) = \mathbb{E}_x\left[\sum_{t=0}^{\infty} \beta^t r(X_t)\right] \quad (7)$$

Conversely, if $R(x)$ satisfies (7) then M_t , $t \in \mathbb{Z}_+$, is a martingale.

Proof. If M_t is a Martingale then

$$0 = \mathbb{E}_x[M_t] = R(X_t) - \mathbb{E}[\beta^T R(X_T)] - \mathbb{E}\left[\sum_{t=0}^{T-1} \beta^t r(X_t)\right]$$

The term $\mathbb{E}[\beta^T R(X_T)]$ goes to zero, by the Dominated Convergence Theorem. So

$$R(x) = \mathbb{E}_x\left[\sum_{t=0}^{\infty} \beta^t r(X_t)\right].$$

Conversely, if

$$R(x) = \mathbb{E}_x\left[\sum_{t=0}^{\infty} \beta^t r(X_t)\right],$$

then, by Prop 15, $R(x) = r(x) + \beta\mathbb{E}_x[R(X_1)]$. Applying this gives

$$\begin{aligned} \mathbb{E}[M_{t+1} - M_t | X_t] &= \mathbb{E}[\beta^t R(X_t) - \beta^{t+1} R(X_{t+1}) - \beta^t r(X_t) | X_t] \\ &= \beta^t [R(X_t) - \beta\mathbb{E}_{X_t}[R(X_{t+1})] - r(X_t)] = 0. \end{aligned}$$

□

Equilibrium Distributions*

This section can be skipped on first reading but later we will discuss the limit behavior of a Markov chain as time gets large. Under suitable assumptions, a Markov chain's distribution will converge to a distribution known as the stationary distribution or equilibrium distribution. Often this convergence is exponentially fast.

Equilibrium Distribution and Ergodicity. Let's assume for now our Markov chain has a finite number of states.³ If we ran the Markov chain for infinite time, some states must be visited infinitely often. So we might imagine there might be some limiting distribution over these states. E.g.

$$\frac{1}{T} \sum_{t=1}^T \mathbb{I}[X_t = x] \xrightarrow{T \rightarrow \infty} \mu(x)$$

³Much of this analysis follows for a countable number of states and, with some work, for a continuum of states.

or

$$\mathbb{P}(X_T = x) \xrightarrow{T \rightarrow \infty} \mu(x). \quad (8)$$

A Markov chain that has convergence to some distribution of this type is called ergodic. The limiting distribution is called the equilibrium distribution or stationary distribution.

Irreducibility and Ergodicity. There are some caveats: we may want the limiting distribution μ to be unique and we want each state to have positive probability in μ (i.e. we don't include states which we eventually stop visiting). This will not happen if we can get stuck in some set of states. Specifically, there is a set of states \mathcal{X}_1 for which there is zero probability of visiting some other state x . In this case the Markov chain is called reducible.

Conversely if there is a positive probability of reaching each state from every other state, then the Markov chain is called *irreducible*. Notice that means if we visit some state x infinitely often, the (by irreducibility) there is positive probability of visiting another state x' from x and so we must visit x' infinitely often too. Irreducibility also implies that the Matrix P has a unique largest eigenvalue equal to and of size 1.⁴ (We know there is an eigenvalue of size 1 since $P\mathbf{1} = \mathbf{1}$ where $\mathbf{1}$ is the vector of all ones) Irreducibility also implies uniqueness of μ .⁵

Rate of Convergence to Equilibrium. We now assume that P is an irreducible transition matrix such that convergence to μ , (8), holds. Notice that since $\mathbb{P}(X_t = x) = \lambda P^t$ and $\lim_{t \rightarrow \infty} \lambda P^t = \mu$, we have that

$$\mu = \lim_{T \rightarrow \infty} \mathbb{P}(X_{T+1} = \cdot) = \lim_{T \rightarrow \infty} \lambda P^{T+1} = \lim_{T \rightarrow \infty} \lambda P^T P = \mu P$$

We now briefly discuss how fast is this convergence is for the following metric:

$$\|\mu - \lambda P^t\|_{TV} := \frac{1}{2} \sum_{x \in \mathcal{X}} |\mu(x) - \lambda P^t(x)|.$$

The norm $\|\cdot\|_{TV}$ is called the total variation distance.

⁴This is the classical Perron-Frobenius Theorem.

⁵This can be proven with an argument that is somewhat elementary and similar to roll out in MDP's.

We study P^t . Since a probabilities are expressed in terms of powers P^t , we know (from linear algebra) that we can always express a matrix in Jordan Normal Form

$$P = UJU^{-1}$$

where U is an invertible matrix, and J is block diagonal for sets of distinct eigenvalues. I.e. The non-zero entries are

$$J = \begin{pmatrix} J_1(\alpha_1) & & & \\ & J_2(\alpha_2) & & \\ & & \ddots & \\ & & & J_n(\alpha_n) \end{pmatrix} \quad \text{where} \quad J_i(\alpha_i) = \begin{pmatrix} \alpha_i & 1 & & \\ & \alpha_i & 1 & \\ & & \ddots & \ddots \\ & & & \alpha_i & 1 \\ & & & & \alpha_i \end{pmatrix}$$

Here $1 = \alpha_1 > \alpha_2 \geq \dots \geq \alpha_n > -1$ are the set of eigenvalues of our irreducible transition matrix P .

We are interested in powers P^t for t large. Notice then that

$$P^t = UJ^tU$$

and J^t is the matrix J above where each block $J_i(\alpha_i)$ is multiplied out t times. A quick check gives that

$$[J(\alpha)^t]_{ij} = \binom{t}{j-i} \alpha^{t-j+i}$$

for $i \geq j$ and $[J(\alpha)^k]_{ij} = 0$ otherwise. Convergence is dominated by the largest term in $\lambda P^t - \mu$ which is given by $J(\alpha_2)^t$ since α_2 is the second largest eigenvalue. (Note, the largest has been cancelled out by $\mu = P^t \mu$.) This gives the following

Theorem 1. *For an irreducible finite time Markov chain*

$$\|\lambda P^t - \mu\|_{TV} \leq Ct^{N-1} \alpha_2^t$$

where C is a constant, $N = |\mathcal{X}|$ and α_2 is the 2nd largest eigenvalue of P .

A more formal proof of this can be found in [40]. However, the argument above should be sufficient for the reader to fill in the relevant mathematical details.

Further Proofs.

We give a proof of Proposition 13 which we skipped earlier.

Proof of Prop 13. a) Using the Markov property (Markov) gives

$$\begin{aligned} & \mathbb{P}(X_0 = x_0, X_1 = x_1, \dots, X_t = x_t) \\ &= \mathbb{P}(X_t = x_t | X_0 = x_0, \dots, X_{t-1} = x_{t-1}) \mathbb{P}(X_0 = x_0, \dots, X_{t-1} = x_{t-1}) \\ &= P_{x_{t-1}x_t} \mathbb{P}(X_0 = x_0, \dots, X_{t-1} = x_{t-1}) \end{aligned}$$

Continuing inductively gives part a).

b)

$$\begin{aligned} \mathbb{P}(X_t = x) &= \sum_{x_0, \dots, x_{t-1}} \mathbb{P}(X_0 = x_0, \dots, X_{t-1} = x_{t-1}, X_t = x) \\ &= \sum_{x_0, \dots, x_{t-1}} \lambda_{x_0} P_{x_0x_1} \dots P_{x_{t-1}x} && \text{(Part a)} \\ &= [\lambda P^t]_x \end{aligned}$$

c)

$$\mathbb{E}_x [r(X_t)] = \sum_y P_x(X_t = y) r(y) = \sum_y [P^t]_{xy} r(y) = P^t r(x).$$

□

Markov Chain Exercises.

The following is an alternative formulation of the previous proposition.

Ex 19. Let $\partial\mathcal{X}$ be a subset of \mathcal{X} and let T be the hitting time on $\partial\mathcal{X}$ i.e. $T = \inf\{t : X_t \in \partial\mathcal{X}\}$ and take $f : \partial\mathcal{X} \rightarrow \mathbb{R}_+$ argue that

$$R(x) = \mathbb{E}_x \left[\sum_{t < T} r(X_t) + f(X_T) \mathbb{I}[T < \infty] \right]$$

solves the equation

$$R(x) = (PR)(x) + r(x), \quad x \notin \partial\mathcal{X} \quad (9)$$

$$R(x) = f(x), \quad x \in \mathcal{X}. \quad (10)$$

There is a close connection between Markov chains and Martingales that we will want to use later when considering Markov Decision Processes.

Ex 20 (Markov Chains and Martingale Problems). *Show that a sequence of random variables $X = (X_t : t \in \mathbb{Z}_+)$ is a Markov chain if and only if, for all bounded functions $f : \mathcal{X} \rightarrow \mathbb{R}$, the process*

$$M_t^f = f(X_t) - f(X_0) - \sum_{\tau=0}^{t-1} (P - I)f(X_\tau)$$

is a Martingale with respect to the natural filtration of X . Here for any matrix, say Q , we define

$$Qf(x) := \sum_{y \in \mathcal{X}} Q_{xy}f(y).$$

References

This section is intended as a brief introductory recap of Markov chains. A much fuller explanation and introduction is provided in standard texts e.g. Norris [36], Bremaud [12], or Levin & Peres [31]. The analysis of rewards and Markov processes is particularly studied by Doob [17].

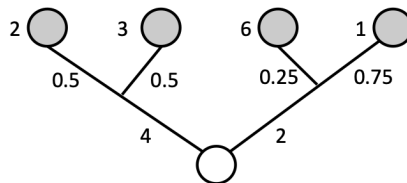
0.3 Markov Decision Processes

- Markov Decision Process (definition).
 - Bellman's Equation.
-

Markov decision processes are the randomized equivalent of a dynamic program. Let's first consider how to randomize the tree example introduced in Section 0.1.

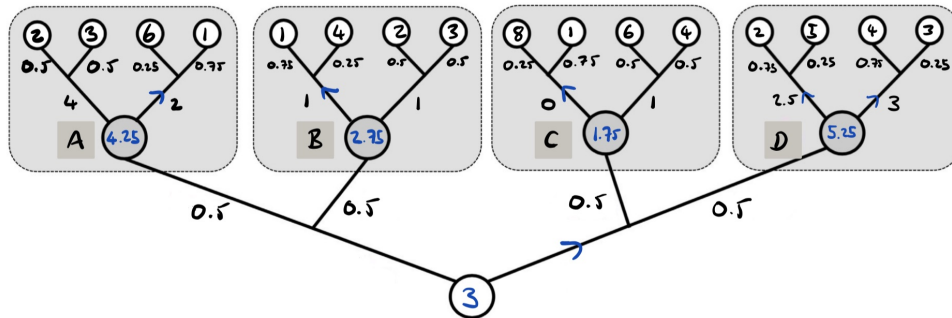
A Random Example

Below is a tree with a root node and four leaf nodes colored grey. At the root node you choose to go left or right. This incurs costs 4 and 2, respectively. Further, after making this decision there is a probability for reaching a leaf node. Namely, after going left the probabilities are 0.5 & 0.5, and for turning right, the probabilities are 0.25 & 0.75. For each leaf node there is a cost, namely, 2, 3, 6, and 1.

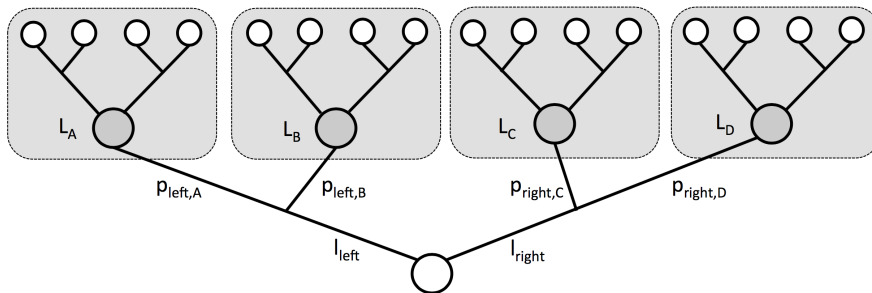


Given you only know the probabilities (and not which outcome happens when you choose left or right), you'd want to take the decision with lowest expected cost. The expected cost for left is $4 + 0.5 \times 2 + 0.5 \times 3 = 5.5$ and for right is $2 + 0.25 \times 6 + 0.75 \times 1 = 4.25$. So go right.

We can then combine these together to make a sequence of such calculations working from the leaves of the tree to the root. Just like we did for our introductory example for dynamic programming. See below.



We now replace the numbers above with symbols. At the route node you can choose the action to go left or right. These, respective, decisions incur costs of l_{left} and l_{right} . After choosing left, you will move to state A with probability $p_{\text{left},A}$ or to state B with probability $p_{\text{left},B}$ and similarly choosing right states C & D can be reached with probabilities $p_{\text{left},C}$ & $p_{\text{left},D}$. After reaching node A (resp. B, C, D) the total expected cost thereafter is L_A (resp. L_B, L_C, L_D).



The cost from choosing “left” is :

$$l_{\text{left}} + p_{\text{left},A}L_A + p_{\text{left},B}L_B = l_{\text{left}} + \mathbb{E}_{\text{left}}[L_{\text{left}}]$$

and the cost for choosing “right” is:

$$l_{\text{right}} + p_{\text{right},A}L_A + p_{\text{right},B}L_B = l_{\text{right}} + \mathbb{E}_{\text{right}}[L_{\text{right}}].$$

The optimal cost is the minimum of these two is

$$L_R = \min_{a \in \{\text{left}, \text{right}\}} \{l_a + \mathbb{E}_a [L_{X_a}]\}.$$

where here the random variable X_a denotes the state in $\{A, B, C, D\}$ reached after each action is taken. Notice how we abstracted away the future behaviour after arriving at A, B, C, D . Into a single cost for each state: L_A, L_B, L_C, L_D . And we can propagate this back to get the costs at the route state R . I.e. We can essentially apply the same principle as dynamic programming here.

Definitions

A Markov Decision Process (MDP) is a Dynamic Program where the state evolves in a random (Markovian) way.

Like with a dynamic program, we consider discrete times $t = 0, 1, \dots, T$, states $x \in \mathcal{X}$, actions $a \in \mathcal{A}$ and rewards $r(x, a)$. A policy, π , at time t when in state x chooses an action

$$\pi_t(x) \in \mathcal{A}.$$

I.e. a policy is a function $\pi : \mathcal{X} \times \{0, 1, \dots, T\} \rightarrow \mathcal{A}$. We let \mathcal{P}_T be the set of policies over T time steps, and we let $\mathcal{P}_T(x)$ be the set of policies over T time steps started at state $X_0 = x$.⁶

The plant equation is slightly different. Like with a Markov chain, the state evolves as a random function, but now the action also influences the next state. Specifically

$$X_{t+1} = f(X_t, A_t; U_t)$$

where X_t is the state at time t and action A_t is the action at time t and where U_t is an independent random variable. This is called the Plant Equation. (Note, under policy π , $A_t = \pi_t(X_t)$).

A policy, a plant equation, and the resulting sequence of states and rewards describe a Markov Decision Process.

Remark 21 (Some further notation).

- We will usually suppress dependence on U_t , and instead write:

$$X_{t+1} = f(X_t, A_t).$$

We let the random variable \hat{x} be the next state (after state x and action a):

$$\hat{x} = f(x, a).$$

- We let

$$P_{xy}^a := P(y|x, a) := \mathbb{P}(X_1 = y | X_0 = x, A_0 = a)$$

which the probability of next moving to state y , when in state x and taking action a .

- We will often suppress the dependence of X_t and write $\pi_t = \pi_t(X_t)$.
- We use the notation

$$\mathbb{E}_{x,a}[G(\hat{x})] = \mathbb{E}[G(f(x, a; U))] = \sum_y P_{xy}^a G(y)$$

⁶We could let a policy depend on past states and decisions. See Remark 24.

and

$$\begin{aligned}\mathbb{E}_{x_t, a_t}[G(X_{t+1})] &:= \mathbb{E}[G(X_{t+1})|X_t = x_t, A_t = a_t] \\ &= \mathbb{E}[G(f(x_t, a_t; U))] = \sum_y P_{x_t, y}^{a_t} G(y)\end{aligned}$$

(Notice in both equalities above, the expectation depends on only one independent random variable, U .)

Our objective is to find a policy that optimizes the expected reward.

Def 22 (Markov Decision Problem). *Given initial state x_0 , a Markov Decision Problem is the following optimization*

$$\begin{aligned}V_T(x_0) = \text{Maximize} \quad & R_T(x_0, \pi) := \mathbb{E} \left[\sum_{t=0}^{T-1} r(X_t, \pi_t) + r(X_T) \right] \quad (\text{MDP}) \\ \text{over} \quad & \pi \in \mathcal{P}_T(x_0).\end{aligned}$$

We often call $V_T(x)$ to value function of the MDP.

We want to derive the Bellman equation for a Markov decision process just like we did for dynamic programming. The key idea there was that we could separate the current decision we made from future decisions. The same holds here and the following lemma helps with this:

Lemma 2. *a) The set of policies satisfies $\mathcal{P}_t(x) = \mathcal{A} \times \mathcal{P}_{t-1}$,*

$$\max_{\pi \in \mathcal{P}_t(x)} f(\pi) = \max_{a \in \mathcal{A}} \max_{\hat{\pi} \in \mathcal{P}_{t-1}} f(a, \hat{\pi})$$

for any $f : \mathcal{P}_t \rightarrow \mathbb{R}$ and $\hat{\pi} = (\pi_1, \dots, \pi_T)$.

b) For policy π , the states X_t are a Markov chain⁷. So if $\pi_0(x) = a$ then

$$\mathbb{E}_{x, a}[Y] = \mathbb{E}_{x, a}[\mathbb{E}_{X_1, \pi_1}[Y]]$$

for any random variable Y depending on states and decision made after time $t = 0$.

c) If $X_0 = x$ and $\pi_0(x) = a$

$$\sum_{s=0}^{t-1} r(X_s, \pi_s) + r(X_t) = r(x, a) + \sum_{s=1}^{t-1} r(X_s, \pi_s) + r(X_t)$$

⁷Note that this Markov chain may not be time homogenous but the Markov property does hold.

The proof is straight forward (and part c is trivial) so we postpone it until the end of the section. The main thing is we can separate present states and decisions from the future when maximizing, when taking expectations and when summing.

The next result shows that the Bellman equation follows essentially as before but now we have to take account for the expected value of the next state.

Thrm 23 (Bellman Equation). *The optimal value function $V_t(x)$ satisfies $V_0(x) = r(x)$ and*

$$V_t(x) = \max_{a \in \mathcal{A}} \{r(x, a) + \mathbb{E}_{x,a} [V_{t-1}(\hat{x})]\} \quad \text{for } t = 1, \dots, T. \quad (\text{Bell})$$

Moreover, in state x at time t , an optimal policy π^* must chose

$$\pi_t^*(x) \in \operatorname{argmax}_{a \in \mathcal{A}} \{r(x, a) + \mathbb{E}_{x,a} [V_{t-1}(\hat{x})]\}$$

Above, the equation (Bell) is the Bellman equation for a finite time Markov Decision Process.

Proof. For $\pi \in \mathcal{P}_t(x)$ we let $\pi = (a, \hat{\pi})$ where $\pi_0(x) = a$ and $(\pi_1, \dots, \pi_t) = (\hat{\pi}_0, \dots, \hat{\pi}_{t-1})$. Also we define $(\hat{X}_1, \dots, \hat{X}_{t-1}) = (X_2, \dots, X_t)$ and $\hat{X}_0 = X_1 = \hat{x}$.

Applying Lemma 2 parts a) and b) gives

$$\begin{aligned} V_t(x) &= \max_{\pi \in \mathcal{P}_t} \mathbb{E}_{x, \pi_0} \left[\sum_{s=0}^{t-1} r(X_s, \pi_s) + r(X_t) \right] \\ &= \max_{a \in \mathcal{A}} \max_{\hat{\pi} \in \mathcal{P}_{t-1}} \mathbb{E}_{x,a} \left[\mathbb{E}_{X_1, \pi_1} \left[r(x, a) + \sum_{s=1}^{t-1} r(X_s, \pi_s) + r(X_T) \right] \right] \\ &= \max_{a \in \mathcal{A}} r(x, a) + \mathbb{E}_{x,a} \left[\max_{\hat{\pi} \in \mathcal{P}_{t-1}} \mathbb{E}_{X_1, \pi_1} \left[\sum_{s=1}^{t-1} r(X_s, \pi_s) + r(\hat{X}_{t-1}) \right] \right] \\ &= \max_{a \in \mathcal{A}} r(x, a) + \mathbb{E}_{x,a} \left[\underbrace{\max_{\hat{\pi} \in \mathcal{P}_{t-1}} \mathbb{E}_{\hat{x}, \hat{\pi}_0} \left[\sum_{s=0}^{t-2} r(\hat{X}_s, \pi_s) + r(\hat{X}_t) \right]}_{=V_{t-1}(\hat{x})} \right]. \end{aligned}$$

In the second equality, we apply Lemma 2 separating the present and future variables in the maximization, the expectation and the sum. In the third equality, we move, as far as possible, terms for the present to the left and terms for the future to the right. The fourth equality, just changes notation to make it clear that the term in brackets is $V_{t-1}(\hat{x})$. \square

Rmk 24. *It is probably worth remarking that we could have let the policy choice depend on past variables and decisions. I.e. the action we choose at time t , π_t , could depend on $X_0, A_0, \dots, X_{t-1}, A_{t-1}$, in addition to X_t . However, by the Markov property these extra terms do not to gain us any additional information when optimizing the future rewards. It only lengthens the proof as given above. See Ex 29.*

Further Proofs.

Proof of Lemma 2. a) This is really just the definition: for $\pi \in \mathcal{P}_t(x)$,

$$\pi = (\pi_0(x), \pi_1, \dots, \pi_{t-1}) = (a, \hat{\pi})$$

for $a \in \mathcal{A}$ and $\hat{\pi} \in \mathcal{P}_{t-1}$.

b) Notice if π is fixed then $X_{t+1} = f(X_t, \pi_t(X_t); U_t)$. Thus by Proposition 14, $(X_t : t = 0, \dots, T)$ is a Markov chain (albeit not necessarily a time homogeneous one). Applying the Markov property, we have

$$\begin{aligned} \mathbb{E}_{x, \pi_0}[Y] &= \mathbb{E}_{x, \pi_0}[\mathbb{E}[Y|X_1, A_1 = \pi_1(X_1), X_0 = x, A_0 = \pi_0(x)]] \\ &= \mathbb{E}_{x, \pi_0}[\mathbb{E}[Y|X_1, A_1 = \pi_1(X_1)]] && \text{(Markov Property)} \\ &= \mathbb{E}_{x, \pi_0}[\mathbb{E}_{X_1, \pi_1}[Y]]. \end{aligned}$$

c) This part is trivial. □

MDP Examples

Ex 25. You need to sell a car. At every time $t = 0, \dots, T - 1$, you set a price p_t and a customer then views the car. The probability that the customer buys a car at price p is $D(p)$. If the car isn't sold by time T then it is sold for fixed price V_T , $V_T < 1$. Maximize the reward from selling the car and find the recursion for the optimal reward when $D(p) = (1 - p)_+$.

Ex 26 (Call Option). You own a call option with strike price p . Here you can buy a share at price p making profit $X_t - p$ where x_t is the price of the share at time t . The share must be exercised by time T . The price of stock X_t satisfies

$$X_{t+1} = X_t + \epsilon_t$$

for ϵ_t IIDRV with finite expectation. Show that there exists a decreasing sequence $\{a_t\}_{0 \leq t \leq T}$ such that it is optimal to exercise whenever $X_s \geq a_s$ occurs.

Ex 27. You own an expensive fish. Each day you are offered a price for the fish according to a distribution density $f(x)$. You make the accept or reject this offer. With probability $1 - p$ the fish dies that day. Find the policy that maximizes the profit from selling fish.

Ex 28. Indiana Jones is trapped in a room in a temple. There are n passages that he can try and escape from. If he attempts to escape from passage $i \in \{1, \dots, n\}$ then either: he escapes with probability p_i ; he dies with probability q_i ; or with probability $r_i = 1 - p_i - q_i$ the passage is a deadend and he returns to the room which he started from. Determine the order of passages which Indiana Jones must try in order to maximize his probability of escape.

Ex 29. We develop slightly the point made in Remark 24.

a) Show that for a MDP with states and actions given by $X_1, A_1, \dots, A_{T-1}, X_T$ that

$$\mathbb{E}[r(X_t, A_t) + V(X_{t+1}) | X_t, A_t, \dots, X_0, A_0] = \mathbb{E}[r(X_t, A_t) + V(X_{t+1}) | X_t, A_t]$$

for any function $r(x, a)$ and $V(x)$.

b) Show that for any optimal policy for the MDP whose current action depends on previous states and actions, there is also a policy that depends only on the current state and action.

[Hint: inductively apply part a) in the proof of the Bellman equation.]

c) Reread the principle of optimality.

References

Markov decision processes were studied as the natural probabilistic analog of dynamic programs. An early text on MDPs is Howard [24]. Standard texts on dynamic programming [6] or on Markov chains [36] cover MDPs. A good account of MDPs is Puterman [37].

0.4 Infinite Time Horizon

- Discounted Programming, Positive Programming, Negative Programming, Average Programming.
 - Conditions for the Optimality of the Bellman Equation.
 - Martingales and Optimality; Occupancy Measure.
-

Thus far we have considered finite time Markov decision processes. We now want to solve MDPs of the form

$$V(x) = \underset{\pi \in \mathcal{P}}{\text{maximize}} \quad R(x, \pi) := \mathbb{E}_{x_0} \left[\sum_{t=0}^{\infty} \beta^t r(X_t, \pi_t) \right].$$

In the above equation the term β is called the *discount factor*.

We can generalize Bellman's equation to infinite time. A correct guess at the form of the equation would, for instance, be

$$V(x) = \max_{a \in \mathcal{A}} \left\{ r(x, a) + \beta \mathbb{E}_{x,a} [V(\hat{x})] \right\}, \quad x \in \mathcal{X}.$$

Previously we solved Markov Decisions Processes inductively with Bellman's equation. In infinite time, we can not directly apply induction; however, we see that Bellman's equation still holds and we can use this to solve our MDP.

Discounted Programming

For now we will focus on the case of discounted programming:

Def 30 (Discounted Program). *A discounted program is a MDP with bounded rewards and a discount factor that is smaller than 1*

$$\max_{x \in \mathcal{X}, a \in \mathcal{A}} |r(x, a)| < \infty \quad \text{and} \quad \beta \in (0, 1).$$

We will cover other cases where $\beta = 1$ later. At this point it is useful to define the concept of a Q-factor. A Q-factor of a policy π is the reward that arises when we take action a from state x and then follow policy π .

Def 31 (Q-Factor). *The Q-factor of reward function $R(\cdot)$ is the value for taking action a in state x and then at the next step receiving reward $R(\hat{X})$:*

$$Q_R(x, a) = \mathbb{E}_{x,a}[r(x, a) + \beta R(\hat{x})].$$

Similarly the Q-factor for a policy π , denoted by $Q_\pi(x, a)$, is given by the above expression with $R(x) = R(x, \pi)$. The Q-factor of the optimal policy is given by

$$Q^*(x, a) = \max_{\pi} Q_\pi(x, a).$$

The following result shows that if we have solved the Bellman equation then the solution and its associated policy is optimal.

Thrm 32. *If we find a function $R(x)$ that solves the Bellman equation:*

$$R(x) = \max_{a \in \mathcal{A}} \{r(x, a) + \beta \mathbb{E}_{x,a} [R(\hat{x})]\}$$

then $R(x) = V(x)$, where $V(x)$ is the optimal value function. Moreover, if we take

$$\pi(x) \in \operatorname{argmax}_{a \in \mathcal{A}} \{r(x, a) + \beta \mathbb{E}_{x,a} [R(\hat{x})]\}, \quad x \in \mathcal{X}$$

then π is optimal.

Proof. We will first show that V solves the Bellman equation [with inequality \leq then \geq]. Then we will argue that its solution is unique with a contraction argument. Then we will argue that a stationary policy associated with the Bellman equation is optimal by applying Proposition 15.

We know that $R_t(x, \pi) = r(x, \pi_0) + \beta \mathbb{E}[R_{t-1}(\hat{x}, \hat{\pi})]$, where $\hat{\pi}$ is the policy π take from time 1 onwards. We know that $R_t(x, \pi) \rightarrow R(x, \pi)$ as $t \rightarrow \infty$ by bounded convergence theorem. Letting $t \rightarrow \infty$ on both sides gives that

$$\begin{aligned} R(x, \pi) &= r(x, \pi_0) + \beta \mathbb{E}_{x, \pi_0} [R(\hat{x}, \hat{\pi})] \\ &\leq r(x, \pi_0) + \beta \mathbb{E}_{x, \pi_0} [V(\hat{x})]. \end{aligned}$$

For the inequality, above, we maximize $R(\hat{x}, \hat{\pi})$ over $\hat{\pi}$. Now maximizing the left hand side over π gives

$$V(x) \leq \sup_{\pi_0 \in \mathcal{A}} \{r(x, \pi_0) + \beta \mathbb{E}_{x, \pi_0} [V(\hat{x})]\}.$$

At this point we have that the Bellman equation holds but with an inequality. We need to prove the inequality in the other direction. For this, we let π_ϵ be the policy that chooses action a and then, from the next state \hat{X} , follows a policy $\hat{\pi}_\epsilon$ which satisfies

$$R(\hat{X}, \hat{\pi}_\epsilon) \geq V(\hat{\pi}) - \epsilon.$$

We then have that

$$\begin{aligned} V(x) &\geq R(x, \pi_\epsilon) = r(x, a) + \beta \mathbb{E}_{x,a} [R(\hat{x}, \hat{\pi}_\epsilon)] \\ &\geq r(x, a) + \beta \mathbb{E}_{x,a} [V(\hat{x})] - \epsilon\beta \end{aligned}$$

The first inequality holds by the sub-optimality of π_ϵ and the second holds by the assumption on $\hat{\pi}_\epsilon$. Maximizing over $a \in \mathcal{A}$, and taking $\epsilon \rightarrow 0$ gives

$$V(x) \geq \max_{a \in \mathcal{A}} \{r(x, a) + \beta \mathbb{E}_{x,a} [V(\hat{x})]\}.$$

Thus we now have that

$$V(x) = \max_{a \in \mathcal{A}} \{r(x, a) + \beta \mathbb{E}_{x,a} [V(\hat{x})]\}.$$

So at this point we know that the optimal value function satisfies the Bellman equation. For the next part of the result we need to show that the solution to the Bellman equation is unique.

Suppose that $R(x)$ is another solution to the Bellman equation. From the definition of a Q -factor and the Bellman recursion, $R(x) = \max_a Q_R(x, a)$ and $V(x) = \max_a Q_V(x, a)$. Thus note that

$$Q_V(x, a) - Q_R(x, a) = \beta \mathbb{E}[V(\hat{X}) - R(\hat{X})] = \beta \mathbb{E}[\max_{a'} Q_V(\hat{X}, a) - \max_{a'} Q_R(\hat{X}, a')]$$

Thus

$$\|Q_V - Q_R\|_\infty \leq \beta \max_{\hat{x}} |\max_{a'} Q_V(\hat{x}, a') - \max_{a'} Q_R(\hat{x}, a')| \leq \beta \|Q_V - Q_R\|_\infty. \quad (11)$$

(In the last equality above, we use the fact that

$$|\max_{a'} Q_V(x, a') - \max_{a'} Q_R(x, a')| \leq \max_a |Q_V(x, a) - Q_R(x, a)|.$$

This is an elementary result which we prove in Lemma 33 after we complete this proof.) Now since $0 < \beta < 1$ the only solution to the inequality, (11), is $Q_V = Q_R$ and thus

$$R(x) = \max_a Q_R(x, a) = \max_a Q_V(x, a) = V(x).$$

So solutions to the Bellman equation are unique for discounted programming. Finally we must show that if we can find a policy that solves the Bellman equation, then it is optimal.

If we find a function $R(x)$ and a function $\pi(x)$ such that

$$R(x) = \max_{a \in \mathcal{A}} \left\{ r(x, a) + \beta \mathbb{E}_{x, a} [R(\hat{X})] \right\}, \quad \pi(x) \in \operatorname{argmax}_{a \in \mathcal{A}} \left\{ r(x, a) + \beta \mathbb{E}_{x, a} [R(\hat{X})] \right\}$$

then note that the MDP induced by π is a Markov chain (with transition matrix $P_{xy}^{\pi(x)}$). Both $R(x, \pi)$ and $R(x)$ solve the equation $\tilde{R}(x) = r(x, \pi(x)) + \beta \mathbb{E}_{x, \pi(x)} [\tilde{R}(\hat{X})]$. By Prop 15 this solution is unique and so, $R(x) = R(x, \pi)$. \square

For the above proof we required the following technical lemma.

Lem 33. For any two real valued function f and g

$$|\max_a f(a) - \max_a g(a)| \leq \max_a |f(a) - g(a)|$$

Proof. Assume wlog that $\max_a f(a) \geq \max_a g(a)$ then

$$f(a) - g(a) \geq f(a) - \max_{a'} g(a')$$

therefore

$$|f(a) - g(a)| \geq f(a) - \max_{a'} g(a')$$

Now maximize both sides over a to complete the proof. \square

It is worth collating together a similar result for Q -factors. Given the facts accrued about the value function and Bellman's equation. The following Proposition should not be too great a surprise (and can be skipped on first reading).

Prop 34. a) Stationary Q -factors satisfy the recursion

$$Q_{\pi}(x, a) = r(x, a) + \beta \mathbb{E}_{x, a} [Q_{\pi}(\hat{x}, \pi(\hat{x}))].$$

b) Bellman's Equation can be re-expressed in terms of Q -factors as follows

$$Q^*(x, a) = r(x, a) + \beta \mathbb{E}_{x, a} \left[\max_{\hat{a}} Q^*(\hat{x}, \hat{a}) \right].$$

The optimal value function satisfies

$$V(x) = \max_{a \in \mathcal{A}} Q^*(x, a).$$

c) *The operation*

$$F_{x,a}(\mathbf{Q}) = \mathbb{E}_{x,a}[r(x,a) + \beta Q_\pi(\hat{X}, \pi(\hat{X}))]$$

is a contraction with respect to the supremum norm, that is,

$$\|F(\mathbf{Q}_1) - F(\mathbf{Q}_2)\|_\infty \leq \|\mathbf{Q}_1 - \mathbf{Q}_2\|_\infty.$$

Proof. a) We can think of extending the state space of our MDP to include states $\mathcal{X}_0 = \{(x,a) : x \in \mathcal{X}, a \in \mathcal{A}\}$ as well as \mathcal{X} . In this new MDP we can assume that initially the MDP starts in state (x,a) then moves to the state $\hat{X} \in \mathcal{X}$ according to the transition probabilities $P_{x\hat{x}}^a$. There after it remains in \mathcal{X} moving according to policy π . Thus by Prop 15

$$Q_\pi(x,a) = \mathbb{E}_{x,a}[r(x,a) + \beta R(\hat{X}, \pi)]$$

where $R(x, \pi)$ is the reward function of policy π . Further since $Q_\pi(x,a)$ is the value from taking a instead of following policy π to should also be clear that

$$Q_\pi(x, \pi) = \mathbb{E}_{x,\pi(x)}[r(x, \pi(x)) + \beta R(\hat{X}, \pi)] = R(x, \pi)$$

Thus, as required,

$$Q_\pi(x,a) = \mathbb{E}_{x,a}[r(x,a) + \beta Q_\pi(\hat{X}, \pi(\hat{X}))].$$

b) Further it should be clear that the optimal value function for the extended MDP discussed has a Bellman equation of the form

$$\begin{aligned} Q^*(x,a) &= \mathbb{E}_{x,a}[r(x,a) + \beta V(\hat{X})] \\ V(x) &= \max_{a \in \mathcal{A}} \mathbb{E}_{x,a}[r(x,a) + \beta V(\hat{X})] \end{aligned}$$

Comparing the first equation above with the second, it should be clear that $V(x) = \max_a Q^*(x,a)$ and substituting this back into the first equation gives as required

$$Q^*(x,a) = \mathbb{E}_{x,a}[r(x,a) + \beta \max_{\hat{a} \in \mathcal{A}} Q^*(\hat{X}, \hat{a})].$$

c) The proof of this part is already embedded in the previous Theorem. Note that

$$F_{x,a}(\mathbf{Q}_1) - F_{x,a}(\mathbf{Q}_2) = \beta \mathbb{E}[\max_{a'} Q_V(\hat{X}, a) - \max_{a'} Q_R(\hat{X}, a')]$$

Thus

$$\begin{aligned} \|F(Q_1) - F(Q_2)\|_\infty &= \max_{x,a} \left| \beta \mathbb{E}_{x,a} \left[\max_{a'} Q_1(\hat{x}, a') \right] - \beta \mathbb{E}_{x,a} \left[\max_{a'} Q_2(\hat{x}, a') \right] \right| \\ &\leq \beta \max_{\hat{x}} \left| \max_a Q_1(\hat{x}, a) - \max_{a'} Q_2(\hat{x}, a') \right| \\ &\leq \beta \|Q_1 - Q_2\|_\infty, \end{aligned}$$

The last inequality follows by Lemma 33, as required. \square

The following shows that if we can get a good approximation of the value function or the Q-function of a discounted MDP then we can construct a good policy from it. This is important as we will construct algorithms to approximate the value and Q-functions.

Proposition 1. *For a discounted MDP with optimal value function V and optimal Q-function Q ,*

a) *If we can find R such that*

$$\|R - V\|_\infty \leq \epsilon$$

and we define a policy π such that

$$\pi(x) \in \operatorname{argmax}_{a \in \mathcal{A}} \{r(x, a) + \beta \mathbb{E}_{x,a}[R(\hat{x})]\},$$

then

$$R_\pi(x) \geq V(x) - \frac{\beta\epsilon}{1-\beta}$$

b) *If we can find Q' such that*

$$\|Q' - Q\| \leq \epsilon$$

and we define a policy π such that

$$\pi(x) \in \operatorname{argmax}_{a \in \mathcal{A}} Q'(x, a)$$

then

$$R_\pi(x) \geq V(x) - \frac{2\beta}{1-\beta} \epsilon.$$

Proof. a) Since V satisfies the Bellman equation

$$\begin{aligned} V(x) &= \max_a r(x, a) + \beta \mathbb{E}_{x,a} [V(\hat{x})] \\ &\leq \max_a r(x, a) + \epsilon\beta + \beta \mathbb{E}_{x,a} [R(\hat{x})] \\ &= r(x, \pi(x)) + \epsilon\beta + \beta \mathbb{E}_{x,\pi(x)} [R(\hat{x})] \\ &\leq r(x, \pi(x)) + 2\epsilon\beta + \beta \mathbb{E}_{x,\pi(x)} [V(\hat{x})] \end{aligned}$$

Applying Rollout (see Remark 17) we have

$$V(x) \leq \mathbb{E} \left[\sum_t \beta^t (r(\hat{x}_t, \pi_t) + 2\epsilon\beta) \right] = R_\pi(x) + \frac{2\epsilon\beta}{1-\beta}.$$

as required.

b) Similar to part a)

$$\begin{aligned} Q(x, a) &= r(x, a) + \beta \mathbb{E}_{x,a} \left[\max_{a'} Q(\hat{x}, a') \right] \\ &\leq r(x, a) + \beta\epsilon + \beta \mathbb{E}_{x,a} \left[\max_{a'} Q'(\hat{x}, a') \right] \\ &= r(x, a) + \beta\epsilon + \beta \mathbb{E}_{x,a} [Q'(\hat{x}, \pi(\hat{x}))] \\ &\leq r(x, a) + 2\beta\epsilon + \beta \mathbb{E}_{x,a} [Q(\hat{x}, \pi(\hat{x}))] \end{aligned}$$

Again applying rollout (or equivalently the expansion given in Proposition 4), we see that

$$Q(x, a) \leq \mathbb{E}_{x,a} \left[\sum_t \beta^t (r(x_t, \pi_t) + 2\beta\epsilon) \right] = R_\pi(x) + \frac{2\beta\epsilon}{1-\beta}$$

as required. □

Remark 35. *It is worth noting that once we get within the error of the optimal solution then we also have exactly the optimal policy. E.g. If Q is the optimal Q -factor and $\Delta = \min_{x,a} \{V(x) - Q(x, a) : Q(x, a) \neq V(x)\}$ is the gap between a sub-optimal action and the optimal value action, then so long as we have an approximation Q' such that $\|Q - Q'\|_\infty < \Delta/2$ then Q' must describe the optimal policy. Of course, when the set of actions is continuous, then it is a distinct possibility that $\Delta = 0$ and so we need to revert to the above proposition.*

Positive Programming*

We now consider the case where all rewards are positive and the discount factor β can be set equal to 1. This is called *Positive Programming*. Notice since $\beta = 1$ is possible, we can no longer use the nice contraction properties that we had for a discounted program. Notice that the optimization of interest is now

$$V(x) = \max_{\pi} \lim_{T \rightarrow \infty} \mathbb{E} \left[\sum_{t=0}^T r(X_t, \pi_t) \right].$$

Since the rewards are positive the sum is always increase in T and so we are effectively maximizing over π and T . For this reason, we can recover some nice properties. Also the proof given below leads us to an algorithm called Value Iteration.

The following result holds for positive programming.

Thrm 36. *Consider a positive program the optimal value function $V(x)$ is the minimal non-negative solution to the Bellman equation*

$$R(x) = \max_{a \in \mathcal{A}} \left\{ r(x, a) + \beta \mathbb{E}_{x,a} [R(\hat{x})] \right\}.$$

Thus if we find a policy π whose reward function $R(x, \pi)$ satisfies the Bellman equation, then it is optimal.

Proof. First we show that V is a solution to the Bellman equation. Suppose that $V_T(x)$ is the optimal value function for the positive program with T time steps. (I.e. we set all rewards equal to zero from time T onwards.) By Thrm 23, Bellman's equation holds

$$V_{T+1}(x) = \max_{a \in \mathcal{A}} \left\{ r(x, a) + \beta \mathbb{E}_{x,a} [V_T(\hat{X})] \right\},$$

with $V_0(x) = 0$. Note $V_T(x)$ is increasing in T , since rewards are positive. Thus, the following limit is well defined $V_{\infty}(x) := \sup_T V_T(x)$. Further note that

$$\begin{aligned} V_{\infty}(x) &= \sup_T \max_{a \in \mathcal{A}} \left\{ r(x, a) + \beta \mathbb{E}_{x,a} [V_T(\hat{X})] \right\} \\ &= \max_{a \in \mathcal{A}} \left\{ r(x, a) + \beta \mathbb{E}_{x,a} \left[\sup_T V_T(\hat{X}) \right] \right\} \\ &= \max_{a \in \mathcal{A}} \left\{ r(x, a) + \beta \mathbb{E}_{x,a} [V_{\infty}(\hat{X})] \right\} \end{aligned}$$

(Above, in the 2nd equality, we swap the sup and the max and use Monotone Convergence Theorem to take the limit inside the expectation.) Thus $V_\infty(x)$ satisfies Bellman's equation.

Note that $V(x) \geq V_T(x)$, since the optimal value function for the infinite time horizon experiences positive rewards after time T . Thus

$$V(x) \geq V_\infty(x) := \lim_{T \rightarrow \infty} V_T(x).$$

Further, for any policy π ,

$$V_T(x) \geq R_T(x, \pi).$$

Now take limits $V_\infty(x) \geq R(x, \pi)$. Now maximize over π to see that $V_\infty(x) \geq V(x)$. So $V_\infty(x)$ equals the optimal value function $V(x)$.

Note that if $R(x)$ is any other positive solution to the Bellman Equation, then $R(x) \geq V_0(x) = 0$. And if $R(x) \geq V_T(x)$ then

$$\begin{aligned} R(x) &= \max_{a \in \mathcal{A}} \left\{ r(x, a) + \beta \mathbb{E}_{x,a} [R(\hat{x})] \right\} \\ &\geq \max_{a \in \mathcal{A}} \left\{ r(x, a) + \beta \mathbb{E}_{x,a} [V_T(\hat{x})] \right\} = V_{T+1}(x) \end{aligned}$$

Thus $R(x) \geq \lim_{T \rightarrow \infty} V_T(x) = V(x)$. So we see that the value function is the minimal positive solution to the Bellman equation.

Finally, if a policy π is such that $R(x, \pi)$ solves the Bellman equation. Then clearly we know that $R(x, \pi) \leq V(x)$. But then since $R(x, \pi)$ is smaller than the minimal non-negative solution to the Bellman equation, and it solves the Bellman equation, it must be that $R(x, \pi) = V(x)$ and so the policy is optimal. \square

Negative Programming*

We now consider case where all rewards are negative and the discount factor β can be set equal to 1. This could also be considered to be the case where you minimize positive costs. This is called *Negative Programming*.

In this section, we phrase this as a minimization problem:

$$L(x) = \underset{\pi \in \mathcal{P}}{\text{minimize}} \quad C(x, \pi) := \mathbb{E}_{x_0} \left[\sum_{t=0}^{\infty} \beta^t c(X_t, \pi_t) \right],$$

for $c(x, a) \geq 0$.

We consider stationary policies, which we recall as follows:

Def 37. A policy π is called a stationary policy if its action only depends on the current state (and is non-random and does not depend on time).

The analogous result to Thrm 36 for Negative programming is weaker.

Thrm 38. Consider a negative program, minimizing positive costs. For the minimal non-negative solution to the Bellman equation

$$L(x) = \min_{a \in \mathcal{A}} \{c(x, a) + \beta \mathbb{E}_{x,a} [L(\hat{x})]\}, \quad (12)$$

any stationary policy π that solves the Bellman equation:

$$\pi(x) \in \operatorname{argmin}_{a \in \mathcal{A}} \{c(x, a) + \beta \mathbb{E}_{x,a} [L(\hat{x})]\}$$

is optimal.

So the Bellman equation is still correct, but as the above result suggests, simply finding a solution to the Bellman equation is not sufficient. We need to find the optimal solution first and then we need to solve with a stationary policy.

Proof. The first part of the argument is identical proof to Thrm 36: by considering the limit of value function the finite time horizon MDP, $L_T(x)$, it can be seen that its limit satisfies the Bellman Equation

$$L(x) = \min_{a \in \mathcal{A}} \{l(x, a) + \beta \mathbb{E}_{x,a} [L(\hat{x})]\},$$

and that $\lim L_T(x) = L(x) \leq C(x)$ for any other solution to the Bellman equation. (See the proof of Thrm 36 for more detail.)

Now for a stationary policy, π , that minimizes the Bellman equation

$$\begin{aligned} L(x) &= \min_{a \in \mathcal{A}} \{c(x, a) + \beta \mathbb{E}_{x,a} [L(X_1)]\} \\ &= c(x, \pi(x)) + \beta \mathbb{E}_{x, \pi(x)} [L(X_1)] \\ &= c(X_0, \pi(X_0)) + \beta \mathbb{E}_{X_0, \pi(X_0)} [c(X_1, \pi(X_1)) + \beta \mathbb{E}_{X_1, \pi(X_1)} [L(X_2)]] \\ &= C_1(x, \pi) + \beta^2 \mathbb{E}_{x, \pi} [L(X_2)] \\ &\vdots \\ &= C_T(x, \pi) + \beta^T \mathbb{E}_{x, \pi} [L(X_{T+1})]. \end{aligned}$$

Thus

$$L(x) = C_T(x, \pi) + \beta^T \mathbb{E}_{x, \pi}[L(X_{T+1})] \geq C_T(x, \pi) \xrightarrow[T \rightarrow \infty]{M.C.T.} C(x, \pi).$$

So the policy has lower cost, and thus is optimal. \square

Average Programming*

We now consider a slightly different approach to dynamic programming without a discount factor. The approach on positive programming and negative programming are applicable to the case where there are a small number of large rewards. Like optimal stopping problems, which we discuss in Section 0.6. However, if there are large numbers of small costs (or rewards) accumulated over time then, without discounting, the total sum of costs will be infinite. So in this case it is better to look at the long run average cost that we get from our policy.

Suppose that for finite-time cost function

$$C_T(x_0, \pi) = \mathbb{E}\left[\sum_{t=0}^{T-1} c(x_t, \pi_t)\right].$$

We look at the limit of the average cost

$$\bar{C}(\pi) = \lim_{T \rightarrow \infty} \frac{C_T(x_0, \pi)}{T},$$

if such a limit exists, and we attempt to find the minimal such policy.

We consider the case of minimizing costs. However, the analogous result for maximizing rewards holds.

Deriving a Bellman Equation. Before proving a formal result, we show how to heuristically get a Bellman equation in this setting.

We could think of the cost function for the optimal policy $C_T(x, \pi)$ as consisting of three components:

$$C_T(x, \pi) \approx T \cdot L + \kappa(x) + \epsilon(x, T) \tag{13}$$

where

- L is the optimal long run average cost.

- $\kappa(x)$ represents the short run cost from starting in state x .
- $\epsilon(x, T)$ is an error term that goes to zero as $T \rightarrow \infty$.

Here if we fix a stationary policy π then the resulting process is a Markov chain. Let's assume that this chain is positive recurrent. Then the process will converge to an equilibrium with an average cost L . The initial condition will be forgotten over time but will have some initial impact $\kappa(x)$.

Now let's consider the impact of the approximation (13) on the finite time Bellman equation:

$$L_{T+1}(x) = \min_a \{c(x, a) + \mathbb{E}_{x,a}[L_T(\hat{x})]\} .$$

After substituting (13), we get

$$(T + 1) \cdot L + \kappa(x) = \min_a \{c(x, a) + \mathbb{E}_{x,a}[T \cdot L + \kappa(\hat{x})]\}$$

So rearranging gives

$$L = \min_{a \in \mathcal{A}} \{c(x, a) + \mathbb{E}_{x,a}[\kappa(\hat{x})] - \kappa(x)\}$$

The above expression is the Bellman equation for minimizing average cost in a Markov decision process. Note we can always add a constant to $\kappa(x)$ in the above theorem. So it is worth specifying that $\kappa(x_0) = 0$ for some arbitrary state x_0 .

Main Result. The following result shows that if we can solve the Bellman equation and find a policy associated with the solution, then we have an optimal solution.

Thrm 39. *If we can find a constant L and a bounded function $\kappa(x)$ such that for all $x \in \mathcal{X}$*

$$L = \min_{a \in \mathcal{A}} \{c(x, a) + \mathbb{E}_{x,a}[\kappa(\hat{x})] - \kappa(x)\} \quad (\text{AvgBell})$$

and if given L and $\kappa(x)$ we can chose a policy such that

$$\pi(x) \in \operatorname{argmin}_{a \in \mathcal{A}} c(x, a) + \mathbb{E}_{x,a}[\kappa(\hat{x})] - \kappa(x)$$

then L is the minimum average reward and π is the optimal policy.

Proof. To prove the result we make the following claim:

Claim: If L' and $\kappa'(x)$ bounded functions are such that

$$L' \leq \min_{x \in \mathcal{X}} \min_{a \in \mathcal{A}} \{c(x, a) + \mathbb{E}_{x,a}[\kappa'(\hat{x})] - \kappa'(x)\}$$

then

$$L' \leq \liminf_{T \rightarrow \infty} \frac{C_T(x, \pi')}{T}$$

for all policies π' .

Proof of claim: Let

$$M_t = \kappa'(X_t) + \sum_{\tau=0}^{t-1} \{c(X_\tau, \pi'_\tau) - L'\}.$$

Our assumption on L' , M_t is a sub-Martingale:

$$\mathbb{E}[M_{t+1} - M_t | X_t = x, \pi'_t = a] = \mathbb{E}_{x,a}[\kappa'(\hat{x})] - \kappa'(x) + c(x, a) - L' \geq 0.$$

Thus

$$\kappa'(x) = \mathbb{E}[M_0] \leq \mathbb{E}[M_T] = \mathbb{E}[\kappa'(X_T)] - L'T + C_T(x, \pi')$$

and so

$$\liminf_{T \rightarrow \infty} \frac{C_T(x, \pi')}{T} \geq L'.$$

✓ *claim proved.*

Clearly L , κ and π satisfy the conditions of our claim. Also, notice for L , κ and π , the process M_t , as defined above, is a Martingale. So

$$\kappa(x) = \mathbb{E}[M_0] = \mathbb{E}[M_T] = \mathbb{E}[\kappa(X_T)] - LT + C_T(x, \Pi)$$

and so

$$\lim_{T \rightarrow \infty} \frac{C_T(x, \tilde{\pi})}{T} = L \leq \liminf_{T \rightarrow \infty} \frac{C_T(x, \pi')}{T}$$

for all policies π' . So π is optimal. \square

Remark 40. The analogous result holds, with an identical proof for maximizing. Here the Bellman equation is

$$V = \max_{a \in \mathcal{A}} \{r(x, a) + \mathbb{E}_{x,a}[\rho(\hat{x})] - \rho(x)\}.$$

Since we established that minimizing (negative programming) is more difficult than maximizing (positive programming), we proved the result for minimizing average costs.

A Martingale Principle of Optimal Control.

We give a Martingale condition for optimal control. This result is analogous to Prop 18 which applies to Markov chains.

Prop 41 (A Martingale Principle of Optimal Control.). *Consider discounted program. Suppose for a bounded function $R : \mathcal{X} \rightarrow \mathbb{R}$ we define a process $(M_t : t \in \mathbb{Z}_+)$ whose increments, $\Delta M(X_t) := M_{t+1} - M_t$, are given by*

$$\Delta M(x) = R(x) - \beta R(\hat{x}) - r(x, \pi(x))$$

If M_t is a submartingale for all policies π' and, for some π , M_t is a martingale, then π is the optimal policy and $R(x) = R(x, \pi)$.

Proof. M_t is a submartingale [resp. martingale] iff

$$M_t^\beta := \sum_{s=0}^t \beta^s \Delta M(X_s)$$

is a submartingale [resp. martingale]. Taking expectations,

$$0 \leq \mathbb{E}_x[M_t^\beta] = \mathbb{E}_x\left[R(x) - \beta^{t+1}R(X_{t+1}) - \sum_{s=0}^t \beta^s r(X_s, \pi'(X_s))\right]$$

Rearranging and letting $t \rightarrow \infty$ gives, for π' ,

$$R(x) \geq \mathbb{E}\left[\sum_{s=0}^{\infty} \beta^s r(X_s, \pi'(X_s))\right],$$

where the inequality above holds with equality if M_t^β is a martingale for some π . Thus we see that $R(x) \geq V(x)$, where $V(x)$ is the value function for the MDP and, for the martingale policy, $R(x) = V(x) = R(x, \pi)$. \square

Occupancy Measure

We can construct a perspective on Markov decision processes based on how often it takes an action and how often it visits each state.

We consider a discounted program. Here we can rewrite the reward function for any policy π as follows

$$\begin{aligned}
R(x_0, \pi) &= \mathbb{E}_{x_0} \left[\sum_{t=0}^{\infty} \beta^t r(X_t, \pi_t) \right] \\
&= \sum_{x \in \mathcal{X}} \sum_{a \in \mathcal{A}} \sum_{t=0}^{\infty} \beta^t \mathbb{E}_{x_0} [\mathbb{I}[X_t = x, \pi_t = a] r(x, a)] \\
&= \sum_{x, a} r(x, a) \underbrace{\sum_t \beta^t \mathbb{P}_{x_0}(X_t = x, \pi_t = a)}_{\eta_{\pi}(x, a)} \\
&= \sum_{x, a} \eta_{\pi}(x, a) r(x, a) \tag{14}
\end{aligned}$$

Def 42 (Occupancy Measure). *The term $\eta_{\pi}(x, a)$ is the discounted occupancy measure:*

$$\eta_{\pi}(x, a) := \sum_{t=0}^{\infty} \beta^t \mathbb{P}_{x_0}(X_t = x, \pi_t = a).$$

It counts how much time we spend in state x taking action a .

We can use this to show that for any policy π (where the decision at time depends on the time and the past states and actions) can be replaced by a simpler stationary policy. (Recall from earlier that a stationary policy is a policy where the probability of choosing an action only depends on the current state.)

Prop 43. *For any policy π , there exists a stationary policy π' such that*

$$\eta_{\pi'} \equiv \eta_{\pi}$$

Proof. Take $\eta_{\pi}(x, a)$ and define $\eta_{\pi}(x) = \sum_a \eta_{\pi}(x, a)$, then define π' such that

$$\pi'(a|x) = \frac{\eta_{\pi}(x, a)}{\eta_{\pi}(x)},$$

(if $\eta_{\pi}(x) = 0$ then we can take $\pi'(a|x)$ equal to some arbitrary policy at a .)

Note that the stationary policy π' defines a Markov chain with transition matrix given by

$$P'(\hat{x}|x) = \sum_a P(\hat{x}|x, a) \pi'(a|x) = \sum_a P(\hat{x}|x, a) \frac{\eta_{\pi}(x, a)}{\eta_{\pi}(x)}.$$

So notice that

$$\begin{aligned}
\eta_\pi(x) &= \sum_{t=0}^{\infty} P_{x_0}(X_t = x) \\
&= \mathbb{I}[x_0 = x] + \beta \sum_{t=0}^{\infty} \beta^t \mathbb{P}_{x_0}(X_{t+1} = x) \\
&= \mathbb{I}[x_0 = x] + \beta \sum_{t=0}^{\infty} \beta^t \sum_{x', a} \mathbb{P}_{x_0}(X_t = x', \pi_t = a) P(x|x, a) \\
&= \mathbb{I}[x_0 = x] + \beta \sum_{x', a} P(x|x', a) \beta \sum_{t=0}^{\infty} \beta^t \mathbb{P}_{x_0}(X_t = x', \pi_t = a) \\
&= \mathbb{I}[x_0 = x] + \beta \sum_{x', a} P(x|x', a) \eta_\pi(x', a) \\
&= \mathbb{I}[x_0 = x] + \beta \sum_{x'} P'(x|x') \eta_\pi(x')
\end{aligned}$$

where in the third equality above we apply the Markov property, and in the final equality, we apply the definition of $P'(x|x')$ given above. It follows by an identical argument that the policy π' and $\eta_{\pi'}$ satisfy the same expression.

Therefore $\eta_\pi = (\eta_\pi(x) : x \in \mathcal{X})$ and $\eta_{\pi'} = (\eta_{\pi'}(x) : x \in \mathcal{X})$ both solve the linear equation

$$\eta = e_{x_0} + \beta \eta P'$$

where e_{x_0} is the zero-one vector with a one in the x_0 component. As proven in Proposition 15, this equation has a unique solution so

$$\eta_\pi = (I - \beta P')^{-1} e_{x_0} = \eta_{\pi'}.$$

Further by definition

$$\eta_{\pi'}(x, a) = \eta_{\pi'}(x, a) \pi'(x|a) = \eta_\pi(x, a).$$

Thus the two occupancy measures are identical. \square

Since the reward function of an MDP is characterized by its occupancy measure (as we saw in (42)), we can now see that a Markov decision process optimizing over all policies can be restricted to an optimization problem just over stationary policies. (This was similarly observed for finite time MDPs after Remark 24)

Infinite Time Examples

Ex 44 (Machine Repair). Each day a machine is either working or broken. If broken, then the day is spent repairing the machine at a cost $8c$. If the machine is working, then it can be either run unattended or attended at a cost 0 or c . In each case the probability of the machine breaking is p and q respectively. Costs are discounted by $\beta \in (0, 1)$.

The objective is to minimize the infinite horizon discounted cost. Letting $F(0)$ and $F(1)$ be the minimal cost starting on a day were the machine starts broken or working, respectively. Show that it is optimal to run the machine unattended iff $7p - 8q \leq \beta^{-1}$.

Ex 45 (Symmetric Random Walk). Consider a symmetric random walk on \mathbb{Z} . We wish to choose a time to stop that minimizes the cost $k(x) = \exp\{-x\}$ where x gives the value of the walk when stopped. Argue that $W_s(x)$ the optimal value function for the s -time horizon problem is constant over s . Argue that

$$\lim_{s \rightarrow \infty} W_s(x) \neq W(x)$$

where $W(x)$ is the optimal value function for the infinite time optimal stopping problem.

(Note for this Negative program we have a solution to the Bellman equation that is not optimal.)

Ex 46 (Repeat Prisoner's Dilemma). Two men are taken prisoner by the authorities. They are interviewed separately and asked to confess to the other prisoners involvement in a crime. A prisoner that does not confess receives 1 year in prison. A prisoner that confesses adds 6 years to the other prisoner's sentence. This game can be expressed by the matrix

$$\begin{array}{cc} & \begin{array}{cc} \text{don't confess} & \text{confess} \end{array} \\ \begin{array}{c} \text{don't confess} \\ \text{confess} \end{array} & \begin{pmatrix} (1, 1) & (7, 0) \\ (0, 7) & (6, 6) \end{pmatrix} \end{array}$$

For each entry (a, b) in the above matrix the left entry a gives the row players sentence and b gives the column player's sentence. Suppose, given what the other prison does, each prison act selfishly to minimize their time in jail.

i) Argue that the each prisoner will confess.

We now assume the criminals from the Prisoner's Dilemma are repeat offenders. They are repeatedly arrested by the police and interviewed. Each time they can choose whether to confess or not and afterwards they find out if their fellow prisoner confessed or not. The payoffs are the same in our previous example, except each time they meet their payoffs are discounted by a multiplicative factor $(1+r)$, for some $r > 0$.

ii) Show that if this repeated game is played for a finite set of times $t = 0, 1, \dots, T$ then it is in the interest of each player to confess at each time. [Hint: Argue for time T and work backwards].

A punishing strategy is a strategy where the prisoner will not confess at every round unless his fellow prisoner confesses. If his fellow prisoner confesses at one time instance then the prisoner will confess for all subsequent time. I.e. the strategy places a heavy penalty on the opponent for confessing.

iii) If this repeated game is played for an infinite set of times $t = 0, 1, 2, 3, \dots$ and if r is suitably small then show that both players playing a punishing strategy is a Nash Equilibrium.

References

The distinction between discounted, positive and negative programming is made by Blackwell [9, 10] and Stauch [45]. Average reward is considered by Howard [24]. The distinction between these cases is quite standard in different textbooks for instance see [37]. A early review of martingale conditions for optimal control is by Davis [16].

0.5 Algorithms for MDPs

- Policy Improvement; Policy Evaluation.
 - Value Iteration; Policy Iteration.
 - Temporal Differences; Q-factors.
 - Linear Program formulation; Asynchronous Value Iteration.
-

We now need some algorithms to solve MDPs. We now know that a solution to Bellman's equation will usually solve an MDP. In the first instance, we can look to solve Bellman's equation. For infinite time MDPs, we cannot exactly solve Bellman's equation from some initial state – like we could for finite time MDP. So we develop some iterative procedures to solve the Bellman equation and more generally to solve an MDP.

At a high level, for a Markov Decision Processes (where the transitions P_{xy}^a are known), an algorithm solving a Markov Decision Process involves two steps:

- (Policy Improvement) Here you take your initial policy π_0 and find a new improved policy π , for instance by solving Bellman's equation:

$$\pi(x) \in \operatorname{argmax}_{a \in \mathcal{A}} \{r(x, a) + \beta \mathbb{E}_{x,a} [R(\hat{x}, \pi_0)]\} .$$

- (Policy Evaluation) Here you find the value of your policy. For instance by finding the reward function for policy π :

$$R(x, \pi) = \mathbb{E}_{x,\pi} \left[\sum_{t=0}^{\infty} \beta^t r(X_t, \pi(X_t)) \right] .$$

Value Iteration

Value Iteration provides an important practical scheme for approximating the solution of an infinite time horizon Markov decision process.

Def 47 (Value iteration). Take $V_0(x) = 0$ for all x and recursively calculate


$$V_{s+1}(x) = \max_{a \in \mathcal{A}} \left\{ r(x, a) + \beta \mathbb{E}_{x,a} \left[V_s(\hat{X}) \right] \right\}$$

for $s = 1, 2, \dots$ this is called value iteration. We can then define a policy by

$$\pi_{s+1}(x) \in \operatorname{argmax}_{a \in \mathcal{A}} \left\{ r(x, a) + \beta \mathbb{E}_{x,a} \left[V_s(\hat{X}) \right] \right\}$$

We can think of the two display equations above, respectively, as the policy evaluation and policy improvement steps. However, that we don't really need to keep a record of the policy found at each iteration.

A Quick Example. Let's do a quick example. In the figure below there is a robot, that can go left or right by one square in each time-step. If it ends in the far left square the robot gets a reward of 1 and we finish. If it ends at the far right square we finish with a reward of 2.

END						END
1						2
$t=0$	0	0	0	0	0	
$t=1$	1	0	0	0	0	2
$t=2$	1	1	0	2	2	2
$t=3$	1	1	2	2	2	2
$t=4$	1	2	2	2	2	2
$t=5$	2	2	2	2	2	2
$t=6$	2	2	2	2	2	2

When implementing value iteration. We start at time $t = 0$ with a row of zeros. We then look at whether we should go left or right by looking at the value of the assigned to that square in the previous iteration (note the value of the end squares does not change). At

time $t = 1$, we see that the square next to the reward 1 square is 1 and the square next to the reward 2 square is 2, while the other squares remain zero. So we go from a row of all zeros to the row with values $(1, 0, 0, 0, 2)$. Continuing in this way we get the remaining rows. At the final iteration the value function does not change. This means we have found the optimal value function (See Exercise 59).

Notice in this example it is not too clear what is the optimal policy even if we have the optimal value function. This is a good reason to consider including a discount factor just less than 1, as it will encourage the use of shorter paths.

Value Iteration for positive programming. The following result shows that Value Iteration converges to the optimal policy.

Thrm 48. *For positive programming, i.e. where all rewards are positive and the discount factor β belongs to the interval $(0, 1]$, then*

$$0 \leq V_s(x) \leq V_{s+1}(x) \nearrow V(x), \quad \text{as } s \rightarrow \infty.$$

Here $V(x)$ is the optimal value function.

The following lemma is the key property for value iterations convergence, as well as a number of other algorithms.

Lemma 3. *For reward function $R(x)$ define*

$$\mathcal{T}R(x) = \max_{a \in \mathcal{A}} \{r(x, a) + \beta \mathbb{E}_{x,a} [R(\hat{x})]\}.$$

Show that if $R(x) \geq \tilde{R}(x)$ for all $x \in \mathcal{X}$ then $\mathcal{T}R(x) \geq \mathcal{T}\tilde{R}(x)$ for all $x \in \mathcal{X}$

Proof. Clearly,

$$r(x, a) + \beta \mathbb{E}_{x,a} [R(\hat{x})] \geq r(x, a) + \beta \mathbb{E}_{x,a} [\tilde{R}(\hat{x})].$$

Now maximize both sides over $a \in \mathcal{A}$. □

We now prove Theorem 48.

Proof of Thrm 48. Note that $V_1(x) = \max_a r(x, a) \geq 0 = V_0(x)$. Now, since $V_{s+1}(x) = \mathcal{T}V_s(x)$, repeatedly applying Lemma 3 to the inequality $V_1(x) \geq V_0(x)$ gives that

$$V_{s+1}(x) \geq V_s(x).$$

Since $V_s(x)$ is increasing $V_s(x) \nearrow V_\infty(x)$ for some function V_∞ . We must show that V_∞ is the optimal value function from the MDP.

Note that $V_s(\cdot)$ is the optimal value function for the finite time MDP with rewards $r(x, a)$ and s time steps. So $V(x) \geq V_s(x)$ and thus $V(x) \geq V_\infty(x)$. Further, for any policy π ,

$$V_s(x) \geq R_s(x, \pi).$$

Now take limits $V_\infty(x) \geq R(x, \Pi)$. Now maximize over π to see that $V_\infty(x) \geq V(x)$. So $V_\infty(x) = V(x)$ as required. \square

Value Iteration: Discounted Programming. We now consider value iteration on a discounted MDP. The proof uses the contraction property that we found in Theorem 32.⁸ We prove this in Lemma 4 below.

The theorem below shows that the value iteration converges fast to the solution of a discounted program.

Theorem 2. *For a discounted program with discount factor $\beta \in (0, 1)$ it holds that*

$$\|V_s - V\|_\infty \leq \beta^s \|V_0 - V\|_\infty$$

where here V is the optimal value function.

For the above proof we require the following lemma.

Lemma 4. *If we define*

$$\mathcal{T}V(x) = \max_{a \in \mathcal{A}} \{r(x, a) + \beta \mathbb{E}_{x,a}[V(\hat{x})]\}$$

then

$$\|\mathcal{T}V - \mathcal{T}V'\|_\infty \leq \beta \|V - V'\|_\infty$$

Proof. Notice if we let Q and Q' be the Q -function of V and V' . I.e.

$$Q(x, a) := r(x, a) + \beta \mathbb{E}_{x,a}[V(\hat{x})]$$

then

$$|Q(x, a) - Q'(x, a)| = \beta |\mathbb{E}_{x,a}[V(\hat{x}) - V'(\hat{x})]| \leq \beta \|V - V'\|_\infty \quad (15)$$

⁸This is analogous to the way that we used the monotonicity property from Theorem 36 in the positive programming proof above.

Finally

$$\begin{aligned} |\mathcal{T}V(x) - \mathcal{T}V'(x)| &= |\max_a Q(x, a) - \max_{a'} Q(x, a')| \\ &\leq \max_a |Q(x, a) - Q(x, a')| \\ &\leq \beta \|V - V'\|_\infty \end{aligned}$$

as required. (If the first inequality above is not immediately obvious please see Lemma 33. The 2nd equality above is (15).) \square

Proof of Theorem 2. We know from Theorem 32, that the optimal value function satisfies $\mathcal{T}V = V$. Also, by definition, value iteration performs the update $V_s = \mathcal{T}V_{s-1}$. Thus

$$\|V_s - V\| = \|\mathcal{T}V_{s-1} - \mathcal{T}V\| \leq \beta \|V_{s-1} - V\| \leq \dots \leq \beta^s \|V_0 - V\|$$

as required. \square

Code for Value iteration.

```
def Value_Iteration (V,P,r ,discount ,time) :
    ''' Value Iteration - a numerical solution to a MDP

    # Arguments:
        P - P[a][x][y] gives probability of x -> y for action a
        r - r[a][x][y] gives reward for x -> y for action a
        V - V[x] gives value for state x
        discount - discount factor
        time - number of iterations

    # Returns:
        ... Value function and policy from value iteration
    ...

    number_of_actions = len(P)
    number_of_states = len(P[0])

    Q = np.zeros ((number_of_actions , number_of_states))

    for _ in range(time) :
        for a in range(number_of_actions) :
            for x in range(number_of_states) :
                Q[a][x] = np.dot(P[a][x] , r[a][x]+discount*V)
            V_new = np.amax(Q, axis=0)

    pi = np.argmax(Q, axis=0)

    return V_new, pi
```

Policy Iteration

We consider a discounted program with rewards $r(x, a)$ and discount factor $\beta \in (0, 1)$. For policy iteration, we update policies and after assessing their values. The idea is we take with a policy π ; We evaluate its reward $R(x, \pi)$; We assume that we are going to follow policy π from time $t = 1$ onwards; We look for the best action that we can make *now* at time $t = 0$. This "best action" defines a new policy.

The reward for following action a for one step and then following policy π thereafter is:

$$Q_\pi(x, a) := r(x, a) + \beta \mathbb{E}_{x,a} [R(\hat{x}, \pi)],$$

(which you might recall is the Q-factor for policy π , (31)). We choose the action that maximizes this for each x :

$$\hat{\pi}(x) \in \operatorname{argmax}_{a \in \mathcal{A}} r(x, a) + \beta \mathbb{E}_{x,a} [R(\hat{X}, \pi)].$$

The change from π to $\hat{\pi}$ gives one iteration of policy iteration.

Def 49 (Policy Iteration). *Given a stationary policy π ,*

1. *We calculate the reward of policy π , $R(x, \pi)$.*
2. *We define a new stationary policy, $\mathcal{I}\pi$, by*

$$\mathcal{I}\pi(x) \in \operatorname{argmax}_{a \in \mathcal{A}} r(x, a) + \beta \mathbb{E}_{x,a} [R(\hat{x}, \pi)]$$

Policy iteration is the algorithm that takes

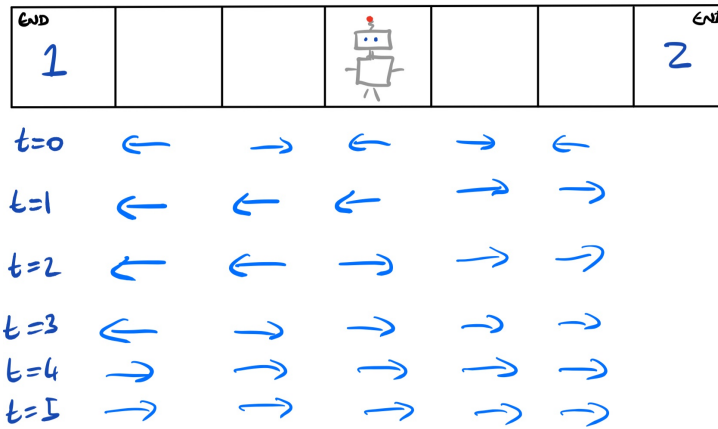
$$\pi_{n+1} = \mathcal{I}\pi_n$$

Step 1 corresponds to the policy evaluation step and step 2 corresponds to the policy improvement step. We need to evaluate $R(x, \pi)$ for a stationary policy π which we can do as follows:

Remark 50 (Calculating $R(x, \pi)$). *If we think of the policy as a matrix $P = (P_{xy}^\pi : x, y \in \mathcal{X})$ and rewards as a vector $r = (r(x, \pi(x)) : x \in \mathcal{X})$ then $R = (R(x, \pi) : x \in \mathcal{X})$ is the vector such that*

$$R = r + \beta P R$$

which is solved by $R = (I - \beta P)^{-1} r$. I.e. evaluating a policy is really a little matrix algebra.



Example Again. We go back to our robot example from before, but now for policy iteration. See the figure below.

We start with an arbitrary policy at $t = 0$. Notice for the far-left square (next to the reward of 1) if we follow the policy then we get a reward of 1. While are at the far right square, the robot moves left then right indefinitely. Thus it receives zero reward from this square. For one iteration of policy iteration, we take a square (lets take the far right square), we then look what happens if we change its direction and then follow the previous policy. In the case of the right-hand square it is clearly better to change from left (with long-run reward of 0) to right (with a reward of 2). Similar for the square to the left of the robot. It was right at time $t = 0$ but left at time $t = 1$. This is because going left and then following the policy from time $t = 0$ gives it a reward of 1 rather than 0 which it is what it was getting previously.

Notice in the final iteration the policy does not change. This implies Bellman’s equation holds and thus we have found an optimal policy. (See Exercise 59.)

Convergence of Policy Iteration. The following result proves that policy iteration converges to the optimal solution.

Thrm 51. For a positive program (or discounted program), under Policy Iteration

$$R(x, \pi_n) \nearrow V(x) \quad \text{as } n \rightarrow \infty$$

Proof. By the optimality of $\mathcal{I}\pi$ with respect to π , we have

$$R(x, \pi) = r(x, \pi(x)) + \beta \mathbb{E}_{x, \pi(x)} [R(\hat{x}, \pi)] \leq r(x, \mathcal{I}\pi(x)) + \beta \mathbb{E}_{x, \mathcal{I}\pi(x)} [R(\hat{x}, \pi)] .$$

Thus from the last part of Thrm 15, we know that $R(x, \pi) \leq R(x, \mathcal{I}\pi)$. This shows that Policy iteration improves solutions. Now we must show it improves to the optimal solution.

First note that

$$\begin{aligned} r(x, a) + \beta \mathbb{E}_{x,a} [R(\hat{x}, \pi)] &\leq r(x, \mathcal{I}\pi(x)) + \beta \mathbb{E}_{x, \mathcal{I}\pi(x)} [R(\hat{x}, \pi)] \\ &\leq r(x, \mathcal{I}\pi(x)) + \beta \mathbb{E}_{x, \mathcal{I}\pi(x)} [R(\hat{x}, \mathcal{I}\pi)] = R(x, \mathcal{I}\pi). \end{aligned}$$

We can use the above inequality to show that the following process is a supermartingale

$$M_t = \sum_{s=0}^{t-1} \beta^s r(X_s, \pi^*(X_s)) + \beta^t R(X_t, \pi_{T-t})$$

where $\pi^*(x)$ is the optimal policy. The interpretation of M_t is the reward of following π^* for t steps and π_{T-t} thereafter.⁹ To see taking expectations with respect to the optimal policy π^* gives

$$\begin{aligned} &\mathbb{E}^* [M_{t+1} - M_t | \mathcal{F}_t] \\ &= \beta^t \mathbb{E}^* \left[\beta R(X_{t+1}, \pi_{T-t-1}) + r(X_t, \pi^*(X_t)) - R(X_t, \pi_{T-t}) \middle| \mathcal{F}_t \right] \\ &= \beta^t \mathbb{E}^* \left[\beta \mathbb{E}_{X_t, \pi^*(X_t)}^* \left[\beta R(\hat{X}, \pi_{T-t-1}) + r(X_t, \pi^*(X_t)) - R(X_t, \pi_{T-t}) \right] \middle| \mathcal{F}_t \right] \\ &\leq 0. \end{aligned}$$

So M_t is a supermartingale. (I.e. π_{T-t} improves the policy in the sense that it is better to do π^* for t steps and then π_{T-t} than do π^* for $t+1$ steps and then do π_{T-t-1} .)

Since M_t is a supermartingale:

$$R(x, \pi_T) = \mathbb{E}_x^* [M_0] \geq \mathbb{E}_x^* [M_T] = \underbrace{\mathbb{E}_x^* \left[\beta^T R(X_T, \pi_0) \right]}_{\xrightarrow{T \rightarrow \infty} 0} + \underbrace{R_T(x, \pi^*)}_{\xrightarrow{T \rightarrow \infty} V(x)}. \quad (16)$$

Therefore, as required, $\lim_{T \rightarrow \infty} R(x, \pi_T) \geq V(x)$. \square

Remark 52 (Discounted Programming.). *Notice with the above theorem, we have also proven convergence for a discounted program. Take a discounted program, we can redefine rewards*

$$r_{new}(x, a) := r(x, a) + r_{\max} \geq 0$$

⁹Note we are implicitly assuming an optimal stationary policy exists. We can remove this assumption by considering a ϵ -optimal (non-stationary) policy. However, the proof is a little cleaner under our assumption.

where $r_{\max} = \max_{x', a'} |r(x', a')|$ then these reward define an equivalent a positive program. Moreover, the Policy Iteration updates are unaffected by this constant factor change in rewards. So the above result implies convergence for discounted programs. Further note that the both terms in (16) converge at rate β^T .


```

def Policy_Iteration(pi,P,r,discount):
    ''' Policy Iteration - a numerical solution to a MDP

    # Arguments:
        P - P[a][x][y] gives probability of x -> y for action a
        r - r[a][x][y] gives reward for x -> y for action a
        pi - pi[x] gives action for state x
        discount - discount factor

    # Returns:
        policy from one policy iteration
        value function of input policy
    ...

    # Collate array of states and actions
    number_of_actions, number_of_states = len(P), len(P[0])
    Actions, States = np.arange(number_of_actions), np.arange(
number_of_states)

    # Get transitions and rewards of policy pi
    P_pi = np.array([P[pi[x]][x] for x in States ])
    r_pi = np.array([r[pi[x]][x] for x in States ])
    Er_pi = [ np.dot(P_pi[x], r_pi[x]) for x in States]

    # Calculate Value of pi
    I = np.identity(number_of_states)
    A = I - discount * P_pi
    R_pi = np.linalg.solve(A, Er_pi)

    # Calculate Q_factors of pi
    Q = np.zeros((number_of_actions, number_of_states))
    for a in range(number_of_actions):
        for x in range(number_of_states):
            Q[a][x] = np.dot(P[a][x], r[a][x]+discount*R_pi)

    # policy iteration update
    pi_new = np.argmax(Q, axis=0)

    return pi_new, R_pi

```

Linear Programming Approach.

Although the previous two methods are specific to dynamic programming. We can reduce the dynamic programming problem into a linear optimization problem. After this there are a host of optimization algorithms to solve this class of problems, the simplex method and interior point methods being two of the most popular. (We do not cover these optimization algorithms here. There are however plenty of good references for more details).

Recall that for any function $(R(x) : x \in \mathcal{X})$, we define

$$\mathcal{T}R(x) = \max_{a \in \mathcal{A}} \{r(x, a) + \beta \mathbb{E}_{x, a}[R(\hat{x})]\} .$$

We want to solve the Bellman equation:

$$V(x) = \max_{a \in \mathcal{A}} r(x, a) + \beta \mathbb{E}_{x, a}[V(\hat{x})]$$

where $(V(x) : x \in \mathcal{X})$. In other words,

$$\mathcal{T}V(x) = V(x) .$$

Further it is not hard to show that

$$R(x) \geq \mathcal{T}R(x), \quad \forall x \quad \implies \quad R(x) \geq V(x), \quad \forall x$$

(We prove this in Lemma 5 after Proposition 53 below.) This shows that $V(x)$ is the smallest vector such that

$$V(x) \geq r(x, a) + \beta \sum_{\hat{x}} V(\hat{x})P(\hat{x}|x, a) .$$

In other words, we want to solve

$$\begin{aligned} & \text{Minimize} && \sum_{x \in \mathcal{X}} \xi(x)V(x) && \text{(LP)} \\ & \text{subject to} && V(x) \geq r(x, a) + \beta \sum_{\hat{x}} V(\hat{x})P(\hat{x}|x, a), \quad \forall x \in \mathcal{X}, a \in \mathcal{A} \\ & \text{over} && (V(x) : x \in \mathcal{X}) \in \mathbb{R}^{\mathcal{X}} \end{aligned}$$

We can apply a linear programming or convex optimization algorithm, such as the Simplex Algorithm, to solve this linear program.¹⁰

¹⁰We may need to apply two stage simplex as the initial zero solution is not feasible.

As we will prove shortly, the dual of this linear program is

$$\begin{aligned}
& \text{Maximize} && \sum_{x \in \mathcal{X}} \eta(x, a) r(x, a) && \text{(Dual)} \\
& \text{subject to} && \sum_{\hat{a} \in \mathcal{A}} \eta(\hat{x}, \hat{a}) = \xi(\hat{x}) + \beta \sum_{x \in \mathcal{X}} \sum_{a \in \mathcal{A}} \eta(x, a) P(\hat{x}|x, a), && \forall \hat{x} \in \mathcal{X} \\
& \text{over} && (\eta(x, a) : x \in \mathcal{X}, a \in \mathcal{A}) \in \mathbb{R}_+^{\mathcal{X} \times \mathcal{A}}.
\end{aligned}$$

In the above $\eta(x, a)$ is the discounted occupancy measure of the chain, which we recall from Definition 42 in Section 0.4. A brief refresher on Lagrangian optimization and duality is given in the appendix.

Prop 53. *The Dual of (LP) is (Dual).*

Proof. The Lagrangian is

$$\begin{aligned}
L(V; \eta) &= \sum_{x \in \mathcal{X}} \xi(x) V(x) - \sum_{x \in \mathcal{X}} \sum_{a \in \mathcal{A}} \eta(x, a) \left(V(x) - r(x, a) - \beta \sum_{\hat{x}} P(\hat{x}|x, a) V(\hat{x}) \right) \\
&= \sum_x V(x) \left[\xi(x) - \sum_a \eta(x, a) + \beta \sum_{\hat{x}, a} \eta(\hat{x}, a) P(x|\hat{x}, a) \right] + \sum_{x, a} \eta(x, a) r(x, a)
\end{aligned}$$

For a well-defined Lagrangian, we need the term in the square brackets to be zero. This gives the constraint in the dual problem. Also for complementary slackness, we require that $\eta(x, a)$ is non-negative. After taking account for this, we see that we arrive at the dual problem (Dual). \square

Lemma 5. *For a positive program or a discounted program it holds that*

$$R(x) \geq \mathcal{T}R(x), \quad \forall x \quad \implies \quad R(x) \geq V(x), \quad \forall x.$$

Proof. We prove the result with a rollout argument (cf. remark 17). Specifically let π^* be an optimal stationary policy and let $(\hat{x}_t : t \in \mathbb{Z}_+)$

be the Markov chain defined by this policy.¹¹ Then, by assumption

$$\begin{aligned}
 R(x) &\geq r(x, \pi^*(x)) + \beta \mathbb{E}_{x, \pi^*(x)}[R(\hat{x})] \\
 &\geq \mathbb{E}_{x, \pi^*}[r(\hat{x}_0, \pi^*(\hat{x}_0)) + \beta r(\hat{x}_1, \pi^*(\hat{x}_1))] + \beta^2 \mathbb{E}[R(\hat{x}_2)] \\
 &\quad \vdots \\
 &\geq R_T(x, \pi^*) + \beta^{T+1} \mathbb{E}_{x, \pi^*}[R(\hat{x}_{t+1})] \\
 &= V(x) + o(1)
 \end{aligned}$$

Letting $T \rightarrow \infty$ gives the result. □

¹¹As usual if there is no optimal stationary policy we can take an ϵ -optimal stationary policy and let ϵ go to zero at the end of the proof.

Asynchronous Value Iteration.

Value iteration assumed that we updated all states at each iteration. If the number of states is large then this can mean each iteration is expensive. However, this is not required for convergence. We now consider the setting where states are updated asynchronously. Here we assume that each state is updated one at a time in an arbitrary order.

We also allow for these iterations to occur without up to date value function estimates. This is important because if the value iteration algorithm is implemented in parallel over multiple processors. As there may need to be communication between different components.

The model setting. We focus on the case of discounted programming. Here for each state x we maintain a value function estimate

$$V_s = (V_s(x) : x \in \mathcal{X})$$

Here s counts the number of iterations our algorithm has performed so far.

We update states asynchronously. Specifically we let \mathcal{T}_x be the set of times where state x is updated. We assume that $|\mathcal{T}_x| = \infty$ and, these sets are disjoint, i.e. $\mathcal{T}_x \cap \mathcal{T}_{x'} = \emptyset$. For times $s \in \mathcal{T}_x$, the set of value function estimates that we have might not be the most recent values. We let $\tau_{x'}(s)$ be the last iteration before iteration s that state x received the current value estimate of state x' .

Thus we update x based on estimates

$$V_s^x = (V_{\tau_{x'}(s)}(x') : x' \in \mathcal{X})$$

Asynchronous value iteration. Asynchronous value iteration performs the following updates: at iteration s if $s \in \mathcal{T}_x$ then we set

$$V_s(x) = \max_{a \in \mathcal{A}} \{r(x, a) + \beta \mathbb{E} [V_s^x(\hat{x})]\}$$

Convergence Result. Here we see that under pretty mild conditions that asynchronous value iteration converges.

Theorem 3. *If $\tau_{x'}(s) \rightarrow \infty$ as $s \rightarrow \infty$ then*

$$V_s \rightarrow V$$

where V is the optimal value function.

The following shows that we maintain the contraction property when we update the value function at state x according to this estimate.

Lemma 6. *The operation for $V = V_s^x$ the operation*

$$TV(x) = \max_{a \in \mathcal{A}} \{r(x, a) + \mathbb{E}_{x,a}[V(\hat{x})]\}$$

is a contraction in the sense that

$$|TV(x) - TV'(x)| \leq \beta \|V - V'\|_\infty.$$

The proof here is given in Lemma 4. (So we omit the proof and refer the reader there.)

Remark 54. *As powerful and general as the statement of Theorem 3 is, once we have Lemma 6 in place the rest of the proof is a book-keeping exercise. Basically Lemma 6 shows that if we make progress updating any state x then that progress is not going to be lost.*

Proof. We prove the result by induction. On the induction hypothesis that there exists a time σ_k such that for all $s \geq \sigma_k$

$$\|V_s^x - V\| \leq \beta^k \|V_0 - V\|, \quad \forall x \in \mathcal{X}.$$

So each x has a close estimate to the true value function. (Notice that Lemma 6 implies that $|V_s(x) - V(x)|$ can only decrease. Thus if the above holds then it also holds that $\|V_s - V\| \leq \beta^k \|V_0 - V\|$. So the most upto date value function is also close estimate of the true value function.)

Let x be the next time that we update state x , i.e. $s \in \mathcal{T}_x$ s.t. $s \geq \sigma_k$. Then, by Lemma 6 and the induction hypothesis, it holds that

$$|V_s(x) - V(x)| \leq \beta \|V_s^x - V\|_\infty \leq \beta^{k+1} \|V_0 - V\|_\infty.$$

So if we let t_{k+1} be the next time after time σ_k that all states have been updated in this way, then for all $s \geq t_{k+1}$

$$\|V_s - V\|_\infty \leq \beta^{k+1} \|V_0 - V\|_\infty. \quad (17)$$

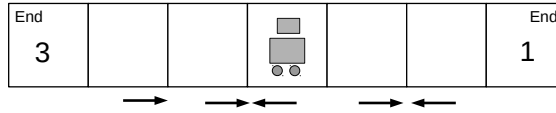
Further, we let σ_{k+1} be the first time after t_k that each state x' has been updated at each x , i.e. the first time s such that $\tau_{x',x}(s) \geq t_k$ holds for all x' and x . Then, since (17) holds, it also holds that for all $s \geq \sigma_{k+1}$

$$\|V_s^x - V\|_\infty \leq \beta^{k+1} \|V_0 - V\|_\infty,$$

as required. □

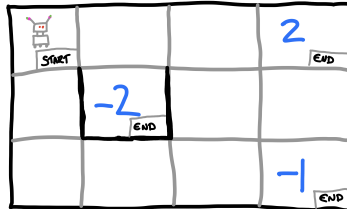
MDP Algorithms – Examples

Ex 55. Apply the policy iteration algorithm to the following problem:



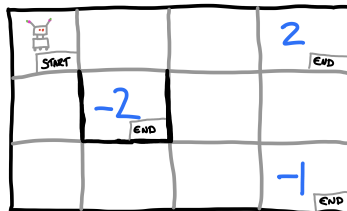
Here a robot must navigate to either end point. It receives a reward of 3 for reaching the lefthand side and 1 for the righthand side. An initial policy from which you must start is provided by the arrows above.

Ex 56 (GridWorld). A robot is placed on the following grid.



The robot can chose the action to move left, right, up or down provided it does not hit a wall, in this case it stays in the same position. (Walls are colored black.) With probability 0.8, the robot does not follow its chosen action and instead makes a random action. The rewards for the different end states are colored above. Write a program that uses, Value Iteration to find the optimal policy for the robot.

Ex 57 (GridWorld, again). Write a program that uses, Policy iteration to find the optimal policy for the robot in [56].



Ex 58. Show that for discounted programming,

$$V_s(x) + \frac{\beta^{s+1} r_{\max}}{1 - \beta} \geq V(x) \geq V_s(x) - \frac{\beta^{s+1} r_{\min}}{1 - \beta}$$

Ex 59. Here we consider a positive programming problem

a) Let $\mathcal{V}R$ give the value function reached after one iteration of the value iteration algorithm. Argue that if $\mathcal{V}R = R$ then $R(x)$ is optimal.

b) Let $\mathcal{I}\pi$ give the policy reached after one iteration of the policy improvement algorithm. Argue that if $\mathcal{I}\pi = \pi$ then π is optimal.

References.

Value iteration is due to Richard Bellman, and Policy Iteration is due to Howard [24]. Both are now standard text book methods [6, 37].

0.6 Optimal Stopping

- Optimal Stopping Problems; One-Step-Look-Ahead Rule.
 - The Secretary Problem.
 - Infinite Time Stopping; Stopping Random Walks.
-

An Optimal Stopping Problem is an Markov Decision Process where there are two actions: $a = 0$ meaning to stop, and $a = 1$ meaning to continue. Here there are two types of costs

$$c(x, a) = \begin{cases} \kappa(x), & \text{for } a = 0 \quad (\text{the stopping cost}) \\ c(x), & \text{for } a = 1 \quad (\text{the continuation cost}), \end{cases}$$

A MDP with these costs is called an *optimal stopping problem*.

Assuming that time is finite, the Bellman equation is

$$C_s(x) = \min \{k(x), c(x) + \mathbb{E}_x[C_{s-1}(\hat{x})]\}$$

for $s \in \mathbb{N}$ and $C_0(x) = k(x)$.

One Step Look Ahead

The One Step Look Ahead policy is the policy where we stop if stopping now has lower cost than continuing one step further and then stopping. This simple policy can be optimal for a wide range of optimal stopping policies. In particular, this is the case when the set of states for which we stop is closed.

Def 60 (OLSA rule). *In the One Step Look Ahead (OLSA) rule we stop whenever the state belongs to the set*

$$S = \{x : k(x) \leq c(x) + \mathbb{E}_x[k(\hat{x})]\}.$$

We call S the stopping set. In words, you stop whenever it is better stop now rather than continue one step further and then stop.

Def 61 (Closed Stopping Set). We say the set $S \subset \mathcal{X}$ is closed, if once inside that said you cannot leave, i.e.

$$P_{xy} = 0, \quad \forall x \in S, y \notin S.$$

Prop 62. For the finite time stopping problem, if the stopping set given by the one step lookahead rule is closed then the one step lookahead rule is an optimal policy.

Proof. Given the set S is closed, we argue that if $C_{s-1}(x) = k(x)$ for $x \in S$ then $C_s(x) = k(x)$: If $x \in S$ then since S is closed $\hat{X} \in S$. In other words $C_{s-1}(\hat{X}) = k(\hat{X})$. Therefore, in this case, Bellman's equation becomes

$$C_s(x) = \min\{k(x), c(x) + \mathbb{E}_x[C_{s-1}(\hat{X})]\} = \min\{k(x), c(x) + \mathbb{E}_x[k(\hat{X})]\} = k(x).$$

The last inequality above follows by the definition of $x \in S$.

We now proceed by induction. The OSLA rule is optimal for $s = 1$ steps, since OSLA is exactly the optimal policy for one step.

Suppose that the result holds for upto $s-1$ steps. Now consider the Optimal Stopping Problem with s steps. If $x \in S$ then $C_s(x) = k(x)$. So it is better to stop. If $x \notin S$, then clearly it's better to continue. \square

OSLA in Infinite Time.* We now give conditions for the one step look ahead rule to be optimal for infinite time stopping problems.

Prop 63. If the following two conditions hold

- $K = \max_x k(x) < \infty, \min_x k(x) \geq 0,$
- $C = \min_x c(x) > 0,$

then the One-Step-Lookahead-Rule is optimal.

Proof. Suppose that a policy π stops at time τ and is better than or equal to stopping at time 0 then

$$(s+1)C\mathbb{P}(\tau > s) \leq \mathbb{E} \left[\left\{ \sum_{t=0}^{\tau-1} c(x_t) + k(x_\tau) \right\} \mathbb{I}[\tau > s] \right] \leq k(x_0) \leq K.$$

Therefore if we follow optimal policy¹² π but for the s time horizon problem and stop at s if $\tau \geq s$ then

$$L(x) \leq L_s(x) \leq L(x) + K\mathbb{P}(\tau > s) \leq L(x) + \frac{K^2}{C(s+1)} \xrightarrow{s \rightarrow \infty} L(x)$$

¹²If an optimal policy doesn't exist, you make take a policy that is within ϵ of the optimal cost and then let ϵ go to zero at the end of the proof.

The first inequality holds since $L_s(x)$ minimizes over s time steps but $L(x)$ minimizes over all time steps. The second inequality holds since stopping at time τ on the event $\{\tau < s\}$ has a cost less than $L(x)$ and stopping at time s on the event $\{\tau \geq s\}$ has a cost of at most K . Thus the loss from this policy is less than the optimal rule over s time steps, namely $L_s(x)$.

Thus $L_s(x) \rightarrow L(x)$.

As before (for the finite time problem), it is not optimal to stop if $x \notin S$ and for the finite time problem $L_s(x) = k(x)$ for all $x \in S$. Therefore, since $L_s(x) \rightarrow L(x)$, we have that $L(x) = k(x)$ for all $x \in S$ and there for it is optimal to stop for $x \in S$. \square

Stopping a Random Walk

The one step lookahead rule is not always the correct solution to an optimal stopping problem. Another approach is to use an excessive majorant. We consider stopping a random walk because it direct interpretation in terms of concavity, but the reader can check that the same idea works for other Markov chains.

Def 64 (Concave Excessive Majorant). *For a function $r : \{0, \dots, N\} \rightarrow \mathbb{R}_+$ a concave majorant is a function G such that*

- $G(x) \geq \frac{1}{2}G(x-1) + \frac{1}{2}G(x+1)$
- $G(x) \geq r(x)$.

Prop 65 (Stopping a Random Walk). *Let X_t be a symmetric random walk on $\{0, \dots, N\}$ where the process is always stopped at 0 and N . For each $x \in \{0, \dots, N\}$, there is a positive reward of $r(x)$ for stopping. We are asked to maximize*

$$\mathbb{E}[r(X_T)]$$

where T is our chosen stopping time. The optimal value function $V(x)$ is the minimal concave majorant. Thus it is optimal to stop whenever $V(x) = r(x)$.

Proof. The Bellman equation is

$$V(x) = \max \left\{ r(x), \frac{1}{2}V(x-1) + \frac{1}{2}V(x+1) \right\}$$

with $V(x) = r(0)$ and $V(N) = r(N)$. Thus the optimal value function is a concave majorant.

We will show that the optimal policy is the minimal concave majorant of $r(x)$. We do so by, essentially applying induction use value iteration. First $R_0(x) = 0 \leq G(x)$ for any concave majorant of $r(x)$. Now suppose that R_{s-1} , the function reached after $s - 1$ value iterations, satisfies $R_{s-1}(x) \leq G(x)$ for all x , then, for time s ,

$$\begin{aligned} R_s(x) &= \max \left\{ r(x), \frac{1}{2}R_{s-1}(x-1) + \frac{1}{2}R_{s-1}(x+1) \right\} \\ &\leq \max \left\{ r(x), \frac{1}{2}G(x-1) + \frac{1}{2}G(x+1) \right\} \\ &\leq \max \{r(x), G(x)\} = G(x). \end{aligned}$$

Since value iteration converges $R_s(x) \nearrow V(x)$, where $V(x)$ satisfies $V(x) \leq G(x)$. So, as required, $V(x)$ is the minimal concave majorant.

Finally observe that from the Bellman equation the optimal stopping rule is to stop whenever $V(x) = r(x)$ for the minimal concave majorant. \square

Remark 66. Note if we have a more general Markov chain then we can replace the convexity condition with the statement

$$G(x) \geq \mathbb{E}_x[G(\hat{x})].$$

Exercises

Ex 67 (The Secretary Problem). *There are N candidates for a secretary job. You interview candidates sequentially. After each interview, you must either accept or reject the candidate. We assume each candidate has the rank: $1, 2, \dots, N$ And arrive for interview uniformly at random. Find the policy that maximises the probability that you hire the best candidate.*

Ex 68 (Optimal Parking). *You look for a parking space on street, each space is free with probability $p = 1 - q$. You can't tell if space is free until you reach it. Once at space you must decide to stop or continue. From position s (s spaces from your destination), the cost of stopping is s . The cost of passing your destination without parking is D .*

Ex 69. In a game show a contestant is asked a series of 10 questions. For each question $q = 1, \dots, 10$ there is a reward r_q for answering the question correctly. With probability p_q the contestant answers the question correctly. After correctly answering a question, the contestant can choose to stop and take their total winnings home or they can continue to the next question $q + 1$. However, if the contestant answers a question incorrectly then the contestant loses all of their winnings. The probability of winning each round is decreasing and is such that the expected reward from each round, $p_q r_q$, is constant.

i) Write down the Bellman equation for this problem.

ii) Using the One-Step-Look-Ahead rule, or otherwise, find the optimal policy of the contestant.

Ex 70 (Burglar). A burglar robs houses over N nights. At any night the burglar may choose to retire and thus take home his total earnings. On the t th night house he robs has a reward r_t where r_t is an iidrv with mean \bar{r} . Each night the probability that he is caught is p and if caught he loses all his money. Find the optimal policy for the burglar's retirement.

Ex 71 (Bruss' Odds Algorithm). You sequentially treat patients $t = 1, \dots, T$ with a new trail treatment. The probability of success is $p_t = 1 - q_t$. We must minimize the number of unsuccessful treatments while treating all patients for which the trail is will be successful. (i.e. if we label 1 for success and 0 for failure, we want to stop on the last 1). Argue, using the One-Step-Look-Ahead rule that the optimal policy is the stop treating at t^* the largest integer such that

$$\frac{p_{t^*}}{q_{t^*}} + \dots + \frac{p_T}{q_T} \geq 1.$$

This procedure is called Bruss' Odds Algorithm.

Ex 72. You own an asset that must be sold in T days. Each day you are offered a price for the asset according to a probability distribution density $f(x)$. You may the accept any offer that you have received so far. Once the asset is sold the money is invested in a bank account which multiplies the invested money by β^{-1} each day. Here $\beta \in (0, 1)$. Your task is the maximize your profit at time T .

Ex 73. You own a “toxic” asset its value, x_t at time t , belongs to $\{1, 2, 3, \dots\}$. The daily cost of holding the asset is x_t . Every day the value moves up to $x + 1$ with probability $1/2$ or otherwise remains the same at x . Further the cost of terminating the asset after holding it for t days is $C(1 - \alpha)^t$. Find the optimal policy for terminating the asset.

References.

An early account of optimal stopping is Chow, Robbins and Siegmund [15]. An authoritative texts mostly focusing on stopping diffusions is Shiryaev [43]. Ferguson provides a good step of unpublished notes on his website.¹³ Again most standard texts on stochastic control cover optimal stopping, see for example Whittle [56].

¹³<https://www.math.ucla.edu/~tom/Stopping/Contents.html>

0.7 Inventory Control.

- S-Inventory Control
- (s, S) -Inventory Control; K -Convexity.

We consider the problem where there is an amount of stock x_t at time t . You can perform the action to order a_t units of stock. Further the demand at time t is d_t . We assume d_t is independent over t . The change in the amount of stock follows the dynamic:

$$x_{t+1} = x_t + a_t - d_t.$$

It is possible for x_t to be negative, which corresponds to backordered stock.

There is a cost for ordering $a \in \mathbb{R}$ units of stock:

$$c(a) = K + ca.$$

Note here $c(a) = 0$ if $a = 0$. There is a cost $\kappa(x)$ for holding stock when $x > 0$ or for unmet demand when $x < 0$. Specifically

$$\kappa(x) = \begin{cases} px & \text{if } x \geq 0, \\ hx & \text{if } x < 0. \end{cases}$$

for positive constants h and p . If we consider the minimization problem over T time steps then we have a dynamic program:

$$L_T(x) = \min_{a_0, \dots, a_T} \mathbb{E} \left[\sum_{t=0}^T c(a_t) + \kappa(x_t + a_t - d_t) \right].$$

The Bellman equation for this optimization problem is

$$\begin{aligned} L_t(x) &= \min_{a \geq 0} \{c(a) + \mathbb{E}_d[\kappa(x + a - d)] + \mathbb{E}[L_{t-1}(x + a - d)]\} \\ &= \min_{y \geq x} \{c(y) + \mathbb{E}_d[\kappa(y - d)] + \mathbb{E}[L_{t-1}(y - d)]\} - cx. \end{aligned}$$

Here $y = x + a$. If we define $B(x)$ as follows

$$B_t(y) = cy + \mathbb{E}_d[\kappa(y - d)] + \mathbb{E}[L_{t-1}(y - d)],$$

then (removing the subscript t for now) notice the above Bellman Equation becomes

$$L(x) = \min \left\{ B(x), \min_{y>x} \{K + B(x)\} \right\} - cx. \quad (18)$$

There exists an explicit closed form solution to this optimization. Specifically there exists values s and S such that if the stock level goes below s then we order enough to have S units of stock. (We will show that S minimizes $B(x)$ and s is such that $B(s) = K + B(S)$.)

We give a proof for the case where $K = 0$. This gives the necessary intuition for the more general case where $K > 0$, which we subsequently discuss. (The formal argument is given in a series of exercises: Ex 75–Ex 78.)

S-Inventory control.

We consider the equation (18) with $K = 0$. Notice if $B(x)$ is convex and $B(x) \rightarrow \infty$ as $|x| \rightarrow \infty$ then S , the minimum of B , is finite. Thus from (18) it holds that

$$L(x) + cx = \min_{y \geq x} B(y) = \begin{cases} B(S) & \text{if } x \leq S, \\ B(x) & \text{if } x > S. \end{cases}$$

Notice if $B(x)$ is convex then relatively straight forward to check that $L(x)$ is convex and $L(x) \rightarrow \infty$ and $|x| \rightarrow \infty$ (see Ex 75). Also see Figure 3.

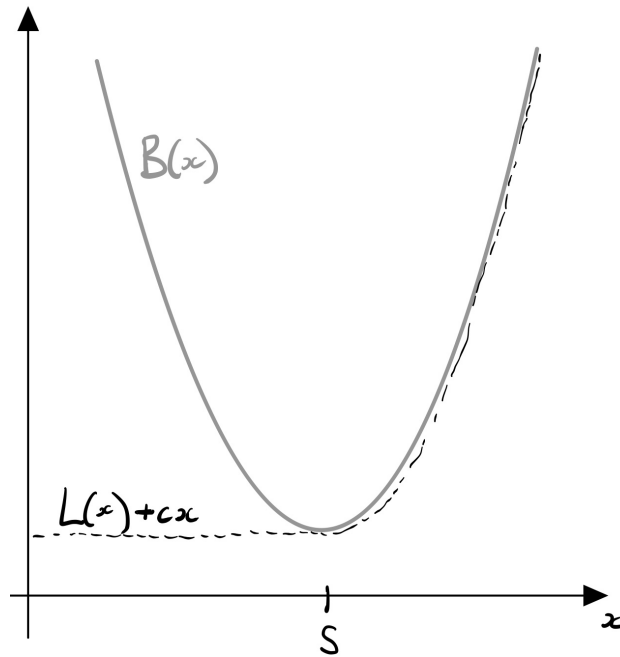
Further, if, given the function $L(x)$ above, we now update $B(x)$ by taking

$$B(y) = cy + \mathbb{E}_d[\kappa(y - d)] + \mathbb{E}_d[L(y - d)],$$

then it is straight-forward to check that the new function B is also convex and $B(x) \rightarrow \infty$ as $|x| \rightarrow \infty$ (see Ex 75).

Thus (putting sub-script t back and noting that $L_0(x)$ is convex) we see that the sequence of functions $L_t(x)$ and $B_t(x)$ are convex and that S_t , the minimum of $B_t(x)$, defines the optimal control

- If $x < S_t$ then order so you have S_t units of stock, $x + a = S_t$.
- If $x \geq S_t$ then do nothing.

Figure 3: $B(x)$ and $L(x)$ for $K = 0$. **(s, S) -Inventory control.**

We consider the equation (18) with $K > 0$. That is

$$L(x) + cx = \min \left\{ B(x), \min_{y>x} \{K + B(x)\} \right\}. \quad (19)$$

In general $B(x)$ will not be convex, but let's assume it is for a moment in order to gain intuition. Again let S be the minimum of $B(x)$. In this case the minimization over $y > x$ above is solved at $y = S$ if $x \leq S$ and $y = x$ if $x > S$. (Note if $y = x$ in (19) it is obviously better to take $B(x)$ rather than $B(x) + K$.) So we have that

$$L(x) + cx = \min \{B(x), K + B(S)\}. \quad (20)$$

Now assuming $B(x)$ is continuous then there will be a value of $x = s < S$ such that

$$B(s) = K + B(S). \quad (21)$$

This will be the point for which for $x < s$ then $K + B(S)$ is the minimum in (19), while for $x \geq s$, $B(x)$ will be the minimum in (19). I.e. In this

case the optimal control will be

$$x + a = \begin{cases} S & \text{if } x < s, \\ x & \text{if } x \geq s. \end{cases} \quad (22)$$

For a convex function $B(x)$ we plot $L(x) + cx$ in Figure 4. Notice that $L(x)$ is clearly not convex. Essentially, the additional K term has introduced a "bump" of size K . So rather than assume convexity motivates, this motivates the idea of K -convexity.

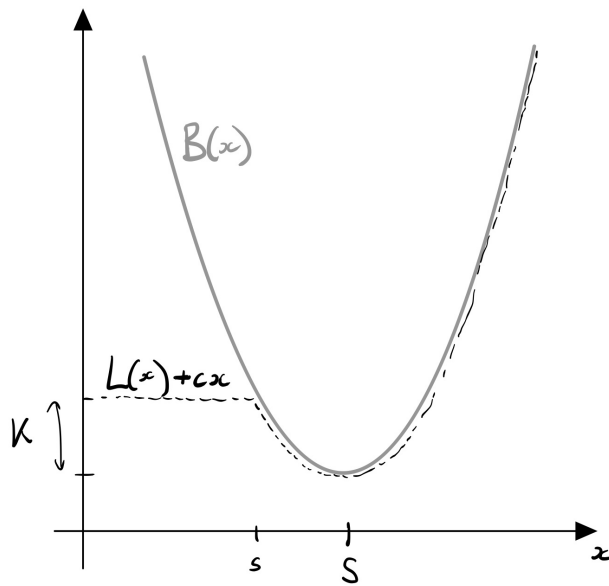


Figure 4: $B(x)$ and $L(x) + cx$ for $K > 0$.

Definition 1 (K -convex). A function B is K -convex if

$$K + B(z + y) \geq B(z) + z \left[\frac{B(y) - B(y - b)}{b} \right]$$

Remark 74. The definition above is not terribly intuitive. It can be shown that K -convexity means that for $x < z$ the line segment from $(z, B(z) + K)$ to $(x, B(x))$ does not intersect $(y, B(y))$ for any value of y between x and z . I.e. if you are standing at point $(z, B(z) + K)$ you have an unobstructed view of $(x, B(x))$.

If $B(x)$ is K -convex then we can show that (19) is solved by (22), just like in the convex case. (This is Ex 76.) Further, we can show

that if $B(x)$ is K -convex then $L(x)$ defined by (20) is K -convex. (This is Ex 78). Thus it can be shown that when we update $B(x)$ according to the rule

$$B(y) = cy + \mathbb{E}_d[\kappa(y - d)] + \mathbb{E}[L(y - d)]$$

then B is also K -convex. (This is Ex 77.)

Exercises

Ex 75. a) Given $B(x)$ is convex and $B(x) \rightarrow \infty$ for $|x| \rightarrow \infty$, show that

$$L(x) + cx = \min_{y \geq x} B(y) = \begin{cases} B(S) & \text{if } x \leq S, \\ B(x) & \text{if } x > S. \end{cases}$$

is such that $L(x)$ is convex and $L(x) \rightarrow \infty$ for $|x| \rightarrow \infty$.

b) Given $L(x)$ is convex and $L(x) \rightarrow \infty$ for $|x| \rightarrow \infty$, show that

$$B(y) = cy + \mathbb{E}_d[\kappa(y - d)] + \mathbb{E}_d[L(y - d)],$$

is such that $B(x)$ is convex and $B(x) \rightarrow \infty$ for $|x| \rightarrow \infty$.

Ex 76. We let $B(x)$ be K -convex, continuous with $B(x) \rightarrow \infty$ as $|x| \rightarrow \infty$. Also let $S = \min B(x)$ and s is the smallest number such that $B(s) = B(S) + K$.

a) Show that, for $x < s$, $B(x) > B(S) + K$.

(Hint: show $B(x) > B(s)$.)

b) Show that for $s < x < y$, $B(x) \leq B(y) + K$.

(Hint: consider two cases $x < S$ and $x > S$.)

c) Show that (19):

$$L(x) + cx = \min \left\{ B(x), \min_{y > x} \{K + B(y)\} \right\}$$

is solved by (22):

$$x + a = \begin{cases} S & \text{if } x < s, \\ x & \text{if } x \geq s. \end{cases}$$

The following exercise establishes some properties of K -convex functions

Ex 77. Show that

- a) The sum of K -convex functions is K -convex.
- b) A convex function is K -convex.
- c) If $B(x)$ is K -convex then so is $B(x + a)$ is K -convex.
- d) The convex combination of K -convex functions is K -convex.
- e) If B is K -convex then $\mathbb{E}_Z[B(x + Z)]$ is K -convex.

Ex 78. If we let

$$L(x) = \begin{cases} K + B(S) - cx & \text{if } x < s, \\ B(x) - cx & \text{if } x \geq s. \end{cases}$$

where S minimizes $B(x)$ and s is the smallest value such that $B(s) = K + B(S)$. We will show that L is K -convex:

$$K + L(z + y) \geq L(z) + z \left[\frac{L(y) - L(y - b)}{b} \right]. \quad (23)$$

Verifying (23) by moving s through the following cases:

- a) $s \leq y - b$
- b) $y - b < s < y$ and $B(y) \geq B(s)$
- c) $y - b < s < y$ and $B(y) < B(s)$
- d) $y < s < y + z$
- e) $y + z < s$

References.

The idea of K -convexity is due to Scarf [42]. A full account of all the calculations above is given in the excellent text of Bertsekas [6].

0.8 Partially Observable MDPs

- Partially Observable MDP; Belief States.
- Reduction to MDP.

It is common that we do not observe the state of a Markov decision process. For instance, we want to control the path of a helicopter. We never know the exact state instead we have to take noisy measurements from sensors and then use these. A Partially Observable is the extension of the MDP framework accounts for this feature.

The state of a POMDP evolves a Markov decision process but we only receive an observation of the current state rather than the actual state of the chain. See Figure 5.

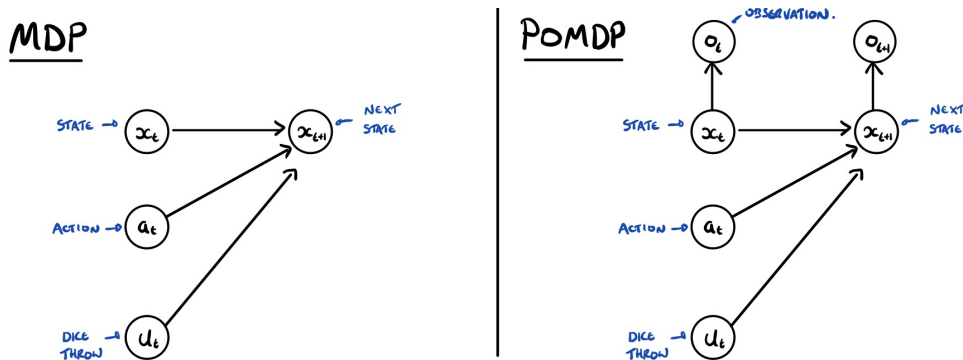


Figure 5:

Definition of POMDP.

States $x \in \mathcal{X}$, actions $a \in \mathcal{A}$, and rewards $r(x, a)$ are the same as we defined for an MDP, see Section 0.3. The state evolves as before, specifically

$$X_{t+1} = f(X_t, A_t; U_t)$$

where X_t is the state at time t and action A_t is the action at time t and where U_t is an independent random variable. As a function of the current the state, we receive an observation

$$\mathcal{O}_t = g(X_t, U_t)$$

where X_t is the state at time t and where U_t is an independent random variable.¹⁴ We let Ω be the set of observations. A policy must chose an action as a function of the observations, i.e. we select

$$\pi_t(\mathcal{O}_t, \dots, \mathcal{O}_0) \in \mathcal{A} \quad \forall t \in \mathbb{Z}_+. \quad (24)$$

This defines a Partially Observable Markov Decision Process (POMDP).¹⁵

Although the sequence (X_t, \mathcal{O}_t) , $t \in \mathbb{Z}_+$, forms a Markov chain, the sequence of observations \mathcal{O}_t , $t \in \mathbb{Z}_+$, is no longer Markov. The loss of the Markov property means that a policy cannot simply select an action, π_t , as a function of the current observation, which we could do for an MDP. So we let \mathcal{P} be the set of policies, i.e. sequences of functions of the form (24).

The objective of a POMDP is the same as for a MDP but, as just discussed, the set of policies is different:

$$\text{maximize} \quad \mathbb{E} \left[\sum_{t=0}^{\infty} \beta^t r(X_t, \pi_t) \right] \quad \text{over} \quad \Pi \in \mathcal{P}. \quad (25)$$

Note that we could also take the objective to depend on the observation \mathcal{O}_t instead of X_t .

Reduction to an Equivalent MDP.

It is, in fact, possible to reduce a POMDP to an MDP. This is perhaps not too surprising as in informal terms: if we condition on enough state information then what is left is independence, and thus the for a suitably large state the evolution is Markov. However, for POMDP this has a particularly nice formulation in terms of Bayes rule.

Def 79 (Belief state). *We define a belief state, $b = (b(x) : x \in \mathcal{X})$, to be a probability distribution over the set of states \mathcal{X} . We let \mathcal{B} the set of belief states.*

¹⁴We could generalize slightly to let \mathcal{O}_t also be a function of the previous action A_{t-1} . However, for notation simplicity, we do not assume this here.

¹⁵This is pronounced *pomm-dee-pee*.

A belief state summarizes all the information we have about where we think the state is going to be at the next time. So, at time t , we think of the belief state as being

$$b_t(x) = \mathbb{P}(X_{t+1} = x | A_t = a_t, \mathcal{O}_t = \mathcal{o}_t, \dots, A_0 = a_0, \mathcal{O}_0 = \mathcal{o}_0)$$

where a_0, \dots, a_t are the actions and $\mathcal{o}_0, \dots, \mathcal{o}_t$ are the observations taken up until time t .

We will apply some shorthand notation

$$P(\mathcal{o}|x) = \mathbb{P}(\mathcal{O}_0 = \mathcal{o} | X_0 = x) \quad \text{and} \quad P(\hat{x}|x, a) = \mathbb{P}(X_1 = \hat{x} | X_0 = x, A_0 = a).$$

Theorem 4. *The POMDP (25) is equivalent to an MDP over belief states $b \in \mathcal{B}$ with instantaneous reward function*

$$\rho(b, a) = \sum_{x \in \mathcal{X}} b(x) r(x, a)$$

and the plant equation, which gives the next belief state from observation $\hat{\mathcal{o}}$, action a and current belief state b , is

$$\hat{b}(\hat{x}) = \frac{P(\hat{\mathcal{o}}|\hat{x}) \sum_{x \in \mathcal{X}} P(\hat{x}|x, a) b(x)}{P(\hat{\mathcal{o}}|a, b)}. \quad (26)$$

and the transition probabilities for \hat{b} given belief state b and action a are given by

$$P(\hat{b}|b, a) := \sum_{\hat{\mathcal{o}} \in \Omega} P(\hat{b}|b, a, \hat{\mathcal{o}}) P(\hat{\mathcal{o}}|b, a) \quad (27)$$

above $P(\hat{\mathcal{o}}|a, b)$ after taking action $a_t = a$ can be interpreted as a normalizing constant for $\hat{b}(x)$

$$P(\hat{\mathcal{o}}|b, a) = \sum_{\hat{x} \in \mathcal{X}} P(\hat{\mathcal{o}}|\hat{x}) \sum_{x \in \mathcal{X}} P(\hat{x}|x, a) b(x).$$

and

$$P(\hat{b}|b, a, \mathcal{o}) = \begin{cases} 1 & \text{if } \hat{b} \text{ is given by formula (26),} \\ 0 & \text{otherwise.} \end{cases} \quad (28)$$

Remark 80. *The subtle point here is that the belief state b summarizes all past information required to determine the distribution of the next state \hat{x} . I.e. so long as we know $b_t = b$ and $a_t = a$, then the past actions and observations do not effect our belief about the state $x_{t+1} = \hat{x}$. Expressed one way we have recovered the Markov property by considering a suitably large state vector b . Expressed another way the belief vector is a sufficient statistics, in that its value summarizes all information above the past actions and observations.*

Proof. We let $\boldsymbol{o}_t = (o_s : s \leq t)$, $\boldsymbol{a}_t = (a_s : s \leq t)$ and $a_t = a$. We proceed inductively. We take $b_0(x) = \mathbb{P}(X_0 = x | \mathcal{O}_0 = \boldsymbol{o}_0)$. Given $b_t(x) = b(x) = P(x | \boldsymbol{o}_t, \boldsymbol{a}_{t-1})$, the next belief state $b_{t+1} = \hat{b}$ can be calculated as follows:

$$\begin{aligned} \hat{b}(\hat{x}) &= P(\hat{x} | \hat{\phi}, \boldsymbol{o}_t, \boldsymbol{a}_t) \\ &= \underbrace{P(\hat{\phi} | \hat{x}, \boldsymbol{o}_t, \boldsymbol{a}_t)}_{P(\hat{\phi} | \hat{x})} \cdot P(\hat{x} | \boldsymbol{o}_t, \boldsymbol{a}_t) \cdot \frac{1}{P(\hat{\phi} | \boldsymbol{o}_t, \boldsymbol{a}_t)} \\ &= P(\hat{\phi} | \hat{x}) \cdot \frac{1}{P(\hat{\phi} | \boldsymbol{o}_t, \boldsymbol{a}_t)} \cdot \sum_x \underbrace{P(\hat{x} | x, \boldsymbol{o}_t, \boldsymbol{a}_t)}_{P(\hat{x} | x, a)} \underbrace{P(x | \boldsymbol{o}_t, \boldsymbol{a}_t)}_{b(x)} \\ &= P(\hat{\phi} | \hat{x}) \frac{\sum_x P(\hat{x} | x, a) b(x)}{P(\hat{\phi} | \boldsymbol{o}_t, \boldsymbol{a}_t)} \end{aligned}$$

The first equality above is Bayes rule. In the second equality we note that $\hat{\phi}$ is only dependent of \hat{x} . The third equality again applies conditional independence. The final expression gives the required formula in 26.

By a similar calculation we can see that the denominator can be calculated

$$P(\hat{\phi} | \boldsymbol{o}_t, \boldsymbol{a}_t) = \sum_{\hat{x}} P(\hat{\phi} | \hat{x}) \sum_x P(\hat{x} | x, a) b(x) =: P(\hat{\phi} | a, b) \quad (29)$$

Thus we see that the required expression (26) holds

$$\hat{b}(\hat{x}) = \frac{P(\hat{\phi} | \hat{x}) \sum_{x \in \mathcal{X}} P(\hat{x} | x, a) b(x)}{P(\hat{\phi} | a, b)}.$$

Thus if we are given a , b and $\hat{\phi}$ we can calculate the next belief state.

Given a , b and $\hat{\phi}$, the value of \hat{b} is deterministic. We have also calculated the probability of $\hat{\phi}$ from a and b above (29). So given b , a we can thus determine probability of \hat{b} :

$$P(\hat{b} | b, a) = \sum_{\hat{\phi} \in \Omega} P(\hat{b} | b, a, \hat{\phi}) P(\hat{\phi} | b, a).$$

where $P(\hat{b} | b, a, \phi)$ is the indicator function defined in 28. Note above b summarizes all past actions and states. So the transitions on b are now Markov. And the process described is a Markov decision process.

Finally we note that

$$\begin{aligned}
& \mathbb{E} \left[\sum_{t=0}^{\infty} \beta^t r(X_t, a_t) \right] \\
&= \mathbb{E} \left[\sum_{t=0}^{\infty} \beta^t \mathbb{E}[r(X_t, a_t) | \mathcal{O}_t = \mathcal{o}_t, \dots, \mathcal{O}_0 = \mathcal{o}_0, A_{t-1} = a_{t-1}, \dots, A_0 = a_0] \right] \\
&= \mathbb{E} \left[\sum_{t=0}^{\infty} \beta^t \sum_x b_t(x) r(x, a_t) \right].
\end{aligned}$$

From this we can see (similar to discussions on MDPs) that an optimal policy only requires information summarized by the current state b_t in order to choose an action at time t . \square

In general the reduction to belief states is conceptually useful. The set of belief states is continuous and so we may leave the world of finite state Markov decision processes to solve problems of this type. The use of Bayes rule above suggests a close link with Bayesian statistics and statistics in general. From a practical standpoint, although we can view a POMDP as an MDP it is often worth developing specialized algorithms for the POMDP setting.

References.

The discussion on belief states is primarily based on the paper of Kaelbling et al. [26]. However, the idea of belief states goes back much earlier to Astrom [1] and Striebel [46]. A general discussion and further reference can be found in [41]. From the above it is clear that Bayesian interpretation of the update and this is regularly used in the study of Hidden Markov Models see Cappe et al. [14] for a text book reference. The update above simplifies significantly for linear-quadratic optimizations. This is covered in our discussion on the Kalman Filter and LQG and references therein.

0.9 LQR and the Kalman Filter

-
- Linear-Quadratic Regularization (LQR); Riccati Equation.
 - Linear Quadratic Gaussian.
 - Certainty Equivalence; Kalman Filter.
-

Linear Quadratic Regularization (LQR) is a special case of dynamic programming where we have a quadratic objective and a linear dynamic. [Note many smooth dynamics are linear over small time steps and smooth objectives are quadratic close to their minimum.] LQR has a solution with a relatively simple form given by the Riccati equation. It can be generalized in a few different ways: random noise and incomplete state information. Even in these settings the optimal control remain essentially the same. However, with incomplete state information we may need to replace the state variable, x with its mean, \bar{x} . This is called certainty equivalent control. If noise is Gaussian, then estimating x is a relatively straight forward recursion which is given by the Kalman filter. We define and discuss each of these steps in subsequent sections below.

Linear Quadratic Regularization.

The objective below is quadratic and its constraints are linear. For this reason, this problem is called a Linear Quadratic Regulator problem and its solution is a Linear Quadratic Regular (LQR).

Def 81 (Linear Quadratic Regularization). *We consider the following optimization:*

$$\begin{aligned}
 L_0(x_0) = \text{minimize} \quad & \sum_{t=0}^{T-1} \{x_t^\top R x_t + a_t^\top Q a_t\} + x_T^\top R x_T & \text{(LQR)} \\
 \text{subject to} \quad & x_t = A x_{t-1} + B a_{t-1}, \quad t = 1, \dots, T \\
 \text{over} \quad & a_0, \dots, a_{T-1} \in \mathbb{R}^m
 \end{aligned}$$

Here the actions a belong to \mathbb{R}^m and the states x belong to \mathbb{R}^n . Here A and B are matrices and R and Q are positive semi-definite matrices.¹⁶

¹⁶Recall, a matrix M is positive semi-definite if $x^\top M x > 0$ for $x \neq 0$.

Why LQR? This optimization is very common in control. This is because many dynamical systems are approximately linear [over small time steps] and many [smooth] objectives are approximately quadratic when close to their minima. So a wide variety of control problems are approximately LQR problems.

Riccati Equation. An important recursion that is needed to solve LQR problems is the Riccati Equation:

Def 82 (Riccati Equation and Gain Matrix). *The Riccati equation is the following matrix recursion*

$$\Lambda_t = R + A^\top \Lambda_{t+1} A - A^\top \Lambda_{t+1} B [Q - B^\top \Lambda_{t+1} B]^{-1} B^\top \Lambda_{t+1} A \quad (\text{Riccati})$$

for $t = 0, \dots, T - 1$ and with $\Lambda_T = R$. The gain matrix is defined to be

$$G_t = [Q + B^\top \Lambda_t B]^{-1} B^\top \Lambda_t A \quad (\text{Gain})$$

From this $\mathbf{a} = -G_t \mathbf{x}$ gives the optimal control at time t .

Solution for LQR. We let $V_\tau(\mathbf{x})$ be the optimal solution to (LQR), where the summation is started from time $t = \tau$ in state $\mathbf{x}_\tau = \mathbf{x}$. The following result gives the solution to an LQR problem.

Thrm 83. *The value function for (LQR) satisfies*

$$L_t(\mathbf{x}) = \mathbf{x}^\top \Lambda_t \mathbf{x}$$

where Λ_t is the solution to the Riccati Equation, see (Riccati). Moreover, the optimal control action is given by

$$\mathbf{a}_t^* = -G_t \mathbf{x}_t$$

where G is the Gain Matrix, see (Gain).

Proof. The Bellman equation is

$$L_{t-1}(\mathbf{x}) = \min_{\mathbf{a}} \{ \mathbf{x}^\top R \mathbf{x} + \mathbf{a}^\top Q \mathbf{a} + L_t(A \mathbf{x} + B \mathbf{a}) \}$$

We now argue by induction that $L_t(\mathbf{x}) = \mathbf{x}^\top \Lambda_t \mathbf{x}$ for all t . This is certainly true at time T where $L_T(\mathbf{x}) = \mathbf{x}^\top R \mathbf{x}$.

Assuming by induction that $L_t(\mathbf{x}) = \mathbf{x}^\top \Lambda_t \mathbf{x}$, we have that

$$\begin{aligned} L_{t-1}(\mathbf{x}) &= \min_{\mathbf{a}} \{ \mathbf{x}^\top R \mathbf{x} + \mathbf{a}^\top Q \mathbf{a} + (A \mathbf{x} + B \mathbf{a})^\top \Lambda_t (A \mathbf{x} + B \mathbf{a}) \} \\ &= \min_{\mathbf{a}} \{ \mathbf{x}^\top R \mathbf{x} + \mathbf{a}^\top Q \mathbf{a} + \mathbf{x}^\top A^\top \Lambda_t A \mathbf{x} + 2 \mathbf{a}^\top B^\top \Lambda_t A \mathbf{x} + \mathbf{a}^\top B^\top \Lambda_t B \mathbf{a} \}. \end{aligned}$$

Differentiating the above objective with respect to \mathbf{a} and setting equal to zero, minimizes the above objective and gives the condition

$$0 = 2Q\mathbf{a} + 2B^\top \Lambda_t A \mathbf{x} + 2B^\top \Lambda_t B \mathbf{a}$$

This implies that the optimal action is

$$\mathbf{a}^* = -[Q + B^\top \Lambda_t B]^{-1} B^\top \Lambda_t A \mathbf{x}.$$

In other words we see that $\mathbf{a}_t^* = -G_t \mathbf{x}_t$, as require above. However, we still need to verify that $V_{t-1}(\mathbf{x}) = \mathbf{x}^\top \Lambda_{t-1} \mathbf{x}$ holds for to complete the induction step. Substituting our expression for \mathbf{a}^* into the above minimization gives.

$$\begin{aligned} L_{t-1}(\mathbf{x}) &= \mathbf{x}^\top R \mathbf{x} + \mathbf{x}^\top A^\top \Lambda_t A \mathbf{x} + \mathbf{a}^{*\top} Q \mathbf{a}^* + 2\mathbf{a}^{*\top} B^\top \Lambda_t A \mathbf{x} + \mathbf{a}^{*\top} B^\top \Lambda_t B \mathbf{a}^* \\ &= \mathbf{x}^\top R \mathbf{x} + \mathbf{x}^\top A^\top \Lambda_t A \mathbf{x} - \mathbf{x}^\top A^\top \Lambda_t B [Q + B^\top \Lambda_t B]^{-1} B^\top \Lambda_t A \mathbf{x} \\ &= \mathbf{x}^\top \Lambda_{t-1} \mathbf{x} \end{aligned}$$

where the last inequality follows by our definition of Λ_{t-1} and gives the required Riccati equation. \square

LQR with Noise.

We consider a variation on the LQR problem. In particular we assume that \mathbf{x}_t is now randomly perturbed by some additional noise ϵ_{t-1} . We consider following optimization:

$$\begin{aligned} L_0(\mathbf{x}_0) = \text{minimize} \quad & \sum_{t=0}^{T-1} \{\mathbf{x}_t^\top R \mathbf{x}_t + \mathbf{a}_t^\top Q \mathbf{a}_t\} + \mathbf{x}_T^\top R \mathbf{x}_T \quad (\text{Noisy LQR}) \\ \text{subject to} \quad & \mathbf{x}_t = A \mathbf{x}_{t-1} + B \mathbf{a}_{t-1} + \epsilon_{t-1}, \quad t = 1, \dots, T \\ \text{over} \quad & \mathbf{a}_0, \dots, \mathbf{a}_{T-1} \in \mathbb{R}^m \end{aligned}$$

The only change with respect to (LQR) is that we add a random variable ϵ_{t-1} . Here we assume that ϵ_t is independent over time and has mean zero and covariance matrix N . That is

$$\mathbb{E}[\epsilon_t] = \mathbf{0} \quad \text{and} \quad \mathbb{E}[\epsilon_t^\top \epsilon_t] = N.$$

The next result shows that the optimal control remains the same when we add noise, only the value function changes a bit.

Thrm 84. *The optimal control for the noisy LQR problem is identical to the LQR problem [without noise] in Theorem 83 that is*

$$\mathbf{a}_t^* = G_t \mathbf{x}_t$$

where G_t is the Gain matrix defined in Gain . The value function now has the form

$$L_t(x) = \mathbf{x}^\top \Lambda_t \mathbf{x} + \gamma_t$$

where $\gamma_{t-1} = \text{tr}(\Lambda_t N) + \gamma_t$ and $\gamma_T = 0$.

Proof. The Bellman equation is

$$L_{t-1}(x) = \min_a \{ \mathbf{x}^\top R \mathbf{x} + \mathbf{a}^\top Q \mathbf{a} + \mathbb{E}[L_t(A\mathbf{x} + B\mathbf{a} + \epsilon)] \} .$$

We now argue by induction that $L_t(x) = \mathbf{x}^\top \Lambda_t \mathbf{x} + \gamma_t$ for all t . This is certainly true at time T where $L_T(x) = \mathbf{x}^\top R \mathbf{x}$.

Assuming by induction that $L_t(x) = \mathbf{x}^\top \Lambda_t \mathbf{x} + \gamma_t$, we have that

$$\begin{aligned} L_{t-1}(x) &= \min_a \{ \mathbf{x}^\top R \mathbf{x} + \mathbf{a}^\top Q \mathbf{a} + \mathbb{E}[(A\mathbf{x} + B\mathbf{a} + \epsilon)^\top \Lambda_t (A\mathbf{x} + B\mathbf{a} + \epsilon)] + \gamma_t \} \\ &= \min_a \{ \mathbf{x}^\top R \mathbf{x} + \mathbf{a}^\top Q \mathbf{a} + \mathbf{x}^\top A^\top \Lambda_t A \mathbf{x} + 2\mathbf{a}^\top B^\top \Lambda_t A \mathbf{x} + \mathbf{a}^\top B^\top \Lambda_t B \mathbf{a} \\ &\quad + \mathbb{E}[\epsilon^\top \Lambda_t \epsilon] + \gamma_t \} \end{aligned}$$

First, observe the minimization above is identical to the LQR minimization in Theorem 83, and so equals $\mathbf{x}^\top \Lambda_{t-1} \mathbf{x}$ by the proof given in Theorem 83. Second observe, a quick calculation shows that

$$\mathbb{E}[\epsilon^\top \Lambda_t \epsilon] = \sum_{ij} \mathbb{E}[\epsilon_i \epsilon_j \Lambda_{t,ij}] = \text{tr}(N \Lambda).$$

Thus $\gamma_t + \mathbb{E}[\epsilon^\top \Lambda_t \epsilon] = \gamma_{t-1}$ as defined above. These two observations give that

$$L_{t-1}(x) = \mathbf{x}^\top \Lambda_{t-1} \mathbf{x} + \gamma_{t-1}$$

as required. □

Linear Quadratic Gaussian.

We consider a Linear Quadratic Regularization problem but were there is both noise and imperfect state observation. In particular,

we do not directly observe the state x but instead some measurement y which we must use to control x . Further both x and y are subject to noise.

$$\begin{aligned}
 L_0(x_0) = & \text{minimize} && \sum_{t=0}^{T-1} \{x_t^\top R x_t + a_t^\top Q a_t\} + x_T^\top R x_T && \text{(LQG)} \\
 & \text{subject to} && x_t = A x_{t-1} + B a_{t-1} + \epsilon_{t-1}, \\
 & && y_t = C x_{t-1} + \delta_{t-1} \quad t = 1, \dots, T \\
 & \text{over} && a_0, \dots, a_{T-1}
 \end{aligned}$$

In addition to the terms in the definition of LQR and LQR with Noise, we introduce a matrix C and a noise term δ_t which is independent over time, has mean zero and covariance M . That is

$$\mathbb{E}[\delta] = 0, \quad \text{and} \quad \mathbb{E}[\delta\delta^\top] = M.$$

Later [when considering the Kalman Filter], we will need the random variables for δ and ϵ to be Gaussian [hence the name LQG] but we do not require this assumption yet.

Here the state x is not directly observed. So we must base decisions from data of past decision and measurements, that is

$$F_t = (y_t, \dots, y_1, a_{t-1}, \dots, a_0).$$

Result on LQG. The key result on LQG is that if we can estimate the mean state given F_t , i.e. $x_t := \mathbb{E}[x|F_t]$, then the optimal control is that same as from the LQR problem i.e. $a^* = -G\bar{x}_t$.

Thrm 85. For an LQG problem the optimal control at time t is

$$a_t^* = -G_t \bar{x}_t$$

where $\bar{x}_t = \mathbb{E}[x_t|F_t]$ and G_t is the Gain matrix (Gain). Further, the optimal value function is

$$L_t(F_t) = \mathbb{E}[x_t^\top \Lambda_t x_t | F_t] + I_t + \gamma_t$$

where

$$I_t = \sum_{\tau=t}^{T-1} \mathbb{E}[\Delta_\tau^\top (R + A^\top \Lambda_{\tau+1} A - \Lambda_\tau) \Delta_\tau | F_t] \quad \text{and} \quad \Delta_t = x_t - \bar{x}_t.$$

Before proving this result, we take a moment to discuss.

Certainty Equivalence. The last result is interesting because even though there is noise and we do not observe the system state. We still apply the same control as in the case where we have full information for a deterministic system. When we treat the mean as if it was the "true" state, we call this *certainty equivalence*.

In general applying a certainty equivalent estimate is not optimal, but it is for LQG systems. So why is certainty equivalence optimal here. In particular, if we look at the new term I_t in the value function, it looks like we need to estimate future values of Δ_τ which in principle should depend on the future actions and states that we visit. This would likely make for a complex dependence on the current action taken. However, it turns out, because the problem is linear, that Δ_t does not depend on the actions taken. So in the Bellman equation I_t is effectively a constant as far as the action taken is concerned. This simplifies the problem considerably and means we are still within the scope of our original LQR solution.

The following lemma shows that Δ_t does not depend on actions taken and states visited.

Lemma 7. Δ_τ is a constant with respect to $\mathbf{a}_0, \dots, \mathbf{a}_\tau$.

Proof. We recursively consider the update equation for \mathbf{x}_τ . Note that

$$\begin{aligned} \mathbf{x}_\tau &= A\mathbf{x}_{\tau-1} + B\mathbf{a}_{\tau-1} + \epsilon_{\tau-1} \\ &= A[A\mathbf{x}_{\tau-2} + B\mathbf{a}_{\tau-2} + \epsilon_{\tau-2}] + B\mathbf{a}_{\tau-1} + \epsilon_{\tau-1} \\ &= A^2\mathbf{x}_{\tau-2} + AB\mathbf{a}_{\tau-2} + B\mathbf{a}_{\tau-1} + A\epsilon_{\tau-2} + \epsilon_{\tau-1} \\ &\vdots \\ &= A^\tau\mathbf{x}_0 + \sum_{t=0}^{\tau-1} A^{\tau-1-t}B\mathbf{a}_t + \sum_{t=0}^{\tau-1} A^{\tau-1-t}\epsilon_t. \end{aligned}$$

Consequently notice,

$$\bar{\mathbf{x}}_\tau = \mathbb{E}[\mathbf{x}_\tau | F_\tau] = A^\tau\mathbf{x}_0 + \sum_{t=0}^{\tau-1} A^{\tau-1-t}B\mathbf{a}_t + \mathbb{E}\left[\sum_{t=0}^{\tau-1} A^{\tau-1-t}\epsilon_t \middle| F_\tau\right]$$

So

$$\mathbf{x}_\tau - \bar{\mathbf{x}}_\tau = \sum_{t=0}^{\tau-1} A^{\tau-1-t}\epsilon_t - \mathbb{E}\left[\sum_{t=0}^{\tau-1} A^{\tau-1-t}\epsilon_t \middle| F_\tau\right].$$

It seems like we are done, we have removed all dependence on the actions taken. But remember $F_t = (\mathbf{y}_t, \dots, \mathbf{y}_1, \mathbf{a}_{t-1}, \dots, \mathbf{a}_0)$ which we condition on above. In principle, we could modify the set of actions that we take to infer information about $\sum_{t=0}^{\tau-1} A^{\tau-1-t} \epsilon_t$ and thus there would be dependence on the actions taken in the conditional expectation above. However, fortunately, this turns out not to be the case.

To see this, first, let \mathbf{y}_t^0 be the sequence of observations made when actions are chosen to be zero. That is, when

$$\mathbf{y}_t = C\mathbf{x}_t = CA^\tau \mathbf{x}_0 + \sum_{t=0}^{\tau-1} CA^{\tau-1-t} B \mathbf{a}_t + \sum_{t=0}^{\tau-1} CA^{\tau-1-t} \epsilon_t$$

then \mathbf{y}_t^0 is given by

$$\mathbf{y}_t^0 = CA^\tau \mathbf{x}_0 + \sum_{t=0}^{\tau-1} CA^{\tau-1-t} \epsilon_t + \delta_t.$$

Since we know which actions are taken, we can always construct \mathbf{y}_t^0 from \mathbf{y}_t and vice versa. I.e. conditioning on $F_t = (\mathbf{y}_t, \dots, \mathbf{y}_1, \mathbf{a}_{t-1}, \dots, \mathbf{a}_0)$ is the same as conditioning on $F_t^0 = (\mathbf{y}_t^0, \dots, \mathbf{y}_1^0, \mathbf{a}_{t-1}, \dots, \mathbf{a}_0)$. Further, suppose we let π be the policy that we use to select the actions. In particular, suppose that $\mathbf{a}_t = \pi(F_t)$ where here π is some deterministic function. However given the discussion above, equally we could view actions as a function of F_t^0 , that is $\mathbf{a}_t = \pi^0(F_t^0)$ where π^0 is again a deterministic. Thus all information required to choose each action is determined by the function π^0 and the vectors $(\mathbf{y}_t^0, \dots, \mathbf{y}_1^0)$. In other words, conditioning on F_t^0 is the same as conditioning on $(\mathbf{y}_t^0, \dots, \mathbf{y}_1^0)$ and the deterministic function π^0 . However, π^0 is deterministic and thus independent of the random variable $\sum_{t=0}^{\tau-1} A^{\tau-1-t} \epsilon_t$, thus it plays no role in determining its conditional expectation. In summary we have found that

$$\begin{aligned} \mathbb{E}\left[\sum_{t=0}^{\tau-1} A^{\tau-1-t} \epsilon_t \middle| F_\tau\right] &= \mathbb{E}\left[\sum_{t=0}^{\tau-1} A^{\tau-1-t} \epsilon_t \middle| F_\tau^0\right] \\ &= \mathbb{E}\left[\sum_{t=0}^{\tau-1} A^{\tau-1-t} \epsilon_t \middle| \mathbf{y}_t^0, \dots, \mathbf{y}_1^0, \pi^0\right] = \mathbb{E}\left[\sum_{t=0}^{\tau-1} A^{\tau-1-t} \epsilon_t \middle| \mathbf{y}_t^0, \dots, \mathbf{y}_1^0\right] \end{aligned}$$

The right hand expression does not depend on the action taken and so the same is true of

$$\mathbf{x}_\tau - \bar{\mathbf{x}}_\tau = \sum_{t=0}^{\tau-1} A^{\tau-1-t} \epsilon_t - \mathbb{E}\left[\sum_{t=0}^{\tau-1} A^{\tau-1-t} \epsilon_t \middle| F_\tau\right].$$

□

A further more minor observation is the following Lemma.

Lemma 8.

$$\mathbb{E}[\mathbf{x}_t^\top M \mathbf{x}_t | F_t] = \bar{\mathbf{x}}_t^\top M \bar{\mathbf{x}}_t + \mathbb{E}[\Delta_t^\top M \Delta_t | F_t]$$

Proof.

$$\begin{aligned} \mathbb{E}[\mathbf{x}_t^\top M \mathbf{x}_t | F_t] &= \mathbb{E}[(\bar{\mathbf{x}}_t + \Delta)^\top M (\bar{\mathbf{x}}_t + \Delta) | F_t] \\ &= \bar{\mathbf{x}}_t^\top M \bar{\mathbf{x}}_t + 2\bar{\mathbf{x}}_t^\top M \mathbb{E}[\Delta_t | F_t] + \mathbb{E}[\Delta_t^\top M \Delta_t | F_t] \\ &= \bar{\mathbf{x}}_t^\top M \bar{\mathbf{x}}_t + \mathbb{E}[\Delta_t^\top M \Delta_t | F_t] \end{aligned}$$

□

Note this lemma is essentially just $\text{Var}(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2$.

Proof of Theorem 85. We can now use the above lemma to prove Theorem 85.

Proof of Theorem 85. The result of the Theorem is certainly true at time T , where $L(F_T) = \mathbf{x}_T^\top R \mathbf{x}_T$. Let's work back inductively assuming the form $L_{t+1}(F_{t+1})$ holds.

$$\begin{aligned} L_t(F_t) &= \min_{\mathbf{a}_t} \mathbb{E}[\mathbf{x}_t^\top R \mathbf{x}_t + \mathbf{a}_t^\top Q \mathbf{a}_t + L_{t+1}(F_{t+1}) | F_t] \\ &= \min_{\mathbf{a}_t} \mathbb{E} \left[\underbrace{\mathbf{x}_t^\top R \mathbf{x}_t + \mathbf{a}_t^\top Q \mathbf{a}_t}_{(a)} + \underbrace{\mathbb{E}[\mathbf{x}_{t+1}^\top \Lambda_{t+1} \mathbf{x}_{t+1} | F_{t+1}]}_{(b)} + \underbrace{L_{t+1}(F_{t+1})}_{(c)} \middle| F_t \right] \end{aligned}$$

Let's deal with the three terms (a), (b) and (c) above.

Firstly, for (a) we have by Lemma 8 that

$$\mathbb{E}[\mathbf{x}_t^\top R \mathbf{x}_t | F_t] = \bar{\mathbf{x}}_t^\top R \bar{\mathbf{x}}_t + \mathbb{E}[\Delta_t^\top R \Delta_t | F_t]$$

Second, for term (b):

$$\begin{aligned} &\mathbb{E}[\mathbf{x}_{t+1}^\top \Lambda_{t+1} \mathbf{x}_{t+1} | F_{t+1}] \\ &= \mathbb{E}[(A \mathbf{x}_t + B \mathbf{a}_t + \boldsymbol{\epsilon}_t)^\top \Lambda_{t+1} (A \mathbf{x}_t + B \mathbf{a}_t + \boldsymbol{\epsilon}_t) | F_{t+1}] \\ &= \mathbb{E}[\mathbf{x}_t^\top A^\top \Lambda_{t+1} A \mathbf{x}_t | F_t] + 2\bar{\mathbf{x}}_t^\top A^\top \Lambda_{t+1} B \mathbf{a}_t + \mathbf{a}_t^\top B^\top \Lambda_{t+1} B \mathbf{a}_t + \text{tr}(N \Lambda_{t+1}) \\ &= \bar{\mathbf{x}}_t^\top A^\top \Lambda_{t+1} A \bar{\mathbf{x}}_t + \mathbb{E}[\Delta_t^\top A^\top \Lambda_{t+1} A \Delta_t | F_t] \\ &\quad + 2\bar{\mathbf{x}}_t^\top A^\top \Lambda_{t+1} B \mathbf{a}_t + \mathbf{a}_t^\top B^\top \Lambda_{t+1} B \mathbf{a}_t + \text{tr}(N \Lambda_{t+1}) \end{aligned}$$

Third, for term (c),

$$\begin{aligned}\mathbb{E}[I_{t+1} + \gamma_{t+1}|F_t] &= \sum_{\tau=t+1}^{T-1} \mathbb{E} \left[\mathbb{E} \left[\Delta_\tau^\top (R + A^\top \Lambda_{\tau+1} A - \Lambda_\tau) \Delta_\tau | F_{t+1} \right] | F_t \right] + \gamma_{t+1} \\ &= \sum_{\tau=t+1}^{T-1} \mathbb{E} \left[\Delta_\tau^\top (R + A^\top \Lambda_{\tau+1} A - \Lambda_\tau) \Delta_\tau | F_t \right] + \gamma_{t+1}.\end{aligned}$$

Applying the last three terms to $L_t(F_t)$, above, we get that

$$\begin{aligned}L_t(F_t) &= \min_a \{ \bar{\mathbf{x}}_t^\top R \bar{\mathbf{x}}_t + \mathbf{a}^\top Q \mathbf{a} + \bar{\mathbf{x}}_t^\top A^\top \Lambda_{t+1} A \bar{\mathbf{x}}_t + 2 \mathbf{a}^\top B^\top \Lambda_{t+1} A \bar{\mathbf{x}}_t + \mathbf{a}^\top B^\top \Lambda_{t+1} B \mathbf{a} \} \\ &\quad + \mathbb{E}[\Delta_t^\top [R + A^\top \Lambda_{t+1} A] \Delta_t | F_t] + \sum_{\tau=t+1}^{T-1} \mathbb{E} \left[\Delta_\tau^\top (R + A^\top \Lambda_{\tau+1} A - \Lambda_\tau) \Delta_\tau | F_t \right] \\ &\quad + \text{tr}(N \Lambda_{t+1}) + \gamma_{t+1}\end{aligned}$$

Critically, we have applied Lemma 7, to take terms involving Δ_t from the minimization. An important consequence is that the minimization above is the same as for deterministic LQR problems. So the optimal control is $\mathbf{a}_t^* = -G_t \bar{\mathbf{x}}_t$, by the same calculation done in Theorem 83. So it is equal to $\bar{\mathbf{x}}_t^\top \Lambda_t \bar{\mathbf{x}}_t$. So applying this and Lemma 8, i.e. that $\bar{\mathbf{x}}_t^\top \Lambda_t \bar{\mathbf{x}}_t = \mathbb{E}[\mathbf{x}_t^\top \Lambda_t \mathbf{x}_t | F_t] - \mathbb{E}[\Delta_t^\top \Lambda_t \Delta_t | F_t]$. This gives that

$$\begin{aligned}L_t(F_t) &= \mathbb{E}[\mathbf{x}_t^\top \Lambda_t \mathbf{x}_t | F_t] \\ &\quad + \mathbb{E}[\Delta_t^\top [R + A^\top \Lambda_{t+1} A - \Lambda_t] \Delta_t | F_t] + \sum_{\tau=t+1}^{T-1} \mathbb{E} \left[\Delta_\tau^\top (R + A^\top \Lambda_{\tau+1} A - \Lambda_\tau) \Delta_\tau | F_t \right] \\ &\quad + \text{tr}(N \Lambda_{t+1}) + \gamma_{t+1} \\ &= \mathbb{E}[\mathbf{x}_t^\top \Lambda_t \mathbf{x}_t | F_t] + I_t + \gamma_t,\end{aligned}$$

where we apply the definitions of I_t and γ_t . This gives the required expression of $L_t(F_t)$. \square

Kalman Filter

Kalman filtering (and filtering in general) considers the following setting: we have a sequence of states \mathbf{x}_t , which evolves under random perturbations over time. Unfortunately we cannot observe \mathbf{x}_t ,

we can only observe some noisy function of x_t , namely, y_t . Our task is to find the best estimate of x_t given our observations of y_t .

For LQG we saw that we needed to have the estimate \hat{x}_t to make an optimal control decision. Assuming that perturbations are Gaussian, the Kalman filter can correctly iteratively calculate the mean of x_t , \hat{x}_t , and its variance.

Consider the equations

$$\begin{aligned}x_{t+1} &= Ax_t + Ba_t + \epsilon_t \\y_{t+1} &= Cx_{t+1} + \delta_{t+1}.\end{aligned}$$

where $\epsilon_t \sim \mathcal{N}(0, \Sigma^\epsilon)$, $\delta_{t+1} \sim \mathcal{N}(0, \Sigma^\delta)$ and ϵ_t and ν_t are independent. (We let Σ^ϵ be the sub-matrix of the covariance matrix corresponding to ϵ and so forth...)

The Kalman filter has two update stages: a prediction update and a measurement update. These are

$$\bar{x}_{t+1|t} = A\bar{x}_{t|t} + Ba_t, \quad (\text{Predict-1})$$

$$P_{t+1|t} = AP_{t|t}A^\top + \Sigma_t^\epsilon, \quad (\text{Predict-2})$$

and

$$\bar{x}_{t+1} = \bar{x}_{t+1|t} + K_t(y_{t+1} - C\bar{x}_{t+1|t}), \quad (\text{Measure-1})$$

$$P_{t+1} = P_{t+1|t} - K_tCP_{t+1|t}, \quad (\text{Measure-2})$$

where

$$K_t = P_{t+1|t}C^\top(CP_{t+1|t}C^\top + \Sigma_t^\delta)^{-1}.$$

The matrix K_t is often referred to as the Kalman Gain. Assuming the initial state x_0 is known and deterministic $P_{0|0} = 0$ in the above.

We will use the following proposition, which is a standard result on normally distributed random vectors, variances and covariances,

Prop 86. *Let u be normally distributed vector with mean \bar{u} and covariance Σ_u , i.e.*

$$u \sim \mathcal{N}(\bar{u}, \Sigma_u).$$

i) *For any matrix A and (constant) vector c , we have that*

$$Au + c \sim \mathcal{N}(A\bar{u} + c, A\Sigma_uA^\top).$$

ii) *If we take $u = (v, w)$ then w conditional on v is*

$$(w|v) \sim \mathcal{N}(\bar{w} + \Sigma_{wv}\Sigma_{vv}^{-1}(v - \bar{v}), \Sigma_{ww} - \Sigma_{wv}\Sigma_{vv}^{-1}\Sigma_{vw})$$

iii) *$\text{Var}(Au) = A\Sigma_uA^\top$, $\text{Cov}(Au, Bu) = A\Sigma_uB^\top$.*

We can justify the Kalman filtering steps by proving that the conditional distribution of x_{t+1} is given by the Prediction and measurement steps. Specifically we have the following.

Thrm 87.

$$\begin{aligned} (x_{t+1} | \mathbf{y}_{[0:t]}, \mathbf{a}_{[0:t]}) &\sim \mathcal{N}(\bar{x}_{t+1|t}, P_{t+1|t}) \\ (x_{t+1} | \mathbf{y}_{[0:t+1]}, \mathbf{a}_{[0:t]}) &\sim \mathcal{N}(\bar{x}_{t+1}, P_{t+1}) \end{aligned}$$

where $\mathbf{y}_{[0:t]} := (\mathbf{y}_0, \dots, \mathbf{y}_t)$ and $\mathbf{a}_{[0:t]} := (\mathbf{a}_0, \dots, \mathbf{a}_t)$. Thus

$$\mathbb{E}[x_{t+1} | F_{t+1}] = \bar{x}_{t+1}$$

where \bar{x}_{t+1} is given by (Measure-1).

Proof. We show the result by induction supposing that

$$(x_t | \mathbf{y}_{[0:t]}, \mathbf{a}_{[0:t-1]}) \sim \mathcal{N}(\bar{x}_t, P_t).$$

Since x_{t+1} is a linear function of x_t , we have that

$$(x_{t+1} | \mathbf{y}_{[0:t]}, \mathbf{a}_{[0:t]}) \sim \mathcal{N}(\bar{x}_{t+1|t}, P_{t+1|t}).$$

where, by Prop 86ii), we have that

$$\bar{x}_{t+1|t} = A\bar{x}_t + Ba_t, \quad P_{t+1|t} = AP_tA^\top + \Sigma^\epsilon.$$

Given $\mathbf{y}_{t+1} = Cx_{t+1} + \delta_t$, we have by Prop 86iii) that $\text{Var}(\mathbf{y}_{t+1} | \mathbf{y}_{[0:t]}, \mathbf{a}_{[0:t]}) = CP_{t+1|t}C^\top$ and $\text{Cov}(x_{t+1}, \mathbf{y}_{t+1} | \mathbf{y}_{[0:t]}, \mathbf{a}_{[0:t]}) = P_{t+1|t}C^\top$. Thus

$$((x_{t+1}, \mathbf{y}_{t+1}) | \mathbf{y}_{[0:t]}, \mathbf{a}_{[0:t]}) \sim \mathcal{N}\left((\bar{x}_{t+1|t}, C\bar{x}_{t+1|t}), \begin{pmatrix} P_{t+1|t} & P_{t+1|t}C^\top \\ CP_{t+1|t} & CP_{t+1|t}C^\top + \Sigma_t^\delta \end{pmatrix}\right).$$

Thus applying Prop 86ii), we get that

$$\begin{aligned} (x_{t+1} | \mathbf{y}_{[0:t+1]}, \mathbf{a}_{[0:t]}) &= ((x_{t+1} | \mathbf{y}_{[0:t]}, \mathbf{a}_{[0:t]}) | \mathbf{y}_{t+1}) \\ &\sim \mathcal{N}\left(\bar{x}_{t+1|t} + P_{t+1|t}C^\top [CP_{t+1|t}C^\top + \Sigma_t^\delta]^{-1}(\mathbf{y}_{t+1} - C\bar{x}_{t+1|t}), \right. \\ &\quad \left. P_{t+1|t} - P_{t+1|t}C^\top [CP_{t+1|t}C^\top + \Sigma_t^\delta]^{-1}CP_{t+1|t}\right). \end{aligned}$$

That is, as required, $(x_{t+1} | \mathbf{y}_{[0:t+1]}, \mathbf{a}_{[0:t]}) \sim \mathcal{N}(\bar{x}_{t+1}, P_{t+1})$ for

$$\begin{aligned} \bar{x}_{t+1} &= \bar{x}_{t+1|t} + K_t(\mathbf{y}_{t+1} - C\bar{x}_{t+1|t}) \\ P_{t+1} &= P_{t+1|t} - K_tCP_{t+1|t} \end{aligned}$$

where $K_t = P_{t+1|t}C^\top (CP_{t+1|t}C^\top + \Sigma_t^\delta)^{-1}$. □

Exercises

Ex 88 (Joint state-action costs). Suppose we consider an LQR problem whose objective is

$$\sum_{t=0}^{T-1} (\mathbf{x}_t, \mathbf{a}_t) D (\mathbf{x}_t, \mathbf{a}_t)^\top + \mathbf{x}_T^\top R \mathbf{x}_T$$

where

$$D = \begin{pmatrix} R & U \\ U^\top & Q \end{pmatrix}$$

then show that the Riccati equation becomes

$$\Lambda_t = R + A^\top \Lambda_{t+1} A - (A^\top \Lambda_{t+1} B + U) [Q - B^\top \Lambda_{t+1} B]^{-1} (B^\top \Lambda_{t+1} A + U)$$

for $t = 0, \dots, T-1$ and with $\Lambda_T = R$. The gain matrix is defined to be

$$G_t = [Q + B^\top \Lambda_t B]^{-1} (B^\top \Lambda_t A + U)$$

References

Bucy and Kalman developed the Kalman filter [13]. It is used extensively in control theory, for a recent text see Grenwal and Andrews [21]. For a machine learning and Bayesian perspective see Murphy [35].

Chapter 1

Continuous Time Control

1.1 Continuous Time Dynamic Programming

- The Hamilton-Jacobi-Bellman equation; a heuristic derivation; and proof of optimality.
 - Linear Quadratic Regularization.
-

Discrete time Dynamic Programming was given in Section 0.1. We now consider the continuous time analogue.

Time is continuous $t \in \mathbb{R}_+$; $x_t \in \mathcal{X}$ is the state at time t ; $a_t \in \mathcal{A}$ is the action at time t ; Given function $f : \mathcal{X} \times \mathcal{A} \rightarrow \mathcal{X}$, the state evolves according to a differential equation

$$\frac{dx_t}{dt} = f(x_t, a_t). \quad (1.1)$$

This is called the Plant Equation. A policy π chooses an action π_t at each time t . The (instantaneous) cost for taking action a in state x at time t is $c(a, x)$ and $c(x)$ is the reward for terminating in state x at time T .

Def 89 (Dynamic Program). *Given initial state x_0 , a dynamic program is the optimization*

$$\begin{aligned} L(x_0) := \text{Minimize} \quad & C(\mathbf{a}) := \int_0^T e^{-\alpha t} c(x_t, a_t) dt + e^{-\alpha T} c(x_T) & (\text{DP}) \\ \text{subject to} \quad & \frac{dx_t}{dt} = f(x_t, a_t), & t \in \mathbb{R}_+ \\ \text{over} \quad & a_t \in \mathcal{A}, & t \in \mathbb{R}_+ \end{aligned}$$

Further, let $C_\tau(\mathbf{a})$ (Resp. $L_\tau(x_\tau)$) be the objective (Resp. optimal objective) for (1.1) when the summation is started from $t = \tau$, rather than $t = 0$.

When a minimization problem where we minimize loss given the costs incurred is replaced with a maximization problem where we maximize winnings given the rewards received. The functions L , C and c are replaced with notation W , R and r .

Def 90 (Hamilton-Jacobi-Bellman Equation). *For a continuous-time dynamic program (1.1), the equation*

$$0 = \min_{a \in \mathcal{A}} \{c(x, a) + \partial_t L_t(x) + f(x, a) \partial_x L_t(x) - \alpha L_t(x)\} \quad (\text{HJB})$$

is called the Hamilton-Jacobi-Bellman equation. It is the continuous time analogue of the Bellman equation [2].

A Heuristic Derivation of the HJB Equation

We now argue why the Hamilton-Jacobi-Bellman equation is a good candidate for the Bellman equation in continuous time.

A good approximation to the plant equation (1.1) is

$$x_{t+\delta} - x_t = \delta f(x_t, a_t) \quad (1.2)$$

for $\delta > 0$ small, and a good approximation for the objective is

$$C(\mathbf{a}) := \sum_{t \in \{0, \delta, \dots, (T-\delta)\}} (1 - \alpha\delta)^{t/\delta} c(x_t, a_t) \delta + (1 - \alpha\delta)^{T/\delta} c(x_T) \quad (1.3)$$

This follows from the definition of the Riemann Integral and we further use the fact that $(1 - \alpha\delta)^{t/\delta} \rightarrow e^{-\alpha t}$ as $\delta \rightarrow 0$.

The Bellman equation for the discrete time dynamic program with objective (1.3) and plant equation (1.2) is

$$L_t(x) = \min_{a \in \mathcal{A}} \{c(x, a) \delta + (1 - \alpha\delta) L_{t+\delta}(x + \delta f(x, a))\}$$

If we minus $L_t(x)$ from each side in this Bellman equation and then divide by δ and let $\delta \rightarrow 0$ we get that

$$0 = \min_{a \in \mathcal{A}} \{c(x, a) + \partial_t L_t(x) + f(x, a) \partial_x L_t(x) - \alpha L_t(x)\},$$

where here we note that, by the Chain rule,

$$\frac{(1 - \alpha\delta) L_{t+\delta}(x + \delta f) - L_t(x)}{\delta} \xrightarrow{\delta \rightarrow 0} \partial_t L_t(x) + f(x, a) \partial_x L_t(x) - \alpha L_t(x).$$

Thus we derive the HJB equation as described above.

The following result shows that if we solve the HJB equation then we have an optimal policy.

Thrm 91 (Optimality of HJB). *Suppose that a policy Π has a value function $C_t(x, \Pi)$ that satisfies the HJB-equation for all t and x then, Π is an optimal policy.*

Proof. Using shorthand $C = C_t(\tilde{x}_t, \Pi)$:

$$\begin{aligned} -\frac{d}{dt} \left(e^{-\alpha t} C_t(\tilde{x}_t, \Pi) \right) &= e^{-\alpha t} \{ c(\tilde{x}_t, \tilde{\pi}_t) - [c(\tilde{x}_t, \tilde{\pi}_t) - \alpha C + f(\tilde{x}_t, \tilde{\pi}_t) \partial_x C + \partial_t C] \} \\ &\leq e^{-\alpha t} c(\tilde{x}_t, \tilde{\pi}_t) \end{aligned}$$

The inequality holds since the term in the square brackets is the objective of the HJB equation, which is *not* maximized by $\tilde{\pi}_t$. \square

Linear Quadratic Regularization

Def 92 (LQ problem). We consider a dynamic program of the form

$$\begin{aligned}
 &\text{Minimize} && \int_0^T [x_t^\top Q x_t + a_t^\top R a_t] dt + x_T^\top Q_T x_T && \text{(LQ)} \\
 &\text{subject to} && \frac{dx_t}{dt} = Ax_t + Ba_t, && t \in \mathbb{R}_+ \\
 &\text{over} && a_t \in \mathbb{R}^m, && t \in \mathbb{R}_+.
 \end{aligned}$$

Here $x_t \in \mathbb{R}^n$ and $a_t \in \mathbb{R}^m$. A and B are matrices. Q and R symmetric positive definite matrices. This an Linear-Quadratic problem (LQ problem).

Def 93 (Riccati Equation). The differential equation with

$$\dot{\Lambda}(t) = -Q - \Lambda(t)A - A^\top \Lambda(t) + \Lambda(t)BR^{-1}B^\top \Lambda(t) \quad \text{and} \quad \Lambda(T) = Q_T. \quad (\text{RicEq})$$

is called the Riccati equation.

Thrm 94. For each time t , the optimal action for the LQ problem is

$$a_t = -R^{-1}B^\top \Lambda(t)x_t,$$

where $\Lambda(t)$ is the solution to the Riccati equation.

Proof. The HJB equation for an LQ problem is

$$0 = \min_{a \in \mathbb{R}^m} \{x^\top Q x + a^\top R a + \partial_t L_t(x) + (Ax + Ra)^\top \partial_x L_t(x)\}$$

We now “guess” that the solution to above HJB equation is of the form $L_t(x) = x^\top \Lambda(t)x$ for some symmetric matrix $\Lambda(t)$. Therefore

$$\partial_x L_t(x) = 2\Lambda(t)x \quad \text{and} \quad \partial_t L_t(x) = x^\top \dot{\Lambda}(t)x$$

Substituting into the Bellman equation gives

$$0 = \min_{a \in \mathbb{R}^m} \{x^\top Q x + a^\top R a + x^\top \dot{\Lambda}(t)x + 2x^\top \Lambda(t)(Ax + Ba)\}.$$

Differentiating with respect to a gives the optimality condition

$$2Ra + 2x^\top \Lambda(t)B = 0$$

which implies

$$a = -R^{-1}B^\top \Lambda(t)x.$$

Finally substituting into the Bellman equation, above, gives the expression

$$0 = x^\top \left[Q + \dot{\Lambda}(t) + \Lambda(t)A + A^\top \Lambda(t) - \Lambda(t)BR^{-1}B^\top \Lambda(t) \right] x.$$

Thus the solution to the Riccati equation has a cost function that solves the Bellman equation and thus by Theorem 91 the policy is optimal. \square

1.2 Calculus of Variations

We have developed the HJB equation. In this section we develop an alternative view of minimizing the integral of costs. This is called the calculus of variations and it goes back to the early days of calculus. Here we can see where the *Hamilton* and *Jacobi* come in the name Hamilton-Jacobi-Bellman Equation. It also helps us understand the Pontryagin's Maximum Principle in Section ??, which can be seen as a calculus of variations characterization of continuous time optimal control.

We now provide a different approach to minimizing (or maximizing) the objective

$$C(x) = \int_0^T c(t, x(t), \dot{x}(t)) dt \quad (1.4)$$

with boundary conditions

$$x(0) = x_0 \quad \text{and} \quad x(T) = x_T, \quad (1.5)$$

where x_0 and x_T are constants. As in previous sections, we let

$$L(t, x) = \min \int_t^T c(s, x(s), \dot{x}(s)) ds \quad (1.6)$$

where we minimize over paths satisfying our boundary conditions. (That is $x(t) = x$ and $x(T) = x_T$.)

We can interpret the above objective as a continuous time control problem, as discussed in Section 1.1, but where the actions can be freely chosen to induce any change in x that we like. So our action at each state is the velocity i.e. $\dot{x}(t) = f(x, a) = a$ with $a \in \mathbb{R}^d$. Since our action has a very direct interpretation as velocity, we will use v as an index, for example, costs are $c(t, x, v)$.

Remark 95 (Lagrangian Mechanics). *We do not concern ourselves with the underlying physics here, but it is worth noting that many physically systems can be seen to minimize an objective (1.4). This is often referred to as Lagrangian mechanics, and can be seen as a reformulation of classical mechanics (e.g. from Newton). In this context, the cost function $c(t, x, v)$ is often called the Lagrangian.¹ However, in this text, we only refer to the Lagrangian when Lagrange multipliers are used, as discussed in Section A.7.*

¹In physics literature letter L is often used rather than c . However, to be consistent with earlier section, we use c for cost and L for optimal loss.

The Euler-Lagrange Equation

Specifically, we can characterize stationary points with the Euler-Lagrange equation:

Definition 2 (Euler-Lagrange Equation). *For a continuously differentiable real-valued function $c(t, x, v)$ and a continuously differentiable function $x(t)$,*

$$\frac{\partial c}{\partial x} - \frac{d}{dt} \frac{\partial c}{\partial v} = 0$$

By a stationary point, we mean the following.

Definition 3 (Stationary Point). *We say that a continuously differentiable real-valued function x is a stationary point of (1.4), if it satisfies the boundary condition (1.5) and for any continuously differentiable $z : [0, T] \rightarrow \mathbb{R}$ satisfying $z(0) = z(T) = 0$ it holds that*

$$C(x + \epsilon z) = C(x) + o(\epsilon),$$

as $\epsilon \rightarrow 0$.

Given that $C(x)$ changes smoothly with x , a local maximum or minimum of the objective must be a stationary point.

We see that the Euler-Lagrange equation characterizes stationary points.

Theorem 5. *Any stationary point of the objective (1.4) satisfies the Euler-Lagrange equation*

$$\frac{\partial c}{\partial x} - \frac{d}{dt} \frac{\partial c}{\partial v} = 0$$

Proof. Suppose $x(t)$ is stationary and let $z(t)$ be continuously differentiable with $z(0) = z(T) = 0$ then, for $\epsilon > 0$,

$$\begin{aligned} C(x + \epsilon z) &= \int_0^T c(t, x(t) + \epsilon z(t), \dot{x}(t) + \epsilon \dot{z}(t)) dt \\ &= \int_0^T c(t, x, \dot{x}) + \epsilon z(t) \partial_x c(t, x, \dot{x}) + \epsilon \dot{z}(t) \partial_v c(t, x, \dot{x}) + o(\epsilon) dt \\ &= C(x) + o(\epsilon) + \epsilon \int_0^T z(t) \partial_x c(t, x, \dot{x}) dt + \epsilon \int_0^T \dot{z}(t) \partial_v c(t, x, \dot{x}) dt. \end{aligned} \tag{1.7}$$

Since $x(t)$ is stationary, $C(x + \epsilon z) - C(x) = o(\epsilon)$. So, we see that

$$o(\epsilon) = \int_0^T \epsilon z(t) \partial_x c(t, x, \dot{x}) + \epsilon \dot{z}(t) \partial_v c(t, x, \dot{x}) dt.$$

Dividing by ϵ and letting $\epsilon \rightarrow 0$ gives that

$$0 = \int_0^T z(t) \partial_x c(t, x, \dot{x}) dt + \int_0^T \dot{z}(t) \partial_v c(t, x, \dot{x}) dt \quad (1.8)$$

Applying integration-by-parts to the final integral on the right,

$$\begin{aligned} \int_0^T \dot{z}(t) \partial_v c(t, x, \dot{x}) dt &= \int_0^T z(t) \frac{d}{dt} \{ \partial_v c(t, x, \dot{x}) \} dt + [z(t) \partial_v c(t, x, \dot{x})]_0^T \\ &= \int_0^T z(t) \frac{d}{dt} \{ \partial_v c(t, x, \dot{x}) \} dt \end{aligned} \quad (1.9)$$

The term in the square brackets is zero since $z(0) = z(T) = 0$. Thus substituting (1.9) into the righthand term in (1.8) gives that

$$\int_0^T z(t) \left[\partial_x c(t, x, \dot{x}) + \frac{d}{dt} \partial_v c(t, x, \dot{x}) \right] dt$$

Since the above condition must hold for all functions $z(t)$, it must be that the term in square-brackets is zero. That is, for all t ,

$$\partial_x c(t, x, \dot{x}) + \frac{d}{dt} \{ \partial_v c(t, x, \dot{x}) \} = 0$$

as required. □

We can use Euler-Lagrange equations to show results such as the following:

Proposition 2.

$$\partial_x L = -\partial_v c$$

Proof. We let $x(t)$ be the optimal solution started from x and $x(t) + \epsilon z(t)$ be the optimal solution started from $x + \epsilon$.² Note since both solutions

²Here we are assuming that optimal solutions change smoothly.

end at x_T , $z(t) = 0$ and $z(T) = 1$. By an identical Taylor expansion to the argument that gave (1.7), we have that

$$L(t, x + \epsilon) - L(t, x) = \epsilon \int_t^T z(s) \partial_x c(s, x, \dot{x}) + \dot{z}(s) \partial_v c(s, x, \dot{x}) ds + o(1).$$

Since $x(t)$ obeys the Euler-Lagrange equations $\partial_x c = d(\partial_v c)/dt$. So

$$\begin{aligned} \epsilon^{-1}[L(t, x + \epsilon) - L(t, x)] &= \int_t^T z(s) \frac{d}{ds} [\partial_v c(s, x, \dot{x})] + \frac{dz}{ds} \partial_v c(s, x, \dot{x}) ds + o(1) \\ &= \int_t^T \frac{d}{ds} [z(s) \partial_v c(s, x, \dot{x})] ds + o(1) \\ &= z(T) \partial_v c(T, x, \dot{x}) - z(t) \partial_v c(t, x, \dot{x}) + o(1) \\ &= -\partial_v c(t, x, \dot{x}) + o(1). \end{aligned}$$

Letting $\epsilon \rightarrow 0$ now gives the result. \square

The Hamilton-Jacobi Equation.

Just as there is a Lagrangian formulation of mechanics. There is a Hamilton-Jacobi formulation. There are multiple equations associated with this. But one that stands out from an optimal control perspective is that the optimal solution to (1.6) satisfies

$$\partial_t L + \min_v \{c(t, x, v) + v \partial_x L\} = 0 \quad (1.10)$$

This is called the Hamilton-Jacobi Equation, and the minimization term above is called the Hamiltonian. Notice this is basically the same as the HJB equation. (This a connection first made by Kalman, see references.) We will discuss this in more detail now and then prove that the equation holds.

Definition 4 (Hamiltonian). *The Hamiltonian and is defined*

$$H(t, x, \rho) := \min_v \{c(t, x, v) - \rho v\}$$

The term ρ is often called the momentum.

A few observations on the Hamiltonian:

- The above supremum is maximized at

$$\rho = \partial_v c(t, x, v).$$

(So Proposition 2 now states that $\partial_x L(t, x) = -\rho(t)$)

- So, with (t, x) fixed, for each velocity v , we can define a momentum ρ . Further if $v \mapsto \partial_v c(t, x, v)$ is increasing, then for each momentum ρ we can define a velocity v . Thus a one-to-one correspondence between velocity and momentum is often assumed.³
- Notice by Proposition 2, the optimal solution satisfies

$$\begin{aligned} H(t, x(t), \rho(t)) &= -\dot{x} \partial_v c(t, x, \dot{x}) + c(t, x, \dot{x}) \\ &= c(t, x, \dot{x}) + \dot{x} \partial_x L \end{aligned} \quad (1.11)$$

(Thus the Hamiltonian is the term in square bracket in the Hamilton-Jacobi Equation (1.12))

- Notice $\rho \mapsto -H(t, x, \rho)$ is the Legendre transform of $v \mapsto c(t, x, v)$. Specifically,

$$-H(t, x, \rho) = \sup_v \{ \rho v - c(t, x, v) \}$$

Given $c(t, x, a)$ is convex in a then there are lots of nice properties associated with this. For instance

$$c(t, x, v) = \sup_\rho \{ \rho v + H(t, x, \rho) \}$$

and $\partial_v c(t, x, \cdot)$ is the inverse of $\partial_\rho H(t, x, \cdot)$.

We can now show that the Hamilton-Jacobi equation (1.12) holds.

Theorem 6. *The optimal solution to the calculus of variations problem (1.6) satisfies the Hamilton-Jacobi equation:*

$$\partial_t L + H = 0. \quad (1.12)$$

Proof. Applying the chain rule to $L(t, x(t))$ gives

$$\frac{dL}{dt} = \partial_t L + \dot{x} \partial_x L.$$

By the Fundamental Theorem of Calculus, we know $\frac{dL}{dt} = -c(t, x, \dot{x})$. So applying this to the above equality gives,

$$\partial_t L = -\dot{x} \partial_x L - c.$$

Thus by (1.11), $\partial_t L + H = 0$, as required. \square

³Though formally we need $v \mapsto c(t, x, v)$ to be convex in v for this to work.

1.3 Pontryagin's Maximum Principle

We consider a continuous time dynamic program as described in Definition 89. That is we aim to solve

$$L(x_0) = \min_{a(t): t \geq 0} \int_0^T c(x(t), a(t)) dt + \kappa(x(T)) \quad \text{s.t.} \quad \dot{x}(t) = f(x(t), a(t)).$$

Here we do not assume that T is a fixed constant. Instead we assume that T is the first time that $x(t)$ visits a set S . We note that in this case the HJB equation is given by

$$\begin{aligned} 0 &= \min_a \{c(x, a) + f(x, a) \partial_x L\} && \text{for } x \notin S, \\ L(x) &= c(x) && \text{for } x \in S. \end{aligned}$$

As given in our discussion on Hamiltonian's in Section 1.2, the Pontryagin approach considers the Hamiltonian

$$H(x, a, \lambda) = \lambda^\top f(x, a) - c(x, a).$$

Theorem 7 (Pontryagin's Maximum Principle). *If $x(t)$ and $a(t)$ are the optimal state and control functions, then*

$$\dot{\lambda}(t) = -\partial_x H \tag{1.13}$$

$$\dot{x}(t) = \partial_\lambda H \tag{1.14}$$

and

$$0 = H(x(t), a(t), \lambda(t)) = \max_a H(x(t), a, \lambda(t)) \tag{1.15}$$

and at time T

$$\lambda(T) = -\partial_x \kappa(x(T)).$$

We give an informal proof to get the main ideas across. We refer the reader to [32] for instance for a fuller account of the original proof.

Sketch Proof. Throughout the proof we let

$$\lambda(t) = \partial_x L(x(t)).$$

Note that the condition $\dot{x}(t) = \partial_\lambda H$ just states that $\dot{x}(t) = f(x(t), a(t))$, which is the required dynamics of our model.

Given $\lambda(t)$ as defined above, the condition (1.15) states that

$$0 = c(x(t), a(t)) + f(x(t), a(t))\partial_x L = \min_a \{c(x, a) + f(x, a)\partial_x L\}$$

which is just the HJB equation above.

For the condition on the change in $\lambda(t)$, we Taylor expand the value function over a time-step of size $\epsilon > 0$ started from state x taking action a at time t

$$L(x) = \epsilon c(x, a) + L(x + f(x, a)\epsilon) + o(\epsilon).$$

Differentiating with respect to x gives

$$\frac{d}{dx}L(x) = \epsilon \frac{d}{dx}c(x, a) + \partial_x L(x + f(x, a)\epsilon) \frac{d}{dx}\{x + f(x, a)\epsilon\} + o(\epsilon).$$

Since $\lambda(t) = \partial_x L(x(t))$ and $\lambda(t + \epsilon) = \partial_x L(x + f(x, a)\epsilon) + o(\epsilon)$ it holds that

$$-\lambda(t) = \epsilon \frac{d}{dx}c(x, a) + \lambda(t + \epsilon) + \lambda(t + \epsilon) \cdot \partial_x f(x, a)$$

which after rearranging and letting ϵ go to zero gives

$$\dot{\lambda}(t) = \partial_x c(x, a) - \lambda(t)\partial_x f(x, a) = -\partial_x H$$

□

Given the proof above we see that (1.14) accounts for the dynamics of system, (1.13) shows that we consider an objective that is summing costs and (1.15) ensures that the actions that we are taking are maximizing (the HJB-equation).

1.4 Stochastic Integration

- Heuristic derivation of the Stochastic Integral and Itô's formula.

What follows is a heuristic derivation of the Stochastic Integral, Stochastic Differential Equations and Itô's Formula.

First note that for $(B_t : t \geq 0)$ a standard Brownian motion argue that, for all T and for δ sufficiently small and positive,

$$\sum_{t \in \{0, \delta, \dots, T\}} (B_{t+\delta} - B_t) = B_T \quad \text{and} \quad \sum_{t \in \{0, \delta, \dots, T\}} (B_{t+\delta} - B_t)^2 \approx T \quad (1.16)$$

The 1st sum is an interpolating sum. By independent increments property of Brownian motion, the 2nd sum adds IIDRVs with each with mean δ . Thus the strong law of large numbers gives the approximation. From this it is reasonable to expect that

$$\sum_{t \in \{0, \delta, \dots, T\}} \sigma(X_t) (B_{t+\delta} - B_t) \approx \int_0^T \sigma(X_t) dB_t$$

and

$$\sum_{t \in \{0, \delta, \dots, T\}} \mu(X_t) (B_{t+\delta} - B_t)^2 \approx \int_0^T \mu(X_t) dt.$$

The first sum, above, is approximation from a Riemann-Stieltjes integral, i.e.

$$\int_0^T f(t) dg(t) \approx \sum_{t \in \{0, \delta, \dots, T\}} f(t) (g(t+\delta) - g(t)).$$

So one might expect a integral limit. (This is unrigorous because Riemann-Stieltjes Integration only applies to functions with finite variation – while Brownian motion does not have finite variation.) The second sum is a Riemann integral upon using the approximation $(B_{t+\delta} - B_t)^2 \approx \delta$. This is, very roughly, how a stochastic integral is defined.

We can also define stochastic differential equations. If we inductively define X_t by the recursion

$$X_{t+\delta} - X_t = \sigma(X_t)(B_{t+\delta} - B_t) + \mu(X_t)\delta, \quad t = 0, \delta, 2\delta, \dots \quad (1.17)$$

then, by summing over values of $t \in \{0, \delta, \dots, T - \delta\}$, we expect X_t to approximately obey an equation of the form

$$X_T = X_0 + \int_0^T \sigma(X_t) dB_t + \int_0^T \mu(X_t) dt.$$

This gives a Stochastic Differential Equation.

Often in differential and integration, we apply chain rule, $\frac{df(x_t)}{dt} = f'(x_t) \frac{dx_t}{dt}$. It's the analogous result for Stochastic Integrals. Let X_t be as above. For a twice continuously differentiable function f and $\delta > 0$ small, we can apply a Taylor approximation

$$\begin{aligned} & f(X_{t+\delta}) - f(X_t) \\ &= f(X_t + \sigma(X_t)(B_{t+\delta} - B_t) + \mu(X_t)\delta) - f(X_t) \\ &= f'(X_t) \{ \mu\delta + \sigma \cdot (B_{t+\delta} - B_t) \} + \frac{f''(X_t)}{2} \{ \mu\delta + \sigma \cdot (B_{t+\delta} - B_t) \}^2 + o(\delta) \\ &= f'(X_t) \{ \mu\delta + \sigma \cdot (B_{t+\delta} - B_t) \} + \frac{f''(X_t)}{2} \sigma^2 \cdot (B_{t+\delta} - B_t)^2 + o(\delta) \end{aligned}$$

In the last equality we use that $(B_{t+\delta} - B_t) = o(\delta^{1/2})$ (which follows from (1.16)). Thus we see that

$$f(X_{t+\delta}) - f(X_t) \approx \left[f'(X_t) \mu(X_t) + \frac{\sigma(X_t)^2}{2} f''(X_t) \right] \delta + f'(X_t) \sigma(X_t) (B_{t+\delta} - B_t).$$

Consequently we expect that $f(X_t)$ obeys the following Stochastic Differential Equation:

$$f(X_T) - f(X_0) = \int_0^T \left[f'(X_t) \mu(X_t) + \frac{\sigma(X_t)^2}{2} f''(X_t) \right] dt + \int_0^T f'(X_t) \sigma(X_t) dB_t.$$

This is Ito's formula.

1.5 Diffusion Control Problems

- The Hamilton-Jacobi-Bellman Equation.
 - Heuristic derivation; Davis-Varaiya Martingale Principle of Optimality.
-

We consider a continuous time analogue of Markov Decision Processes from Section 0.3.

Time is continuous $t \in \mathbb{R}_+$; $X_t \in \mathbb{R}^n$ is the state at time t ; $a_t \in \mathcal{A}$ is the action at time t .

Def 96 (Plant Equation). *Given functions $\mu_t(X_t, a_t) = (\mu_t^i(X_t, a_t)) : i = 1, \dots, n$ and $\sigma_t(X_t, a_t) = (\sigma_t^{ij}(X_t, a_t)) : i = 1, \dots, n, j = 1, \dots, m$, the state evolves according to a stochastic differential equation*

$$dX_t = \mu_t(X_t, a_t)dt + \sigma_t(X_t, a_t) \cdot dB_t$$

where B_t is an m -dimensional Brownian motion. This is called the Plant Equation.

A policy π chooses an action π_t at each time t . (We assume that π_t is adapted and previsible.) Let \mathcal{P} be the set of policies. The (instantaneous) cost for taking action a in state x at time t is $c_t(a, x)$ and $c_T(x)$ is the cost for terminating in state x at time T .

Def 97 (Diffusion Control Problem). *Given initial state x_0 , a dynamic program is the optimization*

$$L(x_0) := \underset{\Pi \in \mathcal{P}}{\text{minimize}} C(x_0, \Pi) := \mathbb{E}_{x_0} \left[\int_0^T e^{-at} c_t(X_t, \pi_t) dt + e^{-\alpha T} c_T(X_T) \right] \quad (\text{DCP})$$

Further, let $C_\tau(x, \Pi)$ (Resp. $L_\tau(x)$) be the objective (Resp. optimal objective) for (DCP) when the integral is started from time $t = \tau$ with $X_t = x$, rather than $t = 0$ with $X_0 = x$.

Def 98 (Hamilton-Jacobi-Bellman Equation). *For a Diffusion Control Problem (DCP), the equation*

$$0 = \min_{a \in \mathcal{A}} \left\{ c_t(x, a) + \partial_t L_t(x) + \mu_t(x, a) \cdot \partial_x L_t(x) + \frac{1}{2} [\sigma^T \sigma] \cdot \partial_{xx} L_t(x) - \alpha L_t(x) \right\} \quad (\text{HJB})$$

is called the Hamilton-Jacobi-Bellman equation.⁴ It is the continuous time analogue of the Bellman equation [2].

Heuristic Derivation of the HJB equation

We heuristically develop a Bellman equation for stochastic differential equations using our knowledge of the Bellman equation for Markov decision processes, in Section 0.3 (Theorem 23) and our heuristic derivation of the stochastic integral in Section A.2. This is analogous to continuous time control in Section 1.1.

Perhaps the main thing to remember is that (informally) the HJB equation is

$$0 = \min_{\text{actions}} \{ \text{"instantaneous cost"} + \text{"Drift term from Ito's Formula"} \}.$$

Here Ito's formula is applied to the optimal value function at time t , $L_t(x)$. This is much easier to remember (assuming you know Ito's formula).

We suppose (for simplicity) that X_t belongs to \mathbb{R} and is driven by a one-dimensional Brownian motion. The plant equation in Def 96 is approximated by

$$X_{t+\delta} - X_t = \mu_t(X_t, \pi_t)\delta + \sigma_t(X_t, \pi_t)(B_{T+\delta} - B_t)$$

for small δ (recall (1.17)). Similarly the cost function in (DCP) can be approximated by

$$C_t(x, \Pi) \approx \mathbb{E} \left[\sum_{t \in \{0, \delta, \dots, T-\delta\}} (1 - \alpha\delta)^{\frac{t}{\delta}} c_t(X_t, \pi_t)\delta + (1 - \alpha\delta)^{\frac{T}{\delta}} c_T(X_T) \right].$$

This follows from the definition of a Riemann Integral and since $(1 - \alpha\delta)^{\frac{t}{\delta}} \rightarrow e^{-\alpha t}$. The Bellman equation for this objective function and plant equation is satisfies

$$L_t(x) = \min_{a \in \mathcal{A}} \{ c_t(x, a)\delta + (1 - \alpha\delta)\mathbb{E}_{x,a} [L_{t+\delta}(X_{t+\delta})] \}.$$

or, equivalently,

$$0 = \min_{a \in \mathcal{A}} \left\{ c_t(x, a) + \frac{1}{\delta} \mathbb{E}_{x,a} [L_{t+\delta}(X_{t+\delta}) - L_t(x)] - \alpha \mathbb{E}_{x,a} [L_{t+\delta}(X_{t+\delta})] \right\}.$$

⁴Here $[\sigma^\top \sigma] \cdot \partial_{xx} L_t(x)$ is the dot-product of the Hessian matrix $\partial_{xx} L_t(x)$ with $\sigma^\top \sigma$. I.e. we multiply component-wise and sum up terms.

Now by Ito's formula $L_t(X_t)$ can be approximated by

$$\begin{aligned} & L_{t+\delta}(X_{t+\delta}) - L_t(X_t) \\ & \approx \left[\partial_t L + \mu_t(X_t, \pi_t) \cdot \partial_x L + \frac{\sigma_t(X_t, \pi_t)^2}{2} \partial_{xx} L \right] \delta + \partial_x L \cdot \sigma_t(X_t, \pi_t) \cdot (B_{t+\delta} - B_t) \end{aligned}$$

Thus

$$\frac{1}{\delta} \mathbb{E}_{x,a} [L_{t+\delta}(X_{t+\delta}) - L_t(x)] = \partial_t L + \mu_t(X_t, \pi_t) \cdot \partial_x L + \frac{\sigma_t(X_t, \pi_t)^2}{2} \partial_{xx} L$$

Substituting in this into the above Bellman equation and letting $\delta \rightarrow 0$, we get, as required,

$$0 = \min_{a \in \mathcal{A}} \left\{ c_t(x, a) + \partial_t L + \mu_t(x, a) \cdot \partial_x L + \frac{\sigma_t(x, a)^2}{2} \partial_{xx} L - \alpha L_t(x) \right\}.$$

The following gives a rigorous proof that the HJB equation is the right object to consider for a diffusion control problem.

Thrm 99 (Davis-Varaiya Martingale Principle of Optimality). *Suppose that there exists a function $L_t(x)$ with $L_T(x) = e^{-\alpha T} c_T(x)$ and such that for any policy Π with states X_t*

$$M_t = L_t(X_t) + \int_0^t e^{-\alpha \tau} c_\tau(X_\tau, \Pi) d\tau$$

is a sub-martingale and, moreover that for some policy Π^ , M_t is a martingale then Π^* is optimal and*

$$L_0(X_0) = \min_{\Pi \in \mathcal{P}} \mathbb{E} \left[\int_0^T e^{-\alpha \tau} c_\tau(X_\tau, \pi_\tau) d\tau + c_T(X_T) \right].$$

Proof. Since M_t is a sub-martingale for all Π , we have

$$L_0(X_0) = M_0 \leq \mathbb{E}[M_T] = \underbrace{\mathbb{E}_{X_0} \left[\int_0^T e^{-\alpha \tau} c_\tau(X_\tau, \Pi_\tau) d\tau + \underbrace{L_T(X_T)}_{C_T(X_T)} \right]}_{C(x_0, \Pi)}$$

Therefore $L_0(X_0) \leq C(X_0, \Pi)$ for all policies Π .

If M_t is a Martingale for policy Π^* , then by the same argument $L_0(X_0) = C(X_0, \Pi^*)$. Thus

$$C(X_0, \Pi^*) = L_0(X_0) \leq C(X_0, \Pi)$$

for all policies Π and so Π^* is optimal, and it holds that

$$L_0(X_0) = \min_{\Pi \in \mathcal{P}} \mathbb{E} \left[\int_0^T e^{-\alpha\tau} c_\tau(X_\tau, \pi_\tau) d\tau + c_T(X_T) \right].$$

□

1.6 Merton Portfolio Optimization

- HJB equation for Merton Problem; CRRA utility solution; Proof of Optimality.
- Multiple Assets; Dual Value function Approach.

We consider a specific diffusion control problem. We focus on setting where there is one risky asset and one riskless asset, though we will see that much of the analysis passes over to multiple assets.

Def 100 (The Merton Problem – Plant Equation). *In the Merton problem you wish to optimise your long run consumption. You may invest your wealth in a bank account receiving riskless interest r , or in a risky asset with value S_t obeying the following SDE*

$$dS_t = S_t \{ \sigma dB_t + \mu dt \}$$

where each $B = (B_t : t \geq 0)$ is an independent standard Brownian motion.

Wealth $(W_t : t \geq 0)$ obeys the SDE

$$dW_t = \underbrace{r(W_t - n_t S_t) dt}_{\text{Wealth in bank}} + \underbrace{n_t dS_t}_{\text{Wealth in asset}} - \underbrace{c_t dt}_{\text{consumption}} \quad (1.18a)$$

$$= r(W_t - n_t \cdot S_t) dt + n_t \cdot dS_t - c_t dt \quad (1.18b)$$

You can control c_t your rate of consumption at time t and n_t the number of stocks the risky asset at time t . Also, we define $\theta_t = n_t S_t$ to be the wealth in the risky asset at time t .

Def 101 (The Merton Problem – Objective). *Given the above plant equation, (1.18), the objective is to maximize the long-term utility of consumption*

$$V(w_0) = \max_{(n_t, c_t)_{t \geq 0} \in \mathcal{P}(w_0)} \mathbb{E} \left[\int_0^\infty e^{-\rho t} u(c_t) dt \right].$$

Here ρ is a positive constant and $u(c)$ is a concave increasing utility function. The set $\mathcal{P}(w_0)$ is the set of policies given initial wealth w_0 . Further, let $V(w, t)$ be the optimal objective with the integral starting for time t with $w_t = w$.

Prop 102. The HJB equation for the Merton Problem can be written as

$$0 = \max_c \{u(c) - c\partial_w V\} + \max_\theta \left\{ \theta(\mu - r)\partial_w V + \frac{1}{2}\sigma^2\theta^2\partial_{ww} V \right\} - \rho V + r w \partial_w V$$

Here the optimal θ and c are given by

$$\theta^* = -\frac{\partial_w V}{\partial_{ww} V} \sigma^{-2} (\mu - r), \quad c^* = (u')^{-1}(\partial_w V)$$

Proof. First we note that we can rewrite the SDE for W_t as follows:

$$\begin{aligned} dW_t &= r(W_t - n_t \cdot S_t) dt + \underbrace{n_t \cdot dS_t}_{=n_t S_t \mu dt + n_t S_t \sigma dB_t} - c_t dt \\ &= (rW_t + (\mu - r)\theta_t - c_t) dt + \theta_t \sigma dB_t. \end{aligned}$$

Recall that informally the HJB equation is

$$0 = \max_{\text{actions}} \{ \text{"instantaneous cost"} - \rho V + \text{"Drift term from Ito's Formula"} \}.$$

Notice that if we apply Ito's formula to $V(W_t)$ we get that

$$\begin{aligned} dV(W_t) &= \partial_w V(W_t) dW_t + \frac{1}{2} \partial_{ww} V(W_t) d[W]_t \\ &= \partial_w V(W_t) [r(W_t - n_t \cdot S_t) dt + n_t \cdot dS_t - c_t dt] \\ &\quad + \partial_t V(W_t) dt + \frac{\theta^2 \sigma^2}{2} \partial_{ww} V(W_t) dt \end{aligned}$$

Applying this to the above term gives as required

$$\begin{aligned} 0 &= \max_{\theta, c} \left\{ u(c) - \rho V + (r w + \theta \cdot (\mu - r) - c) \partial_w V + \frac{1}{2} \sigma^2 \theta^2 \partial_{ww} V \right\} \\ &= \max_c \{u(c) - c\partial_w V\} + \max_\theta \left\{ \theta(\mu - r) + \frac{1}{2} \sigma^2 \theta^2 \partial_{ww} V \right\} - \rho V + r w \partial_w V \end{aligned}$$

Differentiating the HJB equation w.r.t. θ gives

$$\sigma^2 \theta \partial_{ww} V = -(\mu - r) \partial_w V.$$

Now rearrange for θ^* . The final part is a straight-forward calculation on $\sup_c \{u(c) - c\partial_w V\}$. \square

Merton for CRRA Utility

We focus on the case of CRRA utility, that is:

$$u(c) = \frac{c^{1-R}}{1-R}$$

for $R > 0$.

$$V(w_0) = \max_{(n_t, c_t)_{t \geq 0}} \mathbb{E} \left[\int_0^\infty e^{-\rho t} \frac{c_t^{1-R}}{1-R} dt \right].$$

Prop 103. For a CRRA utility it holds that:

a) The Value function takes the form

$$V(w) = \gamma \frac{w^{1-R}}{1-R}$$

for some position constant $\gamma > 0$.

b) The HJB equation is optimized by

$$\theta^* = \frac{w}{R} \sigma^{-2} (\mu - r),$$

$$c^* = \gamma^{-\frac{1}{R}} w \quad \text{and} \quad \sup_c \{u(c) - c \partial_w V\} = \frac{R}{1-R} \gamma^{1-\frac{1}{R}} w^{1-R}.$$

c) The HJB equation is satisfied by parameters

$$\gamma^* = R^{-1} \left\{ \rho + (R-1) \left(r + \frac{1}{2} \frac{\kappa^2}{R} \right) \right\}$$

where

$$\kappa = \sigma^{-1} (\mu - r).$$

Proof. a) Note that having a policy for initial wealth λw_0 is the same as having a policy of wealth w_0 and then multiplying each amount invested by λ :

$$\begin{aligned} V(\lambda w) &= \max_{(n_t, c_t)_{t \geq 0} \in \mathcal{P}(\lambda w_0)} \mathbb{E} \left[\int_0^\infty e^{-\rho t} \frac{c_t^{1-R}}{1-R} dt \right] \\ &= \max_{(n_t, c_t)_{t \geq 0} \in \mathcal{P}(w_0)} \mathbb{E} \left[\int_0^\infty e^{-\rho t} \frac{(\lambda c_t)^{1-R}}{1-R} dt \right] = \lambda^{1-R} V(w). \end{aligned}$$

Letting $\lambda = w^{-1}$ and $\gamma = (1-R)V(1)$ gives the result.

b) By part a), $\partial_w V(w) = \gamma w^{-R}$ and $\partial_{ww} V(w) = -R\gamma w^{-xR-1}$. So, by Prop 102,

$$\theta^* = -\frac{\partial_w V}{\partial_{ww} V} \sigma^{-2} (\mu - r) = \frac{w}{R} \sigma^{-2} (\mu - r)$$

Also,

$$\sup_c \{u(c) - c\partial_w V\} \implies u'(c) = \partial_w V = \gamma w^{-R}$$

which since $u'(c) = c^{-R}$, gives that $c^* = \gamma^{-\frac{1}{R}} w$. Further,

$$\begin{aligned} \sup_c \{u(c) - c\partial_w V\} &= \frac{c^{*1-R}}{1-R} - c^* \partial_w V(c^*) \\ &= \frac{(\gamma^{-\frac{1}{R}} w)^{1-R}}{1-R} - (\gamma^{-\frac{1}{R}} w) \gamma w^{-R} \\ &= \frac{R}{1-R} \gamma^{1-\frac{1}{R}} w^{1-R}, \end{aligned}$$

as required.

c) Applying a) and b) to the HJB equation in Prop 102 gives

$$\begin{aligned} 0 &= \frac{R}{1-R} \gamma^{1-\frac{1}{R}} w^{1-R} - \frac{1}{2} \sigma^{-2} (\mu - r)^2 \frac{(\partial_w V)^2}{\partial_{ww} V} - \rho V + r w \partial_w V \\ &= \frac{R}{1-R} \gamma^{1-\frac{1}{R}} w^{1-R} - \frac{1}{2} \sigma^2 (\mu - r)^2 \frac{(\gamma w^{-R})^2}{(-R\gamma w^{-R-1})} - \rho \gamma \frac{w^{1-R}}{1-R} \\ &= \gamma w^{1-R} \left[\frac{R}{1-R} \gamma^{\frac{1}{R}} + \frac{1}{2} \sigma^2 \frac{(\mu - r)^2}{R} - \frac{\rho}{1-R} + r \right]. \end{aligned}$$

Cancelling γw^{1-R} and rearranging gives the required for γ . \square

To summarize: we notice we have shown that the parameters

$$\theta^* = \frac{w}{R} \sigma^{-2} (\mu - r), \quad c^* = \gamma^{-\frac{1}{R}} w, \quad (1.19a)$$

$$\gamma^* = R^{-1} \left\{ \rho + (R-1) \left(r + \frac{1}{2} \frac{\kappa^2}{R} \right) \right\}, \quad \kappa = \sigma^{-1} (\mu - r). \quad (1.19b)$$

give a solution to the HJB equation for the Merton problem. (Although we have not yet proven them to be optimal.) Further note that the wealth under these parameters obeys the SDE

$$dW_t = W_t \left\{ R^{-1} \kappa dW_t + (r + R^{-1}) |\kappa|^2 - \gamma \right\} dt$$

which is a geometric Brownian motion:

$$W_t = W_0 \exp \left\{ R^{-1} \kappa W_t + \left(r + \frac{1}{R^2} \kappa^2 (2R_1) - \gamma \right) t \right\}$$

We now give rigorous argument for the optimality of parameters c^* , θ^* and γ^* for the Merton problem with CRRA utility. (This section can be skipped if preferred.)

Thrm 104. *The parameters in (1.19), above, are optimal for the Merton problem.*

Proof. Since $u(y)$ is concave, $u(y) \leq u(x) + (y - x)u'(x)$. Thus for $\zeta_t = e^{-\rho t} u'(c_t^*) \propto e^{-\kappa B_t - (r + \frac{1}{2} \kappa^2) t}$ we have that

$$\begin{aligned} \mathbb{E} \left[\int_0^\infty e^{-\rho t} u(c_t) dt \right] &\leq \mathbb{E} \left[\int_0^\infty e^{-\rho t} \{ u(c_t^*) + (c_t - c_t^*) u'(c_t^*) \} dt \right] \\ &= \mathbb{E} \left[\int_0^\infty e^{-\rho t} u(c_t^*) dt \right] + \mathbb{E} \left[\int_0^\infty (c_t - c_t^*) \zeta_t dt \right] \end{aligned} \quad (1.20)$$

Next we show that

$$Y_t = \zeta_t W_t + \int_0^t \zeta_s c_s ds$$

is a positive local martingale. It is clear that the function Y_t is positive. Note that

$$\zeta_t = e^{-\rho t} u'(c_t^*) = D e^{-\kappa B_t - (r + \frac{\kappa^2}{2}) t} \quad \text{where} \quad \gamma w_0.$$

Define function

$$f_t(W, B) = W \exp \left\{ -\kappa B - \left(r + \frac{\kappa^2}{2} t \right) \right\}$$

and note that $\zeta_t W_t = D f_t(W_t, B_t)$. Now lets apply Ito's formula to $f_t(W_t, B_t)$. By Ito's formula:

$$df = \partial_t f dt + \partial_w f dW_t + \partial_B f dB_t + \frac{1}{2} \partial_{BB} f d[B]_t + \partial_{Bw} f d[BW]_t + \frac{1}{2} \partial_{ww} f d[W]_t.$$

Now lets check terms.

$$\begin{aligned} \partial_t f &= -\left(r + \frac{1}{2} \kappa^2 \right) W e^{-\kappa B_t - (r + \frac{1}{2} \kappa^2) t} & \partial_B f &= -\kappa W e^{-\kappa B_t - (r + \frac{1}{2} \kappa^2) t} & \partial_w f &= e^{-\kappa B_t - (r + \frac{1}{2} \kappa^2) t} \\ \partial_{BB} f &= \kappa^2 W e^{-\kappa B_t - (r + \frac{1}{2} \kappa^2) t} & \partial_{wB} f &= -\kappa e^{-\kappa B_t - (r + \frac{1}{2} \kappa^2) t} & \partial_{ww} f &= 0 \\ d[B]_t &= dt & d[W, B]_t &= \theta \sigma dt \end{aligned}$$

Substituting these into Ito's formula above gives,

$$df = e^{-\kappa B_t - (r + \frac{1}{2}\kappa^2)t} \left[-W \left(r + \frac{1}{2}\kappa^2 \right) dt + \{rW - c_t + \theta(\mu - r)\} dt \right. \\ \left. + \theta\sigma dB_t - W\kappa dB_t + \frac{W}{2}\kappa^2 dt - \theta\sigma\kappa dt \right]$$

Cancelling (using that $\kappa\sigma = (\mu - r)$) and rearranging we get

$$df + e^{-\kappa B_t - (r + \frac{1}{2}\kappa^2)t} c_t dt = e^{-\kappa B_t - (r + \frac{1}{2}\kappa^2)t} [\theta\sigma - W\kappa] dB_t$$

So

$$\zeta_t W_t + \int_0^t \zeta_s c_s ds = Df_t(W_t, B_t) + \int_0^t Dc_t e^{\kappa B_t - (r + \frac{1}{2}\kappa^2)t} dt$$

is a local-Martingale. Recall from stochastic integration theory that every positive local martingale is a supermartingale.

Doob's Martingale Convergence Theorem applied to Y_t gives

$$\zeta_0 w_0 = Y_0 \geq \mathbb{E}Y_\infty = \mathbb{E} \left[\int_0^\infty \zeta_s c_s ds \right]$$

Since $\zeta_t = e^{-\rho t} u'(c_t^*) = e^{-\rho t} (c_t^*)^{-R}$ and by the definition of $V(w_0)$:

$$\mathbb{E}_{w_0} \left[\int_0^\infty \zeta_s c_s^* ds \right] = \mathbb{E}_{w_0} \left[\int_0^\infty e^{-\rho t} (c_t^*)^{1-R} ds \right] = (1 - R)V(w_0) = \gamma w_0^{1-R} = \zeta_0 w_0$$

The last equality holds since $\zeta_0 = (c_0^*)^{-R}$ and $c_s^* = \gamma^{1/R} W_s^*$.

Combining the last equality and the inequality before that, we see that

$$\mathbb{E} \left[\int_0^\infty (c_t - c_t^*) \zeta_t dt \right] \leq 0.$$

Applying this to (1.20) we see that

$$\mathbb{E} \left[\int_0^\infty e^{-\rho t} u(c_t) dt \right] \leq \mathbb{E} \left[\int_0^\infty e^{-\rho t} u(c_t^*) dt \right]$$

and, as required, c_t^* is optimal. \square

Merton Portfolio Optimization with Multiple Assets

We now note how the above results extend to the case where there aren't many assets. Now suppose that there are d assets that can be invested in. These obey the Stochastic Differential Equation

$$dS_t^i = S_t^i \left\{ \sum_{j=1}^N \sigma^{ij} dB_t^j + \mu^i dt \right\}, \quad i = 1, \dots, d$$

where B_t^j is an independent Brownian motion for each $j = 1, \dots, N$.

Wealth now evolves according to the SDE

$$dW_t = r(W_t - \mathbf{n}_t \cdot \mathbf{S}_t) dt + \mathbf{n}_t \cdot d\mathbf{S}_t - c_t dt$$

where $\mathbf{n}_t = (n_t^i : i = 1, \dots, d)$ gives the amount of each asset $\mathbf{S}_t = (S_t^i : i = 1, \dots, d)$ held in the portfolio at time t . Also we define $\boldsymbol{\theta}_t = (n_t^i S_t^i : i = 1, \dots, d)$ as the wealth in each asset. As given in Def 101, our task is to maximize the objective

$$V(w) = \max_{(\mathbf{n}_t, c_t)_{t \geq 0} \in \mathcal{P}(w_0)} \mathbb{E} \left[\int_0^\infty e^{-\rho t} u(c_t) dt \right].$$

We now proceed through exercises that are very similar to the case with a single risky asset. We go through the proofs somewhat quickly.

Lemma 9. *Show that the HJB equation for the Merton Problem can be written as*

$$0 = \max_c \{u(c) - c \partial_w V\} + \max_{\boldsymbol{\theta}} \left\{ \boldsymbol{\theta} \cdot (\boldsymbol{\mu} - \mathbf{r}) \partial_w V + \frac{1}{2} |\boldsymbol{\sigma} \boldsymbol{\theta}|^2 \partial_{ww} V \right\} - \rho V + r w \partial_w V.$$

where $\mathbf{r} = (r : i = 1, \dots, d)$.

Proof. The proof follows more-or-less identically to Prop 102. Note that in this case Ito's formula applied to $V(W_t)$ gives

$$dV(W_t) = \partial_w V(W_t) dW_t + \frac{1}{2} \partial_{ww} V(W_t) d[W]_t$$

where

$$\begin{aligned} dW_t &= \left(rW_t - \underbrace{r \mathbf{n}_t \cdot \mathbf{S}_t}_{\boldsymbol{\theta} \cdot \mathbf{r}} \right) dt + \underbrace{\mathbf{n}_t \cdot d\mathbf{S}_t}_{\boldsymbol{\theta}^\top [\boldsymbol{\sigma} d\mathbf{B}_t + \boldsymbol{\mu} dt]} - c_t dt = (rW_t + \boldsymbol{\theta}_t (\boldsymbol{\mu} - \mathbf{r}) - c_t) dt + \boldsymbol{\theta}_t^\top \boldsymbol{\sigma} d\mathbf{B}_t \\ d[W]_t &= \sum_{ij} (\boldsymbol{\theta}_t^\top \boldsymbol{\sigma})_i (\boldsymbol{\theta}_t^\top \boldsymbol{\sigma})_j d[B_t^i, B_t^j] = |\boldsymbol{\theta}_t \boldsymbol{\sigma}|^2 dt. \end{aligned}$$

Thus

$$dV(W_t) = \left[(rW_t + \boldsymbol{\theta}_t(\boldsymbol{\mu} - \mathbf{r}) - c_t) \partial_w V(W_t) + \frac{1}{2} |\boldsymbol{\theta}_t \sigma|^2 \partial_{ww} V(W_t) \right] dt + \boldsymbol{\theta}_t^\top \sigma dB_t.$$

This is the drift term applied in the HJB equation. Thus recalling that

$$0 = \max_{\text{actions}} \{ \text{"instantaneous cost"} + \text{"Drift term from Ito's Formula"} \}.$$

This gives the require HJB equation. \square

Lemma 10. *Show the optimal asset portfolio in the HJB equation is given by*

$$\boldsymbol{\theta}^* = -\frac{\partial_w V}{\partial_{ww} V} (\sigma \sigma^\top)^{-1} (\boldsymbol{\mu} - \mathbf{r})$$

and

$$\max_{\boldsymbol{\theta}} \left\{ \boldsymbol{\theta} \cdot (\boldsymbol{\mu} - \mathbf{r}) \partial_w V + \frac{1}{2} (\boldsymbol{\theta}^\top \sigma^\top \sigma \boldsymbol{\theta}) \partial_{ww} V \right\} = -\frac{1}{2} |\boldsymbol{\kappa}|^2 \frac{(\partial_w V)^2}{\partial_{ww} V}$$

Proof. Considering Lemma 9 we have that

$$\max_{\boldsymbol{\theta}} \left\{ \boldsymbol{\theta} \cdot (\boldsymbol{\mu} - \mathbf{r}) \partial_w V + \frac{1}{2} (\boldsymbol{\theta}^\top \sigma^\top \sigma \boldsymbol{\theta}) \partial_{ww} V \right\} \implies (\boldsymbol{\mu} - \mathbf{r}) \partial_w V + \partial_{ww} V (\sigma^\top \sigma) \boldsymbol{\theta}^* = 0.$$

Solving for $\boldsymbol{\theta}^*$ and substituting back into the maximization gives the answer. \square

Lemma 11. *Show that for a CRRA utility the optimal solution to the HJB equation is given by*

$$\boldsymbol{\theta}^* = \frac{w}{R} (\sigma \sigma^\top)^{-1} (\boldsymbol{\mu} - \mathbf{r}), \quad c^* = \gamma^{-\frac{1}{R}} w$$

where

$$\gamma^{-\frac{1}{R}} = R^{-1} \left\{ \rho + (R-1) \left(r + \frac{1}{2} \frac{|\boldsymbol{\kappa}|^2}{R} \right) \right\} \quad \boldsymbol{\kappa} = \sigma^{-1} (\boldsymbol{\mu} - \mathbf{r}).$$

Proof. 11 (In this proof when we refer to Prop 103 we mean that the argument which was applied in the single-asset setting is identical in the multiple asset setting.)

By Prop 103a)

$$V(w) = \gamma \frac{w^{1-R}}{1-R}$$

for some constant γ . Differentiating twice gives

$$\boldsymbol{\theta}^* = -\frac{\partial_w V}{\partial_{ww} V}(\sigma\sigma^\top)^{-1}(\boldsymbol{\mu} - \mathbf{r}) = \frac{w}{R}(\sigma\sigma^\top)^{-1}(\boldsymbol{\mu} - \mathbf{r}).$$

By Prop 103b), $c^* = \gamma^{-\frac{1}{R}}w$. Substituting these solutions into the HJB equation gives

$$\begin{aligned} 0 &= \max_c \{u(c) - c\partial_w V\} + \max_{\boldsymbol{\theta}} \left\{ \boldsymbol{\theta} \cdot (\boldsymbol{\mu} - \mathbf{r})\partial_w V + \frac{1}{2}|\sigma\boldsymbol{\theta}|^2\partial_{ww} V \right\} - \rho V + rw\partial_w V \\ &= \frac{R}{1-R}\gamma^{1-\frac{1}{R}}w^{1-R} + \frac{1}{2}|\boldsymbol{\kappa}|^2 \cdot \frac{\gamma}{R}w^{1-R} - \rho\gamma\frac{w^{1-R}}{1-R} + r\gamma w^{1-R} \end{aligned}$$

Rearranging and solving for γ gives the required solution for γ^* . \square

Def 105 (Merton Portfolio and Market Price Risk Vector). *As given above,*

$$\boldsymbol{\theta}^* = \frac{w}{R}(\sigma\sigma^\top)^{-1}(\boldsymbol{\mu} - \mathbf{r}),$$

is called the Merton Portfolio and

$$\boldsymbol{\kappa} = \sigma^{-1}(\boldsymbol{\mu} - \mathbf{r}).$$

is called the Market Price Risk Vector.

Dual value function approach

We could solve the CRRA utility case because it had a special structure. We now give a method for solving for general utilities $u(t, c)$.

Here we assume that $u(t, c)$ is continuous in t and c , concave in c and satisfies

$$\lim_{c \rightarrow \infty} u'(t, c) = 0$$

The HJB equation for the Merton problem is

$$0 = \max_{\theta, c} \left\{ u(t, c) + \partial_t V + (rw + \boldsymbol{\theta} \cdot (\boldsymbol{\mu} - \mathbf{r}) - c)\partial_w V + \frac{1}{2}|\sigma^\top \boldsymbol{\theta}|^2\partial_{ww} V \right\}.$$

We take the LF transform of u ,

$$u^*(t, z) = \max_c \{u(t, c) - zc\}$$

Further we define

$$J(t, z) = V(t, w) - wz$$

where w is such that $z = \partial_w V(t, w)$.

Thrm 106. *The HJB equation can be written as*

$$0 = u^*(t, c) + \partial_t J - rz\partial_z J + \frac{1}{2}|\kappa|^2 z^2 \partial_{zz} J$$

Moreover if we suppose that $u(t, x) = e^{-\rho t} u(x)$, for $u(x)$ concave and increasing, the HJB equation becomes

$$0 = u^*(y) - \rho j(y) + (\rho - r)yj'(y) + \frac{1}{2}|\kappa|^2 y^2 j''(y)$$

Noticed in the first HJB equation above that we have got rid of the maximization and in the second we have a linear ODE, which can be solved using standard methods.

Proof. First we will show that

$$\partial_z J = -w, \quad \partial_{zz} J = -\frac{1}{\partial_{ww} V}, \quad \partial_t J = \partial_t V \quad (1.21)$$

We can ignore the dependence of t for the first two expressions i.e. take $J(t, z) = J(z)$. Now $J(z) = V((V')^{-1}(z)) - z(V')^{-1}(z)$, so

$$J'(z) = \frac{d}{dz}(V')^{-1}(z) \cdot \underbrace{V'((V')^{-1}(z))}_{=z} - z \frac{d}{dz}(V')^{-1}(z) - (V')^{-1}(z) = -(V')^{-1}(z) = -w$$

and

$$J''(z) = -\frac{d}{dz}(V')^{-1}(z) = -\frac{1}{V''((V')^{-1}(z))} = -\frac{1}{V''(w)}.$$

Now reintroducing dependence on t ,

$$\frac{\partial J}{\partial t} = \partial_t V(t, w) + \frac{dw}{dt} \underbrace{\partial_w V}_{=z} - \frac{dw}{dt} z = \partial_t V$$

This gives the required derivatives in (1.21).

Substituting the expressions in (1.21), the HJB equation is

$$\begin{aligned} 0 &= \max_c \{u(t, z) - c\partial_w V\} + \max_{\theta} \left\{ \theta \cdot (\mu - r)\partial_w V + \frac{1}{2}|\sigma\theta|^2 \partial_{ww} V \right\} + \partial_t V + rw\partial_w V \\ &= u^*(t, \partial_w V) - \frac{1}{2}|\kappa|^2 \frac{(\partial_w V)^2}{\partial_{ww} V} + \partial_t V + rw\partial_w V \\ &= u^*(t, z) + \frac{1}{2}|\kappa|^2 z^2 \partial_{zz} J + \partial_t J - rz\partial_z J. \end{aligned}$$

Now if we suppose that $u(t, x) = e^{-\rho t} u(x)$, for $u(x)$ concave and increasing, then by the same argument as Prop 103a) we have that

$$V(t, w) = e^{-\rho t} V(w).$$

Defining $j(z) = V(w) - wz$ where w is such that $z = \partial_w V(t, w)$, the following are straightforward calculations:

$$\begin{aligned} J(t, z) &= e^{-\rho t} j(y), & \partial_t J &= -\rho e^{-\rho t} j(y) + \rho e^{-\rho t} y j'(y) \\ \partial_z J &= j'(y), & \partial_{zz} J &= e^{\rho t} j''(y) \end{aligned}$$

where $y = ze^{\rho t}$. Now substituting these terms into the HJB equation gives the result:

$$0 = \underbrace{u^*(t, z)}_{=e^{-\rho t} u^*(y)} + \underbrace{\partial_t J}_{=-\rho e^{-\rho t} j(y) + \rho e^{-\rho t} y j'(y)} - \underbrace{rz \partial_z J}_{re^{-\rho t} y j'(y)} + \underbrace{\frac{1}{2} |\kappa|^2 z^2 \partial_{zz} J}_{\frac{1}{2} |\kappa|^2 y^2 e^{-\rho t} j''(y)}.$$

□

Exercises

Ex 107. We consider the standard Merton investment problem. We assume that utility is derived from both consumption and wealth according to the function

$$u(c, w) = \frac{w^\alpha c^\beta}{1 - R}$$

for α and β positive constants such that $1 - R = \alpha + \beta$. Show that the solution to the HJB equation is of the form

$$V(w) = A \frac{w^{1-R}}{1 - R}$$

and write down the HJB equation for this problem and use it to find the constant A .

Ex 108. We consider the standard Merton investment problem. We assume that utility has the form

$$u(c) = -\frac{1}{R} e^{-Rc}.$$

for $R > 0$. Argue that the HJB equation (as a function of wealth and time) has a solution of the form

$$V(t, w) = -Ae^{-\rho t} e^{-rRw}$$

for some positive constant A and that for this there is constant amount of wealth kept in the risky asset.

Ex 109. We consider the Merton investment problem but now the interest rate can vary. Wealth ($W_t : t \geq 0$) satisfies $W_0 = w$ and obeys the stochastic differential equation

$$dW_t = \{r_t W_t + (\mu - r_t)\theta_t - c_t\}dt + \theta_t \sigma dB_t.$$

and the interest rate obeys the stochastic differential equation

$$dr_t = \beta(\bar{r} - r_t)dt + \sigma_r dB_t^r$$

where β , \bar{r} and σ_r are fixed parameters and B_t^r is a standard Brownian motion with covariation $[B_t^r, B_t] = \eta t$.

We maximize the utility of a CRRA utility function:

$$V(w, r) = \max_{(\theta_s, c_s)_{s \geq 0}} \mathbb{E} \left[\int_0^\infty e^{-\rho s} u(c_s) ds \mid W_0 = w, r_0 = r \right], \quad \text{with} \quad u(c_s) = \frac{c_s^{1-R}}{1 - R}.$$

Show that,

$$V(w, r) = \gamma(r) \frac{w^{1-R}}{1-R}$$

for some function $\gamma(r)$, and analyse the HJB equation to find the differential equation that the function $\gamma(r)$ must satisfy.

Chapter 2

Stochastic Approximation

2.1 Lyapunov Functions

Lyapunov functions are an extremely convenient device for proving that a dynamical system converges.

- For some continuous function $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$, we suppose $x(t)$ obeys the differential equation

$$\frac{dx}{dt} = f(x(t)), \quad t \in \mathbb{R}_+. \quad (2.1)$$

- A Lyapunov function is a continuously differentiable function $L : \mathbb{R}^n \rightarrow \mathbb{R}$ with unique minimum at x^* such that

$$f(x) \cdot \nabla L(x) < 0, \quad \forall x \neq x^*. \quad (2.2)$$

- We add the additional assumption that $\{x : L(x) \leq l\}$ is a compact set for every $l \in \mathbb{R}$.

Theorem 8. *If a Lyapunov exists (2.2) for differential equation (2.1) then $L(x(t)) \searrow L(x^*)$ as $t \rightarrow \infty$ and*

$$x(t) \xrightarrow[t \rightarrow \infty]{} x^*.$$

Proof. Firstly,

$$\frac{dL(x(t))}{dt} = f(x(t)) \cdot \nabla L(x(t)) < 0.$$

So $L(x(t))$ is decreasing. Suppose it decreases to \tilde{L} . By the Fundamental Theorem of Calculus

$$\tilde{L} - L(x(t)) = \int_t^\infty \frac{dL(x(s))}{ds} ds \xrightarrow[t \rightarrow \infty]{} 0 \quad (2.3)$$

Thus we can take a sequence of times $\{s_k\}_{k=1}^\infty$ such that $\frac{dL(x(s_k))}{dt} \rightarrow 0$ as $s_k \rightarrow \infty$. As $\{x : L(x) < L(x(0))\}$ is compact, we can take a subsequence of times $\{t_k\}_{k=1}^\infty \subset \{s_k\}_{k=1}^\infty$, $t_k \rightarrow \infty$ such that $\{x(t_k)\}_{k=0}^\infty$ converges. Suppose it converges to \tilde{x} . By continuity,

$$0 = \lim_{k \rightarrow \infty} \frac{dL(x(t_k))}{dt} = \lim_{k \rightarrow \infty} f(x(t_k)) \cdot \nabla L(\tilde{x}) = f(\tilde{x}) \cdot \nabla L(\tilde{x}).$$

Thus by definition $\tilde{x} = x^*$. Thus $\lim_{t \rightarrow \infty} L(x(t)) = L(x^*)$ and thus by continuity of L at x^* we must have $x(t) \rightarrow x^*$. \square

- One can check this proof follows more-or-less unchanged if x^* , the minimum of L , is not unique.

La Salle's Invariance Principle

We generalize and strengthen the Lyapunov convergence result in Theorem Lya:ConvThrm. Here we no longer consider convergence to a unique global minimum of $L(x)$. Instead, we find convergence to points

$$\mathcal{X}^* = \{x : g(x) \cdot \nabla L(x) = 0\}$$

The set \mathcal{X}^* is called the set of *invariant points*. Notice, under the conditions of Theorem 8, $\mathcal{X}^* = \{x^*\}$. In general, \mathcal{X}^* contains all local [and global] minima of $L(x)$. If the dynamics exclude invariant points which are non-local minima [for instance by taking $g(x) = -\eta \nabla L(x)$] then \mathcal{X}^* is exactly the set of local minima. The result which proves convergence to invariant points is called *Salle's Invariance Principle*.

We assume the following

- The process $x(t)$, $t \in \mathbb{R}_+$ obeys the o.d.e.

$$\frac{dx}{dt} = g(x(t))$$

for $g : \mathbb{R}^d \rightarrow \mathbb{R}^d$ a continuous function.

- $x(0) \in \mathcal{X}$ where \mathcal{X} is a compact set such that if $x(0) \in \mathcal{X}$ then $x(t) \in \mathcal{X}$ for all t .
- The Lyapunov function $L : \mathbb{R}^d \rightarrow \mathbb{R}_+$ is a continuously differentiable function such that

$$g(x) \cdot \nabla L(x) \leq 0, \quad \forall x \in \mathcal{X}$$

and we let

$$\mathcal{X}^* = \{x \in \mathcal{X} : g(x) \cdot \nabla L(x) = 0\}.$$

La Salle's Invariance Principle is as follows¹

Theorem 9 (La Salle's Invariance Principle). *As $t \rightarrow \infty$, $x(t)$ converges to \mathcal{X}^* uniformly over initial conditions $x(0) \in \mathcal{X}$. That is: $\forall x(0) \in \mathcal{X} \forall \epsilon > 0 \exists T > 0$ such that*

$$\min_{x^* \in \mathcal{X}^*} |x(T) - x^*| < \epsilon.$$

¹The original result of La Salle proves pointwise convergence rather than uniform convergence, other than this the proof closely follows La Salle's argument.

Proof. If $x(0)$ belongs to a compact set then $L(x(0)) \leq l$ for some $l \geq 0$. For $\delta > 0$, we let $\mathcal{X}^*(\delta) = \{x \in \mathcal{X} : -g(x) \cdot \nabla L(x) < \delta\}$. Since $dL/dt = g(x) \cdot \nabla L(x)$ and $L(x_0) \leq l$, for time $T > l/\delta$ it must be that $x(t) \in \mathcal{X}^*(\delta)$ for all $t \geq T$. [This is the main part of the argument completed.]

It is reasonable to assume that $x(t) \in \mathcal{X}^*(\delta)$ for δ suitably small then $x(t)$ must be close to \mathcal{X}^* . This is true and to show this we prove the following claim: for $\epsilon > 0 \exists \delta > 0$ such that if $x(t) \in \mathcal{X}^*(\delta)$ then $|x(T) - x^*| < \epsilon$ for some $x^* \in \mathcal{X}^*$. The proof is a fairly standard analysis argument: suppose the claim is not true, then there exists a sequence x_n such that $-g(x_n) \cdot \nabla L(x_n) < 1/n$ and $|x_n - x^*| \geq \epsilon$ for all $x^* \in \mathcal{X}^*$. Since \mathcal{X} is compact $x_{n_k} \rightarrow x_\infty$ over some subsequence $\{n_k\}_{k \in \mathbb{N}}$. By continuity $g(x_\infty) \cdot \nabla L(x_\infty) = 0$ and $|x_\infty - x^*| \geq \epsilon$ for all $x^* \in \mathcal{X}^*$, which is a contradiction since $g(x_\infty) \cdot \nabla L(x_\infty) = 0$ means $x_\infty \in \mathcal{X}^*$. This contradiction thus shows that there exists a value n^* such that for all x such that $-g(x) \cdot \nabla L(x) < \delta := 1/n^*$ implies $\min_{x^*} |x - x^*| < \epsilon$. \square

Remark 110. We assume $x(t) \in \mathcal{X}$ for some compact set \mathcal{X} . Notice a natural condition that implies compactness is that

$$\liminf_{|x| \rightarrow \infty} L(x) = \infty.$$

Specifically this implies that $\mathcal{X}_l := \{x : L(x) \leq l\}$ is compact and by assumption that $L(x)$ is decreasing, if $x(0) \in \mathcal{X}_l$ then $x(t) \in \mathcal{X}_l$ for all $t \in \mathbb{R}_+$.

Convex Lyapunov Functions

Next if we assume a bit more about $L(x)$ we can ask more about the rate of convergence. We assume that

- $L(x)$ is convex, $\min_x L(x) = 0$
- $\frac{dx}{dt} = -\gamma_t \nabla L(x(t))$
- $\|x_t\| \leq D$ for some D

Note, before we proceed recall that $L(x)$ is a convex function iff

$$\nabla L(x)(y - x) \leq L(y) - L(x) \leq \nabla L(y)(y - x).$$

Theorem 10. *Given the assumptions itemized above,*

$$L(\bar{x}_T) \leq \frac{D}{T\gamma_T}$$

where $\bar{x}_T = \frac{1}{T} \int_0^T x_t dt$.

Proof. By Jensen's inequality,

$$L(\bar{x}_T) \leq \frac{1}{T} \int_0^T L(x(t)) dt.$$

So we analyse the integral of $L(x(t))$.

$$\int_0^T L(x(t)) dt = \int_0^T L(x(t)) - L(x^*) dt \leq \int_0^T \nabla L(x(t))(x(t) - x^*) dt$$

Notice that since $\frac{dx}{dt} = -\eta_t \nabla L(x(t))$ we have

$$-\frac{1}{2\eta_t} \frac{d}{dt} \|x(t) - x^*\|^2 = \nabla L(x(t))(x(t) - x^*).$$

Substituting this into the integral above gives

$$\begin{aligned} \int_0^T L(x(t)) dt &\leq - \int_0^T \frac{1}{2\eta_t} \frac{d}{dt} \|x(t) - x^*\|^2 dt \\ &= - \left[\frac{1}{2\eta_t} \|x(t) - x^*\|^2 \right]_0^T + \int_0^T \frac{1}{2} \|x(t) - x^*\|^2 \frac{d\eta^{-1}}{dt} dt \\ &\leq \frac{D}{2\eta_T} + \frac{D}{2\eta_T} = \frac{D}{\eta_T} \end{aligned}$$

Finally, applying this to our Jensen bound on $L(\bar{x}_T)$ gives

$$L(\bar{x}_T) \leq \frac{D}{T\eta_T}.$$

□



Contractions.

For Markov decision processes working with contractions is important since the Bellman operator is a contraction in the $\|\cdot\|_\infty$ norm. Thus we give some Lyapunov convergence results in this case.

Recall that $F : \mathcal{R}^d \rightarrow \mathbb{R}^d$ is a contraction if, for $\beta \in (0, 1)$ it holds that

$$\|F(x) - F(y)\|_p \leq \beta \|x - y\|_p$$

in this section we let

$$\|x\|_p := \left(\sum_{i=1}^d w_i |x_i|^p \right)^{\frac{1}{p}}$$

for weights $w_i > 0$ and p such that $1 \leq p \leq \infty$.

Theorem 11. *If we suppose that $x(t)$ obeys the ODE*

$$\frac{dx}{dt} = F(x) - x$$

where $F(x)$ is contraction with fixed point x^* then $L(x) = \|x - x^*\|_p$ is a Lyapunov function and $x(t) \rightarrow x^*$.

Proof. We let $\text{sgn}(x)$ be the sign function that is $\text{sgn}(x) = +1$ if $x > 0$, $\text{sgn}(x) = -1$ if $x < 0$ and $\text{sgn}(x) = 0$ if $x = 0$. For now we assume $1 \leq p < \infty$ (shortly we will extend to allow $p = \infty$).

By the chain rule

$$\begin{aligned} \frac{dL}{dt} &= \sum_i \frac{\partial L}{\partial x_i} \frac{dx_i}{dt} \\ &= \underbrace{\left(\sum_j w_j |x_j - x^*|^p \right)^{\frac{1-p}{p}}}_{\|x - x^*\|_p^{1-p}} \sum_i w_i \text{sgn}(x_i - x_i^*) |x_i - x_i^*|^{1-p} \underbrace{(F_i(x) - x_i)}_{F_i(x) - F_i(x^*) - (x_i - x_i^*)} \\ &= \|x - x^*\|_p^{1-p} \left[\underbrace{\sum_i w_i \text{sgn}(x_i - x_i^*) |x_i - x_i^*|^{p-1} (F_i(x) - F_i(x^*))}_{\leq \|x - x^*\|_p^{p-1} \|F(x) - F(x^*)\|_p} \right] - \|x - x^*\|_p \\ &\leq \|F(x) - F(x^*)\|_p - \|x - x^*\|_p \end{aligned}$$

Therefore integrating the above gives

$$\|x(t) - x^*\|_p - \|x(s) - x^*\|_p \leq \int_s^t \|F(x(u)) - F(x^*)\|_p - \|x(u) - x^*\|_p du$$

Notice we can allow $p = \infty$ in the above expression since $\|z\|_p \rightarrow \|z\|_\infty$ uniformly on compacts. So applying the fact F is a contraction we gain that

$$\|x(t) - x^*\|_p - \|x(s) - x^*\|_p \leq -(1 - \beta) \int_s^t \|x(u) - x^*\|_p du$$

which ensures convergence of $x(t)$ to x^* . □

Exponential Convergence

We now place some assumptions where we can make further comments about rates of convergence.

Theorem 12. *If we further assume that f and L satisfy the conditions*

1. $f(x) \cdot \nabla L(x) \leq -\gamma L(x)$ for some $\gamma > 0$.
2. $\exists \alpha, \eta > 0$ such that $\alpha \|x^* - x\|^\eta \leq L(x) - L(x^*)$.
3. $L(x^*) = 0$.

then there exists a constants $\kappa, K > 0$ such that for all $t \in \mathbb{R}_+$

$$\|x(t) - x^*\| \leq Ke^{-\kappa t}. \quad (2.4)$$

Proof.

$$\frac{dL(x(t))}{dt} = f(x(t)) \cdot \nabla L(x(t)) \leq -\gamma L(x(t)).$$

So long as $x(t) \neq x^*$, $L(x(t)) > 0$, thus dividing by $L(x(t))$ and integrating gives

$$\log L(x(t)) - \log L(x(0)) = \int_0^t \frac{1}{L(x(s))} \frac{dL(x(s))}{dt} ds \leq -\gamma t$$

Rearranging gives

$$L(x(t)) \leq L(x(0))e^{-\gamma t}$$

This gives exponential convergence in $L(x(t))$ and quick application of the bound in the 2nd assumption gives

$$\|x(t) - x^*\| \leq \frac{L(x(0))}{\alpha} e^{-\frac{\gamma}{\eta} t}.$$

□

- We can assume the 2nd assumption only holds on a ball around x^* . We have convergence from Theorem 8, so when $x(t)$ is such that assumption 2 is satisfied we can then apply the same analysis for an exponential convergence rate. Ensuring the 2nd assumption locally is more easy to check, eg. check L is positive definite at x^* .

Linear-Quadratic Lyapunov Control

We consider one of the key examples: a linear dynamic and a quadratic Lyapunov function. We give a very brief treatment here. A fuller text considering various design principles is Khalil [27].

We consider $x(t) \in \mathbb{R}^d$ which satisfies the linear differential equation

$$\frac{dx}{dt} = Ax,$$

Here A is a $d \times d$ matrix. We assume that $\det(A) \neq 0$.

Remark 111. Note the condition $\det(A) \neq 0$ excludes the possibility of multiple equilibria. It also allow us to easily extend to the case where

$$\dot{x} = Ax + b$$

for a constant vector b . Since $Ax^* = b$ for some x^* and thus we consider the equivalent differential equation $\frac{d}{dt}\tilde{x} = A\tilde{x}$ for $\tilde{x}(t) = x(t) - x^*$.

The only equilibrium of the above equation is $x^* = 0$. The key property that determines convergence is if the matrix is Hurwitz:

Definition 5 (Hurwitz Matrix). The non-singular matrix A is said to Hurwitz if all eigenvalues of A have (strictly) negative real part. I.e. $\text{Re}(\alpha) < 0$ for all eigenvalues α .

Theorem 13. If the matrix A is Hurwitz then $x(t) \rightarrow 0$ as $t \rightarrow \infty$.

Remark 112. The converse to the result also holds. Specifically if $x(t) \rightarrow 0$ for all initial conditions then the matrix A must be Hurwitz. We don't cover this part, but the proof should give a good indication of why this is true.

Proof of Theorem 13. It is not too hard to show (this is exercise Ex116 below) that the solution to the above equation is uniquely given by the matrix exponential

$$x(t) = e^{At}x(0)$$

where

$$e^{At} = I + tA + \frac{t^2}{2!}A^2 + \dots$$

We can write A in Jordan normal form $A = UJU^{-1}$. Recall that J is a block diagonal matrix. I.e. The non-zero entries are

$$J = \begin{pmatrix} J_1(\alpha_1) & & & \\ & J_2(\alpha_2) & & \\ & & \ddots & \\ & & & J_n(\alpha_n) \end{pmatrix} \quad \text{where} \quad J_i(\alpha_i) = \begin{pmatrix} \alpha_i & 1 & & \\ & \alpha_i & 1 & \\ & & \ddots & \ddots \\ & & & \alpha_i & 1 \\ & & & & \alpha_i \end{pmatrix}.$$

Here α_i , $i = 1, \dots, n$ are the set of eigenvalues of A . We are interested in powers A^k . Notice then that $A^k = UJ^kU^{-1}$ and thus

$$e^{At} = Ue^{Jt}U^{-1}.$$

Further a quick check gives that

$$[J(\alpha)^k]_{ij} = \binom{k}{j-i} \alpha^{k-j+i}.$$

Thus

$$[e^{J(\alpha)t}]_{ij} = \sum_{k=0}^{\infty} \frac{t^k}{k!} [J(\alpha)^k]_{ij} = \sum_{k=j-i}^{\infty} \frac{1}{(k-j+i)!(j-i)!} \alpha^{k-j+i} t^{k-j+i} t^{j-i} = \frac{t^{j-i}}{(j-i)!} e^{\alpha t}.$$

From this we see that

$$e^{At} = \sum_{l=1}^n \sum_{j=1}^{m_l} t^j e^{\alpha_l t} R_{lj} \quad (2.5)$$

where R_{lj} is a matrix for each l and j . (Here m_l is the multiplicity of the l -th eigenvalue). Notice if $\operatorname{Re}(\alpha_l) \leq -c < 0$ for each eigenvalue α_l then $|e^{\alpha_l t}| < e^{-ct}$. Applying this to our solution for $x(t)$ gives

$$\|x(t)\| \leq Ct^m e^{-ct} \xrightarrow{t \rightarrow \infty} 0,$$

for some positive constant C . □

Remark 113. Note that for a Hurwitz matrix the eigenvalue closest to 0 determines the rate of convergence to equilibrium in (2.5). Notice that when the matrix is not Hurwitz in that $\operatorname{Re}(\alpha) > 0$ for some eigenvalue, then the terms in (2.5) can diverge. Thus we cannot expect convergence.

The previous analysis in Theorem 13 did not involve any Lyapunov function we now introduce this here. We consider Lyapunov functions of the form

$$L(x) = x^\top \Lambda x$$

where Λ is a positive definite matrix. A quick calculation gives

$$\frac{dL}{dt} = x^\top A^\top \Lambda x + x^\top \Lambda A x =: -x^\top R x$$

where $R := -A^\top \Lambda - \Lambda A$.

Definition 6 (Sylvester's equation). *For a matrices R and A , the equation*

$$-R = A^\top \Lambda + \Lambda A$$

is known as Sylvester's equation.

There are standardized numerical routines for solving Sylvester's equation, e.g. Matlab uses the Bartels–Stewart algorithm.

It is worth noting the following

Theorem 14. *If the solution to Sylvester's equation is positive definite then $x(t) \rightarrow 0$ as $t \rightarrow \infty$.*

The above result follows immediately from Theorem 8. Note further since $\|x\| \lambda_{\min}(M) \leq x M x \leq \lambda_{\max}(M) \|x\|$ for any positive definite matrix M then we can also apply Theorem 12 to give an exponential rate of convergence.

We also see that the (unique) solution to

Theorem 15. *If A is Hurwitz then for each positive definite R there exists a (unique) positive definite matrix Λ given by*

$$\Lambda = \int_0^\infty e^{A^\top t} R e^{A t} dt$$

such that

$$-R = A^\top \Lambda + \Lambda A \tag{2.6}$$

Remark 114. *We note that the converse to this result also holds. That is if we solve Sylvester's equation with some positive definite matrix then the matrix A must be Hurwitz.*

Proof. Notice if we take $\Lambda = \int_0^\infty e^{A^\top t} R e^{At} dt$ then

$$\begin{aligned} A^\top \Lambda + \Lambda A &= \int_0^\infty A^\top e^{A^\top t} R e^{At} + e^{A^\top t} R e^{At} A dt \\ &= \int_0^\infty \frac{d}{dt} e^{A^\top t} R e^{At} dt = \left[e^{A^\top t} R e^{At} \right]_{t=0}^\infty = -R \end{aligned}$$

So Λ is a solution to Sylvester's equation. It is possible to show that this Λ is well-defined, positive definite and is the unique solution to Sylvester's equation. (This is exercise Ex 117 below) \square

Remark 115. We note that the above analysis extends to non-linear o.d.e's in the sense that if

$$\frac{dx}{dt} = f(x)$$

for $f(x)$ is continuously differentiable with $f(0) = 0$ and $A = \nabla f(0)$ Hurwitz then $x(t) \rightarrow 0$ for some neighborhood of 0. (This is exercise Ex 118 below.

Ex 116. 1) Show that $e^{At}x(0)$ satisfies the o.d.e.

$$\frac{dx}{dt} = Ax.$$

2) Show that if there are two solutions $x_1(t)$ and $x_2(t)$ to the above o.d.e. then for $z(t) = x_1(t) - x_2(t)$

$$\frac{dz}{dt} = 0$$

and thus argue that there is a unique solution to the o.d.e.

Ex 117. We verify properties of the matrix $\Lambda = \int_0^\infty e^{A^\top t} R e^{At} dt$ for R positive definite:

1) [Positive Definite] For $\Lambda = \int_0^\infty e^{A^\top t} R e^{At} dt$ show that $x^\top \Lambda x = 0$ implies $e^{A^\top t} x = 0$ and thus argue that Λ is positive definite, when A is Hurwitz.

2) [Uniqueness] Suppose that Λ' is another solution to Sylvester's equation, show that

$$0 = \frac{d}{dt} \left\{ e^{A^\top t} (\Lambda - \Lambda') e^{At} \right\}$$

By noting that $e^{At} \rightarrow 0$ as $t \rightarrow \infty$, show that $\Lambda' = \Lambda$.

Ex 118 (Local stability). We prove the claims in Remark 115. We assume that

$$\frac{dx}{dt} = f(x)$$

for $f(x)$ is continuously differentiable with $f(0) = 0$ and $A = \nabla f(0)$ is Hurwitz.

1) Show (by a Taylor expansion) that

$$f(x) = Ax + g(x)$$

where $g(x)/\|x\|_2 \rightarrow 0$.

2) Let Λ be the solution to Sylvester's equation with $R = I$, show that

$$\frac{dL}{dt} = -\|x\|_2^2 + 2x^\top \Lambda g(x)$$

3) Noting that $\lambda_{\min}(\Lambda) > 0$, combine parts 1) and 2) to show that

$$\frac{dL}{dt} = -\|x\|_2^2 + o(\|x\|_2^2)$$

and thus argue that $x(t) \rightarrow 0$ in some neighborhood of $x = 0$.

References

The Lyapunov method goes back to Lyapunov in (1892) [34]. Extensions were considered by La Salle [30]. A widely used textbook treatment is Khalil [27]. Applications to internet congestion control are given by Srikant [44]. The convex Lyapunov function proof is the an o.d.e adaptation to the result of [58].

2.2 Robbins-Monro.

- Robbins Monro step rule.
 - Robbins-Siegmund Theorem.
 - Stochastic Gradient Descent.
 - Asynchronous Implementation.
-

We review a method for finding fixed points then extend it to slightly more general, modern proofs.

Often it is important to find a solution to the equation

$$0 = g(x^*)$$

by evaluating g at a sequence of points. For instance Newton's method would perform the updates $x_{n+1} = x_n - g(x_n)/g'(x_n)$. However, Robbins and Monro consider the setting where we cannot directly observe g but we might observe some random variable whose mean is $g(x)$. Thus we observe

$$y_n = g(x_n) + \epsilon_n \tag{2.7}$$

where ϵ_n is a random variable with mean zero and hope solve for $g(x) = 0$. Notice in this setting, even if we can find $g'(x)$, Newton's method may not converge. The key idea of Robbins and Monro is to use a schema where

$$x_{n+1} = x_n - \alpha_n y_n \tag{RM}$$

where we chose the sequence $\{\alpha_n\}_{n=0}^{\infty}$ so that

$$\sum_n \alpha_n = \infty, \quad \sum_n \alpha_n^2 < \infty.$$

Before proceeding here are a few different use cases:

- **Quartiles.** We want to find x such that $P(X \leq x) = p$ for some fixed p . But we can only sample the random variable X .
- **Regression.** We perform regression $g(x) = \beta_0 + \beta_1 x$, but rather than estimate β we want to know where $g(x) = 0$.

- **Optimization.** We want to optimize a convex function $f(x)$ whose gradient is $g(x)$. Assume that $f(x) = \sum_{k=1}^K f_k(x)$ for some large K . To find the optimum at $g(x) = 0$ we randomly sample (uniformly) $f_k(x)$ whose gradient, $g_k(x)$, is an bias estimate of $g(x)$.

The following result contains the key elements of the Robbins-Monro proof

Lem 119. Suppose that z_n is a positive sequence such that

$$z_{n+1} \leq z_n(1 - a_n) + c_n \quad (2.8)$$

where a_n and c_n are positive sequences such that

$$\sum_n a_n = \infty, \quad \text{and} \quad \sum_n c_n < \infty \quad (2.9)$$

then $\lim_{n \rightarrow \infty} z_n = 0$.

Proof. We can assume that equality holds, i.e., $z_{n+1} = z_n(1 - a_n) + c_n$. We can achieve this by increasing a_n or decreasing c_n in the inequality (2.8); neither of which effect the conditions on a_n and b_n , (2.9).

Now for all n we have the following lower-bound

$$-z_0 \leq z_n - z_0 = \sum_{k=0}^{n-1} (z_{k+1} - z_k) = \sum_{k=0}^{n-1} c_k - \sum_{k=0}^{n-1} a_k z_k$$

Since $\sum c_k < \infty$ it must be that $\sum a_k z_k < \infty$. Thus since both sums converge it must be that $\lim_n z_n$ converges. Finally since $\sum a_k = \infty$ and $\sum a_k z_k < \infty$ it must be that $\lim_n z_n = 0$. \square

An Easy Robbin's Monro Proof

The following lemma is a straight-forward proof based on the original argument of Robbins and Monro. This proof is intentionally not the most general but instead gives the key ideas.

We assume that $\sup_n \mathbb{E}[y_n] < \infty$. The main assumption that we make is that, for some $\kappa > 0$

$$(g(y) - g(x))^\top (y - x) \geq \kappa \|y - x\|^2 \quad (2.10)$$

Here are a couple of instances where this holds:

- $g(x)$ is the gradient of a strongly convex function $f(x)$.²
- $g : [x_{\min}, x_{\max}] \rightarrow [y_{\min}, y_{\max}]$ is differentiable real valued with $g(x_{\min}) < 0$, $g(x_{\max}) > 0$ and there is a unique point x^* such that $g(x^*) = 0$ and $g'(x^*) > 0$.

The first item shows that Robbins-Monro plays nicely with convex optimization problems. The second items shows however, that we don't need g to be the derivative of a convex function for the method to work.

Thrm 120 (Robbins-Monro). *If we chose x_n according to the Robbins-Monro step rule (RM) and we assume (2.10) then we have that*

$$\mathbb{E}[(x_n - x^*)^2] \xrightarrow{n \rightarrow \infty} 0$$

where here x^* satisfies $g(x^*) = 0$.

Proof. Let $z_n = \mathbb{E}[(x_n - x^*)^2]$, $e_n = \mathbb{E}[y_n^2]$ and $d_n = \mathbb{E}[(x_n - x^*)(g(x_n) - g(x^*))]$. Then we have

$$\begin{aligned} z_{n+1} &= \mathbb{E}(x_{n+1} - x_n + x_n - x^*)^2 \\ &= \mathbb{E}(x_{n+1} - x_n)^2 + 2\mathbb{E}[(x_{n+1} - x_n)(x_n - x^*)] + \mathbb{E}(x_n - x^*)^2 \\ &= \alpha_n^2 \mathbb{E}[y_n^2] - 2\alpha_n \mathbb{E}[g(x_n)(x_n - x^*)] + \mathbb{E}(x_n - x^*)^2 \\ &= \alpha_n^2 e_n - 2\alpha_n d_n + z_n \end{aligned}$$

Thus

$$d_n = \mathbb{E}[(x_n - x^*)(g(x_n) - g(x^*))] \geq \kappa \mathbb{E}[(x_n - x^*)(x_n - x^*)] = \kappa z_n.$$

Thus

$$z_{n+1} \leq z_n(1 - 2\alpha_n \kappa) + \alpha_n^2 e_n.$$

Now applying Lemma 119 gives the result. \square

²Note, if $g(x) = \nabla f(x)$ then (2.10) is the definition of $f(x)$ being strongly convex.

Additional Results*

We develop a few additional results. These can be skipped on first reading.

Projection. Notice that if we wish to constrain the vector x_n to a convex set \mathcal{X} , then we can do this. In particular, if we are looking for a point $x^* \in \mathcal{X}$ such that

$$(x - x^*)^\top g(x^*) \geq 0, \quad \forall x \in \mathcal{X},$$

(We are implicitly assuming such a point x^* exists. If $g(x)$ is the derivative of a convex function then x^* exists and this condition says all points x are higher up than x^* .) then we can adapt the Robbins-Monro scheme to consider

$$x_{n+1} = \Pi_{\mathcal{X}}(x_n - \alpha_n y_n) \tag{2.11}$$

where here $\Pi_{\mathcal{X}}(z)$ is the L_2 projection of z on the set \mathcal{X} . Notice that projections always reduce distances specifically Lemma 25 in the appendix proves

$$\|\Pi_{\mathcal{X}}(x) - \Pi_{\mathcal{X}}(y)\|_2 \leq \|x - y\|_2$$

The following proof is almost identical to Theorem 120 and we leave the proof as an exercise for the reader.

Thrm 121 (Projected Robbins-Monro). *If we chose x_n according to the Projected Robbins-Monro step rule (2.11) and we assume (2.10) then we have that*

$$\mathbb{E}[(x_n - x^*)^2] \xrightarrow{n \rightarrow \infty} 0$$

where here x^* satisfies $(x - x^*)^\top g(x^*) \geq 0 \forall x \in \mathcal{X}$.

For this projected algorithm to be practical we are implicitly assuming that we can efficiently calculate the projection on \mathcal{X} . This is possible for certain sets such as the positive orthant \mathbb{R}_+^d , hyperplanes $\mathcal{H} = \{x : n^\top x = c\}$ and, sometimes, simplexes $\mathcal{S} = \{x \in \mathbb{R}_+^d : n^\top x = c\}$ such as the probability simplex.

A recursion for finite time bounds. The following Lemma is also sometimes applied to get tighter bounds on convergence.

Lem 122. *Suppose that z_n is a positive sequence such that*

$$z_{n+1} \leq z_n(1 - a_n) + c_n$$

Then

$$z_{n+1} \leq z_0 \prod_{k=0}^n (1 - a_k) + \sum_{j=0}^n c_j \prod_{k=j+1}^n (1 - a_k)$$

Proof. The proof follows by repeated substitution

$$\begin{aligned} z_{n+1} &\leq z_n(1 - a_n) + c_n \leq (z_{n-1}(1 - a_{n-1}) + c_{n-1})(1 - a_n) + c_n \\ &\leq z_{n-1}(1 - a_{n-1})(1 - a_n) + c_n + c_{n-1}(1 - a_n) \\ &\vdots \\ &\leq z_0 \prod_{k=0}^n (1 - a_k) + \sum_{j=0}^n c_j \prod_{k=j+1}^n (1 - a_k) \end{aligned}$$

□

A Stochastic Recursion for Almost Sure Convergence. The following proposition is a Martingale version of the above result.

Prop 123 (Robbins-Siegmund Theorem). *If*

$$\mathbb{E}[Z_{n+1}|\mathcal{F}_n] \leq (1 - a_n + b_n)Z_n + c_n \quad (2.12)$$

for positive adaptive RVs Z_n, a_n, b_n, c_n such that with probability 1,

$$\sum_n a_n = \infty, \quad \sum_n b_n < \infty, \quad \text{and} \quad \sum_n c_n < \infty$$

then $\lim_{n \rightarrow \infty} Z_n = 0$.

Proof. The results is some manipulations analogous to the Robbins-Monro proof and a bunch of nice reductions to Doob's Martingale Convergence Theorem.

First note the result is equivalent to proving the result with $b_n = 0$ for all n . If we divide both sides of (2.12) by $\prod_{m=0}^n (1 + b_m)$ we get

$$\mathbb{E}[Z'_{n+1}|\mathcal{F}_n] \leq (1 - a'_n)Z'_n + c'_n,$$

where $a'_n = a_n/(1 + b_n)$, $c'_n = c_n/\prod_{m=0}^n (1 + b_m)$ and $Z'_n = Z_n/\prod_{m=0}^n (1 + b_m)$. Notice since $\sum b_n$ converges then so does $\prod(1 + b_n)$. Thus a'_n , c'_n and Z'_n have the same convergence properties as those required for a_n , c_n and Z_n . Thus, we now assume $b_n = 0$ for all n .

Now notice

$$Y_n = Z'_n + \sum_{k=0}^{n-1} a'_k Z'_k - \sum_{k=0}^{n-1} c'_k$$

is a super-martingale. We want to use Doob's Martingale convergence theorem; however, we need to apply a localization argument to apply this. Specifically, let $\tau_C = \inf\{n \geq 0 : \sum_{k=1}^n c'_k > C\}$. This is a stopping time. Notice

$$Y_{n \wedge \tau_C} \geq - \sum_{k=0}^{n \wedge \tau_C - 1} c'_k \geq -C.$$

So $Y_{n \wedge \tau_C}$ is a super-martingale and below by $-C$. Thus by Doob's Martingale Convergence Theorem, $\lim_{n \rightarrow \infty} Y_{n \wedge \tau_C}$ exists for each $C > 0$, and $\tau_C = \infty$ for some C , since $\sum c'_k < \infty$. Thus $\lim_{n \rightarrow \infty} Y_n$ exists.

Now notice

$$\sum_{k=1}^n c'_k - \sum_{k=1}^n a'_k Z'_k = Z'_{n+1} - Y_{n+1} \leq -Y_{n+1}.$$

So like in the last proposition, since $\lim Y_n$ and $\sum c'_k$ exists, we see that $\sum_{k=1}^{\infty} a'_k Z'_k$ converges. And thus Z'_{n+1} converges.

Finally since we assume $\sum_k a'_k = \infty$ and we know that $\sum_{k=1}^{\infty} a'_k Z'_k < \infty$ it must be that Z'_k converges to zero. \square

Example: Sequential Mean-Variance Portfolio Optimization

Here we consider a sequential version the classical mean-variance portfolio optimization of Markowitz. Here we assume that the mean and variance of the stock price change are unknown and we must learn these while constructing an optimal portfolio.

The set up is as follows. At time $t = 0$ there are stocks at prices $s = (s_0^0, s_0^1, \dots, s_0^d) \in \mathbb{R}^{d+1}$. Given $t = 1, 2, \dots$ at time $t + 1$ the are work

$$s_{t+1} = (\Delta_t^0 s_t^0, \dots, \Delta_t^d s_t^d)$$

where $\Delta_t^0, \Delta_t^1, \dots, \Delta_t^d$ are positive independent random variables representing the change in the stock prices. We define the vector μ and the matrix Σ by

$$\mu^i := \mathbb{E}[\Delta_t^i] \quad \text{and} \quad \Sigma_{ij} = \mathbb{E}[(\Delta_t^i - \mu^i)(\Delta_t^j - \mu^j)]$$

for $i, j = 1, \dots, d$. Also we $\mu^0 = (1 + r) = \Delta_t^0$ and so is a risk less asset.

We let θ^i be the proportion of wealth in stock i . We consider the objective

$$\max_{\theta} \quad \mu^\top \theta - \frac{\lambda}{2} \theta^\top \Sigma \theta \quad \text{subject to} \quad \sum_{i=1}^d \theta^i = 1, \quad \theta^i \geq 0.$$

Remark 124. Notice this is equivalent to solving the optimization

$$\max \quad \mu^\top \theta \quad \text{s.t.} \quad \theta^\top \Sigma \theta \leq \sigma, \quad \theta^\top \mathbf{1} = 1, \quad \theta \geq 0$$

for some σ and

$$\min \quad \theta^\top \Sigma \theta \quad \text{s.t.} \quad \mu^\top \theta \geq m \quad \theta^\top \mathbf{1} = 1 \text{ over } \theta \geq 0$$

for some m . (Basically the optimization is the Lagrangian for both the above optimizations.) Both of these are more standard representations of Mean-Variance portfolio optimization.

Now notice that the objective, above, can be rewritten as

$$\begin{aligned} \max \quad & \sum_i \theta^i \mu^i - \frac{\lambda}{2} \mathbb{E} \left[\sum_{ij} \theta^i (\Delta^i - \mu^i) (\Delta^j - \mu^j) \theta^j \right] \\ \text{subject to} \quad & \mu^i = \mathbb{E}[\Delta^i], \quad \sum_i \theta^i = 1 \end{aligned}$$

Thus applying projected Robbins-Monro to this we see that this can be solved with the scheme

$$\begin{aligned}\tilde{\theta}_{t+1}^i &= \theta_t^i + \alpha_t \left(\mu_t^i - \lambda \sum_j \theta_t^i (\Delta_t^i - \mu_t^i) (\Delta_t^j - \mu_t^j) \theta_t^j \right) \\ \theta_{t+1}^i &= \tilde{\theta}_{t+1}^i + 1 - \sum_j \tilde{\theta}_{t+1}^j \\ \mu_{t+1}^i &= \mu_t^i + \alpha_t (\Delta_t^i - \mu_t^i)\end{aligned}$$

Notice the second expression above corresponds to the projection step.

Stochastic Approximation Examples.**Ex 125.** Consider the Stochastic Differential Equation

$$d\theta_t = -\alpha_t g(\theta_t) dt + \alpha_t dB_t.$$

Suppose that for some unique θ^* that $(\theta^* - \theta)g(\theta) \leq -\|\theta - \theta^*\|^2/2$. Use Ito's formula to argue that $z_t = \mathbb{E}\|\theta_t - \theta^*\|^2/2$ obeys the differential equation

$$\frac{dz_t}{dt} \leq -\alpha_t z_t + \frac{1}{2} \alpha_t^2$$

Then show that

$$z_t - z_1 \leq e^{-\int_0^t \alpha_s ds} + \int_0^t \frac{1}{2} \alpha_s^2 e^{-\int_s^t \alpha_u du} ds.$$

Ex 126. Prove Theorem 121.

Rmk 127. A quick remark before proceeding with the solution. Note that the above SDE behaves very similarly to the Robbins-Monro step rule. Notice that θ_t behaves like the following process

$$\theta_t - \theta_0 = G\left(\int_0^t \alpha_s ds\right) + B\left(\int_0^t \alpha_s^2 ds\right)$$

where here $G(t)$ is a solution to the differential equation $\dot{\theta}_t = -g(\theta_t)$ and $B(t)$ is a standard Brownian motion. If we take $\alpha_s = 1/s^\gamma$ for $\gamma \in (0, 1]$ then integral in the Brownian term is finite, so the random part of the process eventually converges. While the integral in the G function goes to infinity, so we observe the entire sample path of $G(t)$ is explored. So we expect to converge to the stationary behaviour of the ODE $\dot{\theta} = -g(\theta)$. This point is quite informal. Basically the work of Kushner and Yin [CITE] argues this point in a somewhat more formal sense.

References

Robbin-Monro introduce the procedure was (unsurprisingly) introduced by Robbins and Monro [38] (A very readable paper). Stochastic approximation has grown enormously, see Krushner and Yin [28] for an excellent text on the topic. The discussion on finite time error is based on Bach and Moulines [2]. Asynchronous update section is based on reading Tsitsiklis [49] (and Bertsekas & Tsitsiklis [7]), here we apply a Robbins & Siegmund [39].

2.3 Stochastic Gradient Decent

Suppose that we have some function $F : \mathbb{R}^p \rightarrow \mathbb{R}$

$$F(\theta) = \mathbb{E}_X[f(X; \theta)]$$

that we wish to minimize. We suppose that the function $f(X; \theta)$ is known and so is its gradient $g(\theta; X)$, where $\mathbb{E}[g(\theta; X)] = G(\theta)$ is the gradient of $F(\theta)$. The difficulty is that we do not have direct access to the distribution of X , but we can draw random samples X_1, X_2, \dots . We can use the Robbins-Munro Schema to optimize $F(\theta)$. Specifically we take

$$\begin{aligned} \theta_{n+1} &= \theta_n - \alpha_n g_n(\theta_n) \\ &= \theta_n - \alpha_n G(\theta_n) + \alpha_n \epsilon_n \end{aligned} \tag{SGD}$$

where $g_n(\theta) = g(\theta; X_n)$ and $\epsilon_n = G(\theta) - g_n(\theta)$. The above sequence is often referred to as *Stochastic Gradient Descent*. We chose the sequence $\{\alpha_n\}_{n=0}^\infty$ so that

$$\sum_n \alpha_n = \infty, \quad \sum_n \alpha_n^2 < \infty.$$

(Note here we may assume that α_n is a function of previous parameters and observations $\theta_1, \dots, \theta_{n-1}$ and X_1, \dots, X_{n-1} .) We let $\|\cdot\|_2$ be the Euclidean norm. We can prove that convergence θ_n to the minimizer of $F(\theta)$.

Thrm 128 (Stochastic Gradient Descent). *Suppose that θ_n , $G(\cdot)$, and ϵ_n in Stochastic Gradient Descent (SGD) satisfy the following conditions*

1. $\exists \theta^*$ such that $\forall \theta, G(\theta) \cdot (\theta - \theta^*) \geq \mu \|\theta - \theta^*\|_2^2$
2. $\|G(\theta_n)\|_2^2 \leq A + B\|\theta_n\|_2^2$
3. $\mathbb{E}[\|\epsilon_n\|_2^2 | \mathcal{F}_n] \leq K$

Then $\lim_n \theta_n = \theta^$ where $\theta^* = \operatorname{argmin}_\theta F(\theta)$ and assuming α_n are deterministic then $\lim \mathbb{E}[\|\theta_n - \theta^*\|_2^2] = 0$*

Let's quickly review the conditions above. First consider Condition 1. Note condition 1 implies moving in the direction of θ^* always decreases the $F(\theta)$, so θ^* minimizes F . The statement $(G(\theta) - G(\phi)) \cdot$

$(\theta - \phi) \geq \mu \|\theta - \phi\|^2$ is equivalent to $F(\theta)$ being strongly convex. So this is enough to give Condition 1. Condition 2 can be interpreted as a gradient condition, or that the steps θ_n do not grow unboundedly. Condition 3 is natural given our analysis so far.

Proof.

$$\begin{aligned} & \|\theta_{n+1} - \theta^*\|_2^2 - \|\theta_n - \theta^*\|_2^2 \\ &= -\alpha_n G(\theta_n) \cdot (\theta_n - \theta^*) - \alpha_n \epsilon_n \cdot (\theta_n - \alpha_n G(\theta_n) - \theta^*) + \alpha_n^2 \|\epsilon_n\|_2^2 + \alpha_n^2 \|G(\theta_n)\|_2^2 \end{aligned}$$

Taking expectations with respect to $\mathbb{E}[\mathcal{F}_n]$ we get

$$\begin{aligned} & \mathbb{E}[\|\theta_{n+1} - \theta^*\|_2^2 - \|\theta_n - \theta^*\|_2^2 | \mathcal{F}_n] \\ &= -\alpha_n G(\theta_n) \cdot (\theta_n - \theta^*) + \alpha_n^2 \mathbb{E}[\|\epsilon_n\|_2^2 | \mathcal{F}_n] + \alpha_n^2 \|G(\theta_n)\|_2^2 \\ &\leq -\alpha_n \mu \|\theta_n - \theta^*\|_2^2 + \alpha_n^2 K + \alpha_n^2 (A + B \|\theta_n - \theta^*\|_2^2) \end{aligned}$$

Thus, rearranging

$$\mathbb{E}[\|\theta_{n+1} - \theta^*\|_2^2 | \mathcal{F}_n] \leq (1 - \alpha_n \mu + \alpha_n^2 B) \|\theta_n - \theta^*\|_2^2 + \alpha_n^2 (K + A)$$

Thus by Proposition 123, $\theta_{n+1} \rightarrow \theta^*$. Further taking expectations on both sides above we have

$$\mathbb{E}[\|\theta_{n+1} - \theta^*\|_2^2] \leq (1 - \alpha_n \mu + \alpha_n^2 B) \mathbb{E}[\|\theta_n - \theta^*\|_2^2] + \alpha_n^2 (K + A)$$

We can apply Lemma 119 (note that $a_n = \alpha_n \mu + \alpha_n^2 B$ will be positive for suitably large n), to give that $\mathbb{E}[\|\theta_{n+1} - \theta^*\|_2^2] \rightarrow 0$ as $n \rightarrow \infty$ as required. \square

Finally we remark that in the proof we analyzed $\|\theta_n - \theta^*\|_2$ but equally we could have analyzed $F(\theta_n) - F(\theta^*)$ instead.

Finite Time Error Bounds for Stochastic Gradient Descent

Above we established convergence of Stochastic Gradient Descent. However, once we know it works we might want to know how well it works. Considering the same set up as above, we establish some finite time bounds on the error of stochastic gradient descent.

This involves a more exact analysis of the inequality

$$\mathbb{E}[\|\theta_{n+1} - \theta^*\|_2^2] \leq (1 - \alpha_n \mu + \alpha_n^2 B) \mathbb{E}[\|\theta_n - \theta^*\|_2^2] + \alpha_n^2 C \quad (2.13)$$

that we found in our proof of Theorem 128. (Here $C = K + A$.)

Thrm 129. *There exist positive constants a_1, a_2, M_1 and M_2*

$$\mathbb{E}[\|\theta_{n+1} - \theta^*\|_2^2] \leq \frac{4C}{\mu} \frac{1}{n^\gamma} + \mathbb{E}[\|\theta_0 - \theta^*\|_2^2] M_1 e^{-a_1 n^{1-\gamma}} + n^{1-2\alpha} M_2 e^{-a_2 n^{1-\gamma}}. \quad (2.14)$$

Proof. Letting $z_n = \mathbb{E}[\|\theta_n - \theta^*\|_2^2]$. Applying Lemma 119 gives that

$$z_{n+1} \leq z_0 \underbrace{\prod_{k=0}^n (1 - \alpha_k \mu + \alpha_k^2 B)}_{=: F_n} + \underbrace{\sum_{m=0}^n C \alpha_m^2 \prod_{j=m+1}^n (1 - \alpha_j \mu + \alpha_j^2 B)}_{=: G_n}$$

We now bound F_n and G_n . First, F_n , assuming $\frac{\mu}{2} \geq \alpha_k B$ we get

$$F_n \leq \exp \left\{ -\frac{\mu}{2} \sum_k \alpha_k \right\}$$

For G_n we split the sum down the middle and bound the two parts:

$$\begin{aligned} G_n &= \sum_{m=n/2}^n C \alpha_m^2 \prod_{j=m+1}^n (1 - \alpha_j \mu + \alpha_j^2 B) + \sum_{m=0}^{n/2} C \alpha_m^2 \prod_{j=m+1}^n (1 - \alpha_j \mu + \alpha_j^2 B) \\ &\leq \frac{2C}{\mu} \alpha_{n/2} \sum_{m=n/2}^n \underbrace{\frac{\alpha_k \mu}{2} \prod_{j=m+1}^n \left(1 - \frac{\alpha_k \mu}{2}\right)}_{=\prod_{j=m+1}^n (1 - \frac{\alpha_k \mu}{2}) - \prod_{j=m}^n (1 - \frac{\alpha_k \mu}{2})} + \left(\sum_{m=0}^{n/2} C \alpha_m^2 \right) \prod_{j=n/2}^n \left(1 - \frac{\alpha_k \mu}{2}\right) \\ &\leq \frac{2C}{\mu} \alpha_{n/2} \left[1 - \prod_{j=n/2}^n \left(1 - \frac{\alpha_k \mu}{2}\right) \right] + \left(\sum_{m=0}^{n/2} C \alpha_m^2 \right) \exp \left\{ -\frac{\mu}{2} \sum_{k=n/2}^n \alpha_k \right\}. \end{aligned}$$

Putting our bounds on F_n and G_n together then gives

$$z_{n+1} \leq \frac{2C}{\mu} \alpha_{n/2} + z_0 \exp \left\{ -\frac{\mu}{2} \sum_k \alpha_k \right\} + \left(\sum_{m=0}^{n/2} C \alpha_m^2 \right) \exp \left\{ -\frac{\mu}{2} \sum_{j=n/2}^n \alpha_j \right\}$$

Letting $\alpha_n = n^{-\gamma}$ and noting $\sum_{k=1}^n n^{-\gamma} = \int_1^n x^{-\gamma} dx + M = (x^{1-\gamma} - 1)/(1-\gamma) + M$ for a constant M . We get that

$$z_{n+1} \leq \frac{4C}{\mu n^\gamma} + z_0 M_1 \exp \left\{ -\frac{\mu n^{1-\gamma}}{2(1-\gamma)} \right\} + n^{1-2\alpha} M_2 \exp \left\{ -\frac{\mu \left(1 - \left(\frac{1}{2}\right)^{1-\gamma}\right)}{2(1-\gamma)} n^{1-\gamma} \right\}.$$

□

Rmk 130. Notice the order of magnitude achieved is correct (assuming the original inequality in (2.13) is tight). To see this notice that the product for G_n above behaves as the integral to which we can apply integration by parts:

$$\begin{aligned} \int_1^n \frac{1}{x^{2\alpha}} e^{-\int_x^n \frac{1}{y^\alpha} dy} dx &= \int_1^n \underbrace{\frac{1}{x^\alpha}}_u \underbrace{\frac{1}{x^\alpha} e^{-\int_x^n \frac{1}{y^\alpha} dy}}_{dv} dx \\ &= \underbrace{\left[\frac{1}{x^\alpha} e^{-\int_x^n \frac{1}{y^\alpha} dy} \right]_1^n}_{\frac{1}{n^\alpha} e^{-\int_1^n \frac{1}{y^\alpha} dy}} + \int_1^n \frac{\alpha}{x^{\alpha-1}} e^{-\int_x^n \frac{1}{y^\alpha} dy} \geq \frac{1}{n^\alpha} \end{aligned}$$

which suggests that the convergence rate of $\frac{1}{n^\alpha}$ found for this term is of the correct order of magnitude.

Exercises

We now give some variants of Lemma 119 and Theorem 129.

Ex 131. a) Show that if

$$z(t+1) \leq (1 - \alpha(t))z(t) + \alpha(t)\beta(t)z^{\frac{1}{2}} + \alpha(t)^2c(t)$$

then

$$z(T+1) \leq z(0) \prod_{t=1}^T \left(1 - \frac{\alpha}{2}\right) + \sum_{t=1}^T \left(\alpha(t)^2c(t) + \alpha(t)\frac{\beta^2(t)}{2}\right) \prod_{s=t+1}^T \left(1 - \frac{\alpha(t)}{2}\right) \quad (2.15)$$

[Hint: $\beta z^{\frac{1}{2}} \leq \beta^2/2 + z/2$]

b) Show that if we apply a stochastic approximation update where there is a deterministic bias, $\beta(t)$, at each time, i.e.

$$x(t+1) = x(t) + \alpha(t) (G(x(t)) + \epsilon(t) + \beta(t))$$

then show that (2.15) holds with $z(t) = \mathbb{E}[\|x(t) - x^*\|^2]$.

c) Given $\beta(t) = \frac{1}{t^\beta}$ derive a bound similar to Theorem 129.

Although we deal with L^2 convergence in Theorem ???. We apply a concentration bound using the Azuma Hoeffding Inequality.

Ex 132. a) Show that if

$$z(t+1) - z(t) \leq -\alpha(t)z(t) + \alpha_t\epsilon(t) + \alpha(t)^2c(t)$$

where $\epsilon(t)$ is a martingale difference sequence then

$$\begin{aligned} z(T+1) \leq & z(0) \prod_{t=1}^T (1 - \alpha(t)) + \sum_{t=1}^T c(t)\alpha(t)^2 \prod_{s=t+1}^T (1 - \alpha(t)) \\ & + \sum_{t=1}^T \epsilon(t)\alpha(t) \prod_{s=t+1}^T (1 - \alpha(t)) \end{aligned}$$

b) Show with probability at least $1 - \delta$,

$$\begin{aligned} z(T+1) \leq & z(0) \prod_{t=1}^T (1 - \alpha(t)) + \sum_{t=1}^T c(t)\alpha(t)^2 \prod_{s=t+1}^T (1 - \alpha(t)) \\ & + \left(2 \log\left(\frac{2}{\delta}\right) \sum_{t=1}^T \epsilon_{\max}^2 \alpha_t^2 \prod_{s=t+1}^T (1 - \alpha(t))\right)^{\frac{1}{2}} \end{aligned}$$

(Hint: apply Azuma-Hoeffding)

c) Under the assumptions of Theorem 129, show that if

$$\theta(t+1) = \theta(t) + \alpha(t)(G(\theta(t)) + \epsilon(t))$$

then (by expanding $\|\theta(t) - \theta^*\|$) it holds that

$$\begin{aligned} \|\theta(t+1) - \theta^*\| &\leq (1 - \mu\alpha(t))\|\theta(t) - \theta^*\| + \alpha(t)^2\epsilon_{\max} \\ &\quad + \alpha(t)(\theta(t) - \theta^*) \cdot \epsilon(t) + \alpha(t)(G(\theta(t)) - G(\theta^*)) \cdot \epsilon(t). \end{aligned}$$

d) Combine parts a) and b) to give a high probability bound for stochastic gradient descent.

Ex 133 (Proximal Algorithm). 1) Show that the gradient descent step

$$\hat{\theta} = \theta - \alpha\nabla F(\theta),$$

solves to the optimization problem, as follows,

$$\hat{\theta} = \underset{\phi}{\operatorname{argmin}} \left\{ (\phi - \theta)^\top \nabla F(\theta) + \frac{1}{2\alpha} \|\phi - \theta\|^2 \right\}.$$

2) Argue that small values of α this is close to the optimization

$$\hat{\theta} = \min_{\phi} \left\{ F(\phi) + \frac{1}{2\alpha} \|\phi - \theta\|^2 \right\}$$

(The above minimization is often called the proximal operator or proximal optimization. This idea is you want to optimize $F(\phi)$ but there is a cost for the proximity away from start $\theta(t)$.)

3) Show that the proximal optimization is equivalent to the optimization

$$\min_{\phi} F(\phi) \quad \text{subject to} \quad \|\phi - \theta(t)\|^2 \leq c.$$

Ex 134 (Natural Gradients). We consider optimization over a parameterized set of probability distributions ($p_\theta : \theta \in \Theta$). We consider an alternative to gradient descent minimization of some loss function $F(\theta) = L(p_\theta)$. Specifically, gradient descent would give

$$\hat{\theta} = \theta - \alpha\nabla F(\theta).$$

i) Argue that the gradient descent update above is dependent on the parameterization used.

(I.e. if we changed coordinates to $\phi \in \Phi$ then the step rule will change.)

ii) Argue that the following optimization does not depend on the parameterization used (and compare with Ex 133 part 3))

$$\min_{\theta'} F(\theta') := L(p_{\theta'}) \quad \text{subject to} \quad D(p_{\theta} \| p_{\theta'}) \leq c.$$

iii) Argue that the optimization in part i) approximated by the optimization

$$\hat{\theta} = \operatorname{argmin}_{\theta'} \left\{ (\theta' - \theta) \nabla F(\theta) + \frac{1}{2\alpha} D(p_{\theta} \| p_{\theta'}) \right\}.$$

iv) Argue that

$$D(p_{\theta} \| p_{\theta+\delta}) = \delta^T J(\theta) \delta + o(\delta^2),$$

where $J(\theta)$ is the Fisher Information matrix $J(\theta) = \mathbb{E}_{\theta}[-\nabla_{\theta}^2 \log p_{\theta}(x)]$.

v) Argue that the solution to the optimization to ii) can be approximated by the update

$$\hat{\theta} = \theta - \alpha J(\theta)^{-1} \nabla F(\theta).$$

(The update above is known as the natural gradient).

2.4 Asynchronous Update

We now consider Robbins-Munro from a slightly different perspective. Suppose we have a continuous function $F : \mathbb{R}^p \rightarrow \mathbb{R}^p$ and we wish to find a fixed point x^* such that $F(x^*) = x^*$. We assume that $F(\cdot)$ is a contraction namely that, for some $\beta \in (0, 1)$,

$$\|F(x) - F(y)\|_\infty \leq \beta \|x - y\|_\infty. \quad (2.16)$$

Here $\|x\|_\infty = \max_{i=1, \dots, p} |x_i|$. (Note this contraction condition implies the existence of a fixed point). (Note the previous analysis was somewhat restricted to euclidean space.) If we suppose that we do not observe $F(x)$ but instead some perturbed version whose mean is $F(x)$, then we can perform the Robbins-Munro update for each component $i = 1, \dots, p$:

$$x_i(t+1) = x_i(t) + \alpha_i(t)(F_i(x(t)) - x_i(t) + \epsilon_i(t)) \quad (\text{RM-Async})$$

where $\alpha_i(t)$ is a sequence such that for all i

$$\sum_t \alpha_i(t) = \infty, \quad \sum_t \alpha_i^2(t) < \infty. \quad (\text{RM step})$$

Further we suppose that $\epsilon_i(t-1)$ is measurable with respect to \mathcal{F}_t , the filtration generated by $\{\alpha_i(s), x_i(s)\}_{s \leq t}$ measurable and

$$\mathbb{E}[\epsilon_i(t) | \mathcal{F}_t] = 0. \quad (2.17)$$

We assume both the functions $F(x)$ and the noise $\epsilon_i(t)$ are bounded.³

Asynchronous update. Note that in the above we let the step rule depend on i . For instance at each time t we could chose to update one component only at each step, e.g., to update component i only, we would set $\alpha_j(t) = 0$ for all $j \neq i$. Thus we can consider this step rule to be asynchronous.

Convergence result. We can analyze the convergence of this similarly

Theorem 16. *Suppose that $F(\cdot)$ is a contraction with respect to $\|\cdot\|_\infty$ (2.16), suppose the vector $x(t)$ obeys the step rule (RM-Async) with*

³However, we remark that this assumption can be relaxed significantly. At some expense in doubling the length of the proof. See [49].

step sizes satisfying (RM step) and further suppose that $F(x)$ and noise terms are bounded, then

$$\lim_{t \rightarrow \infty} x(t) = x^*$$

where x^* is the fixed point $F(x^*) = x^*$.

Proof of Theorem 16. We need to take some time to set up notation and prove three short Lemmas. After this we can wrap up the proof.

First, we may assume with out loss of generality that $x^* = 0$, since the recursion above is equivalent to

$$x_i(t+1) - x^* = x_i(t) - x^* + \alpha_i(t)(F_i(x(t)) - F_i(x^*) - x_i(t) + x^* + \epsilon_i(t)).$$

Given the assumption on $F_i(x(t)) + \epsilon_i(t)$ being bounded, we have that that $\|x(t)\|_\infty \leq D_0$ for all t , for some $D_0 < \infty$. Further define

$$D_{k+1} = \beta(1 + 2\epsilon)D_k.$$

Here we choose $\epsilon > 0$ so that $(1 + 2\epsilon)\beta < 1$ so that $D_k \rightarrow 0$. By induction, we will show that, given $\|x(t)\|_\infty < D_k$ for all $t \geq \tau_k$ for some τ_l , then there exists a τ_{k+1} such that for all $t \geq \tau_{k+1}$

$$\|x(t)\|_\infty < D_{k+1}$$

We use two recursions to bound the behavior of $x_i(t)$:

$$\begin{aligned} W_i(t+1) &= (1 - \alpha_i(t))W_i(t) + \alpha_i(t)\epsilon_i(t) \\ Y_i(t+1) &= (1 - \alpha_i(t))Y_i(t) + \alpha_i(t)\beta D_k. \end{aligned}$$

for $t \geq \tau_k$, where $W_i(\tau_k) = 0$ and $Y(\tau_k) = 0$. We use $W_i(t)$ to summarize the effect of noise on the recursion for $x_i(t)$ and we use $Y_i(t)$ to bound the error arising from the function $F_i(x)$ in the recursion. Specifically we show that

$$|x_i(t) - W_i(t)| \leq Y_i(t)$$

in Lemma 12 below. Further we notice that is a Robbin-Munro recursion for $W_i(t)$ to go to zero and $Y_i(t)$ to go to βD_k .

Lemma 12. $\forall t_0 \geq \tau_k$

$$|x_i(t) - W_i(t)| \leq Y_i(t)$$

Proof. We prove the result by induction. The result is clearly true for $t = \tau_k$.

$$\begin{aligned} x_i(t+1) &= (1 - \alpha_i(t))x_i(t) + \alpha_i(t)F_i(x(t)) + \alpha_i(t)\epsilon_i(t) \\ &\leq (1 - \alpha_i(t))(Y_i(t) + W_i(t)) + \alpha_i(t)\beta D_k + \alpha_i(t)\epsilon_i(t) \\ &= Y_i(t+1) + W_i(t+1) \end{aligned}$$

In the inequality above with apply the induction hypothesis on $x_i(t)$ and bounds of F_i . The second equality just applies the definitions of Y_i and W_i . Similar inequalities hold in the other direction and give the result. \square

Lemma 13.

$$\lim_{t \rightarrow \infty} |W_i(t)| = 0$$

Proof. We know

$$\mathbb{E}[W_i(t+1)^2 | \mathcal{F}_t] \leq (1 - 2\alpha_i(t) + \alpha_i^2(t))W(t)^2 + \alpha_i(t)^2 \mathbb{E}[\epsilon(t)^2 | \mathcal{F}_t].$$

From the Robbins-Siegmund Theorem (Prop 123), we know that

$$\lim_{t \rightarrow \infty} W(t) = 0.$$

\square

Lemma 14.

$$Y_i(t) \xrightarrow{t \rightarrow \infty} \beta D_k$$

Proof. Notice

$$Y_i(t+1) - \beta D_k = (1 - \alpha_i(t))(Y_i(t) - \beta D_k) = \dots = \left(\prod_{s=1}^t (1 - \alpha_i(s)) \right) (Y_i(0) - \beta D_k)$$

The result holds since $\sum_t \alpha_i(t) = \infty$. \square

We can now prove Theorem 16.

Proof of Theorem 16. We know that $\|x(t)\|_\infty \leq D_0$ for all t and we assume $\|x(t)\|_\infty \leq D_k$ for all $t \geq \tau_k$. By Lemma 12 and then by Lemmas 13 and 14

$$|x_i(t)| \leq Y_i(t) + |W_i(t)| \xrightarrow{t \rightarrow \infty} \beta D_k$$

Thus there exists τ_{k+1} such that $\sup_{t \geq \tau_{k+1}} \|x(t)\|_\infty \leq D_{k+1}$. Thus by induction we see that $\sup_{t \geq \tau_k} \|x(t)\|_\infty$ decreases through sequence of levels D_k as $k \rightarrow \infty$, thus $x(t)$ goes to zero as required. \square

2.5 ODE method for Stochastic Approximation

We consider the Robbins-Monro update

$$x_{n+1} = x_n + \alpha_n [g(x_n) + \epsilon_n]$$

The condition for convergence used was

$$\sum_{n=0}^{\infty} \alpha_n = \infty, \quad \sum_{n=0}^{\infty} \alpha_n^2 < \infty. \quad (\text{RM})$$

Put informally, the condition $\sum_n \alpha_n$ is used to keep the process moving [albeit in decreasingly small increments] while the condition $\sum_n \alpha_n^2$ ensures that the noise from the process goes to zero.

Given that noise is suppressed and increments get small, it is natural to ask how close the above process is to the ordinary differential equation

$$\frac{dz}{dt} = g(z(t)).$$

Moreover, can Lyapunov stability results [that we applied earlier] be directly applied to prove the convergence of the corresponding stochastic approximation scheme? Often, the answer is yes. And this has certain conceptual advantages, since the Lyapunov conditions described earlier can be quite straightforward to establish and also we don't need to directly deal with the compounding of small stochastic errors from the sequence ϵ_n .

The set up. The idea is to let

$$t_n = \sum_{k=0}^n \alpha_k \quad \text{and} \quad \mathcal{T} = \{t_n : n \in \mathbb{Z}_+\}.$$

Here t_n represents the amount of "time" that the process has been running for. We then let

$$x(t) = x_n, \quad \text{for } t = t_n.^4$$

⁴We could also linearly interpolate between these terms to define $x(t)$ for all $t \in \mathbb{R}_+$, but we choose not to do this as it provides no new insight and only serves to lengthen the proof.

We let z_m be the solution above o.d.e. started at x_m at time t_m , that is

$$\frac{dz_m}{dt} = g(z_m(t)), \quad \text{and} \quad z_m(t_m) = x_m.$$

We then compare $x(t)$ and $z_m(t)$ to see how much error has accumulated since time t_m . More specifically we are interested in

$$\sup_{t \in [t_m, t_m + T] \cap \mathcal{T}} \|x(t) - z_m(t)\|_{L_2}.$$

Assumptions. In addition to the Robbins-Monro condition (RM), we make the following assumptions.

- g is Lipschitz continuous.
- For $F_n = (x_m, \epsilon_{m-1} : m \leq n)$

$$\mathbb{E}[\epsilon_n | F_n] = 0$$

$$\text{and } \sup_n \mathbb{E}[\epsilon^2] < \infty.$$

Main result. A key result that we will prove is the following proposition

Proposition 3.

$$\lim_{m \rightarrow \infty} \sup_{t \in [t_m, t_m + T] \cap \mathcal{T}} \|x(t) - z_m(t)\|_{L_2} = 0.$$

where convergence holds with probability 1 and in L^2 .⁵

Proof. Notice

$$z_m(t_n) = x_m + \int_{t_m}^{t_n} z_m(u) du$$

while

$$x(t_n) = x_m + \sum_{k=m}^{n-1} \alpha_k g(x_k) + \sum_{k=m}^{n-1} \alpha_k \epsilon_k = x_m + \int_{t_m}^{t_n} g(x(\lfloor u \rfloor_\alpha)) du + \sum_{k=m}^{n-1} \alpha_k \epsilon_k$$

⁵Convergence in L^2 occur assuming $\|\cdot\|$ is the usual Euclidean distance.

where $\lfloor u \rfloor_\alpha = \max\{t_n : t_n \leq u\}$. So

$$\begin{aligned} \|z_m(t_n) - x(t_n)\| &\leq \left\| \sum_{k=m}^{n-1} \alpha_k \epsilon_k \right\| + \int_{t_m}^{t_n} \|g(z_m(u)) - g(x(\lfloor u \rfloor_\alpha))\| du \\ &\leq \left\| \sum_{k=m}^{n-1} \alpha_k \epsilon_k \right\| + \int_{t_m}^{t_n} \|z_m(u) - x(\lfloor u \rfloor_\alpha)\| du \end{aligned}$$

which implies by Gronwall's Lemma [Theorem 166]

$$\|z_m(t_n) - x(t_n)\| \leq \left\| \sum_{k=m}^{n-1} \alpha_k \epsilon_k \right\| e^{L(t_n - t_m)} \leq \|M_n - M_m\| e^{LT}. \quad (2.18)$$

where the final inequality holds for t_n such that $t_n - t_m \leq T$, and we define $M_n = \sum_{k=1}^n \alpha_k \epsilon_k$. Notice that M_n , $n \in \mathbb{N}$, is a martingale and further

$$\mathbb{E}M_n^2 = \sum_{k=1}^n \alpha \mathbb{E}[\epsilon_k^2] \leq K \sum_{k=1}^{\infty} \alpha_k^2 < \infty$$

where $K = \max_k \mathbb{E}[\epsilon_k^2]$. Thus M_n is an L^2 bounded Martingale and thus convergence with probability 1 and in L^2 . Consequently M_n is a Cauchy sequence, meaning

$$\lim_{m \rightarrow \infty} \sup_{n \geq m} \|M_n - M_m\| = 0$$

with probability 1 and in L^2 . Applying this to (2.18) gives the required result

$$\lim_{m \rightarrow \infty} \sup_{t \in [t_m, t_m + T] \cap \mathcal{T}} \|z_m(t) - x(t)\| = 0.$$

□

Applying the o.d.e limit. Before we focus on the proof of Proposition 3, it's worth explaining how it can be applied. The main idea is to

1. Check that the o.d.e. convergence by showing gets close to the some desired set of points \mathcal{X}^* in T time units for each initial condition x_n , $n \in \mathbb{N}$.
2. Then apply Proposition 3 to show that the stochastic approximation is also close to the o.d.e at time T .

This is sufficient to show that the stochastic approximation converges to \mathcal{X}^* .

Here, we combine La Salle's Invariance Principle, Theorem 9, with Proposition 3, though, in principle, we could have considered any of the ode results from Section 2.1. We assume that the assumptions of Proposition 3 are satisfied. In addition we assume,

- Almost surely,

$$\sup_n \|x_n\| < \infty.$$

Further we recall that for La Salle's Invariance Principle, we assumed there was a Lyapunov function $L : \mathbb{R}^d \rightarrow \mathbb{R}_+$ such that

- L is continuously differentiable .
- $g(x) \cdot \nabla L(x) \leq 0$ for all x .
- The sets $\{x : L(x) \leq l\}$ are compact for all l .

Also we defined $\mathcal{X}^* := \{x : g(x) \cdot \nabla L(x) = 0\}$. Recall that La Salle's Invariance Principle stated that for all solutions to the o.d.e. $dz/dt = g(z(t))$ with $z(0) \in \{x : L(x) \leq l\}$ there exists a T such that $\max_{x^* \in \mathcal{X}^*} |z(t) - x^*| \leq \epsilon$ for all $t \geq T$.

Theorem 17. *For the stochastic approximation scheme*

$$x_{n+1} = x_n + \alpha_n [g(x_n) + \epsilon_n]$$

described in Proposition 3 and given a Lyapunov function L as described above, it holds, with probability 1,

$$x_n \xrightarrow[n \rightarrow \infty]{} \mathcal{X}^*$$

Proof. First we check that the o.d.e solutions $z_m(t)$ considered in Proposition 3 are going to satisfy the conditions of La Salle. In particular, La Salle's result requires o.d.e. solutions to all belong to some compact set. Notice, since we assume that $C := \sup_n \|x_n\| < \infty$ we can let $l = \max\{L(x) : \|x\| \leq C\}$ and take $\mathcal{X} = \{x : L(x) \leq l\}$. Since $L(z(t))$ is decreasing for all solutions to the o.d.e. We see that $z_m(t) \in \mathcal{X}$ for all m and $t \geq t_m$.

Next we set up the bounds from the two results. From La Salle's Invariance Principle, we know that $\forall \epsilon > 0 \exists T > 0$ such that $\forall t \geq T$ and $\forall m$

$$\min_{x^*} |z_m(t_m + t) - x^*| \leq \epsilon.$$

Taking this choice of T , we know from Proposition 3 that there exists m^* s.t. $\forall m > m^*$

$$\sup_{t \in [t_m, t_m + 2T] \cap \mathcal{T}} \|x(t) - z_m(t)\| \leq \epsilon.$$

Notice we can take m^* suitably large so that $\alpha_m =: t_m - t_{m-1} \leq T$ for all $m \geq m^*$. Notice this implies that

$$\bigcup_{m: m \geq m^*} [t_m + T, t_m + 2T] = [t_{m^*} + T, \infty).$$

Now notice that if n is such that $t_n \in [t_m + T, t_m + 2T]$ for some $m \geq m^*$ then combining the inequalities above, gives that

$$\min_{x^*} \|x_n - x^*\| \leq \|x_n - z_m(t_n)\| + \min_{x^*} \|z_m(t_n) - x^*\| \leq 2\epsilon.$$

Thus we see, with the union above, that for all $t_n \geq t_{m^*} + T$ it holds that $\min_{x^*} \|x_n - x^*\| \leq 2\epsilon$. In other words $x_n \rightarrow \mathcal{X}^*$ as required. \square

Exercises

Ex 135 (A Finite Time Bound). *a) Using the notation from Prop 3. Show that*

$$\|x^* - x_n\|_{L_2} \leq \|x^* - z_m(t_n)\|_{L_2} + \left\| \sum_{k=m}^n \alpha_k \epsilon_k \right\|_{L_2} e^{KT}$$

b) Argue that if $\alpha_n = \frac{c}{n}$, $x^ = 0$, $\mathbb{E}[\epsilon_k^2] < \sigma^2$, x_n is bounded and $\dot{z} = -\beta z$ then*

$$\|x^* - x_n\|_{L_2} = O\left(n^{-\frac{\beta}{2(\beta+K+\frac{1}{2})}}\right)$$

(Note the above bound is slightly worse than Theorem 129. This is because Gronwall's lemma is quite a loose bound, where as Theorem 129 applies and integrating factor which dampens stochastic terms.)

References

The o.d.e approach to stochastic approximation was initiated by Ljung [33]. Shortly after it is was extensively developed by Kushner, see [29] and [28] for two text book accounts. The arguments above loosely follow the excellent text of Borkar [11]. We shorten the proof in several ways and consider L^2 convergence. A further text with a general treatment is Benveniste et al. [5].

2.6 Online Convex Optimization

In online convex optimization, we sequentially receive convex functions and our task is to sequentially minimize the aggregate of these. The assumptions on these convex functions can be very mild. Yet, surprisingly, very good bounds can be derived. For instance, stochastic gradient descent can be recast as an online convex optimization, and the bound derived are competitive with the best finite time bounds.

Setup. For a convex set Θ and discrete time $t = 1, 2, \dots$, we consider the setting where at time t :

1. An unknown convex loss function, $l_t : \Theta \rightarrow \mathbb{R}$, is set;
2. The algorithm chooses $\theta_t \in \Theta$, and we incur loss $l_t(\theta_t)$;
3. The gradient $\nabla l_t(\theta_t)$ is revealed.

Since l_t is not known when we chose θ_t , there is little hope of minimizing l_t itself. Instead, we measure the quality of the algorithm in terms of its *regret*:

$$\mathcal{R}_g(T) = \sum_{t=1}^T l_t(\theta_t) - \min_{\theta \in \Theta} \sum_{t=1}^T l_t(\theta). \quad (2.19)$$

The regret, $\mathcal{R}_g(T)$, compares the aggregate loss of the algorithm with the best fixed choice. A good algorithm should have small regret.

Remark 136. *Although minimizing each individual function l_t is not feasible, it is reasonable to expect some algorithms to have low regret. Think of it this way: the best fixed choice $\theta_t^* \in \operatorname{argmin}_{\theta} \sum_{s=1}^t l_s(\theta)$ should have a roughly equal contribution from each loss function. When moving on to the next time step, θ_t^* should not be too far from θ_{t-1}^* and the size of the change should be about $1/t$ (if each term contributes equally to the change). Further if we are to move towards to optimum then direction should not be far off the steepest decent direction $\nabla l_t(\theta_t)$. This suggests using gradient descent with a $1/t$ set size:*

$$\theta_{t+1} = \theta_t - \frac{\alpha}{t} \nabla l_t(\theta_t)$$

should track the changes in θ_t^ . This works out, though we need to be careful about the assumptions that we make as there are counter-examples, and if the functions l_t are not sufficiently smooth then we need to compromise on our rate of convergence.*

We will prove results for both convex and strongly convex functions. Recall that a differentiable function is convex when

$$L(\phi) - L(\theta) \geq (\phi - \theta)\nabla L(\theta).$$

We say $L(\theta)$ is a -strongly convex if

$$L(\phi) - L(\theta) \geq (\phi - \theta)\nabla L(\theta) + \frac{a}{2}\|\phi - \theta\|^2.$$

The following lemma follows quite immediately from the definition.

Lemma 15. *If $l_t : \Theta \rightarrow \mathbb{R}$ is a -strongly convex (including $a = 0$) then*

$$\sum_{t=1}^T (\theta_t - \theta)\nabla l_t(\theta_t) \geq \sum_{t=1}^T \frac{a}{2}\|\theta_t - \theta\|^2 + \sum_{t=1}^T l_t(\theta_t) - l_t(\theta).$$

Proof. Apply the definition of a -strongly convex, rearrange and sum. \square

Notice in the above lemma the final term corresponds to the regret, and the penultimate sum can only act to reduce the regret. So the above lemma shows that if we get a handle on

$$\sum_{t=1}^T (\theta_t - \theta^*)\nabla l_t(\theta_t) \tag{2.20}$$

then low regret may be achievable. The follow algorithm is one of a suite of algorithms that can manage the above expression.

Projected Gradient Descent. We analyse the following projected gradient descent algorithm:

$$\theta_{t+1} = \Pi_{\Theta}(\theta_t - \alpha_t \nabla l_t(\theta_t)).$$

Here $\nabla_t(\theta) = \nabla l_t(\theta)$ and $\Pi_{\Theta}(\theta)$ is the projection of θ onto Θ . The following bounds the gradient sum (2.20) in terms of its step sizes.

Lemma 16. *For any function $\nabla_t(\theta)$ and θ_t , as above, then for all $\theta \in \Theta$*

$$\sum_{t=1}^T (\theta_t - \theta)\nabla_t(\theta_t) \leq \sum_{t=1}^T \frac{\|\theta_t - \theta\|^2}{2} \left(\frac{1}{\alpha_t} - \frac{1}{\alpha_{t+1}} \right) + \sum_{t=1}^T \frac{\alpha_t}{2} \|\nabla_t(\theta_t)\|^2.$$

Proof. For any $\theta \in \Theta$,

$$\begin{aligned}\|\theta_{t+1} - \theta\|^2 &= \|\Pi_{\Theta}(\theta_t - \alpha_t \nabla_t(\theta_t)) - \Pi_{\Theta}(\theta)\|^2 \\ &\leq \|\theta_t - \alpha_t \nabla_t(\theta_t) - \theta\|^2 \\ &\leq \|\theta_t - \theta\|^2 + \alpha_t^2 \|\nabla_t(\theta_t)\|^2 - 2\alpha_t(\theta_t - \theta) \cdot \nabla_t(\theta_t).\end{aligned}$$

Rearranging and summing gives

$$\begin{aligned}&\sum_{t=1}^T (\theta_t - \theta) \cdot \nabla_t(\theta_t) \\ &\leq \sum_{t=1}^T \left[\frac{1}{2\alpha_t} \|\theta_t - \theta\|^2 - \frac{1}{2\alpha_t} \|\theta_{t+1} - \theta\|^2 \right] + \sum_{t=1}^T \frac{\alpha_t}{2} \|\nabla_t(\theta_t)\|^2 \\ &= -\frac{1}{\alpha_T} \|\theta_{T+1} - \theta\|^2 + \sum_{t=1}^T \|\theta_t - \theta\|^2 \left[\frac{1}{\alpha_t} - \frac{1}{\alpha_{t-1}} \right] + \sum_{t=1}^T \frac{\alpha_t}{2} \|\nabla_t(\theta_t)\|^2 \\ &\leq \sum_{t=1}^T \|\theta_t - \theta\|^2 \left[\frac{1}{\alpha_t} - \frac{1}{\alpha_{t-1}} \right] + \sum_{t=1}^T \frac{\alpha_t}{2} \|\nabla_t(\theta_t)\|^2.\end{aligned}$$

In the equality above we rearrange the order of summation in the square brackets (wlog. we take $\|\theta_1 - \theta\|^2/\alpha_0 := 0$). The above expression gives the required bound. \square

With Lemmas 15 and 16 in place, we can prove the following theorem. This result is the discrete analogue of Theorem 10.

Theorem 18. *i) If l_t is convex and we take $\alpha_t = 1/\sqrt{T}$ then*

$$\mathcal{R}_g(T) \leq \left(\frac{\theta_{\max}^2}{2} + \nabla_{\max}^2 \right) \sqrt{T}.$$

ii) If l_t is a -strongly convex for all t and we take $\alpha_t = 1/(aT)$ then

$$\mathcal{R}_g(T) \leq \frac{\nabla_{\max}^2}{2a} (1 + \log T).$$

Here $\theta_{\max} = \max_{\theta, \phi \in \Theta} \|\theta - \phi\|$ and $\nabla_{\max} = \max_t \|\nabla_t(\theta_t)\|$

Proof. Combining Lemma 15 and 16, we have that

$$\begin{aligned}
\mathcal{R}g(T) &= \sum_{t=1}^T l_t(\boldsymbol{\theta}_t) - l_t(\boldsymbol{\theta}) \leq \sum_{t=1}^T (\boldsymbol{\theta}_t - \boldsymbol{\theta}) \nabla_t l_t(\boldsymbol{\theta}_t) - \frac{a}{2} \|\boldsymbol{\theta}_t - \boldsymbol{\theta}\|^2 \\
&\leq \sum_{t=1}^T \frac{\|\boldsymbol{\theta}_t - \boldsymbol{\theta}\|^2}{2} \left(\frac{1}{\alpha_t} - \frac{1}{\alpha_{t+1}} - \frac{a}{2} \right) + \sum_{t=1}^T \frac{\alpha_t}{2} \|\nabla_t l_t(\boldsymbol{\theta}_t)\|^2 \\
&\leq \frac{\theta_{\max}^2}{2} \sum_{t=1}^T \left(\frac{1}{\alpha_t} - \frac{1}{\alpha_{t+1}} - \frac{a}{2} \right) + \frac{\nabla_{\max}^2}{2} \sum_{t=1}^T \alpha_t. \quad (2.21)
\end{aligned}$$

Now for part i), where $a = 0$, if we let $\alpha_t = \frac{1}{\sqrt{t}}$ then

$$\sum_{t=1}^T \left(\frac{1}{\alpha_t} - \frac{1}{\alpha_{t+1}} - \frac{a}{2} \right) = \sum_{t=1}^T (\sqrt{t} - \sqrt{t+1}) \leq \sqrt{T},$$

and

$$\sum_{t=1}^T \alpha_t = \sum_{t=1}^T \frac{1}{\sqrt{t}} \leq \frac{\sqrt{T} - 1/2}{1 - 1/2} \leq 2\sqrt{T}.$$

(The inequality above is proven in Lemma 26) Substituting these two inequalities into (2.21) gives the regret bound in part i).

For part ii) with $a > 0$ and $\alpha_t = \frac{1}{2at}$ we have

$$\sum_{t=1}^T \left(\frac{1}{\alpha_t} - \frac{1}{\alpha_{t+1}} - \frac{a}{2} \right) \leq 0$$

and

$$\sum_{t=1}^T \alpha_t \leq \frac{1}{2a} (1 + \log T).$$

(The inequality above is proven in Lemma 26) Substituting the above two inequalities into (2.21) gives the regret bound in part ii). □

Chapter 3

Bandits and Experts

3.1 Stochastic Multi-Armed Bandit

- Lai and Robbins lower bound.
- UCB policy and optimality.

We discuss a canonical multi-arm bandit setting. The stochastic multi-arm bandit with Bernoulli distributed rewards. We let $a = 1, \dots, N$ index the set of arms. We let $\mathcal{A} = \{1, \dots, N\}$ be the set of arms. If you play the arm a at time $t \in \mathbb{N}$, you receive rewards $r_t(a)$ which are independent and identically distributed in t . However, the distribution between arms may change.

We let $\bar{r}(a)$ be the mean of arm a . We want to find the machine with highest mean reward. We define

$$r^* := \max_a \bar{r}(a) \quad \text{and} \quad a^* \in \operatorname{argmax}_a \bar{r}(a)$$

and also we define

$$\Delta_a = r^* - \bar{r}(a) \quad \text{and} \quad \Delta = \min\{\Delta_a : \Delta_a > 0\}.$$

We do not know these averages in advance and we only know the rewards from the times that we play each machine – we are only allowed to play one machine per unit time.

We let $\hat{r}_t(a)$ denote the empirical mean of arm a after it has been played t times. Given the central limit theorem and the Azumehoeffding bound (see appendix), we can assume the following

Assumption. There is some function $\Lambda(\Delta)$ positive for $\Delta > 0$ such that

$$\mathbb{P}(|\hat{r}_t(a) - \bar{r}_t(a)| \geq \Delta/2) \leq 2e^{-t\Delta^2/c}.$$

Many reasonably behaved distributions obey this condition. For instance, any bounded random variable and Gaussian random variables. This assumption can be relaxed.

A policy chooses a sequence of arms, π_t at times $t \in \mathbb{N}$ as a function of the rewards previously experienced, i.e. as a function of $\{\pi_s, r_s(\pi_s)\}_{s=1}^{t-1}$. As with MDPs, the expected cumulative reward of policy π at time T is

$$R_T(\pi) = \mathbb{E} \left[\sum_{t=1}^T r_t(\pi_t) \right]$$

Def 137 (Regret). *The regret of policy $\pi = \{\pi_t\}_{t=1}^T$ at time T is¹*

$$\mathcal{Rg}(T) = \max_{a=1,\dots,N} R_T(a) - R_T(\pi).$$

The regret is a frequently used metric of choice for bandit problems and many other areas of statistical learning.

The following lemma shows that we can analyse the regret by understanding the number of times we play a sub-optimal arm or equivalently calculating the probability that we play a sub-optimal arm.

Lemma 17. *For a multi-arm bandit problem with Bernoulli rewards*

$$\begin{aligned} \mathcal{Rg}(T) &= \sum_{a=1}^N [r^* - \bar{r}(a)] \mathbb{E}[T_a] \\ &= \sum_{t=1}^T \sum_{a=1}^N [r^* - \bar{r}(a)] \mathbb{P}(\pi_t = a) \end{aligned}$$

where T_a is the number of times arm a is played by time T .

Proof. We can expand the reward of the optimal arm with

$$R_T(a^*) = r^* T = r^* \sum_a \mathbb{E}[T_a] \tag{3.1}$$

and, by independence, we can replace the reward with its expectation as follows

$$R_T(\pi) = \mathbb{E} \left[\sum_{t=1}^T \sum_a p(a) \mathbb{I}[\pi_t = a] \right] = \sum_a p(a) \mathbb{E}[T_a]. \tag{3.2}$$

Now subtracting (3.2) from (3.1) gives the regret and the first equality. The second equality follows immediately from the fact that

$$\sum_{t=1}^T \mathbb{P}(\pi_t = a) = \mathbb{E}[T_a].$$

□

¹Note the regret depends on the policy, but for convenience we do not explicitly include this in our notation when it is clear what policy is being considered.

Explore and Commit.

Explore and commit gives a simple way of understanding the explore-exploit trade-off and why a regret of $O(\log T)$ is achievable.

Suppose we implement the following policy:

Explore-and-Commit.

1. We play each arm t times.
2. We play the best arm there after.

Given we are going to run our bandit experiment for T rounds, we now look for a good choice of t .

Proposition 4. *For $t = C \log T$, the regret of explore-and-commit is*

$$\mathcal{R}g(T) \leq \left[C \log T + \frac{4}{T^{\Delta^2 C/c - 1}} \right] \cdot \sum_{a=1}^N \Delta_a$$

In the above, notice we need $C\Delta^2/c > 1$ to get good results for large T and we don't know Δ in advance. This is a weakness.

Proof. Let $\mathcal{A}^* = \{a : r(a) = \max_{a'} r(a')\}$ be the set of optimal arms. For any $a \notin \mathcal{A}^*$ and $a^* \in \mathcal{A}^*$,

$$\mathbb{P}(\hat{r}_t(a) > \hat{r}_t(a^*)) \leq \mathbb{P}(|\hat{r}_t(a) - \bar{r}(a)| > \Delta/2) + \mathbb{P}(|\hat{r}_t(a^*) - \bar{r}(a^*)| > \Delta/2) \leq 4e^{-t\Delta^2/c}.$$

I.e. either a or a^* must observe rewards away from their expected mean for a to beat a^* in the explore phase.

We can split the regret into its explore and commit phases and then apply the above inequality:

$$\begin{aligned} \mathcal{R}g(T) &\leq t \sum_a \Delta_a + (T - Nt) \sum_{a \notin \mathcal{A}^*} \Delta_a \mathbb{P}(\hat{r}_t(a) > \hat{r}_t(a^*)) \\ &\leq \sum_a \Delta_a \left[t + 4Te^{-t\Delta^2/c} \right]. \end{aligned}$$

(At this point it is worth noting that the term in square brackets can easily be minimized, and it is minimized at $t = (c/\Delta^2) \log(4\Delta^2 T/c)$. So basically $t = O(\log T)$ is the right amount of exploration.) Substituting in $t = C \log T$ to the term in square brackets gives the result. \square

Deterministic Exploration.

The exploration phase in explore and commit does not have to all happen that the beginning. An easy way to deal with exploration is to decide in advance when you are going to explore each arm.

Specifically, for each arm a take $E_a(t) \in \mathbb{Z}_+$ where $E_a(t)$ gives the number of times arm a is explored by time t .²

Deterministic Exploration. At time t ,

1. If $E_a(t) > E_a(t - 1)$ play arm a .
2. If no such arm exists, play the arm with highest reward found during exploration.

If $E_i(T) \sim C_T \log T$ where C_T is a slowly increasing function (think $C_T = \log(e + \log(T))$) then a very similar proof to the explore and commit proof gives

$$\mathcal{Rg}(T) = O(C_T \log T)$$

This is an exercise.

 ϵ -Greedy.

Rather tracking how long you need to play each arm a simple heuristic, with a similar proof to deterministic exploration is to randomize.

ϵ -Greedy. At time t

1. with probability ϵ_t , play an arm uniformly at random.
2. Else, with probability $1 - \epsilon_t$, play the arm with highest reward.

The proof for this policy is very similar to the deterministic exploration case. (We just need to apply a [Bernstein] concentration bounds to the amount of exploration.) Similar to before if $\epsilon_t = C_t/t$ for a slowly increasing function then

$$\mathcal{Rg}(T) = O(C_T \log T)$$

Again this is an exercise for the reader.

²Here we require that $\sum_a (E_a(t) - E_a(t - 1)) \in \{0, 1\}$.

At this point, it is worth noting that we can get arbitrarily close to $\log T$ with bounds and simple policies. So the moral of the story so far is: no matter what you do, as long as you explore every arm a bit more than $\log T$ times and you spend the rest of your time on the best arm seen so far then you should have a good regret bound.

As we see shortly, with the Lai-Robbins lower bound, $\mathcal{R}_g(T) = O(\log T)$ is the right rate of convergence for regret in the stochastic setting. We can also achieve this bound with a policy called Upper-Confidence Bound (UCB). The idea is to construct a statistical confidence bound and then choose the arm with confidence bound (empirical mean plus confidence error). UCB is a fantastic policy used all over the place. However, it should be noted that in practice it does not always generalize well. Thus one should fall back on simpler policies, remembering the rules of thumb developed so far.

3.2 Upper Confidence Bound.

The proofs so far have relied on the bound

$$\mathbb{P}(|\hat{r}_t(a) - \bar{r}_t(a)| > \epsilon) \leq 2e^{-t\epsilon^2/c}. \quad (3.3)$$

We noted that if we take $t \approx \log T$ then the probability of choosing the wrong arm goes down very fast and does not contribute to the regret bound. Basically, the regret of $\log T$ occurs because of the $t \approx \log T$ arm pulls that were required to get the concentration in the first place.

Based on this, the Upper Confidence Bound algorithm looks at the values of ϵ that guarantee this fast decay in probability. It assesses how confident are we about each estimate on the mean and then chooses the arm with highest mean plus error. Specifically, for t and $t_a \leq t$, we define $\epsilon_{t_a,t}$ to be such that

$$\mathbb{P}(|\hat{r}_{t_a}(a) - \bar{r}(a)| > \epsilon_{t_a,t}) \leq \frac{1}{t^2}. \quad (3.4)$$

Given (3.3) this holds for ϵ such that

$$2e^{-t_a\epsilon^2/c} = \frac{1}{t^2} \quad \text{which implies} \quad \epsilon_{t_a,t} = \sqrt{\frac{c \log 2t}{t_a}}.$$

From now on we will take $\epsilon_{t_a,t}$ as given above, where t is the number of rounds of the algorithm and t_a is the number of times arm a has been selected by time t .

UCB choses the highest empirical mean plus error:

Upper-Confidence Bound (UCB).

- Choose any arm such that

$$a \in \operatorname{argmax}_{a \in \mathcal{A}} \hat{r}_{t_a}(a) + \epsilon_{t_a,t} \quad \text{where} \quad \epsilon_{t_a,t} = \sqrt{\frac{c \log 2t}{t_a}}.$$

Notice for UCB algorithm so long as arm a is not played the $\log t$ will deterministically keep increasing (and eventually) will dominate forcing the policy to explore a . Notice we need to monitor what is going on with all arms at all times implement an iteration of UCB. The following result shows that we get a much more crisp regret bound for UCB.

Theorem 19. *We regret of UCB is*

$$\mathcal{R}g(T) \leq \sum_{a \in \mathcal{A}} \frac{4c}{\Delta_a} \log 2T + 4 \sum_{a \in \mathcal{A}} \Delta_a.$$

Proof. Given Lemma 17, we investigate the probability that UCB might select a sub-optimal arm. We will see that it occurs either because of insufficient exploration (which we will see is an issue that can only occur a small –order $\log T$ – number of times) or because of insufficient concentration (which is unlikely to happen because of the way we cooked-up $\epsilon_{t_a,t}$).

Specifically, suppose at time t the policy chooses $\pi_t = a$ for $a \notin \operatorname{argmax} \bar{r}(a')$ then

$$\hat{r}_{t_a}(a) + \epsilon_{t_a,t} \geq \hat{r}_{t_{a^*}}(a^*) + \epsilon_{t_{a^*},t} \quad (3.5)$$

where a^* is some arm in $\operatorname{argmax} \bar{r}(a')$.

In addition to playing arm a at time t , let's first suppose that both arms are concentrated about their means. Specifically, the following event holds

$$E_t = \{|\hat{r}_{t_a}(a) - \bar{r}(a)| \leq \epsilon_{t_a,t}\} \cap \{|\hat{r}_{t_{a^*}}(a^*) - \bar{r}(a^*)| \leq \epsilon_{t_{a^*},t}\}$$

Then using the bounds in E_t to replace \hat{r} with \bar{r} in (3.5) implies that

$$\bar{r}(a) + 2\epsilon_{t_a,t} \geq \hat{r}_{t_a}(a) + \epsilon_{t_a,t} \geq \hat{r}_{t_{a^*}}(a^*) + \epsilon_{t_{a^*},t} \geq \bar{r}(a^*),$$

which³ in turn implies

$$\Delta_a = \bar{r}(a^*) - \bar{r}(a) \leq 2\epsilon_{t_a,t} = 2\sqrt{\frac{c \log 2t}{t_a}} \leq 2\sqrt{\frac{c \log 2T}{t_a}}.$$

So we have a bound on how much we explored arm a :

$$t_a \leq \frac{4c}{\Delta_a^2} \log 2T. \quad (3.6)$$

We can interpret this as saying: when we have good concentration, we choose the wrong arm, a , because the amount of exploration t_a was too small. And critically we can see what the right amount of exploration for UCB is quantified in (3.6).

³Note we are fortunate that we don't need to know $\epsilon_{t_{a^*},t}$ in the above bound.

Now we can also bound the probability that the event E does not hold. That is, we a union bound and (3.4), we have

$$\mathbb{P}(E_t^c) \leq \mathbb{P}(|\hat{r}_{t_a}(a) - \bar{r}(a)| > \epsilon_{t_{a,t}}) + \mathbb{P}(|\hat{r}_{t_{a^*}}(a^*) - \bar{r}(a^*)| > \epsilon_{t_{a^*,t}}) \quad (3.7)$$

$$\leq \frac{2}{t^2} \quad (3.8)$$

Putting everything together,

$$\begin{aligned} \mathbb{E}[T_a] &= \mathbb{E}\left[\sum_{t=1}^T \mathbb{I}[\pi_t = a]\right] \\ &= \mathbb{E}\left[\sum_{t=1}^T \mathbb{I}[\pi_t = a, E_t] + \mathbb{I}[\pi_t = a, E_t^c]\right] \\ &\leq \frac{4c}{\Delta_a^2} \log 2T + \sum_{t=1}^T \mathbb{P}(\pi_t = a, E_t^c) \\ &\leq \frac{4c}{\Delta_a^2} \log 2T + \sum_{t=1}^T \frac{2}{t^2} \\ &\leq \frac{4c}{\Delta_a^2} \log 2T + 4. \end{aligned}$$

In the first inequality above apply (3.6), i.e. there can be at most $\frac{4c}{\Delta_a^2} \log 2T$ times where a is played and E_t holds. In the second inequality we apply (3.7) [after noting that $\mathbb{P}(\pi_t = a, E_t^c) \leq \mathbb{P}(E_t^c)$]. In the final inequality, we use the fact that $\sum_{t=1}^{\infty} 1/t^2 \leq 1 + \int_1^{\infty} \frac{1}{t} dt = 2$.

The result then follows by applying the above bound to the identity in Lemma 17, that is

$$\mathcal{Rg}(T) = \sum_a \Delta_a \mathbb{E}[T_a] \leq \sum_a \frac{4c}{\Delta_a} \log 2T + 4 \sum_a \Delta_a.$$

□

3.3 Lai and Robbins Lower Bound.

The results so far have suggest that, for independent identically distributed arms, the correct size of the regret is of the order $\log T$. We now more formally prove this with the Lai and Robbins lower-bound .

We suppose that there are N arms as described at the very beginning of this section. We let P_a be the probability distribution of the rewards from arm $a \in \mathcal{A}$. For the Lai and Robbins lower-bound, it is useful to think of \mathcal{A} as a subset of size N with values from a set of parameters Θ , i.e. $\mathcal{A} \subset \Theta$, and that P_θ is the probability distribution of rewards under the parameter choice θ . (To be concrete, think of Θ being the interval $[0,1]$ and P_θ being a Bernoulli distribution with parameter θ .)

Asymptotic Consistency. To have a reasonable lower-bound we need to assume a reasonable set of polices. For instance, we need to exclude polices that already know the reward of each arm. Lai and Robbins consider amongst polices that are asymptotically consistent, which means, for each set of arms \mathcal{A} , for all $a \notin \mathcal{A}^*$

$$\mathbb{E}[T_a] = o(T^\eta) \tag{3.9}$$

for every $\eta \in (0,1)$. I.e. a policy is a "good" policy if it can do better than any power T^η in playing sub-optimal arms. (Recall here \mathcal{A}^* is the set of optimal arms, and T_a is the number of times we play arm a by time T .)

The Lower Bound. The Lai–Robbins Lower Bound is the following:

Theorem 20 (Lai and Robbins).

$$\liminf_{T \rightarrow \infty} \frac{\mathbb{E}[T_a]}{\log T} \geq \frac{1}{D(P_a \| P_\star)}$$

and thus

$$\liminf_{T \rightarrow \infty} \frac{\mathcal{R}g(T)}{\log T} \geq \sum_{a \in \mathcal{A}} \frac{\Delta_a}{D(P_a \| P_\star)}$$

where here $D(P_a \| P_\star)$ is the Relative Entropy (defined in the appendix) between the distribution of arm a , P_a and the distribution of the optimal arm, P_\star .

For independent rewards $r_1, \dots, r_t \sim P$ and independent rewards $r'_1, \dots, r'_t \in P'$, under mild conditions, there exists a function $D(r')$ such that

$$\mathbb{P}((r_1, \dots, r_t) \in C) = \mathbb{E}' \left[e^{\sum_{s=1}^t D(r'_s)} \mathbb{I}[(r_1, \dots, r_t) \in C] \right] \quad (3.10)$$

and

$$D(P||P') = \mathbb{E}[D(r)]. \quad (3.11)$$

(The term $e^{D(r)}$ is just the ratio between $P(r)$ and $P'(r)$ and is formally called Radon-Nikodym derivative.) For the Lai Robbins lower-bound, we also require the assumption that

$$D(P||P_{a'}) \rightarrow D(P||P_a)$$

as $a' \rightarrow a$. This is required to get the constants right in the theorem but is not critical to the overall analysis.

We now prove the Lai and Robbins Lower-bound.

Proof. We take an increasing sequence of integers n_T . We will specify it soon but for now it is enough to assume that $n_T = o(T)$. We can lower-bound the expected number of times arm a is played as follows:

$$\mathbb{E}[T_a] \geq n_T \mathbb{P}(T_a > n_T) = n_T (1 - \mathbb{P}(T_a \leq n_T)). \quad (3.12)$$

Here we see that we want to take n_T as large as possible while keeping the probability $\mathbb{P}(T_a \leq n_T)$ away from 1. When arm a was replaced by an arm a' whose mean reward is the larger than each arm in \mathcal{A} (i.e. $\bar{r}' > r_*$) then we can analyze this probability:

$$\mathbb{P}'(T_{a'} \leq n_T) = \mathbb{P}'(T - T_{a'} \geq T - n_T) \leq \frac{\mathbb{E}[T - T_{a'}]}{T - n_T} = \frac{o(T^\eta)}{T - n_T} = o(T^{\eta-1}). \quad (3.13)$$

In the inequality above we apply Markov's Inequality and the asymptotic consistency assumption (3.9). (Note the above bound holds for every $\eta \in (0, 1)$ so in informal terms we could think of the above bound as being $O(T^{-1})$.)

We can change arm a for arm a' through the change of measure given above in (3.10). This involves the sum of independent random variables $\sum_{k=1}^{n_T} D(r_k)$ which by (3.11) and the weak law of large numbers satisfies:

$$\delta_T := \mathbb{P}\left(\left|\sum_{k=1}^{n_T} \{D(r_k) - D(P||P')\}\right| > \epsilon\right) \xrightarrow{T \rightarrow \infty} 0.$$

Here we apply the shorthand $P = P_a$ and $P' = P_{a'}$.

We now have various pieces in place where we can now analyze the probabilities $\mathbb{P}(T_a \leq n_T)$ as follows:

$$\begin{aligned} & \mathbb{P}(T_a \leq n_T) \\ &= \mathbb{P}(T_a \leq n_T, |\sum_{k=1}^{n_T} \{D(r_k) - D(P||P')\}| > \epsilon) \\ & \quad + \mathbb{P}(T_a \leq n_T, |\sum_{k=1}^{n_T} \{D(r_k) - D(P||P')\}| \leq \epsilon) \end{aligned} \quad (3.14)$$

$$\begin{aligned} & \leq \mathbb{P}(|\sum_{k=1}^{n_T} \{D(r_k) - D(P_a||P_{a'})\}| > \epsilon) \\ & \quad + \mathbb{E}' \left[e^{\sum_{k=1}^{n_T} D(r'_k)} \mathbb{I} [T_{a'} \leq n_T, |\sum_{k=1}^{n_T} \{D(r'_k) - D(P||P')\}| \leq \epsilon] \right] \end{aligned} \quad (3.15)$$

$$\leq \delta_T + e^{n_T(D(P||P')+\epsilon)} \mathbb{P}'(T_{a'} \leq n_T) \quad (3.16)$$

$$\leq \delta_T + e^{n_T(D(P||P')+\epsilon)} o(T^{\eta-1}). \quad (3.17)$$

In the first equality (3.14), we split the probability according to how close $\sum_{k=1}^{n_T} D(r_k)$ is to $n_T D(P||P')$. In the inequality (3.15), we remove the condition $\{T_a \leq n_T\}$ from the first term and apply the change of measure to the second term by using equality (3.10). The condition on $\sum_{k=1}^{n_T} D(r_k)$ and the definition of δ_T yields (3.16). Finally (3.17) follows from the bound on $\mathbb{P}'(T_{a'} \leq n_T)$ given in (3.13).

So, we have

$$P(T_a \leq n_T) \leq \delta_T + e^{n_T(D(P||P')+\epsilon)} \mathbb{P}'(T_{a'} \leq n_T)$$

and recall we want the probability above to stay away from 1. The only term that can grow is in the exponent $e^{n_T(D(P||P')+\epsilon)}$. So the largest we can make n_T without this growing is by taking

$$n_T = (1 - 2\eta) \frac{\log(T)}{D(P||P') + \epsilon}.$$

This gives

$$P(T_a \leq n_T) \leq \delta_T + o(T^{-\eta}) = o(1).$$

which applied to the bound (3.12) gives

$$\mathbb{E}[T_a] \geq n_T(1 - o(1)) = (1 - 2\eta) \frac{\log(T)}{(D(P||P') + \epsilon)} (1 - o(1))$$

Thus

$$\liminf_{T \rightarrow \infty} \frac{\mathbb{E}[T_a]}{\log T} \geq \frac{1}{D(P||P_\star)}$$

which is the required bound on $\mathbb{E}[T_a]$. (Some technical details: Here after taking the limit we set η to zero, ϵ to zero and set P' to P_\star . We can do this because the bound holds for all $\eta > 0$ and $\epsilon > 0$. Also a' is an arbitrary parameter such that $r' > r_\star$ so we apply our assumption that $D(P_a||P_{a'}) \rightarrow D(P_a||P_\star)$ as $a' \rightarrow a^\star$.)

Finally, for the regret bound we recall from Lemma 17 that

$$\mathcal{Rg}(T) = \sum_{a \notin \mathcal{A}^\star} \Delta_a \mathbb{E}[T_a]$$

Thus applying the bound on each term $\mathbb{E}[T_a]$ gives the regret bound

$$\liminf_{T \rightarrow \infty} \frac{\mathcal{Rg}(T)}{\log T} \geq \sum_{a \in \mathcal{A}} \frac{\Delta_a}{D(P_a||P_\star)}.$$

□

3.4 Gittins' Index Theorem*

There are many formulations of the multi-arm bandit problem (MAB). The Gittins' Index theorem considers a MDP formulation of MAB. Hence there is less focus on learning aspects. (Though that of course can be added in later.) Although perhaps the statistical approach is most relevant to us, the MDP formulation that we consider here has a particularly elegant solution.

We consider the following problem:

- There are bandit arms $a = 1, \dots, N$ each in state x_t^a at time $t \in \mathbb{Z}_+$. For each a , the set \mathcal{X}_a gives the (finite, disjoint) set of states of each bandit and \mathcal{X} is the union of these sets.
- If bandit a is chosen at state x at time t then the bandit is played for $\sigma_a(x)$ units of time (an i.i.d. random variables chosen each time bandit a is selected in state x). A reward $r_a(x)$ is received at each time for the following $\sigma_a(x)$ units time and afterwards the bandit moves to state y with probability $P_a(x, y)$. All other arms remain unchanged.
- We let a_t be the index of the bandit that is currently active and we let x_t be its state.
- Started from $x \in \prod_a \mathcal{X}_a$ and with discount factor $\beta \in (0, 1)$, the objective is to choose $a_t, t \in \mathbb{Z}_+$ to maximize the expected discounted reward process:

$$R(x) = \mathbb{E}_x \left[\sum_{t=0}^{\infty} r_{a_t}(x_t) \beta^t \right].$$

The above problem is a Markov Decision Processes which we will see has a solution with special structure. Note the probabilities P_a are assumed to be known. So this is not a statistical setup.

Remark 138 (A deterministic MAB). *To get quickly get a flavour for this MAB setting and its solution, suppose we know the current reward of each arm and each arm is played for one unit of time, and assume that we have no information about the future rewards of the arms (not even their distribution). Since there is no information about future rewards, you might as well play the arm with the highest indexed reward. All other arms remain paused whilst it is played. So,*

to avoid comparing with arms at each step, you might as well keep playing this arm until this index is less than the index when you originally started playing the arm. When the index goes below its initial level, you can reassess the indices and continue as described.

The stochastic setting is similar to the above example, there is an index with each arm and we only need to switch arms after the index goes below its initial value. However, since we have information about future rewards in the stochastic setting, we cannot use the instantaneous reward as an index (e.g. we might know that a loss today leads to a big win tomorrow). So a different index is required...

Let's quantify the reward need required to switch arms. If playing arm a repeatedly completing this play after τ time units opposed to receiving a fixed reward γ for that time period, i.e. we compare

$$e_x(\tau) = \mathbb{E}_x \left[\sum_{t=0}^{\tau-1} r_a(x_t^a) \beta^t \right] \quad \text{with} \quad f_x(\tau) = \mathbb{E} \left[\sum_{t=0}^{\tau-1} \gamma \beta^t \right] \quad (3.18)$$

Basically, the first term is bigger, then it is worth trying out a for a while. When we say "for a while", we mean play a until first time the state x_t hits, at time τ , some specified set of bad values. Let \mathcal{T} be the set of such stopping times. Now, given τ then we could chose the value of γ where the two above terms are equal. In some sense, this gives the value for playing a until stopping at time τ . The best value is achieved by maximizing over \mathcal{T} , giving the following index

$$G(x) = \max_{\tau \in \mathcal{T}} \frac{\mathbb{E}_x \left[\sum_{t=0}^{\tau-1} r_a(x_t^a) \beta^t \right]}{\mathbb{E}_x \left[\sum_{t=0}^{\tau-1} \beta^t \right]} \quad (3.19)$$

We call $G(x)$ the *Gittin's Index* of arm a in state x . Observe

- The Gittin's index for bandit a does not depend on the states of other bandits.
- In (3.19), the stopping time solving the maximization is given by $\tau = \min\{t > 0 : G(x_t^a) \leq G(x)\}$ as we now prove

Proof. The index $\gamma = G(x)$ gives the biggest ratio between e_x and f_x (see (3.18)). So, $0 \geq e_x(\tau) - \gamma f_x(\tau)$ is negative and is achieves

its maximum value of zero at a maximizer of (3.19). So

$$\begin{aligned} 0 &= \max_{\tau \in \mathcal{T}} \mathbb{E}_x \left[\sum_{t=0}^{\tau-1} r_a(x_t^a) \beta^t - \sum_{t=0}^{\tau-1} \gamma \beta^t \right] \\ &= \mathbb{E}_x \left[(r_a(x_0^a) - \gamma) + \beta \sum_{x_1 \in \mathcal{X}} P_a(x, x_1) \cdot \max_{\tau \in \mathcal{T}} \mathbb{E}_{x_1} \left[\sum_{t=0}^{\tau-1} (r_a(x_t^a) - \gamma) \beta^t \right] \right] \end{aligned}$$

The second inequality, emphasizes that the stopping time problem is a Markov Decision Process, with two decisions stop or continue. Since a value of zero can always be achieved by stopping, it is only worth continuing from a state x' if there exists a $\tau \in \mathcal{T}$ which gives a positive future reward. I.e. (given the calculation with γ above) we stop when $G(x') \leq \gamma = G(x)$, as required. \square

- The Gittin's indices for an arm, a , can be calculated in with the following algorithm

Calculating Gittin's Indices

1. Let $x^* \in \operatorname{argmax}_{x \in \mathcal{X}_a} r_a(x)$ and take $G(x^*) = r_a(x^*)$.
2. We re-adjust the transition probabilities for $x, y \neq x^*$ so that x^* is skipped over:

$$\tilde{P}_a(x, y) = P_a(x, y) + P_a(x, x^*) \frac{P_a(x^*, y)}{1 - P_a(x^*, x^*)} \quad (3.20)$$

Let $\tilde{\sigma}_a(x)$ be the first time the bandit moves to a state not equal to x^* (started from x) and define

$$\tilde{r}_a(x) = \frac{\mathbb{E}_x \left[\sum_{t=0}^{\tilde{\sigma}_a(x)-1} r_a(x_t^a) \beta^t \right]}{\mathbb{E}_x \left[\sum_{t=0}^{\tilde{\sigma}_a(x)-1} \beta^t \right]}, \quad x \in \mathcal{X}_a \setminus \{x^*\}. \quad (3.21)$$
3. Reset \mathcal{X}_a to $\mathcal{X}_a \setminus \{x^*\}$, P_a to \tilde{P}_a , $\sigma_a(\cdot)$ to $\tilde{\sigma}_a(\cdot)$ and $r_a(\cdot)$ to $\tilde{r}_a(\cdot)$ and repeat from step 1, until $\mathcal{X}_a = \emptyset$.

The main result about this index is the following

Theorem 21 (Gittin's Index Theorem). *For the Multiarmed bandit problem (above), the optimal policy is to play at each time the Bandit whose arm has the highest index.*

Proof. We prove the results in the following three steps. Step 1, we argue by an interchange argument that if there is a bandit with the best possible reward is available then, due to our time discount, its is optimal to play this arm now rather than later. Step 2, we apply a reduction that sequentially removes and orders these states under an optimal strategy (this yields the Gittin's Index and the above algorithm). Step 3, we argue that the indices found are increasing in order, and so the optimal policy always chooses the arm with the highest Gittin's index.

Step 1: Suppose that π is an optimal policy, and that the system starts in state $(x_a : i \in \mathcal{I})$ and bandit a^* is in a state x^* maximizing $r_a(x)$ over $a \in \mathcal{I}$ and $x \in \mathcal{X}$. We now argue that it is always optimal to play arm a^* first rather than later.

Let π^* be the policy that plays a^* first, then after exiting state x^* plays π as the policy would have played from time 0. We let π^* do this until π first plays a^* , at time τ ($\tau = \infty$ is in principle possible, if π never plays a^*). At this point, the policy π^* skips the play of a^* and follows hereafter the behaviour of π . This leads to comparing the rewards under π and π^* , which only differ up to time $\tau + \sigma_{i^*}(x^*)$, and are respectively:

$$\mathbb{E}_x^\pi \left[\sum_{t=0}^{\tau-1} r_{\pi_t}(x_t) \beta^t + \beta^\tau \sum_{t=0}^{\sigma_{i^*}(x^*)-1} r_{i^*}(x^*) \beta^t \right]$$

and

$$\mathbb{E}_x^\pi \left[\sum_{t=0}^{\sigma_{i^*}(x^*)-1} r_{i^*}(x^*) \beta^t + \beta^{\sigma_{i^*}(x^*)} \sum_{t=0}^{\tau-1} r_{\pi_t}(x_t^{\pi_t}) \beta^t \right].$$

the latter term is bigger than the form, because after a bit of algebra, this is equivalent to the inequality

$$r_{i^*}(x^*) \geq \frac{\mathbb{E}_x^\pi \left[\sum_{t=0}^{\tau-1} r_{\pi_t}(x_t^{\pi_t}) \beta^t \right]}{\mathbb{E}_x^\pi \left[\sum_{t=0}^{\tau-1} \beta^t \right]},$$

which holds by assumption of a^*, x^* . This shows that there is an optimal policy that chooses a^* whenever it is in state x^* .

Step 2: We can reduce the problem and remove x^* from the state-space of bandit a^* , by performing one iterations of the algorithm above . Notice the reward from one bandit play on \tilde{r} , (3.21), in the reduced problem is equivalent to the reward from r for under a policy with x^* is given priority

$$\tilde{\mathbb{E}}_x \left[\sum_{t=0}^{\bar{o}_a(x)-1} \tilde{r}_a(x) \beta^t \right] = \mathbb{E}_x \left[\sum_{t=0}^{\bar{o}_a(x)-1} r_{\pi_t}(x_t) \beta^t \right].$$

So when we play with bandit rewards \tilde{r} at x , we are effectively playing with rewards r as it visits x and any subsequent visits to x^* . Since x^* has priority under an optimal policy, an optimal policy on the reduced problem recovers the optimal policy under the original (unreduced) problem, and the average rewards are the same. Thus by induction, we can reduce the problem to until a states remain. This single state system is clearly optimal, thus the priority that can be derived (backward) though the stages of this induction must be optimal. It remains to show that this policy corresponds to the Gittins' index policy.

Step 3: Notice that the reduced reward \tilde{r} involves taking an average of rewards $r_a(x)$, $x \in \mathcal{X}_a$, which are less than $r_a(x^*)$ by the definition of x^* . Thus, $r_a(x) \geq \tilde{r}_a(x)$ at each stage of the reduction. We can argue that under the reduction argument, the reward of the state removed is the Gittins index of that state. It is clear that the state with maximum reward is the state with highest Gittins index. So the first step of this repeated reduction, defines a stopping rule for each state that says keep playing until you reach a state with index lower that x^* and thus the highest reduced reward, if payed until the stopping rule is executed, then has rewards are equal to the original bandit if played until an index lower than this reduced reward. In otherwords, the highest reduced reward is the Gittin's index for that bandit. \square

3.5 Non-Stochastic Multi-armed Bandits*

We now consider a slightly less classical formulation of the Multi-arm bandit problem. Up until now we assumed rewards or costs were generated at random. However, it is possible to consider the setting where there is some arbitrary mechanism generating costs. In this case it is possible to design algorithms that do roughly as well as choosing the best arm.

The Setup. We consider the following setting. There are a finite set of arms \mathcal{A} . There is an environment that chooses outcomes ω from some set Ω . We do not specify the specific mechanism that chooses outcomes.⁴ At each time, after choosing an arm a , an outcome ω occurs and you receives a cost $c(a; \omega) \in [0, 1]$.⁵ We assume that the set of arms in of size N .

A policy π_t choses arm is a function of the previous actions chosen $a_s, s = 1, \dots, t - 1$ and their costs $c(a_s; \omega_s)$ I.e. you cannot observe what cost would have happened if you had chosen a different action.

Interaction: policy and environment. A policy and environment do the following at each time t :

1. The environment chooses $\omega_t \in \Omega$ (unknown to the policy)
2. Policy choses an action $\pi_t \in \mathcal{A}$
3. The cost $c(\pi_t, \omega_t)$ is revealed to the policy.

Regret. The regret of a policy is defined to be

$$\mathcal{R}g(T) = \mathbb{E} \left[\sum_{t=1}^T c(\pi_t, \omega_t) \right] - \min_{a \in \mathcal{A}} \sum_{t=1}^T c(a, \omega_t)$$

I.e. The regret is the additional cost incurred by following policy π rather than the best arm in \mathcal{A} . The expectation above is taken over any randomization employed by out policy to choose arms. As previously, a good algorithm should (at least) have regret that exhibits sub-linear growth in its regret.

⁴The environment is arbitrary other than to say that it is unaware of any randomization that we may employ in the design of our algorithm.

⁵We do our analysis for costs rather than rewards as the maths is more clean this way.

Exponential Explore Exploit. We now describe an algorithm has sub-linear regret for bandit problems. It is known as *Exponential Explore Exploit* or *Exp³*, for short. It is the bandit analogue of an algorithm called the exponential weighted forecaster. This, in turn, can be thought of as an online convex optimization algorithm, which we discuss in Section ??.

Exp³ depends on a parameter η and maintains weights $w_t(a)$ and from these constructs probabilities

$$P_t(a) = \frac{w_t(a)}{\sum_{a' \in \mathcal{A}} w_t(a')}.$$

We select arm π_t with these probabilities. After we play arm π_t and receive cost $c(\pi_t, \omega_t)$, we can construct an unbiased estimator of the cost for each arm

$$\hat{c}_t(a; \omega_t) = \frac{c(a; \omega_t)}{P_t(a)} \mathbb{I}[\pi_t = a].$$

(Observe that $\mathbb{E}_{\pi_t}[\hat{c}_t(a; \omega_t)] = c(a, \omega_t)$.) With this estimates we can update the weights as follows

$$w_{t+1}(a) = e^{-\eta \hat{c}_t(a)} w_t(a).$$

The algorithm is given in the text box below.

Exponential Explore-Exploit, *Exp³*

For parameter $\eta > 0$ and costs $0 \leq c(a; \omega) \leq 1$, *Exp³* is a randomized policy choses action π_t at time t according to weights $w_1(a) = 1$

$$w_t(a) = e^{-\eta \hat{c}_{t-1}(a)} w_{t-1}(a) \quad \text{and} \quad W_t = \sum_{a \in \mathcal{A}} w_t(a)$$

and $\pi_t = a$ with probability

$$P_t(a) = \frac{w_t(a)}{W_t}.$$

Here

$$\hat{c}_t(a; \omega_t) = \frac{c(a; \omega_t)}{P_t(a)} \mathbb{I}[\pi_t = a].$$

Regret bound. We prove the following

Theorem 22. *For the Exp³ algorithm with parameter η the following bound holds*

$$\mathcal{Rg}(T) \leq \frac{1}{\eta} \log N + \frac{\eta}{2} TN$$

and for appropriate η

$$\mathcal{Rg}(T) \leq 2 \sqrt{2TN \log N}.$$

Proof. We collect together a few facts, specifically (3.22)–(3.26), then the proof proceeds by analyzing and bounding the change in the sum of the weights.⁶

First note that by a Taylor expansion that it holds that for $c \geq 0$

$$e^{-c} \leq 1 - l + \frac{c^2}{2}. \quad (3.22)$$

and

$$\log(1 + x) \leq x. \quad (3.23)$$

As discussed above note that

$$\mathbb{E}_{\pi_t}[\hat{c}_t(a; \omega_t)] = c(a; \omega_t) \quad (3.24)$$

Also it holds that

$$P_t(a) \mathbb{E}[\hat{c}_t(a, \omega_t)^2] \leq 1 \quad (3.25)$$

because

$$P_t(a) \mathbb{E}_{\pi_t}[\hat{c}_t(a, \omega_t)^2] = P_t(a) \cdot P_t(a) \frac{c(a; \omega)^2}{P_t(a)^2} = c(a; \omega)^2 \leq 1.$$

Finally, note that

$$\sum_{a \in \mathcal{A}} P_t(a) \hat{c}_t(a; \omega) = c(\pi_t; \omega) \quad (3.26)$$

because

$$\sum_a P_t(a) \hat{c}_t(a; \omega_t) = \sum_a P_t(a) \frac{c(a; \omega_t)}{P_t(a)} \mathbb{I}[\pi_t = a] = c(\pi_t; \omega_t).^7$$

⁶This proof is similar to the proof of the weighted majority algorithm.

⁷A subtle point to note here is that we are *not* taking an expectation with respect to π_t but we are just taking a sum weighted by $P_t(a)$.

We now have note the elementary bounds that we need for the main proof. We analyse $W_t = \sum_a w_t(a)$, as follows,

$$\begin{aligned} \frac{W_{t+1}}{W_t} &= \sum_{a \in \mathcal{A}} \frac{w_{t+1}(a)}{W_t} = \sum_{a \in \mathcal{A}} \frac{w_t(a)}{W_t} e^{-\eta \hat{c}_t(a; \omega_t)} \\ &\leq \sum_{a \in \mathcal{A}} P_t(a) \left(1 - \eta \hat{c}_t(a; \omega_t) + \frac{\eta^2}{2} \hat{c}_t(a; \omega_t)^2 \right) \\ &= 1 - \eta c(\pi_t, \omega_t) + \frac{\eta^2}{2} \sum_{a \in \mathcal{A}} P_t(a) \hat{c}_t(a; \omega_t)^2. \end{aligned} \quad (3.27)$$

The inequality applies the Taylor expansion (3.22) which we can apply since the costs are bounded above by 1 from (3.24). The final equality applies (3.26).

Now taking logs in (3.27), applying inequality (3.23), and summing for $t = 1, \dots, T$ gives

$$\log W_{T+1} = \sum_{t=1}^T \log \frac{W_{t+1}}{W_t} \leq -\eta \sum_{t=1}^T c(\pi_t, \omega_t) + \frac{\eta^2}{2} \sum_{t=1}^T \sum_{a \in \mathcal{A}} P_t(a) \hat{c}_t(a; \omega_t)^2.$$

We can also lower bound $\log W_{T+1}$ as follows

$$\log W_{T+1} = \log \sum_a w_t(a) \geq \log \bar{w}_t(a) = \eta \sum_{t=1}^T \hat{c}_t(a; \omega_t)$$

Combining the upper bound and the lower bound on $\log W_{T+1}$ gives

$$\sum_{t=1}^T \hat{c}_t(\pi_t; \omega_t) - \sum_{t=1}^T \hat{c}_t(a; \omega_t) \leq \frac{1}{\eta} \log N + \eta \sum_{t=1}^T \sum_{a \in \mathcal{A}} P_t(a) \hat{c}_t(a; \omega_t)^2$$

Taking expectation on both sides above and applying the equality (3.24) and the inequality (3.25) gives

$$\begin{aligned} \mathcal{Rg}(T) &= \sum_{t=1}^T \mathbb{E}_{\pi_t} [c_t(\pi_t; \omega_t)] - \sum_{t=1}^T c_t(a; \omega_t) \\ &\leq \frac{1}{\eta} \log N + \frac{\eta}{2} \sum_{t=1}^T \sum_{a \in \mathcal{A}} P_t(a) \mathbb{E}_{\pi_t} [\hat{c}_t(a; \omega_t)^2] \\ &\leq \frac{1}{\eta} \log N + \eta TN, \end{aligned}$$

as required. Notice that if we minimize the above bound over η by taking $\eta = \sqrt{2 \log N / TN}$. This give the second bound required in the theorem. \square

3.6 Stochastic Regression

We consider the following formulation of Lai, Robbins and Wei (1979), and Lai and Wei (1982). Consider the following regression problem,

$$y_t = \beta^\top x_t + \epsilon_t$$

for $t = 1, 2, \dots$ where ϵ_t are unobservable random errors and $\beta = (\beta_1, \dots, \beta_p) \in \mathbb{R}^p$ are unknown parameters.

Typically for a regression problem, it is assumed that inputs x_1, \dots, x_t are given and errors are IID random variables. However, we now want to consider a setting where we sequentially chose inputs $x_t \in \mathbb{R}^p$ and then get observations $y_t \in \mathbb{R}$ and errors ϵ_i are a martingale difference sequence with respect to the filtration generated by $\{x_i, y_{i-1} : i \leq t\}$.

We let $X_t = (x_{ij} : i = 1, \dots, t, j = 1, \dots, p)$ be the matrix of inputs and $y_t = (y_i : 1 \leq i \leq t)$ be the matrix of outputs. Further we let b_t be the least squares estimate of β given X_t and y_t . The following result gives a condition on the eigenvalues of the design matrix $X_t^\top X_t$ for b_t to converge to β and also gives a rate of convergence.

Thrm 139. *If $\lambda_{\min}(t)$ and $\lambda_{\max}(t)$ are, respectively, the minimum and maximum eigenvalues of the design matrix $X_t^\top X_t$ and if we assume that almost surely, $\sup_t \mathbb{E}[\epsilon_t^4 | \mathcal{F}_{t-1}] < \infty$,⁸ then whenever we have*

$$\lambda_{\min}(t) \xrightarrow[n \rightarrow \infty]{} \infty \quad \text{and} \quad \frac{\log(\lambda_{\max}(t))}{\lambda_{\min}(t)} \xrightarrow[t \rightarrow \infty]{} 0$$

then b_t converges to β and

$$\|b_t - \beta\|^2 = O\left(\frac{\log(\lambda_{\max}(t))}{\lambda_{\min}(t)}\right).$$

In what follows, $\|\cdot\|$ is the Euclidean norm for a vector and for a matrix $\|A\| = \sup_{v: \|v\|=1} \|Av\|$. (Note that it is well known that $\|A\| = \lambda_{\max}(A)$ the maximum eigenvalue of A .)

Proof. The least squares estimate to the above regression problem is given by

$$b_t := (X_t^\top X_t)^{-1} X_t^\top y_t \quad \text{and} \quad \beta := (X_t^\top X_t)^{-1} X_t^\top (y_t - \epsilon_t).$$

⁸This assumption can be weakened but the proof is a little cleaner with a 4th moment.

So $\mathbf{b}_t - \boldsymbol{\beta} = (\mathbf{X}_t^\top \mathbf{X}_t)^{-1} \mathbf{X}_t^\top \boldsymbol{\epsilon}_t$ where $\boldsymbol{\epsilon}_t = (\epsilon_i : i = 1, \dots, t)$. To prove the above theorem first note that

$$\begin{aligned} \|\mathbf{b}_t - \boldsymbol{\beta}\|^2 &= \left\| (\mathbf{X}_t^\top \mathbf{X}_t)^{-1} \sum_{i=1}^t \mathbf{x}_i \epsilon_i \right\|^2 \\ &\leq \|(\mathbf{X}_t^\top \mathbf{X}_t)^{-1/2}\|^2 \left\| (\mathbf{X}_t^\top \mathbf{X}_t)^{-1/2} \sum_{i=1}^t \mathbf{x}_i \epsilon_i \right\|^2 \\ &= \lambda_{\min}(t)^{-1} \times \underbrace{\boldsymbol{\epsilon}_t^\top \mathbf{X}_t (\mathbf{X}_t^\top \mathbf{X}_t) \mathbf{X}_t^\top \boldsymbol{\epsilon}_t}_{=: Q_t} \end{aligned} \quad (3.28)$$

In the inequality above we apply the Cauchy-Schwartz inequality. We bound Q_t using the Sherman-Morrison formula. Specifically in Proposition 140, below, it is shown that

$$Q_T - Q_0 + a_T = o(a_T) + \sum_{k=0}^T \epsilon_k^2 \mathbf{x}_k^\top V_k \mathbf{x}_k$$

where a_T is some positive increasing sequence. So eventually it holds that

$$Q_T \leq Q_0 + \sum_{k=0}^T \epsilon_k^2 \mathbf{x}_k^\top V_k \mathbf{x}_k. \quad (3.29)$$

So the convergence is determine by the rate of convergence of the sequence $\sum_{k=0}^T \mathbf{x}_k^\top V_k \mathbf{x}_k$. Specifically in Lemma 20 we show that

$$\sum_{k=0}^T \epsilon_k^2 \mathbf{x}_k^\top V_k \mathbf{x}_k \leq \sigma^2 \sum_{k=0}^T \mathbf{x}_k^\top V_k \mathbf{x}_k + o\left(\sum_{k=0}^T \mathbf{x}_k^\top V_k \mathbf{x}_k\right) + O(1). \quad (3.30)$$

In Lemma 19, we also show that

$$\sum_{k=p}^T \mathbf{x}_k^\top V_k \mathbf{x}_k \leq p \log \lambda_{\max}(T) - c \quad (3.31)$$

for some constant c . Combining (3.29), (3.30), and (3.31) gives

$$Q_T \leq Q_0 + \sigma^2 p \log \lambda_{\max}(T) + o\left(\log \lambda_{\max}(T)\right) + O(1).$$

I.e. $Q_t \leq O(\log \lambda_{\max}(t))$. Applying this to (3.28), we arrive at a bound of the required form

$$\|\mathbf{b}_t - \boldsymbol{\beta}\|^2 \leq O\left(\frac{\log(\lambda_{\max}(t))}{\lambda_{\min}(t)}\right).$$

□

In what follows we must study the asymptotic behavior of Q_t . What we will show is

Prop 140. *Almost surely*

$$Q_T - Q_p = o(a_T) - a_T + O(1) + \sum_{t=p+1}^T \epsilon_t^2 \mathbf{x}_t^\top V_t \mathbf{x}_t$$

where a_T is a positive, increasing sequence.

Proof. To prove this proposition we will require some lemmas, such as the Sherman-Morris formula. These are stated and proven after the proof of this result.

The Sherman-Morrison Formula states that:

$$(A + uv^\top)^{-1} = A^{-1} - \frac{A^{-1}uv^\top A^{-1}}{1 + v^\top A^{-1}u}.$$

Note that

$$V_t := (X_t^\top X_t)^{-1} = (X_{t-1}^\top X_{t-1} + \mathbf{x}_t \mathbf{x}_t^\top)^{-1} = V_{t-1} - \frac{V_{t-1} \mathbf{x}_t \mathbf{x}_t^\top V_{t-1}}{1 + \mathbf{x}_t^\top V_{t-1} \mathbf{x}_t}.$$

Thus

$$\begin{aligned} Q_t &= \epsilon_t^\top X_t V_t X_t^\top \epsilon_t \\ &= \epsilon_{t-1}^\top X_t V_t X_t^\top \epsilon_{t-1} + 2\epsilon_{t-1}^\top X_t V_t X_t^\top \epsilon_t + \epsilon_t^2 \mathbf{x}_t^\top V_t \mathbf{x}_t \\ &= \epsilon_{t-1}^\top X_t \left(V_{t-1} - \frac{V_{t-1} \mathbf{x}_t \mathbf{x}_t^\top V_{t-1}}{1 + \mathbf{x}_t^\top V_{t-1} \mathbf{x}_t} \right) X_t^\top \epsilon_{t-1} \\ &\quad + 2\epsilon_{t-1}^\top X_t \left(V_{t-1} - \frac{V_{t-1} \mathbf{x}_t \mathbf{x}_t^\top V_{t-1}}{1 + \mathbf{x}_t^\top V_{t-1} \mathbf{x}_t} \right) X_t^\top \epsilon_t \\ &\quad + \epsilon_t^2 \mathbf{x}_t^\top V_t \mathbf{x}_t \\ &= Q_{t-1} - \frac{(\mathbf{x}_t^\top V_{t-1} X_t^\top \epsilon_{t-1})^2}{1 + \mathbf{x}_t^\top V_{t-1} \mathbf{x}_t} \\ &\quad + 2\epsilon_{t-1}^\top X_t V_{t-1} X_t^\top \epsilon_t \left(\frac{1}{1 + \mathbf{x}_t^\top V_{t-1} \mathbf{x}_t} \right) \\ &\quad + \epsilon_t^2 \mathbf{x}_t^\top V_t \mathbf{x}_t \end{aligned}$$

Thus summing and rearranging a little

$$\begin{aligned} Q_T - Q_p + \underbrace{\sum_{t=p+1}^T \frac{(\mathbf{x}_t^\top V_{t-1} X_t^\top \epsilon_{t-1})^2}{1 + \mathbf{x}_t^\top V_{t-1} \mathbf{x}_t}}_{=a_T} \\ = 2 \sum_{t=p+1}^T x_t V_{t-1} X_t^\top \epsilon_{t-1} \left(\frac{1}{1 + \mathbf{x}_t^\top V_{t-1} \mathbf{x}_t} \right) \epsilon_t + \sum_{t=p+1}^T \epsilon_t^2 \mathbf{x}_t^\top V_t \mathbf{x}_t \end{aligned}$$

Notice in the above, the first summation (before the equals sign) only acts to decrease Q_T , while on the right hand side, the first term is a martingale difference sequence and the second term is a quadratic form.

We recall from the results on Martingales stated in Section A.1, that for any L^2 -martingale it holds, almost surely, that

$$M_t = o(\langle M \rangle_t) + O(1)$$

where $\langle M \rangle_t - \langle M \rangle_{t-1} = \mathbb{E}[(M_t - M_{t-1})^2 | \mathcal{F}_t]$. Applying this to our case, the Martingale difference sequence is a L^2 martingale and so we have that

$$\begin{aligned} 2 \sum_{t=p+1}^T x_t V_{t-1} X_t^\top \epsilon_{t-1} \left(\frac{1}{1 + \mathbf{x}_t^\top V_{t-1} \mathbf{x}_t} \right) \epsilon_t &= o \left(\sum_{t=p+1}^T \frac{(x_t V_{t-1} X_t^\top \epsilon_{t-1})^2}{(1 + \mathbf{x}_t^\top V_{t-1} \mathbf{x}_t)^2} \right) + O(1) \\ &= o(a_T) + O(1) \end{aligned}$$

In the second equality above, we use that $(1 + \mathbf{x}_t^\top V_{t-1} \mathbf{x}_t)^{-1} \leq 1$. Thus we have that

$$Q_T - Q_p + a_T = o(a_T) + O(1) + \sum_{t=p+1}^T \epsilon_t^2 \mathbf{x}_t^\top V_t \mathbf{x}_t.$$

□

Lemma 18 (Sherman-Morrison Formula). *For an invertible Matrix A and two vectors u and v*

$$(A + uv^\top)^{-1} = A^{-1} - \frac{A^{-1}uv^\top A^{-1}}{1 + v^\top A^{-1}u} \quad (\text{Sherman-Morrison})$$

Proof. Recalling that the outer-product of two vectors wv^\top is the matrix $(w_i v_j)_{i=1, j=1}^{t, t}$ it holds that

$$(wv^\top)(wv^\top) = (w^\top v)(wv^\top)$$

(Nb. This is matrix multiplication: each column is a constant times u and every row is a constant time v , so the dot product comes out.)

Using this identity note that

$$\begin{aligned} \left(I - \frac{wv^\top}{1 + v^\top w}\right) \left(1 + wv^\top\right) &= I + wv^\top - \frac{wv^\top}{1 + v^\top w} + \frac{1}{1 + v^\top w} (wv^\top)(wv^\top) \\ &= I + wv^\top - \frac{wv^\top}{1 + v^\top w} [1 + w^\top v] \\ &= I. \end{aligned}$$

So $(I + wv^\top)^{-1} = I - \frac{wv^\top}{1 + v^\top w}$. Now letting $u = Aw$,

$$(A + uv^\top)^{-1} = (I + wv^\top)^{-1} A^{-1} = \left(I - \frac{wv^\top}{1 + v^\top w}\right) A^{-1} = A^{-1} + \frac{A^{-1} u v^\top A^{-1}}{1 + v^\top A^{-1} u}.$$

as required. \square

The following in some sense repeatedly analyses to the determinant under the Sherman-Morrison formula.

Lemma 19. *If w_1, w_2, \dots are a sequence of vectors and we let $A_t = \sum_{k=1}^t w_k^\top w_k$ then*

$$w_k^\top A w_k \leq 1$$

and

$$\sum_{k=p+1}^T w_k^\top A_k^{-1} w_k \leq p \log \lambda_{\max}(T) - \log |A_p|.$$

Proof. First note that if $A = B + w^\top w$ then, as was also in the Sherman-Morrison formula

$$|B| = |A - w^\top w| = |A| (1 - w^\top A^{-1} w)$$

Thus

$$w^\top A w = \frac{|A| - |B|}{|A|}$$

which should remind you of the derivative of the logarithm. Also this implies $w^\top A w \leq 1$ as requires. (Also note that this tells us that

determinant is increasing and that $w^\top A w \leq 1$.) If we apply this to the above sum and apply the concavity of the logarithm

$$\begin{aligned} \sum_{k=p+1}^T \mathbf{w}_k^\top A_k^{-1} \mathbf{w}_k &= \sum_{k=p+1}^T \frac{|A_k| - |A_{k-1}|}{|A_k|} \leq \sum_{k=p+1}^T \log |A_k| - \log |A_{k-1}| \\ &= \log |A_T| - \log |A_p| \end{aligned}$$

Since $|A|$ is the product of all eigenvalues $\lambda_{\max}(T)^p \geq |A_T|$. So we see that

$$\sum_{k=p+1}^T \mathbf{w}_k^\top A_k^{-1} \mathbf{w}_k \leq p \log \lambda_{\max}(T) - \log |A_p|,$$

which then gives the result. \square

Lemma 20.

$$\sum_{k=0}^T \epsilon_t^2 \mathbf{x}_t^\top V_t \mathbf{x}_t = \sum_{k=0}^T \mathbf{x}_t^\top V_t \mathbf{x}_t + o\left(\sum_{k=0}^T \mathbf{x}_t^\top V_t \mathbf{x}_t\right) + O(1)$$

Proof. We let $\sigma^2 = \max_t \mathbb{E}[\epsilon_t^2 | \mathcal{F}_t]$ (which we know is finite since, by assumption, the 4th moment is finite).

$$\begin{aligned} \sum_{k=0}^T \epsilon_t^2 \mathbf{x}_t^\top V_t \mathbf{x}_t &= \sum_{k=0}^T (\epsilon_t^2 - \mathbb{E}[\epsilon_t^2 | \mathcal{F}_t]) \mathbf{x}_t^\top V_t \mathbf{x}_t + \sum_{k=0}^T \mathbb{E}[\epsilon_t^2 | \mathcal{F}_t] \mathbf{x}_t^\top V_t \mathbf{x}_t \\ &\leq \sum_{k=0}^T \delta_t \mathbf{x}_t^\top V_t \mathbf{x}_t + \sigma^2 \sum_{k=0}^T \mathbf{x}_t^\top V_t \mathbf{x}_t. \end{aligned} \quad (3.32)$$

Here we define $\delta_t = \epsilon_t^2 - \mathbb{E}[\epsilon_t^2 | \mathcal{F}_{t-1}]$. Notice since we assume $\sup_t \mathbb{E}[\epsilon_t^4 | \mathcal{F}_{t-1}] < \infty$ then δ_t is L^2 bounded.

We recall Section A.1, that for any L^2 -martingale it holds, almost surely, that $M_t = o(\langle M \rangle_t) + O(1)$ where $\langle M \rangle_t - \langle M \rangle_{t-1} = \mathbb{E}[(M_t - M_{t-1})^2 | \mathcal{F}_t]$, and we recall that $\mathbf{x}_t^\top V_t \mathbf{x}_t \leq 1$, which is proven below in Lemma 19. From this it follows

$$\begin{aligned} \sum_{k=0}^T \delta_t \mathbf{x}_t^\top V_t \mathbf{x}_t &= o\left(\sum_{k=0}^T (\mathbf{x}_t^\top V_t \mathbf{x}_t)^2\right) + O(1) \\ &= o\left(\sum_{k=0}^T \mathbf{x}_t^\top V_t \mathbf{x}_t\right) + O(1). \end{aligned}$$

Applying this to (3.32) gives the required bound

$$\sum_{k=0}^T \epsilon_t^2 \mathbf{x}_t^\top V_t \mathbf{x}_t = \sum_{k=0}^T \mathbf{x}_t^\top V_t \mathbf{x}_t + o\left(\sum_{k=0}^T \mathbf{x}_t^\top V_t \mathbf{x}_t\right) + O(1).$$

□

Chapter 4

Tabular Reinforcement Learning

4.1 Principles of Reinforcement Learning

- Overview of Reinforcement learning and terminology.
 - Policy evaluation & policy improvement; exploration-exploitation trade-off; model free control; function approximation.
-

First we discuss at a high level a few of the key concepts in Reinforcement learning. These will then be discussed in more precise mathematical detail for specific examples and algorithms in subsequent sections.

Reinforcement Learning: Reinforcement Learning is the setting where we do not know the transition probabilities of a Markov Decision Process (or we might want to approximate a control problem with MDP). For instance, you might be able to simulate a problem with states, actions and rewards but you do not have access to the underlying dynamics of the simulation. Enough information must be gathered to approximate the optimal action for each state.

Policy Evaluation and Policy Improvement: When we look at reinforcement learning algorithms the same principles that applied to MDPs (with known transition probabilities) apply. I.e. we might want to think of the steps of the algorithm either improving the policy:

$$\pi(x) \in \operatorname{argmax}_{a \in \mathcal{A}} \{r(x, a) + \beta \mathbb{E}_{x,a} [R(\hat{x}, \pi_0)]\}$$

or evaluating the reward function of the current policy

$$R(x, \pi) = \mathbb{E}_x^\pi \left[\sum_{t=0}^{\infty} \beta^t r(X_t, \pi(X_t)) \right].$$

Although algorithms might be subject to more noisy estimates.

Exploration-Exploitation trade-off: Because transition probabilities are unknown, when you are at a state, say x , there is a question of whether you should perform the best action a^* given the available information and thus attempt to implement the best known policy; or if you should chose a different (possibly random) action and thus

get better information about the value of that action. I.e. there is a trade-off between doing what is myopically best given the available information (exploitation) and trying something new in case it might be better (exploration). (This is similar to policy evaluation and improvement, but here we are interested in finding the statistical properties of each action rather than performing computations on a function.) Problems that investigate exploration and exploitation tradeoff in isolation are often called Multi-armed Bandit problems, and there is a vast recent literature on these topics as well as a very well developed theoretical basis preceding this.

Model Free Control: Here we are especially interested in methods that are *model free*. A method is model free when it does not require an explicit estimation of the system dynamics, specifically, we don't try to estimate the transition probabilities P_{xy}^a for each action. For instance, if we perform policy improvement based on an estimation of the value function to V ,

$$\pi(x) \in \operatorname{argmax}_{a \in \mathcal{A}} r(x, a) + \sum_{\hat{x}} P_{x\hat{x}}^a V(\hat{x})$$

then this is not model free, because we need to estimate $P_{x\hat{x}}^a$ in addition to our estimate of the value function V . Instead we might consider the Q -function of the MDP. This is the function $Q(x, a)$ which gives the value function for taking action a in state x and then afterward follow the optimal policy. If we perform policy improvement based on an estimation of the Q -function

$$\pi(x) \in \operatorname{argmax}_{a \in \mathcal{A}} Q(x, a)$$

then this is model free. We will discuss this in more detail in the next section.

Function Approximation: If the set of state and actions is moderately small then we can store functions of interest such as the Q -function $Q(x, a)$ as a table (or matrix) in computer memory. These algorithms are often called table based methods. But for larger problems or of problems with continuous state spaces and action spaces, then it is not possible to store this information. Further the likelihood of revisiting exactly the same state twice is vastly reduced. So often we have to infer relationships between states that are "close" and hope that the value function is suitably continuous

that this forms a good approximation. So here we might for instance replace the value function $Q(x, a)$ with some approximation $Q_w(x, a)$ which is of lower dimension than $Q(x, a)$. Here w represents a weights that we use to parameterize our approximation (e.g. we could approximate continuous real valued function with a polynomial). Then we might look to find the best approximation:

$$\min_w \mathbb{E}[(\hat{Q}(x, a) - Q_w(x, a))^2].$$

Here we let $\hat{Q}(x, a)$ be the Q -values of the current policy as observed from the data seen so far and we look to find the weights that give the best approximation. Above we minimize the mean-squared-error of the loss function, but we could consider other metrics and we could approximate other functions e.g. policies $\pi_w(x) \approx \pi_w(x)$.

Further Terminology.

Def 141 (Episode). *When we run a sample path of an MDP under a policy π we call this an episode.*

Here we implicitly assume that each episode terminates, or refreshes after some finite time.

In reinforcement learning we are often fitting functions $R(x, \pi)$, $Q(x, a)$ and $\pi(x)$ using simulation data. Here performing updates of the form

$$R(x) \leftarrow R(x) + \alpha d(\hat{x})$$

We need to specify when and how often we perform these updates. These give different variants of each algorithm that we consider.

Def 142 (Offline and Online update). *If we perform the update (4.1) at the end of each episode simultaneously for each x then we say that the update is offline. If we update (4.1) for each x in the order visited by the episode, then we say this is online.*

Note that we can perform online updates while we simulate an episode, while offline we must wait for the episode to end. Note that the offline algorithm updates are synchronous – we update all components of $R(x)$ simultaneously – while online algorithms asynchronously update.

Def 143 (First Visit and Every Visit update). *If we perform the update (4.1) only once for the first visit to x then we say this is the first visit. If we perform an update (4.1) for each visit to x we say this is the every visit update.*

In an offline every-visit algorithm, we assume an update of the form

$$R(x) \leftarrow R(x) + \sum_{v=1}^N \alpha d(\hat{x}^{(v)})$$

i.e. we only update R once at the end of the episode, but the update applies a term for every visit $v = 1, \dots, N$ to x . While in an online every-visit algorithm, we update

$$R(x) \leftarrow R(x) + \alpha d(x)$$

When we talk about different policy evaluation algorithms we can talk about offline & online and first-visit & every-visit variants. From a theoretical perspective offline first-visit algorithms are easier to deal with. While from an implementation perspective, every-visit online algorithms are more straight-forward to program (as we don't need to remember anything).

References.

The book of Sutton and Barto is the gold standard on reinforcement learning [48]. Though to go a little deeper, I have benefited a lot from reading the more mathematically rigorous text of Bertsekas and Tsitsiklis [7]. Bertsekas has a new book on reinforcement learning which I will likely reference once I have a copy!

4.2 Policy Evaluation: MC and TD methods

- Monte-Carlo and Temporal differences.
 - TD(0), n-step TD, TD(λ).
 - Importance Sampling, Stopping Time and Tree back up.
-

We now begin to consider algorithms for Markov decision problems where the rewards are not known and, now, these need to be estimated either through simulation or data. Recall from Section 0.5 that a MDP algorithm consists of two parts: policy evaluation and policy improvement. Here we begin to see policy evaluation as a statistical procedure rather than just linear algebra.

Our task in this section is to estimate the reward function

$$R(x, \pi) := \mathbb{E}_{x_0} \left[\sum_{t=0}^{\infty} \beta^t r(X_t, \pi_t) \right].$$

for a stationary policy π by generating episodes under the policy π .

Some Terminology

Since our policy π will not change in this section we will often suppress the dependence on π our notation. To estimate $R(\cdot)$, we will be applying updates of the form

$$R(x) \leftarrow R(x) + \alpha d(\hat{x}) \tag{4.1}$$

for each state x , where $\hat{x} = (\hat{x}_0, \hat{x}_1, \dots)$ is the set of states visited from $\hat{x}_0 = x$ onwards and where d is a function of some of these states and the current reward function.

Monte-Carlo Policy Evaluation

Monte-Carlo policy evaluation is the simplest method of evaluating a policy. Here you simply run a number of episodes under a policy and calculate the mean future reward for each state. As is shown in Lemma 21 below, it is not hard to see that we calculate the mean of N data points empirically through the recursion:

$$\bar{R} \leftarrow R + \frac{1}{N}(\tilde{R} - \bar{R}).$$

Def 144 (Monte-Carlo Policy Evaluation). *When x is visited in an episode update*

$$\begin{aligned} N(x) &\leftarrow N(x) + 1 \\ \bar{R}(x) &\leftarrow \bar{R}(x) + \frac{1}{N(x)} (\tilde{R}(x) - \bar{R}(x)) \end{aligned}$$

where

$$\tilde{R}(x) = r(\hat{x}_0) + \beta r(\hat{x}_1) + \dots + \beta^T r(\hat{x}_T)$$

is the observed reward after visiting state x to the end of the episode. Also $N(x)$ is the number of visits to x and $\bar{R}(x)$ is the mean.

This update can be done on every visit to state x or the first time x is visited in an episode. Monte-Carlo Policy Evaluation convergences to the reward function

Proposition 5.

$$\bar{R}(x) \xrightarrow[N(x) \rightarrow \infty]{} R(x) = \mathbb{E}[\tilde{R}(x)].$$

The proof follows immediately from the strong law of large numbers.

Advantages and disadvantages. Monte-Carlo policy evaluation has the advantage that it is simple and intuitive. Further it is an unbiased estimate of the true reward. However, it requires a full episode to perform an update. The variance of a full episode's reward can be quite big.

¹It the environment is continuing you can choose a state to be a "starting state", and assuming that state is recurrent then you can reset the episode every time that state is visited.

Remark 145 (Forgetting the past). We can also perform an update where we don't divide by the number of visits:

$$\bar{R}(x) \leftarrow \bar{R}(x) + \alpha(\tilde{R}(x) - \bar{R}(x)).$$

Note that after N updates we get that:

$$\bar{R}(x) = \alpha\tilde{R}_N + \alpha(1 - \alpha)\tilde{R}_{N-1} + \alpha(1 - \alpha)^2\tilde{R}_{N-2} + \dots + \alpha(1 - \alpha)^{N-1}\tilde{R}_1.$$

This puts a focus on the most recent rewards. (This can be useful if the policy has been changing a small amount over each step, and we care about the more recent information.)

Lemma 21. For data a_1, a_2, \dots , we let \bar{a}_N be the mean of the first N pieces of data, i.e.

$$\bar{a}_N = \frac{1}{N} \sum_{n=1}^N a_n$$

Notice \bar{a}_n obeys the recursion:

$$\bar{a}_{N+1} = \frac{1}{N}(a_{N+1} - \bar{a}_N)$$

Proof. After N iterations, the algorithm update gives

$$\begin{aligned} N \times \bar{a}_N &= N \times \left(\bar{a}_{N-1} + \frac{1}{N} (a_N - \bar{a}_{N-1}) \right) \\ &= a_N + (N - 1)\bar{a}_{N-1} \\ &= \dots = \sum_{n=1}^N \tilde{a}_n. \end{aligned}$$

Above we can repeat the same substitution on $(N - 1)\bar{a}_{N-1}$ as we did for $N\bar{a}_N$. □

Temporal Difference Learning

Like Monte-Carlo the temporal difference method is a way of estimating the value function of a dynamic program. While Monte-Carlo required us to evaluate a whole episode of a simulation. Temporal difference methods cut this short, so that, in principle, we can update our reward estimate at simulation step.

Recall, that for any Markov chain the reward function satisfies

$$R(x) = \mathbb{E}_x[r(x, \hat{x}) + \beta R(\hat{x})] \quad (4.2)$$

or, equivalently,

$$R(x) = R(x) + \mathbb{E} [r(x, \hat{x}) + \beta R(\hat{x}) - R(x)]$$

which is a fixed point of the operation

$$R(x) \leftarrow \mathbb{E} [r(x, \hat{x}) + \beta R(\hat{x}) - R(x)]$$

which, in turn, can be approximated by

$$R(x) \leftarrow r(x, \hat{x}) + \beta R(\hat{x}) - R(x).$$

The update term, above, is called a Temporal Difference, and the algorithm given described is called TD(0). The above recursion is an asynchronous Robbins-Munro step rule and so by Theorem 16 the algorithm converges to the correct reward function. We summarize each of these points below.

Def 146 (Temporal Differences). *The above term*

$$d(x, x') = r(x, x') + \beta R(x') - R(x)$$

is called a temporal difference.

Def 147 (TD(0)). *The algorithm where on every visit to x we perform the update:*

$$R(x) \leftarrow r(x, \hat{x}) + \beta R(\hat{x}) - R(x)$$

is called TD(0). Here TD stands for Temporal Difference.

Theorem 23. *If $\alpha_t(x)$ the learning rate applied in TD(0) at state x after t iterations is such that*

$$\sum_{t=0}^{\infty} \alpha_t(x) = \infty, \quad \sum_{t=0}^{\infty} \alpha_t^2(x) < \infty,$$

and if each state $x \in \mathcal{X}$ is visited infinitely often then the TD(0) reward estimate after t iterations, $R_t(x)$, is such that

$$R_t(x) \xrightarrow[t \rightarrow \infty]{} R(x) := \mathbb{E}_x \left[\sum_{s=0}^{\infty} \beta^s r(x_s, x_{s+1}) \right]$$

where the reward function, satisfying (4.2).

Proof. Notice that $R_t(x)$ follows the update

$$R_{t+1}(x_t) = \alpha_t(x_t) [r(x_t, x_{t+1}) + \beta R(x_{t+1})]$$

and $R_{t+1}(x') = R_t(x')$ for all $x' \neq x_t$. This is an asynchronous Robbins-Munro scheme, and by Theorem 16 convergence almost surely to $R(x)$ satisfying the condition

$$R(x) = \mathbb{E}_x[r(x, \hat{x}) + \beta R(\hat{x})]$$

which by our result of rewards for Markov chains, Proposition 15 (see the proposition subsequent remark too), is equal to $\mathbb{E}_x[\sum_{s=0}^{\infty} \beta^s r(x_s, x_{s+1})]$. \square

Remark 148. *Very similar convergence proof exists for the other TD methods mentioned in this section: n -step TD and TD(λ). For instance the proof of n -step TD is almost identical. The proof for TD(λ) follows in a similarly straightforward manner, we also refer the reader to [49, Section 5.3] for a proof.*

n-Step TD

In (every-visit) Monte-Carlo Policy Evaluation, to estimate the reward we added the whole sequence of future rewards whereas in TD(0) we only add one reward at a time. Therefore we can presume that TD(0) has considerably lower variance, but perhaps at the cost bias in our estimate. We can extend the TD update to include more than one reward.

We can expand out the Bellman equation over n -steps:

$$\begin{aligned}
R(\hat{x}_0) &= R(\hat{x}_0) + \mathbb{E} [r(\hat{x}_0, \hat{x}_1) + \beta R(\hat{x}_1) - R(\hat{x}_0)] \\
&\vdots \\
&= R(\hat{x}_0) + \mathbb{E} \left[\sum_{t=0}^{n-1} \beta^t r(\hat{x}_t, \hat{x}_{t+1}) + \beta^n R(\hat{x}_n) - R(\hat{x}_0) \right] \\
&= R(\hat{x}_0) + \mathbb{E} \left[\sum_{t=0}^{n-1} \beta^t d(\hat{x}_t, \hat{x}_{t+1}) \right]
\end{aligned}$$

In TD(0) we update by adding the first temporal difference after visiting x discounting each by a factor β . Suppose now we update by adding the next n temporal differences after visiting x .

Def 149 (n -Step TD). *The update*

$$R(x) \leftarrow \sum_{t=0}^{n-1} \beta^t d(\hat{x}_t, \hat{x}_{t+1}) = \sum_{t=0}^{n-1} \beta^t r(\hat{x}_t, \hat{x}_{t+1}) + \beta^n R(\hat{x}_n) - R(\hat{x}_0)$$

is called n -step TD.

Notice that ∞ -step TD is exactly Monte-carlo policy evaluation. And notice 1-step TD is TD(0).

A Bias-Variance Decomposition. We can look at the n -step TD update as moving a prediction $R(x)$ towards a target observation

$$\text{TD target} = \sum_{t=1}^n \beta^t r(\hat{x}_t) + \beta^{n+1} R(\hat{x}_{n+1})$$

Thus, if $R^\pi(x)$ is the true reward of the policy being evaluated then the temporal difference error is

$$TD_n := \sum_{t=1}^n \beta^t r(\hat{x}_t) + \beta^{n+1} R(\hat{x}_{n+1}) - R^\pi(\hat{x}_0)$$

Like with regression, we can analyze the bias and variance of these predictions.

Lemma 22 (Bias-Variance Decomposition for n -step TD).

$$\mathbb{E}[TD_n^2] = \underbrace{\beta^{2n} \mathbb{E}_x [R^\pi(\hat{x}_n) - R(\hat{x}_n)]^2}_{\text{Bias}} + \underbrace{\beta^{2n} \mathbb{V}(R(\hat{x}_n)) + \mathbb{V}\left(\sum_{t=0}^{n-1} \beta^t r(\hat{x}_t)\right)}_{\text{Variance}}$$

Notice this splits the n -step TD error into two terms one bias term and two variance terms: one for the time n predicted future reward, $R(\hat{x}_n)$; and one for the cumulated reward over n steps $\sum_{t=0}^{n-1} \beta^t r(\hat{x}_t)$.

Proof.

$$\begin{aligned}
\mathbb{E}_x[TD_n^2] &= \mathbb{E}_x \left[\left(R^\pi(\hat{x}_0) - \beta^n R(\hat{x}_n) - \sum_{t=0}^{n-1} \beta^t r(\hat{x}_t) \right)^2 \right] \\
&= \mathbb{E}_x \left[\left(R^\pi(\hat{x}_0) - \beta^n R(\hat{x}_n) - \sum_{t=0}^{n-1} \beta^t r(\hat{x}_t) \right)^2 \right] \\
&\quad + \mathbb{E}_x \left[\left(\beta^n R(\hat{x}_n) + \sum_{t=0}^{n-1} \beta^t r(\hat{x}_t) - \mathbb{E}_x \left[\beta^n R(\hat{x}_n) - \sum_{t=0}^{n-1} \beta^t r(\hat{x}_t) \right] \right)^2 \right] \\
&= \beta^{2n} \mathbb{E}_x [R^\pi(\hat{x}_0) - R(\hat{x}_n)]^2 \\
&\quad + \mathbb{E} [(\beta^n R(\hat{x}_n) - \mathbb{E}[\beta^n R(\hat{x}_n)])^2] \\
&\quad + \mathbb{E} \left[\left(\sum_{t=0}^{n-1} \beta^t r(\hat{x}_t) - \mathbb{E} \left[\sum_{t=0}^{n-1} \beta^t r(\hat{x}_t) \right] \right)^2 \right] \\
&= \beta^{2n} \mathbb{E}_x [R^\pi(\hat{x}_n) - R(\hat{x}_n)]^2 + \beta^{2n} \mathbb{V}(R(\hat{x}_n)) + \mathbb{V} \left(\sum_{t=0}^{n-1} \beta^t r(\hat{x}_t) \right).
\end{aligned}$$

In the 2nd equality, we add and subtract $\mathbb{E}_x[\beta^n R(\hat{x}_n) - \sum_{t=0}^{n-1} \beta^t r(\hat{x}_t)]$ and then expand. In the 3rd equality, we note for the 1st expectation that $R^\pi(x) - \mathbb{E}_x[\sum_{t=0}^{n-1} \beta^t r(x_t)] = \beta^n \mathbb{E}[R^\pi(\hat{x}_n)]$ and we expand the 2nd expectation into two terms. \square

Remark 150. Notice that if rewards are roughly IID with variance σ^2 then $\mathbb{V} \left(\sum_{t=0}^{n-1} \beta^t r(\hat{x}_t) \right) = \sigma^2 (1 - \beta^{2n}) / (1 - \beta)$. Thus the decomposition takes the form:

$$\mathbb{E}[TD_n^2] = \beta^{2n} \left[\mathbb{E}_x [R^\pi(\hat{x}_n) - R(\hat{x}_n)]^2 + \mathbb{V}(R(\hat{x}_n)) - \frac{\sigma^2}{1 - \beta} \right] + \frac{\sigma^2}{1 - \beta}$$

This gives some intuition on the number of steps n . Basically, depending on whether the term in square brackets is positive or negative we should choose n large or small. Specifically if there is high error or high variance in the future expected reward then n should be increased in size. However, if there is small error and small variance in future expected reward relative to the variance in individual rewards, then n should be small. This could for instance imply that the number of steps in the TD algorithm should be reduced for later stages of training.

TD-Lambda.*

Similar to n -step TD, we want to consider continuous update where we can trade off bias and variance. Again like n -step TD, TD-Lambda will represent a set of methods with Monte-Carlo policy evaluation on the one extreme and TD(0) on the other. TD-Lambda continuously parameterizes this range (rather than the discrete way that n -step TD).

We break this into two pieces first describing an off-line TD(λ) update and then adapt it to give a more practical online update.

Remark 151. *At a practical level, n -step TD methods are much more simpler to understand and code up. They achieve much the same goal as TD-lambda methods. TD-Lambda is elegant in their use of the memoryless property of the geometric distributions. However, (in my opinion) TD-lambda methods are arguably a marginal improvement on n -step TD methods.*

TD(λ) – Offline

Under n -step TD we need to look forward through the next n -steps of the algorithm and then perform an update. TD(λ), which we describe shortly, makes use of the memoryless property of the geometric distribution to give a simplified update equation. In particular, suppose that we apply n -Step TD with weight $(1 - \lambda)\lambda^n$ then at the end of each episode we perform the update

$$R(x) \leftarrow \sum_{n=0}^{\infty} (1 - \lambda)\lambda^n \sum_{k=0}^n \beta^k d(\hat{x}_k, \hat{x}_{k+1}) = \sum_{k=0}^{\infty} \lambda^k \beta^k d(\hat{x}_k, \hat{x}_{k+1})$$

for each x . Here we let \hat{x}_0 be the first visit to x and we let $\hat{x}_1, \hat{x}_2, \dots$ be the subsequent states.

Def 152 (TD(λ) – Offline). *The above update equation above gives the Offline update for TD(λ) under a first visit update.*

If we perform the above update at the end of each episode for every visit to x , we have

$$R(x) \leftarrow \sum_{v=1}^{\infty} \sum_{k=0}^{\infty} \lambda^k \beta^k d(\hat{x}_k^{(v)}, \hat{x}_{k+1}^{(v)})$$

where $\hat{x}_0^{(v)}$ be the v th visit to x and we let $\hat{x}_1^{(v)}, \hat{x}_2^{(v)}, \dots$ be the subsequent states, then we gain the every visit update formulation.

Notice $\lambda = 0$ corresponds to $TD(0)$ and $TD(1)$ gives Monte-Carlo policy evaluation. Like n -step TD, trades bias and variance through its parameter λ .

TD(λ) – Online

The TD(λ) algorithm appears to only work offline, as we need to record the chain's transitions over the whole episode and then update each term. However we can see a much simpler view exists by extracting the contribution of each term to the update. We now construct an update at every time step rather than at every visit.

Suppose we visited x just once at time τ . The contribution to the update at the end of the episode is

$$R(x) \leftarrow R(x) + \alpha \sum_{t=\tau}^{\infty} (\lambda\beta)^{t-\tau} d(x_t, x_{t+1})$$

We could view this single update as a sequence of updates occurring at each time. So if we update at every time t , the contribution from this visit to x at time τ would be

$$R(x) \leftarrow R(x) + \underbrace{\alpha(\lambda\beta)^{t-\tau}}_{=:E(t)} d(x_t, x_{t+1})$$

We could express the recursion that $E(x)$ satisfies more compactly as follows:

$$E(x) \leftarrow (\lambda\beta)E(x) + \alpha\mathbb{I}[x_t = x] \quad (4.3)$$

Notice, if we wanted to implement the every-visit update, the above recursion would stay the same. If we wanted to implement the first-visit update the indicator function above would only be applied at the first visit to x (and be zero there-after).

Def 153 (Eligibility Trace). $E(x)$ as described above, (4.3), is called the eligibility trace of the episode

The eligibility trace records a weighted count of how many times x has been visited so far. Notice, because λ is applied geometrically in $TD(\lambda)$, we do not need to record how long since x was last visited.

We can now perform an online version of $TD(\lambda)$

Def 154 (TD(λ) – Online). *At each time step with current state x and next state \hat{x} , perform the following update to every state x'*

$$\begin{aligned} E(x') &\leftarrow (\lambda\beta)E(x') + \alpha\mathbb{I}[x' = x] \\ R(x') &\leftarrow R(x') + E(x')d(x, \hat{x}). \end{aligned}$$

Here $d(x, \hat{x})$ is the current temporal difference, and we start initially with $E(x) = 0$.

Further TD methods.

We briefly mention a few quick generalization of the TD methods.

Importance Sampling. Suppose that we want to evaluate $R(x) = \mathbb{E}_x^P \left[\sum_t \beta^t r(\hat{x}_t) \right]$ where P_{xy} gives the probability of transitions. However, the simulator used does transitions with probability Q_{xy} . Then

$$\mathbb{E}_x^P[f(x, \hat{x})] = \sum_y P_{xy} f(x, y) = \sum_y Q_{xy} \frac{P_{xy}}{Q_{xy}} f(x, y) = \mathbb{E}_x^Q \left[\frac{P_{x\hat{x}}}{Q_{x\hat{x}}} f(x, \hat{x}) \right]$$

for any function $f : \mathcal{X}^2 \rightarrow \mathbb{R}$. For instance, TD(0) is searching for the fixed point

$$0 = \mathbb{E}_x^P [r(x) + \beta R(\hat{x}) - R(x)] = \mathbb{E}_x^Q \left[r(x) + \beta R(\hat{x}) \frac{P_{x\hat{x}}}{Q_{x\hat{x}}} - R(x) \right].$$

Thus, given the simulator generates transitions under Q , an importance sampled TD(0) w.r.t. P would be

$$R(x) \leftarrow r(x) + \beta R(\hat{x}) \frac{P_{x\hat{x}}}{Q_{x\hat{x}}} - R(x).$$

Notice importance sampling for more general functions would be,

$$\mathbb{E}_x^P[f(\hat{x}_0, \dots, \hat{x}_t)] = \mathbb{E}_x^Q \left[f(\hat{x}_0, \dots, \hat{x}_t) \prod_{s=0}^{t-1} \frac{P_{\hat{x}_s, \hat{x}_{s+1}}}{Q_{\hat{x}_s, \hat{x}_{s+1}}} \right].$$

We leave it to the reader to use this to figure more general importance sampling update rules e.g. for Monte-carlo, n -step, TD(λ).

Stopping Times. If σ is a stopping time then we can update at a stopping time and the appropriate TD update is

$$R(\hat{x}_0) \stackrel{\alpha}{\leftarrow} r(\hat{x}_0) + \dots + \beta^{\sigma-1} r(\hat{x}_{\sigma-1}) + \beta^\sigma R(\hat{x}_\sigma) - R(\hat{x}_0)$$

Notice TD(λ) is essential this update with the stopping time being a geometrically distributed with parameter λ . We include the possibility that $\sigma = \infty$ in which case Monte-carlo methods are included. Further n -step and TD(0) are instances where the stopping time is constant.

Resampling and Branching. Suppose τ is a stopping time. For example, τ could be the time the total reward so far goes above some predetermined level. We can resample the trajectory from the point τ and perform two or more update. Thus we can search around that point for good trajectories.

Insert Picture here

For two updates, suppose $\hat{x}_{\tau+1}^{(1)}, \hat{x}_{\tau+2}^{(1)}, \dots, \hat{x}_{\sigma^{(1)}}^{(1)}$ and $\hat{x}_{\tau+1}^{(2)}, \hat{x}_{\tau+2}^{(2)}, \dots, \hat{x}_{\sigma^{(2)}}^{(2)}$ are two trajectories for stopping times $\sigma^{(1)}$ and $\sigma^{(2)}$ after time τ . We can then perform two TD updates

$$R(\hat{x}_0) \stackrel{\alpha}{\leftarrow} \sum_{t=0}^{\tau} \beta^t r(\hat{x}_t) + \sum_{t=\tau+1}^{\sigma^{(1)}-1} \beta^{\tau+1} r(\hat{x}_{\tau+1}^{(1)}) + \beta^{\sigma^{(1)}} R(\hat{x}_{\sigma^{(1)}}^{(1)}) - R(\hat{x}_0)$$

$$R(\hat{x}_0) \stackrel{\alpha}{\leftarrow} \sum_{t=0}^{\tau} \beta^t r(\hat{x}_t) + \sum_{t=\tau+1}^{\sigma^{(2)}-1} \beta^{\tau+1} r(\hat{x}_{\tau+1}^{(2)}) + \beta^{\sigma^{(2)}} R(\hat{x}_{\sigma^{(2)}}^{(2)}) - R(\hat{x}_0).$$

Even if the event τ does not occur, (and thus $\sigma^{(1)}, \sigma^{(2)}$ and τ are all infinite) then you still need to update the objective twice, otherwise you introduce bias into the system.

TD Trees. The above argument gives a simple method for resampling and branching. With this branching argument, we can in principle branch an arbitrary predetermined number of times. Here we can create a tree and we must perform an update for each leaf in that tree (even if the associated stopping time is infinite). This somewhat related to the idea of Monte-Carlo Tree search which we will discuss later.

References.

Temporal difference methods were introduced by Sutton [47]. So the text of Sutton and Barto is the best place to go to to read more

on this [\[48\]](#).

4.3 Q-learning

- Q-learning and a proof of convergence.

Q-learning is an algorithm, that contains many of the basic structures required for reinforcement learning and acts as the basis for many more sophisticated algorithms. The Q-learning algorithm can be seen as an (asynchronous) implementation of the Robbins-Munro procedure for finding fixed points. For this reason we will require results from Section 2.2 when proving convergence.

A key ingredient is the notion of a Q-factor as described in Section 0.4. Recall that optimal Q-factor, $Q(x, a)$, is the value of starting in state x taking action a and thereafter following the optimal policy. In Prop 34 we showed that this solved the recursion:

$$Q(x, a) = \mathbb{E}_{x,a}[r(x, a) + \beta \max_{\hat{a}} Q(\hat{X}, \hat{a})]. \quad (4.4)$$

Def 155 (Q-learning). *Given a state x , an action a , its reward $r(x, a)$ and the next state \hat{x} , Q-learning performs the update*

$$Q(x, a) \stackrel{\alpha}{\leftarrow} r(x, a) + \beta \max_{a' \in \mathcal{A}} Q(\hat{x}, a') - Q(x, a)$$

where α is a positive (learning rate) parameter. Recall $x \stackrel{\alpha}{\leftarrow} dx$ means reset x with x' such that $x' = x + \alpha dx$.

To implement this as an algorithm, we assume that we have a sequence of state-action-reward-next_state quadruplets $\{(x_t, a_t, r_t, \hat{x}_t)\}_{t=0}^{\infty}$ and we apply the above update to each of the terms in this sequence.

Thm 156. *For a sequence of state-action-reward triples $\{(x_t, a_t, r_t, \hat{x}_t)\}_{t=0}^{\infty}$ Consider the Q-learning update for $(x, a, r, \hat{x}) = (x_t, a_t, r_t, \hat{x}_t)$*

$$Q_{t+1}(x, a) = Q_t(x, a) + \alpha_t(x, a) \left(r + \max_{a'} Q_t(x', a') - Q_t(x, a) \right)$$

if the sequence of state-action-reward triples visits each state and action infinitely often, and if the learning rate $\alpha_t(x, a)$ is an adapted sequence satisfying the Robbins-Munro condition

$$\sum_{t=1}^{\infty} \alpha_t(x, a) = \infty, \quad \sum_{t=1}^{\infty} \alpha_t^2(x, a) < \infty$$

then, with probability 1,

$$Q_t(x, a) \rightarrow Q^*(x, a)$$

where $Q^*(x, a)$ is the optimal value function.

Proof. We essentially show that the result is a consequence of Theorem 16 in Section 2.2. We note that the optimal Q -function, $\mathbf{Q} = (Q(x, a) : x \in \mathcal{X}, a \in \mathcal{A})$ satisfies a fixed point equation

$$\mathbf{Q} = \mathbf{F}(\mathbf{Q}),$$

with

$$F_{x,a}(\mathbf{Q}) = \mathbb{E}_{x,a}[r(x, a) + \beta \max_{\hat{a}} Q(\hat{X}, \hat{a})],$$

for each $x \in \mathcal{X}$ and $a \in \mathcal{A}$. We know from Prop 34 that for discounted programming $\mathbf{F}(\cdot)$ is a contraction. I.e.

$$\|\mathbf{F}(\mathbf{Q}_1) - \mathbf{F}(\mathbf{Q}_2)\|_\infty \leq \beta \|\mathbf{Q}_1 - \mathbf{Q}_2\|_\infty.$$

Now notice that the Q -learning algorithm performs the update

$$Q_{t+1}(x, a) = Q_t(x, a) + \alpha_t(x, a)(F(\mathbf{Q})(x, a) - Q_t(x, a) + \epsilon(x, a)),$$

where

$$\epsilon(x, a) = r + \beta \max_{\hat{a}} Q(\hat{X}, \hat{a}) - \mathbb{E}_{x,a}[r(x, a) + \beta \max_{\hat{a}} Q(\hat{X}, \hat{a})]$$

for $(x_t, a_t, r_t, \hat{x}_t) = (x, a, r, \hat{x})$. The update above is a Robbin's Munro update. FurtherbNotice $Q(x', a')$ remains the same for all other values of x, a , the update is asynchronous. It is not hard to see that when we condition on \mathcal{F}_t the set of previous actions and states that

$$\mathbb{E}[\epsilon_t(x_t, a_t) | \mathcal{F}_t] = 0$$

and, a quick calculation shows,² that

$$\mathbb{E}[\epsilon_t(x_t, a_t)^2 | \mathcal{F}_t] \leq 2r_{\max}^2 + 2\beta^2 \max_{x,a} Q_t(x, a)^2.$$

From this we see that we are working in the setting of Theorem 16 and that the condtions of that theorem are satisfied. Thus it must be that

$$Q_t(x, a) \xrightarrow[t \rightarrow \infty]{} Q^*(x, a)$$

where $Q^*(x, a)$ satisfies $\mathbf{Q}^* = \mathbf{F}(\mathbf{Q}^*)$. In otherwords, as required, it satisfies the Bellman equation for the optimal Q -function and thus is optimal. \square

²Note $(x + y)^2 \leq 2x^2 + 2y^2$

Q-learning code with learning step and ϵ -greedy exploration.³

```
class Q_learning(Q_function):
    def __init__(self, lr=0.1):
        self.lr = lr

    def learn(self, state, action, reward, next_state,
              done=False, discount=1.):

        self.add(state, action)
        self.add(next_state)

        dQ = reward \
            + discount * self.max(next_state) \
            - self[state][action]

        if done :
            dQ = reward - self[state][action]

        self[state][action] += self.lr * ( dQ )

    def action(self, state, explore_prob=0.):

        if random.random() > explore_prob :
            return self.argmax(state)
        else :
            Actions = list(self[state].keys())
            random_action = random.choice(Actions)
            return random_action
```

Other variants.

We will discuss some variants in separate sections, but we discuss a few simple variants of Q-learning.

Double Q-learning. Double Q-learning, as the name suggests, is a variation of Q-learning where you maintain two Q-functions $Q^{(A)}$ and $Q^{(B)}$ and you update one in terms of the other:

$$Q^{(A)}(x, a) \stackrel{\alpha}{\leftarrow} r(x, a) + \beta Q^{(A)}(\hat{x}, \hat{b}) - Q^{(A)}(x, a)$$

$$Q^{(B)}(x, a) \stackrel{\alpha}{\leftarrow} r(x, a) + \beta Q^{(B)}(\hat{x}, \hat{a}) - Q^{(B)}(x, a)$$

³The code is subclass of a dictionary object with .max, argmax, .add methods. Here .add adds new states or actions to the Q-function.

where

$$\hat{a} = \operatorname{argmax}_{a \in \mathcal{A}} Q^{(A)}(x, a) \quad \text{and} \quad \hat{b} = \operatorname{argmax}_{b \in \mathcal{A}} Q^{(B)}(x, b).$$

It's not really clear at first why this should help. The problem it tries to resolve is this: in traditional Q-learning there is a maximization over actions and the Q-factor is a noisy estimate of the optimal Q-factor. In general, it is true that

$$\mathbb{E}[\max_{\hat{a}} Q(\hat{x}, \hat{a})] \geq \max_{\hat{a}} \mathbb{E}[Q(\hat{x}, \hat{a})]$$

with the inequality being increasingly strict the more random the Q-function estimate $Q(\hat{x}, \cdot)$ is.

The reason it helps is the following: suppose A_a and B_a are independent identical random variables with mean \bar{A} for each $a \in \mathcal{A}$. Suppose $\hat{a} = \operatorname{argmax}_a A_a$ and $\hat{b} = \operatorname{argmax}_a B_a$. Notice

$$\mathbb{E}[A_{\hat{a}}] > \max_a \bar{A}_a,$$

but $\mathbb{E}[A_{\hat{b}}] = \bar{A}_{\hat{b}}$ (here we take the expectation over A given B) so

$$\mathbb{E}[A_{\hat{b}}] = \mathbb{E}[\bar{A}_{\hat{b}}] < \max_b \bar{A}_b$$

In summary, we go from over-estimating the maximum A to under-estimating. In general, it depends if over-estimating is worse than under-estimating. However, over estimating tends to occur due to outliers that can mess up training (particularly if function approximation is being used). So to achieve training with more modest updates double Q-learning is generally a good idea. Further convergence is guaranteed by much the same analysis as for Q-learning. (We leave this proof as an exercise for the reader).

Advantage Updating / Duelling. For Q-learning the value function V which is the largest Q-factor determines the optimal policy. However, the magnitude of the value function V can differ from the relative sizes of the Q-factors. Specifically, during training the Q-factors of several action can get be close to V but V can be really big. The value of the Q-factor update is dominated by the size of V and so this can mean sub-optimal actions can often fluctuate above their true values and thus inhibit convergence.

The idea of advantage updating is to separate our the task of finding V from the task of finding how much less Q is relative to V . The difference between V and Q is called the advantage and it is defined as follows:

Def 157 (Advantage). For a value function $V(x)$ and Q-factors $Q(x, a)$, the advantage function is defined by

$$A(x, a) := Q(x, a) - V(x), \quad x \in \mathcal{X}, a \in \mathcal{A}.$$

Notice for optimal Q-factors and values, the advantage function is negative (when maximizing) and equal to zero for optimal actions. Under a general policy, if the advantage function is positive for some a then this suggests that improvement to the current policy can be made by increasing the probability of playing action a .

An advantage updating algorithm does the following steps when each state action-pair is visited:

$$A(x, a) \leftarrow A(x, a) - A_{\max}(x) \tag{4.5a}$$

$$A(x, a) \stackrel{\alpha}{\leftarrow} r + \beta V(\hat{x}) - V(x) + A_{\max}(x) - A(x, a) \tag{4.5b}$$

$$V(x) \stackrel{\gamma}{\leftarrow} \Delta A_{\max}(x) / \alpha \tag{4.5c}$$

t where $A_{\max}(x) := \max_a A(x, a)$ and $\Delta A_{\max}(x)$ is size of the last change in the value value of $A_{\max}(x)$ (from step (4.5b)).

If we take $\gamma = \alpha$, the following algorithm is really just the Q-learning algorithm.

Proposition 6. If $\gamma = \alpha$ then, when (4.5) is applied,

$$Q(x, a) = V(x) + A(x, a) - A_{\max}(x), \tag{4.6}$$

obeys the Q-learning update. Consequently, the policy implied by the advantage function $A(x, a)$ converges to the optimal policy.

Proof. Clearly,

$$V(x) = \max_a Q(x, a). \tag{4.7}$$

Notice that step (4.5a) of the algorithm sets $A_{\max}(x) = 0$. Also notice that any shift in $A(x, a)$ applied equally to all a will not effect the change applied in the update (4.5b). The update (4.5c) ensures that

$$V(x) - A_{\max}(x) = \tilde{V}(x) - \tilde{A}_{\max}(x) \tag{4.8}$$

where \tilde{V} and \tilde{A} denotes the values before the update are made. So here $V(x) - A_{\max}(x)$ stays constant. If we add this to both sides of (4.8) to the final update (4.5c) and then apply (4.6) and (4.7) we recover:

$$\begin{aligned} Q(x) &= V(x) + A(x, a) - A_{\max}(x) \\ &\leftarrow V(x) + A(x, a) - A_{\max}(x) \\ &\quad + \alpha (r + \beta V(x) - V(x) - A(x, a) + A_{\max}(x)) \\ &= Q(x, a) + \alpha \left(r + \beta \max_a Q(x, a) - Q(x, a) \right). \end{aligned}$$

Q-learning converges under the appropriate step size choice given in Theorem 156. So since $Q(x, a)$ convergence and $V(x) = \max_a Q(x, a)$, $V(x)$ converges. Thus $A(x, a) - A_{\max}$ from (4.6) convergence and since $A_{\max} = 0$ we see that the advantage functions converge, and these (along with $Q(x, a)$) imply the policy given by $A(x, a)$ in the limit is optimal. \square

Although advantage updating above is more-or-less identical to Q-learning, the changes are more pronounced when we later apply function approximation with a Neural Network. In this situation, a Q-learning step only applies an update to each state action pair, but advantage updating applies an update to both $V(x)$ and one value of $A(x, a)$. Thus each update has an impact on every Q-value through $V(x)$. The combination of advantage updating and function approximation have come to be called these are called Duelling architectures.

References.

This section is based on reading Tsitsiklis [49]. An alternative proof is given by Jaakkola et al. [25] which applies a slightly esoteric fixed point method of Dvoretzky.

Double Q-learning is first proposed by Hasselt [23]. The Advantage updating algorithm [with some minor modification to (4.5a)] is first given by Baird [3]. Both ideas have been successfully applied with neural network function approximation [52, 54].

4.4 SARSA

We consider a simple variant of Q -learning called Sarsa. Here SARSA stands for State, Action, Reward, (next) State, (next) Action. This is defined as follows.

Def 158 (Sarsa). *Under Sarsa, starting in state x take action a (as say ϵ -greedy from Q), then observe \hat{x} and reward $r(x, \hat{x})$. Next, take the action for that \hat{a} (under the same rule derived from Q . then update*

$$Q(x, a) \leftarrow r(x, \hat{x}) + \beta Q(\hat{x}, \hat{a}) - Q(x, a)$$

and continue from state and action \hat{x}, \hat{a} .

Notice, unlike with Q -learning, the value of the update depends on the policy generating states and actions. For instance, if action are chosen uniformly at random then the Sarsa update will converge to the reward function of the randomized policy, while Q -learning will still converge to the optimal value function.

To account for this we have to let our choice of actions converge to the optimal action for each state. Such policies are called GLIE where GLIE stands for Greedy in the Limit with Infinite Exploration.

Def 159 (GLIE). *A policy is GLIE if*

- *each action is chosen infinitely often for every state,*
- *the greedy action with respect to the Q -function is chosen with probability 1 in the limit.*

The most straight forward choice of GLIE policy is ϵ -greedy (recall the section on Bandits) where $\epsilon = 1/N_t$ where N_t is the number of episodes simulated by time t .

Given the policy is GLIE and each state is visited infinitely often then Sarsa will converge to the optimal Q -function.

Theorem 24. *For a discounted program, if each state is visited infinitely often and actions are chosen by a GLIE policy at each state then Sarsa converges to the optimal value function with probability 1.*

Proof. The proof is much the same as for Q -learning. Note that Sarsa makes the update

$$Q_{t+1}(x, a) = Q_t(x, a) + \alpha_t(x, a) \{r(x_t, a_t) + \beta Q_t(x_{t+1}, a_{t+1}) - Q_t(x_t, a_t)\}$$

where here $\alpha_t(x, a) = 0$ for $(x, a) \neq (x_t, a_t)$. Now focusing on $x = x_t$ and $a = a_t$, we can rearrange the above expression as follows

$$\begin{aligned} & Q_{t+1}(x_t, a_t) \\ &= Q_t(x_t, a_t) + \alpha_t(x_t, a_t) \left\{ r(x_t, a_t) - \beta \max_{\hat{a}} Q_t(x_{t+1}, \hat{x}) - Q_t(x_t, a_t) \right\} \\ &\quad + \alpha_t(x_t, a_t) \beta \left[\max_{\hat{a}} Q_t(x_{t+1}, \hat{x}) - Q_t(x_{t+1}, a_{t+1}) \right] \\ &= Q_t(x_t, a_t) + \alpha_t(x_t, a_t) [F_{x_t, a_t}(\mathbf{Q}_t) - Q_t(x_t, a_t)] + \alpha_t(x_t, a_t) \delta_t + \alpha_t(x_t, a_t) \beta e_t \end{aligned}$$

where here

$$F_{x,a}(Q) := \mathbb{E}_{x,a}[r(x, a) + \beta \max_{\hat{a}} Q(\hat{x}, \hat{a})]$$

which is the same β -contraction used in the proof for Q-learning; and

$$\begin{aligned} \delta_t &:= r(x_t, a_t) + \beta \max_{\hat{a}} Q_t(x_{t+1}, \hat{a}) - Q(x_t, a_t) - \mathbb{E}[r(x_t, a_t) + \beta \max_{\hat{a}} Q(x_t, \hat{a}) - Q(x_t, a_t) | \mathbf{F}_t] \\ &\quad + \max_{\hat{a}} Q_t(x_{t+1}, \hat{a}) - Q_t(x_{t+1}, a_{t+1}) - \mathbb{E}[\max_{\hat{a}} Q_t(x_{t+1}, \hat{a}) - Q_t(x_{t+1}, a_{t+1}) | \mathbf{F}_t] \end{aligned}$$

where here $\mathbf{F}_t = (x_s, a_s : s \leq t)$ and we note that δ_t is a bounded Martingale difference sequence; and

$$e_{t+1} = \mathbb{E} \left[\max_{\hat{a}} Q_t(x_{t+1}, \hat{a}) - Q_t(x_{t+1}, a_{t+1}) \middle| \mathbf{F}_t \right]$$

which satisfies

$$e_t \rightarrow 0$$

as $t \rightarrow \infty$, since $Q_t(x, a)$ is bounded (note each update is bounded for a discounted program) and our policy is GLIE (so the probability of choosing the maximizing action goes to one).

Thus if we define $\epsilon_t = \delta_t + \beta e_t$, then we satisfy exactly the conditions of the Asynchronous Robbins-Munro update 2.4 in Theorem 16, that is

$$Q_{t+1}(x, a) = Q_t(x, a) + \alpha_t(x, a) [F_{x,a}(\mathbf{Q}_t) - Q_t(x, a) + \epsilon_t]$$

and by that result, for all x, a ,

$$Q_t(x, a) \xrightarrow[t \rightarrow \infty]{} Q^*(x, a)$$

where $Q^* = F(Q^*)$ or, in other words,

$$Q^*(x, a) = \mathbb{E}_{x,a}[r(x, a) + \beta \max_{\hat{a}} Q(\hat{x}, \hat{a})].$$

So the Bellman equation is satisfied and so the limit $Q^*(x, a)$ is the optimal value function. \square

So it is good that we have a convergence proof. However, it is easy to construct simple examples of GLIE policies where SARSA does not converge (as states are not automatically visited infinitely often for GLIE policies)

Remark 160. Consider the following example. *Insert Picture*
Here there are two actions, and two states with rewards where the process terminates. Suppose we apply an ϵ -greedy policy with $\epsilon = 1/N$ where N is the number of episodes so far. If we assume that initially $Q(x,0) > Q(x,1)$ for all x , the probability of visiting state $x = 2$ is $1/N^2$ and the sum of these probabilities $\sum_N 1/N^2$ is finite. Therefore by the Borel-Cantelli Lemma there can only be finitely many visits to $x = 2$ while $Q(x,0) > Q(x,1)$. Thus there is a positive probability that SARSA will stop visiting $x = 2$ and thus the Q function will not converge to the optimal Q -function.

Chapter 5

Reinforcement Learning with Function Approximation

5.1 Temporal Differences: Linear Approximation

Temporal difference methods evaluate the value of states (and state action pairs) of a given policy. Previously, when we introduced temporal difference learning, we needed to record a value function estimate for every state visited. However, when the set of states is large (or infinite) this is not feasible. Instead, we can interpolate between the states visited in much the same way that we do in linear regression. Thus, we approximate the value of each state. Consequently, there is some degradation in the quality of the solution attained, but we can achieve a practical method for policy evaluation.

First we give the high level idea behind linear function approximation. Then we give an analysis of $TD(0)$.

For a Markov chain $\hat{x} = (\hat{x}_t : t \in \mathbb{Z}_+)$, our aim is to estimate the reward function

$$R(x) := \mathbb{E}_x \left[\sum_{t=0}^{\infty} \beta^t r(\hat{x}_t) \right] \quad (5.1)$$

here rewards given by $(r(x) : x \in \mathcal{X})$. We use the states x_t and their rewards $r(x_t)$ as data and with this we approximate the reward function $R(x)$ with a linear approximation,

$$R(x; \mathbf{w}) = \mathbf{w}^\top \phi(x) = \sum_{j \in \mathcal{J}} w_j \phi_j(x).$$

Here we have taken our state x and extracted features, $\phi_j(x)$ for j in finite set \mathcal{J} . Interpreting each $\phi_j = (\phi_j(x) : x \in \mathcal{X})$ as a (very long) vector, we assume $\{\phi_j : j \in \mathcal{J}\}$ are linearly independent. We think of the set of features $(\phi_j(x) : j \in \mathcal{J})$ as giving a low dimensional representation of the state x . We then apply a vector of weights $\mathbf{w} = (w_j : j \in \mathcal{J})$ to each of these features.

Our job is to find weights that give a good approximation to $R(x)$. We know, for instance, that $R(x)$ is a solution to the fixed point equation

$$R(x) = \mathbb{E}_x \left[\underbrace{r(x) + \beta R(\hat{x})}_{=: \text{Target}(x)} \right], \quad x \in \mathcal{X}. \quad (5.2)$$

The target, $\text{Target}(x)$, is an estimate of the true value of $R(x; \mathbf{w})$. Other targets can be used, e.g. the Monte-carlo target, n -step TD or $TD(\lambda)$.

In function approximation, we cannot get the expected reward to equal its target. So we attempt to minimize the difference between our reward function estimate and its target. For example

$$\underset{w}{\text{minimize}} \quad \mathbb{E}_\mu \left[(\text{Tg}(x) - R(x; w))^2 \right].$$

Here the expectation is over $(\mu(x) : x \in \mathcal{X})$, the stationary distribution of our Markov process. We do not know the stationary distribution μ , but, asymptotically, we sample from it with the Markov chain $(x_t : t \in \mathbb{Z}_+)$. Thinking of this as a linear regression problem, we receive input data $\phi(x)$ and we need to apply weights to these, $w^\top \phi(x)$ to predict the output $\text{Tg}(x)$. We can apply a Stochastic Gradient Descent update to w

$$w \leftarrow w + \alpha (\text{Target}(x) - R(x; w)) \nabla_w R(x; w).$$

Just like with tabular methods these updates can be applied online, offline, first-visit, every-visit. Note we've conveniently ignored the dependence on w in our targets. Both empirically and theoretically this works out surprisingly well.

Analysis of TD(0).

We analyze linear function approximation for the $TD(0)$ target.

Linear TD(0) Algorithm. The $TD(0)$ target for a reward function R is $r(x) + \beta R(\hat{x}; w)$, where \hat{x} is the next state after x . We define the operation T by

$$TR(x) = \mathbb{E}_x[r(x) + \beta R(\hat{x}; w)].$$

Under a linear function approximation this gives an update

$$\begin{aligned} w &\stackrel{\alpha}{\leftarrow} (r(x) + \beta R(\hat{x}; w) - R(x; w)) \nabla_w R(x; w) \\ &= (r(x) + \beta w^\top \phi(\hat{x}) - w^\top \phi(x)) \phi(x). \end{aligned}$$

O.D.E. and Convergence. The stochastic approximation scheme above is an approximation to the o.d.e.

$$\frac{dw}{dt} = \mathbb{E}_\mu [\{r(x) + \beta \phi(x)^\top w - \phi(x)w\} \phi(x)]$$

where μ is the stationary distribution of our Markov chain. We also define the norm

$$\|R\|_\mu := \left(\sum_{x \in \mathcal{X}} \mu(x) R(x)^2 \right)^{1/2}.$$

We let Π be the projection with respect to $\|\cdot\|_\mu$ on to the space spanned by $\{\phi_j : j \in \mathcal{J}\}$. Specifically,

$$\Pi R = \operatorname{argmin}_{\phi^\top \mathbf{w} : \mathbf{w} \in \mathbb{R}^{\mathcal{J}}} \mathbb{E}_\mu \left[(R(x) - \phi(x)^\top \mathbf{w})^2 \right].$$

We can prove the following about the convergence and limit of the above o.d.e. .

Theorem 25. *The o.d.e. converges to aa limit \mathbf{w}^* satisfying*

$$R(\mathbf{w}^*) = \Pi TR(\mathbf{w}^*) \tag{5.3}$$

Moreover, when compared with the true reward $R^*(x)$, the approximation $R(x; \mathbf{w}^*)$ satisfies

$$\|R^* - R(\mathbf{w}^*)\|_\mu \leq \frac{1}{1 - \beta} \|R^* - \Pi R^*\|_\mu. \tag{5.4}$$

Before proceeding with the proof, we briefly discuss the result. Notice in (5.1), without the projection we have $R = TR$ which is the condition for the true reward function. Thus we see that $R(\mathbf{w}^*)$ is the fixed point reached when we apply the TD update and then project back into the space spanned by $\phi_j, j \in \mathcal{J}$.

In (5.4), we see that the error between the true reward and our approximation with within a factor of the smallest error we could have received from approximating. This is important, but the quality of the bound does degrade as we let $\beta \rightarrow 1$.

Proof. We first analyse the convergence of solutions of our o.d.e..

The fixed point of the o.d.e. is given by

$$0 = \mathbb{E}_\mu \left[\{r(x) + \beta \phi(\hat{x})^\top \mathbf{w} - \phi(x)^\top \mathbf{w}\} \phi(x) \right] \tag{5.5}$$

Substituting the above for the $\mathbb{E}_\mu[\phi(x)r(x)]$ term in our o.d.e. gives

$$\begin{aligned} \frac{d\mathbf{w}}{dt} &= \mathbb{E}_\mu \left[\{\beta \phi(\hat{x})^\top (\mathbf{w} - \mathbf{w}^*) - \phi(x)^\top (\mathbf{w} - \mathbf{w}^*)\} \phi(x) \right] \\ &= \mathbb{E}_\mu \left[\{\beta P \phi(x)^\top (\mathbf{w} - \mathbf{w}^*) - \phi(x)^\top (\mathbf{w} - \mathbf{w}^*)\} \phi(x) \right] \end{aligned} \tag{5.6}$$

5.1. TEMPORAL DIFFERENCES: LINEAR APPROXIMATION NSW

Above $P = (P_{xy} : x, y \in \mathcal{X})$ is the transition matrix of our Markov chain, and $Pf(x) = \mathbb{E}[f(\hat{x})|x] = \sum_y P_{xy}f(y)$, for any function $f : \mathcal{X} \rightarrow \mathbb{R}$. Shortly, we will require the following inequality:

$$\mathbb{E}_\mu [f(x)Pf(x)] \leq \mathbb{E}_\mu [f(x)^2]. \quad (5.7)$$

We prove this in Lemma 23 immediately after this proof. Now

$$\begin{aligned} \frac{d}{dt} \left\{ \frac{1}{2} \|\mathbf{w}(t) - \mathbf{w}^*\|^2 \right\} &= (\mathbf{w} - \mathbf{w}^*)^\top \frac{d}{dt} \mathbf{w}(t) \\ &\stackrel{(5.6)}{=} \mathbb{E}_\mu [(\mathbf{w} - \mathbf{w}^*)^\top \phi(x) \beta P \phi(x)^\top (\mathbf{w} - \mathbf{w}^*)] \\ &\quad - \mathbb{E}_\mu [(\mathbf{w} - \mathbf{w}^*)^\top \phi(x) \phi(x)^\top (\mathbf{w} - \mathbf{w}^*)] \\ &\stackrel{(5.7)}{\leq} \beta \mathbb{E}_\mu [(\mathbf{w} - \mathbf{w}^*)^\top \phi(x) \phi(x)^\top (\mathbf{w} - \mathbf{w}^*)] \\ &\quad - \mathbb{E}_\mu [(\mathbf{w} - \mathbf{w}^*)^\top \phi(x) \phi(x)^\top (\mathbf{w} - \mathbf{w}^*)] \\ &= -(1 - \beta) \mathbb{E}_\mu [(\phi(x)^\top (\mathbf{w} - \mathbf{w}^*))^2]. \end{aligned}$$

Consequently, if $\mathbf{w}(t) \neq \mathbf{w}^*$ then by linear independence $\phi(x)\mathbf{w} \neq \phi(x)\mathbf{w}^*$ for some x , so the final expression above is negative. Thus by the Lyapunov convergence result (8), $\frac{1}{2} \|\mathbf{w}(t) - \mathbf{w}^*\|^2 \rightarrow 0$ as $t \rightarrow \infty$.

We now have convergence of our o.d.e. to \mathbf{w}^* . We now should how \mathbf{w}^* can be interpreted as a projection. Lets find the project of a reward function $R_0(x)$:

$$\begin{aligned} &\min_{\mathbf{w}} \mathbb{E}_\mu \left[\left(R_0(x) - \phi(x)^\top \mathbf{w} \right)^2 \right] \\ &\implies 0 = \mathbb{E}_\mu \left[\phi(x) \left(R_0(x) - \phi(x)^\top \mathbf{w} \right) \right] \\ &\implies \phi(x')^\top \mathbf{w} = \phi(x')^\top \mathbb{E}_\mu [\phi(x)^\top \phi(x)]^{-1} \mathbb{E}_\mu [\phi(x) R_0(x)] \\ &\quad =: \Pi R_0(x'). \end{aligned}$$

Notice the steps above are the same as we apply when doing linear regression. Notice further that (5.5) is the condition for

$$R(x'; \mathbf{w}^*) = \Pi T R(x; \mathbf{w}^*)$$

as required.

Finally note that the optimal reward function is the fixed point $R^* = TR^*$ and T is a β -contraction (see Prop 34c) and that projections always move distances closer (see Lemma 25). Applying these observations gives

$$\begin{aligned} \|\mathbf{R}^* - \mathbf{R}(\mathbf{w}^*)\|_\mu &\leq \|\mathbf{R}^* - \Pi(\mathbf{R}^*)\|_\mu + \|\Pi(T(\mathbf{R})) - \Pi(T(\mathbf{R}(\mathbf{w}^*)))\|_\mu \\ &\leq \|\mathbf{R}^* - \Pi(\mathbf{R}^*)\|_\mu + \beta \|\mathbf{R}^* - \mathbf{R}(\mathbf{w}^*)\|_\mu. \end{aligned}$$

So, as required,

$$\|\mathbf{R}^* - \mathbf{R}(w^*)\|_\mu \leq \frac{1}{1-\beta} \|\mathbf{R}^* - \Pi(\mathbf{R}^*)\|_\mu.$$

□

Lemma 23.

$$\mathbb{E}_\mu [f(x)Pf(x)] \leq \mathbb{E}_\mu [f(x)^2]. \quad (5.8)$$

Proof. By Cauchy-Schwartz

$$\mathbb{E}_\mu [f(x)Pf(x)] \leq \mathbb{E}_\mu [f(x)^2]^{\frac{1}{2}} \mathbb{E}_\mu [(Pf(x))^2]^{\frac{1}{2}}. \quad (5.9)$$

Now

$$\begin{aligned} \mathbb{E}_\mu [(Pf(x))^2] &= \sum_x \mu(x) \sum_y (P_{xy}f(y))^2 \\ &\leq \sum_x \mu(x) \sum_y P_{xy}f(y)^2 \\ &= \sum_y \sum_x \mu(x)P_{xy}f(y)^2 = \sum_y \mu(y)f(y)^2 = \mathbb{E}_\mu [f(x)^2] \end{aligned}$$

where in the last inequality we use the fact that μ is the stationary distribution of our Markov chain. Substituting this into (5.9) gives the result. □

References

The section is based on reading Tsitsiklis and Van Roy [50], but also see Bertsekas and Tsitsiklis [7].

5.2 Policy Gradients

While almost all of the approaches considered so far in these notes focus on obtaining the value of each state and action in order to find an optimal policy. There is a much more direct approach that can be applied. Here we parameterize the set of policies and differentiate the objective. We can then simply apply gradient ascent to this to optimize.

The approach is appealing. However, from a theoretical perspective, it's a thorny issue: we cannot always exclude local minima or directly guarantee sufficient exploration. Only recently are there results that give conditions for convergence. Further it is not clear how data can be reused as is done in Q -learning. With that said, the results that do exist are elegant and many of the most successful methods in recent years make use of policy gradients.

Parameterized Policies. We let $\pi_\theta(a|x)$ be the probability that we choose action a in state x . Here θ parameterize the set of policies. For example, a popular choice is the soft-max function:

$$\pi_\theta(a|x) = \frac{e^{\theta^\top \phi(a|x)}}{\sum_{a'} e^{\theta^\top \phi(a'|x)}}$$

where like in our analysis of linear function approximation, $\theta(a|x)$ act as basis functions. However, many other choices exist.

Note that the probability of states and actions $\mathbf{x} = (x_0, \dots, x_T)$ and $\mathbf{a} = (a_0, \dots, a_{T-1})$ is

$$\pi_\theta(\mathbf{x}_T, \mathbf{a}_T) := \prod_{t=0}^{T-1} p(x_{t+1}|x_t, a_t) \pi_\theta(a_t|x_t)$$

The expected reward under policy π_θ is

$$V(\theta) := \mathbb{E}_{\pi_\theta} [Q(\hat{\mathbf{x}}, \hat{\mathbf{a}})] = \sum_{\mathbf{x}, \mathbf{a}} Q(\mathbf{x}, \mathbf{a}) \pi_\theta(\mathbf{x}, \mathbf{a})$$

where

$$Q(\mathbf{x}, \mathbf{a}) := \sum_{t=0}^{T-1} \beta^t r(x_t, a_t).$$

Differentiating the Reward Objective. We now cover a number of calculations where we differentiate the reward objective over θ . These are summarized by the following theorem.

Theorem 26 (The Policy Gradient Theorem).

$$\begin{aligned}\nabla_{\theta} V(\theta) &= \mathbb{E}_{\pi_{\theta}} \left[Q(\mathbf{x}, \mathbf{a}) \nabla_{\theta} \log \pi_{\theta}(\mathbf{x}, \mathbf{a}) \right] \\ &= \mathbb{E}_{\pi_{\theta}} \left[\sum_{t=0}^{T-1} \beta^t Q_t \nabla_{\theta} \log \pi_{\theta}(a_t | x_t) \right]\end{aligned}$$

where $Q_t = \sum_{s=t}^{T-1} \beta^{s-t} r(x_s, a_s)$.

Proof. We can differentiate the reward

$$\begin{aligned}\nabla_{\theta} V(\theta) &= \sum_{\mathbf{x}, \mathbf{a}} Q(\mathbf{x}, \mathbf{a}) \nabla_{\theta} \pi_{\theta}(\mathbf{x}, \mathbf{a}) \\ &= \sum_{\mathbf{x}, \mathbf{a}} \{Q(\mathbf{x}, \mathbf{a}) \nabla_{\theta} \log \pi_{\theta}(\mathbf{x}, \mathbf{a})\} \pi_{\theta}(\mathbf{x}, \mathbf{a}) \\ &= \mathbb{E}_{\pi_{\theta}} [Q(\hat{\mathbf{x}}, \hat{\mathbf{a}}) \nabla_{\theta} \log \pi_{\theta}(\hat{\mathbf{x}}, \hat{\mathbf{a}})].\end{aligned}\tag{5.10}$$

Thus we can apply stochastic gradient descent on the above objective by sampling

$$Q(\hat{\mathbf{x}}, \hat{\mathbf{a}}) \nabla_{\theta} \log \pi_{\theta}(\mathbf{x}, \mathbf{a}).$$

Further we note that we can simplify the derivative above,

$$\begin{aligned}\nabla_{\theta} \log \pi_{\theta}(\mathbf{x}, \mathbf{a}) &= \nabla_{\theta} \left\{ \log \prod_{t=0}^{T-1} p(x_{t+1} | x_t, a_t) + \log \prod_{t=0}^{T-1} \pi_{\theta}(a_t | x_t) \right\} \\ &= \sum_{t=0}^{T-1} \nabla_{\theta} \log \pi_{\theta}(a_t | x_t)\end{aligned}$$

This calculation is *important for reinforcement learning* applications, as we do not need to know $p(\hat{x}|x, a)$ to calculate the change in the above likelihood function for our policy.

We will shortly use the following

$$\mathbb{E}[\nabla_{\theta} \log \pi_{\theta}(\hat{a}|x)] = \sum_a \pi_{\theta}(a|x) \frac{\nabla_{\theta} \pi(a|x)}{\pi_{\theta}(a|x)} = \nabla_{\theta} \underbrace{\left\{ \sum_a \pi(a|x) \right\}}_{=1} = 0,\tag{5.11}$$

and, consequently for $s < t$,

$$\mathbb{E}_{\pi_{\theta}} [r(x_s, a_s) \nabla_{\theta} \log \pi_{\theta}(a_t | x_t)] = \mathbb{E}_{\pi_{\theta}} [r(x_s, a_s) \mathbb{E}_{\pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(a_t | x_t) | x_t]] = 0\tag{5.12}$$

Finally, we note that we can rearrange the objective (5.10) to account for the contribution from each individual state and action (x_t, a_t) :

$$\begin{aligned}
\mathbb{E}_{\pi_\theta} [Q(\mathbf{x}, \mathbf{a}) \nabla_\theta \log \pi_\theta(\mathbf{x}, \mathbf{a})] &= \mathbb{E}_{\pi_\theta} \left[\left(\sum_{s=0}^{T-1} \beta^s r(x_s, a_s) \right) \left(\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t | x_t) \right) \right] \\
&= \sum_{t=0}^{T-1} \sum_{s=0}^{T-1} \beta^s \mathbb{E}_{\pi_\theta} [r(x_s, a_s) \nabla_\theta \log \pi_\theta(a_t | x_t)] \\
&\stackrel{\text{by (5.12)}}{=} \sum_{t=0}^{T-1} \sum_{s=t}^{T-1} \beta^s \mathbb{E}_{\pi_\theta} [r(x_s, a_s) \nabla_\theta \log \pi_\theta(a_t | x_t)] \\
&= \mathbb{E}_{\pi_\theta} \left[\sum_{t=0}^{T-1} \beta_t Q_t \nabla_\theta \log \pi_\theta(a_t | x_t) \right]
\end{aligned}$$

as required. \square

Although the proof above suggests that T is finite. The proof will work for $T = \infty$ in a discounted program.

Further for later use it is worth noting that in the above proof we also proved the following Lemma.

Lemma 24.

$$\mathbb{E}[\nabla_\theta \log \pi_\theta(\hat{a}|x)] = 0.$$

Here the expectation is taken over \hat{a} not x .

A further form of the policy gradient theorem is the following.

Corollary 1.

$$\nabla V(\theta) = \mathbb{E}_{\pi_\theta} \left[\sum_{t=0}^{T-1} \beta^t \nabla_\phi Q_{\pi_\theta}(x_t, \pi_\phi) \Big|_{\phi=\theta} \right]$$

Proof.

$$\begin{aligned}
\mathbb{E}[\tilde{Q}_t(\hat{\mathbf{x}}_t, \hat{\mathbf{a}}_t) \nabla_\theta \log \pi_\theta(a_t | x_t) | x_t] &= \mathbb{E}[Q_{\pi_\theta}(x_t, a_t) \nabla_\theta \log \pi_\theta(a_t | x_t) | x_t] \\
&= \sum_a Q_{\pi_\theta}(x_t, a) \nabla_\theta \log \pi_\theta(a | x_t) \\
&= \nabla_\phi Q_{\pi_\theta}(x_t, \pi_\phi) \Big|_{\phi=\theta}
\end{aligned}$$

\square

The above corollary is not directly useful for designing algorithms (as we don't know the true Q -function in the expectation). However, it gives a useful interpretation of the policy gradient update. We are making a small one-step improvement to the Q -function, in much the same way that policy iteration operates.

Some Algorithms.

Now we can use our results to design algorithms.

REINFORCE. Given the Policy Gradient Theorem above the REINFORCE algorithm performs the following per episode update

$$\theta \leftarrow \theta + \gamma \tilde{Q} \nabla_{\theta} \log \pi_{\theta}(\mathbf{x}, \mathbf{a})$$

or as a per-visit update update does

$$\theta \leftarrow \theta + \gamma \tilde{Q}_t \nabla_{\theta} \log \pi_{\theta}(a_t | x_t)$$

where $\tilde{Q} = \sum_{s=0}^T \beta^s r(x_s, a_s)$ and $\tilde{Q}_t = \sum_{s=t}^T \beta^s r(x_s, a_s)$.

REINFORCE with a Baseline. By Lemma 24, any function of x can be added to the REINFORCE update and the mean of the update will not change. I.e.

$$\mathbb{E}[B(x) \nabla_{\theta} \log \pi_{\theta}(\hat{a}|x)] = 0.$$

So

$$\nabla_{\theta} R(\theta) = \mathbb{E}_{\pi_{\theta}} \left[\sum_{t=0}^{T-1} \beta^t (Q_t - B(x_t)) \log \pi_{\theta}(a_t | x_t) \right]$$

Although the mean stays the same the variance can be reduced if we have a decent estimate of the mean of \tilde{Q}_t . We can do this by using a temporal difference function approximation for the mean value of each state

$$w \leftarrow w + \alpha (r + \beta V_w(\hat{x}) - V_w(x)) \nabla_w V_w(x) \quad (5.13a)$$

and we update the policy weights

$$\theta \leftarrow \theta + \alpha (Q_t - V_w(x_t)) \nabla_{\theta} \log \pi_{\theta}(\hat{a}|x) \quad (5.13b)$$

We implement these updates per step.

Actor-Critic. Notice when we apply the updates above (5.13) the terms $r + \beta V_w(\hat{x})$ and Q_t serve the same purpose: they are both estimates of the Q-function for (x_t, a_t) . So we could reduce variance further by replacing Q_t with $r + \beta V_w(\hat{x})$. This gives the following algorithm

$$\delta \leftarrow (r + \beta V_w(\hat{x}) - V_w(x)) \quad (5.14a)$$

$$w \leftarrow w + \alpha \delta \nabla_w V_w(x) \quad (5.14b)$$

$$\theta \leftarrow \theta + \alpha \delta \nabla_\theta \log \pi_\theta(\hat{a}|x) \quad (5.14c)$$

This algorithm is called an Actor-Critic algorithm. Here we within of π_θ as the "actor" that makes the decisions in the simulation. And we think of V_w as the "critic" that evaluates the performance of the actor.

Notice δ is the TD(0) error. However, any TD update could be used. For example, in place of (5.14a), n -step TD would use:

$$\delta \leftarrow r_1 + \beta r_2 + \dots + \beta^n r_n + \beta^{n+1} V_w(\hat{x}_{n+1}) - V_w(x). \quad (5.14a')$$

Convergence Issues with Policy Gradients.

Although they are often very effective, there can be theoretical (and practical) issues with policy gradients. In order for a gradient descent method to converge to a global minimum, we generally require convexity (or some other monotonicity property). However when minimizing costs, the policy gradient algorithm objective might non-convex.¹

An Example. Consider the example in Figure 5.1.

Here there is a high cost for switching between the two local states A and B . If we suppose

$$\pi_\theta(R|A) = \pi_\theta(R|B) = \theta \in [0, 1]$$

Notice in this case the stationary distribution is given by $\mu(A) = 1 - \mu(B) = 1 - \theta$. Thus the average reward is

$$\begin{aligned} R(\theta) &= 1 \cdot \pi(L|A)\mu(A) + 2 \cdot \pi(R|A)\mu(A) + 2 \cdot \pi(L|B)\mu(A) + 0 \cdot \pi(L|A)\mu(A) \\ &= (1 - \theta)^2 + 2\theta(1 - \theta) + 2(1 - \theta)\theta = 1 + 2\theta - 3\theta^2. \end{aligned}$$

¹Similarly when maximizing rewards the objective might be non-concave.

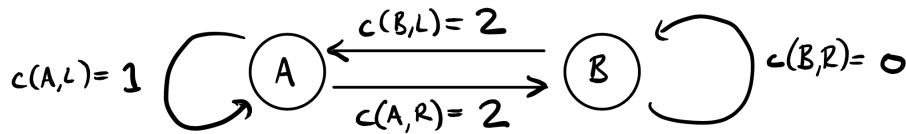


Figure 5.1: Policy gradient counter-example.

A plot of this curve is given in Figure 5.2, and as you can see this is clearly not a convex minimization problem. Thus if θ is initially taken close to zero, a policy gradient algorithm will likely converge on the sub-optimal local minimum at $\theta = 0$.

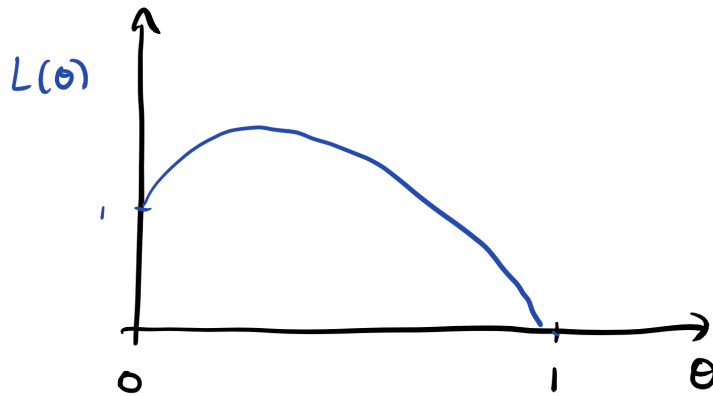


Figure 5.2: The example's non-convex cost function.

What this example shows is that even though the optimal policy is in the parameter set this is not enough to guarantee convergence. Even though our parameterization is linear on a bounded a convex set of parameters. Note this is unlike linear function approximation for the value function which will converge.

A Convergence Proof for Policy Gradients.

Here we give a convergence argument due to recent work of Bhandari and Russo [8]. The idea is to observe that Corollary 1 shows that the policy gradient algorithm is doing a one-step policy iteration update (started a random initial state).

In this we make the following assumptions about the Q -functions and the policy class. Further we assume

- $Q_{\pi_\theta}(x, a)$ is concave in a for all x and θ .
- $\Pi = \{\pi_\theta : \theta \in \Theta\}$ is convex. I.e.

$$\forall \theta, \theta', (1 - \alpha)\pi_\theta + \alpha\pi_{\theta'} \in \Pi.$$

- The policy iteration $\hat{\pi}_\theta$ belongs to Π . I.e.

$$\hat{\pi}_\theta(s) \in \operatorname{argmax}_a Q_{\pi_\theta}(s, a).$$

- From the above $\forall \alpha \in [0, 1] \exists \theta^\alpha$ s.t. $\pi_{\theta^\alpha} = (1 - \alpha)\pi_\theta + \alpha\hat{\pi}_\theta$. We assuming our parameterization is suitably smooth so that the following limit is well defined:

$$\mathbf{u} = \lim_{\alpha \rightarrow 0} \frac{\theta^\alpha - \theta}{\alpha} \quad \text{with} \quad \|\mathbf{u}\|_2 > 0.$$

Notice here \mathbf{u} is the direction that updates in the policy iteration direction:

$$\frac{d\pi_\theta}{d\mathbf{u}} = \hat{\pi}_\theta - \pi_\theta.$$

This is useful given the observation in Corollary 1.

Theorem 27.

$$\frac{dV(\theta)}{d\mathbf{u}} \geq \frac{1}{1 - \beta} \|R(\theta) - \mathcal{V}R(\theta)\|_\theta$$

where \mathcal{V} is the value iteration update: $\mathcal{V}R(x) := \max_a r(x, a) + \beta \mathbb{E}_{x,a}[R(\hat{x})]$, and where

$$\|B\|_\theta = \sum_{t=0}^{\infty} (1 - \beta) \mathbb{E}_{\pi_\theta} [\beta^t |B(\hat{x}_t)|].$$

Proof. Notice that applying the Corollary to the Policy Gradient The-

orem, Corollary 1, we have that

$$\begin{aligned}
\frac{dV(\theta)}{d\mathbf{u}} &= \mathbb{E}_{\pi_\theta} \left[\sum_{t=0}^{\infty} \beta^t \frac{d}{d\mathbf{u}} R_{\pi_\theta}(x_t, \pi_\phi) \right] \\
&= \mathbb{E}_{\pi_\theta} \left[\sum_{t=0}^{\infty} \beta^t \sum_a \frac{\partial Q_{\pi_\theta}}{\partial a}(x_t, a) \frac{d\pi_\theta(a|x)}{d\mathbf{u}} \right] \\
&= \mathbb{E}_{\pi_\theta} \left[\sum_{t=0}^{\infty} \beta^t \sum_a \frac{\partial Q_{\pi_\theta}}{\partial a}(x_t, a) (\hat{\pi}_\theta(a|x) - \pi_\theta(a|x)) \right] \\
&\geq \mathbb{E} \left[\sum_{t=0}^{\infty} \beta^t \{ \mathcal{V}R_{\pi_\theta}(x_t) - R_{\pi_\theta}(x_t) \} \right] \\
&= \frac{1}{1-\beta} \| \mathcal{V}R_{\pi_\theta} - R_{\pi_\theta} \|_\theta.
\end{aligned}$$

□

Now if we assume in addition that

- If $\sum_{t=0}^{\infty} (1-\beta)\beta^t \mathbb{P}_{\pi_\theta}(\hat{x}_t = x) > 0$ for all $x \in \mathcal{X}$.

and we assume the o.d.e dynamics

$$\frac{d\theta}{dt} = \nabla_\theta V(\theta)$$

then the above theorem shows the following

Corollary 2.

$$V(\theta) \rightarrow V^*$$

where V^* is the optimal value function.

Proof. We know from our proof for value iteration that if $R(x, \pi_\theta)$ is sub-optimal then $R(x, \pi_\theta) \neq \mathcal{V}R(x, \pi_\theta)$. Also we know that

$$\begin{aligned}
\frac{dV(\theta)}{dt} &= \|\nabla_\theta V(\theta)\|^2 = \frac{1}{\|\nabla_\theta V(\theta)\|} \max_{v: \|v\|_2=1} v^\top \nabla_\theta V(\theta) \\
&\geq \frac{1}{\|\nabla_\theta V(\theta)\| \cdot \|u_\theta\|} u^\top \nabla_\theta V(\theta) \\
&\geq \frac{\| \mathcal{V}R_{\pi_\theta} - R_{\pi_\theta} \|}{(1-\beta)\|\nabla_\theta V(\theta)\| \cdot \|u_\theta\|} > 0.
\end{aligned}$$

□

5.3 Linear Approximation and TD Learning

First we give the high level idea behind linear function approximation. Then we give a somewhat informal analysis of TD(0).

For a Markov chain $\hat{x} = (\hat{x}_t : t \in \mathbb{Z}_+)$, consider the reward function

$$R(x) := \mathbb{E}_x \left[\sum_{t=0}^{\infty} \beta^t r(\hat{x}_t) \right] \quad (5.15)$$

associated with rewards given by $r = (r(x) : x \in \mathcal{X})$. We approximate the reward function $R(x)$ with a linear approximation,

$$R(x; \mathbf{w}) = \mathbf{w}^\top \boldsymbol{\phi}(x) = \sum_{j \in \mathcal{J}} w_j \phi_j(x).$$

Here we have taken our state x and extracted features, $\phi_j(x)$ for j in finite set \mathcal{J} , that we believe to be important to determining the overall reward function $R(x)$. Interpreting each $\phi_j = (\phi_j(x) : x \in \mathcal{X})$ as a vector, we assume $\{\phi_j : j \in \mathcal{J}\}$ are linearly independent. We then apply a vector of weights $\mathbf{w} = (w_j : j \in \mathcal{J})$ to each of these features. Our job is to find weights that give a good approximation to $R(x)$.

We know for instance that $R(x)$ is a solution to the fixed point equation

$$R(x) = \mathbb{E}_x \left[\underbrace{r(x) + \beta R(\hat{x})}_{=: \text{Target}(x)} \right], \quad x \in \mathcal{X}. \quad (5.16)$$

The target, $\text{Target}(x)$, is an estimate of the true value of $R(x; \mathbf{w})$. Here the target random variable considered is the TD(0) target. Other targets can be used, e.g. the term in the sum, (5.15), would be the Monte-carlo target, and there are various options in between, c.f. TD(λ).

In function approximation, we cannot get the expected reward to equal its target. So we attempt to minimize the difference between them. For example

$$\underset{\mathbf{w}}{\text{minimize}} \quad \mathbb{E}_\mu \left[(\text{Target}(x) - R(x; \mathbf{w}))^2 \right].$$

Here the expectation is over $\boldsymbol{\mu} = (\mu(x) : x \in \mathcal{X})$, the stationary distribution of our Markov process. We can't minimize this since we do not know the stationary distribution $\boldsymbol{\mu}$. We can only get samples

and so we can instead apply Robbin's-Munro/Stochastic Gradient Descent update to w

$$w \leftarrow w + \alpha(\text{Target}(x) - R(x; w))\nabla_w R(x; w).$$

Just like with tabular methods these updates can be applied online, offline, first-visit, every-visit.

Analysis of TD(0).

Let's do an informal analysis of TD(0).

Linear TD(0) Algorithm. For TD(0) our target is $r(x) + \beta R(\hat{x}; w)$, where \hat{x} is the next state after x . Under a linear function approximation this gives an update

$$w \leftarrow w + \alpha(r(x) + \beta w^\top \phi(\hat{x}) - w^\top \phi(x))\phi(x).$$

we let

$$g(x; w) := (r(x) + \beta w^\top \phi(\hat{x}) - w^\top \phi(x))\phi(x).$$

Convergence Result. We argue (informally) that, for this iteration scheme, $w(t) \rightarrow w^*$ where the limit function Φw^* is in a factor of the best approximation

$$\|\mathbf{R} - \mathbf{R}(w^*)\|_\mu \leq \frac{1}{1 - \beta} \|\mathbf{R} - \Pi(\mathbf{R})\|_\mu.$$

Here we interpret \mathbf{R} and $\mathbf{R}(w^*)$ as vectors $\mathbf{R} = (R(x) : x \in \mathcal{X})$ and $\mathbf{R}(w) = (R(x; w) : x \in \mathcal{X})$. Also, we define the norm above by

$$\|\mathbf{R}\|_\mu^2 = \sum_{x \in \mathcal{X}} \mu(x) R(x)^2.$$

Average Behaviour. Suppose that our Markov chain \hat{x} is stationary with stationary distribution $\mu(x)$. If we look at the expected change in our update term we get

$$\begin{aligned} & \mathbb{E}_\mu[g(x; w)] \\ &= \sum_x \mu(x) r(x) \phi(x) + \sum_x \sum_y \mu(x) (\beta \phi(x) P_{xy} \phi(y)^\top - \phi(x) \phi(x)^\top) w \\ &= \Phi^\top M r - \Phi^\top M [I - \beta P] \Phi w. \end{aligned} \tag{5.17}$$

Above Φ is the $\mathcal{X} \times \mathcal{J}$ matrix with entries $\Phi_{xj} = \phi_j(x)$ and M is the $\mathcal{X} \times \mathcal{X}$ diagonal matrix with diagonal entries given by $\mu(x)$. We use these to define the length \mathcal{J} vector b and the $\mathcal{J} \times \mathcal{J}$ matrix A , as defined follows:

$$A := \Phi^\top M[I - \beta P]\Phi \quad \text{and} \quad b := \Phi^\top M r.$$

Differential Equation Analysis. So, roughly, w moves according to the differential equation

$$\frac{dw}{dt} = -Aw + b.$$

Now P is the transition matrix of a Markov chain. Since its rows sum to 1, its biggest eigenvalue is 1. So we can expect that $-(I - \beta P)$ is in some sense "negative", specifically it can be shown that $(\Phi w)^\top M(I - \beta P)\Phi w < -(1 - \beta)\|\Phi w\|_\mu^2$. This then implies that

$$v^\top A v \geq (1 - \beta)\|\Phi v\|_\mu^2.$$

This is proven in Lemma 7 below.

This is sufficient to give convergence of the above differential equation: take w^* such that $Aw^* = b$ and take $L(w) = \frac{1}{2}\|\Phi w - \Phi w^*\|_\mu^2$ then

$$\frac{dL}{dt} = \nabla L(w) \cdot \frac{dw}{dt} = -(w - w^*)^\top A(w - w^*) \leq -(1 - \beta)\|\Phi w - \Phi w^*\|_\mu^2.$$

Thus we see that $R(w(t)) = \Phi w(t) \rightarrow \Phi w^* = R(w^*)$, and since we assume ϕ_j are linearly independent $w(t) \rightarrow w^*$.

Approximation Error. Convergence is great and everything, but we must verify that the solution obtained, w^* , is a "good" solution. First, notice that the reward function $R = (R(x) : x \in \mathcal{X})$ satisfies

$$R = T_0(R), \quad \text{where} \quad T_0(R) := r + \beta P R.$$

This is just (5.16) with R interpreted as a vector and the expectation as a matrix operation with respect to transition matrix P .

Second, notice the approximation $R(w) = (R(x; w) : x \in \mathcal{X})$ that is closest to the rewards R , is given by a projection, specifically

$$\Pi(R) := \Phi(\Phi^\top M \Phi)^{-1} \Phi^\top M R = \underset{R(w)}{\operatorname{argmin}} \|\mathbf{R} - \mathbf{R}(w)\|_\mu^2.$$

Third, we see the equations satisfied by w^* can be expressed in a form somewhat similar to above expression $\mathbf{R} = T_0(\mathbf{R})$. Specifically, we rearrange the expression $Aw^* = b$

$$\begin{aligned} Aw^* = b &\iff \Phi^\top M[I - \beta P]\Phi w^* = \Phi^\top Mr \\ &\iff \Phi^\top M\Phi w^* = \Phi^\top Mr + \Phi^\top M\beta P\Phi w^* \\ &\iff \underbrace{\Phi w^*}_{\mathbf{R}(w^*)} = \underbrace{\Phi[\Phi^\top M\Phi]^{-1}\Phi^\top M}_{\Pi} \underbrace{(\mathbf{r} + \beta P\Phi w^*)}_{T_0(\mathbf{R}(w^*))}. \end{aligned}$$

So, while \mathbf{R} satisfies $\mathbf{R} = T_0(\mathbf{R})$, we see that $\mathbf{R}(w^*)$ satisfies

$$\mathbf{R}(w^*) = \Pi(T_0(\mathbf{R}(w^*))).$$

We can use these identities satisfied by \mathbf{R} and $\mathbf{R}(w^*)$ to show that approximation is comparable to the best approximation of \mathbf{R} . Since T_0 is a β -contraction and that projections always move distances closer (both properties are relatively easy to verify, see Lemma 7):

$$\begin{aligned} \|\mathbf{R} - \mathbf{R}(w^*)\|_\mu &\leq \|\mathbf{R} - \Pi(\mathbf{R})\|_\mu + \|\Pi(T_0(\mathbf{R})) - \Pi(T_0(\mathbf{R}(w^*)))\|_\mu \\ &\leq \|\mathbf{R} - \Pi(\mathbf{R})\|_\mu + \beta\|\mathbf{R} - \mathbf{R}(w^*)\|_\mu. \end{aligned}$$

So

$$\|\mathbf{R} - \mathbf{R}(w^*)\|_\mu \leq \frac{1}{1 - \beta}\|\mathbf{R} - \Pi(\mathbf{R})\|_\mu.$$

Some Formal Analysis. Here are a few formal results that we mention in the discussion above.

Proposition 7. We define $\|\mathbf{R}\|_\mu = \sum_x \mu(x)R(x)^2$ and $\|P\|_\mu = \sup_f \|Pf\|_\mu / \|f\|_\mu$

a) $\|P\|_\mu \leq 1$.

b) The TD(0) map is a contraction that is if for function $R : \mathcal{X} \rightarrow \mathcal{X}$ we let, $T\mathbf{R}(x) = r(x) + \beta P\mathbf{R}(x)$ then

$$\|T\mathbf{R}_1 - T\mathbf{R}_2\|_\mu \leq \beta\|\mathbf{R}_1 - \mathbf{R}_2\|_\mu$$

and

$$\|\Pi T\mathbf{R}_1 - \Pi T\mathbf{R}_2\|_\mu \leq \beta\|\mathbf{R}_1 - \mathbf{R}_2\|_\mu$$

where Π is a projection in $\|\cdot\|_\mu$.

c)

$$(w - w^*)^\top \mathbb{E}_\mu[g(x; w)] \leq -(1 - \beta)\|\Phi w - \Phi w^*\|_\mu^2.$$

Proof. a) Using Jensen's Inequality below,

$$\begin{aligned} \|Pf\|_\mu^2 &= \sum_x \mu(x) \left(\sum_y P_{xy} f(y) \right)^2 \leq \sum_x \mu(x) \sum_y P_{xy} f(y)^2 \\ &= \sum_y f(y)^2 \underbrace{\sum_x \mu(x) P_{xy}}_{=\mu(y)} = \sum_y \mu(y) f(y)^2 = \|f\|_\mu^2. \end{aligned}$$

In the curly brace, we are using that μ is a stationary distribution.
b)

$$\|TR_1 - TR_2\|_\mu = \beta \|P(R_1 - R_2)\|_\mu \leq \beta \|P\|_\mu \|R_1 - R_2\|_\mu$$

Since, $\|P\|_\mu \leq 1$, the result holds. Further since projections reduce distances $\|\Pi R\|_\mu \leq \|R\|_\mu$ the 2nd inequality holds too.

c) From the above calculation in (5.17), we have that

$$\begin{aligned} \mathbb{E}_\mu[g(x; \mathbf{w})] &= \Phi^T M[r + \beta P - I] \Phi \mathbf{w} \\ &= \Phi^T M[T(\Phi \mathbf{w}) - \Phi \mathbf{w}] \\ &= \Phi^T M[(I - \Pi) + \Pi] [T(\Phi \mathbf{w}) - \Phi \mathbf{w}] \\ &= \Phi^T M[\Pi T(\Phi \mathbf{w}) - \Phi \mathbf{w}]. \end{aligned}$$

In the third, equality we use that $\Phi^T(I - \Pi) = 0$ which holds since Π is a projection in $\|\cdot\|_\mu$ onto the space spanned by Φ and thus $(I - \Pi)$ is orthogonal to this space.

Now applying the above inequality

$$\begin{aligned} &(\mathbf{w} - \mathbf{w}^*)^T \mathbb{E}_\mu[g(x; \mathbf{w})] \\ &= (\Phi \mathbf{w} - \Phi \mathbf{w}^*)^T M [\Pi T(\Phi \mathbf{w}) - \Phi \mathbf{w}] \\ &= (\Phi \mathbf{w} - \Phi \mathbf{w}^*)^T M [\Pi T(\Phi \mathbf{w}) - \Pi T(\Phi \mathbf{w}^*) + \Phi \mathbf{w}^* - \Phi \mathbf{w}] \\ &= -\|\Phi \mathbf{w} - \Phi \mathbf{w}^*\|_\mu^2 + (\Phi \mathbf{w} - \Phi \mathbf{w}^*)^T M [\Pi T(\Phi \mathbf{w}) - \Pi T(\Phi \mathbf{w}^*)] \\ &\leq -\|\Phi \mathbf{w} - \Phi \mathbf{w}^*\|_\mu^2 + \|\Phi \mathbf{w} - \Phi \mathbf{w}^*\|_\mu \|\Pi T(\Phi \mathbf{w}) - \Pi T(\Phi \mathbf{w}^*)\|_\mu^2 \\ &\leq -\|\Phi \mathbf{w} - \Phi \mathbf{w}^*\|_\mu^2 + \|\Phi \mathbf{w} - \Phi \mathbf{w}^*\|_\mu \beta \|\Phi \mathbf{w} - \Phi \mathbf{w}^*\|_\mu \\ &= -(1 - \beta) \|\Phi \mathbf{w} - \Phi \mathbf{w}^*\|_\mu^2 \end{aligned}$$

In the second equality above, we use that \mathbf{w}^* satisfied $\Pi T(\Phi \mathbf{w}^*) = \Phi \mathbf{w}^*$. The first inequality is the Cauchy-Schwartz inequality. The second inequality, applies the contraction property from part b). \square

References

The section is based on reading Tsitsiklis and Van Roy [\[50\]](#), but also see Bertsekas and Tsitsiklis [\[7\]](#).

5.4 Linear Approximation and Stopping

We consider an optimal stopping problem as introduced in Section 0.6, and we apply a linear function approximation scheme like in Section 5.3.

It is not straight-forward to prove that the results of Section 5.3 can be extended reinforcement learning methods like Q-learning. However, optimal stopping is an example where this is possible. There are essentially two reasons why:

1. A stopping rule does not interact with the process being estimated. I.e. Changing the stopping rule does not change the underlying process, which is not true for other MDPs.
2. The Q function has a simple form, like $Q = r + \beta \max\{\bar{r}, V\}$, which remains a contraction. (For example, if r, \bar{r}, β and V are real numbers then a short calculation shows that $|Q' - Q| \leq \beta|V - V'|$.)

This suggests that the analysis for approximating rewards R should pass over to optimal stopping problems.

Optimal Stopping Recap. We consider the problem of stopping to maximize rewards rather than minimizing costs. We briefly recall that we wish to stop a Markov chain $\hat{x} = (\hat{x}_t : t \in \mathbb{Z}_+)$ with values in state space \mathcal{X} and transition probabilities given by the matrix $P = (P_{xy} : x, y \in \mathcal{X})$. Here $r(x)$ is the reward for continuing at state x and $\bar{r}(x)$ is the reward for stopping at state x . We consider the MDP

$$V(x) = \max_{\tau} \mathbb{E}_x \left[\sum_{t=0}^{\tau-1} \beta^t r(\hat{x}_t) + \beta^{\tau} \bar{r}(\hat{x}_{\tau}) \right] \quad (5.18)$$

with discount factor $\beta \in (0, 1)$. The above maximization is take over all stopping times τ . The Bellman equation for this problem is

$$V(x) = \max \{ \bar{r}(x), r(x) + \beta E_x[V(\hat{x})] \} .$$

We view the left-hand expression as an operation on a vector. Specifically, for $\mathbf{R} = (R(x) : x \in \mathcal{X})$, we let

$$T\mathbf{R}(x) = \max \{ \bar{r}(x), r(x) + \beta E_x[R(\hat{x})] \} .$$

For the value function V , we define the optimal Q -factor by

$$Q(x) = r(x) + \beta \mathbb{E}_x[V(x)] .$$

and we define the operation S on the \mathcal{Q} -function vector $\mathbf{Q}' = (Q'(x) : x \in \mathcal{X})$ by

$$SQ'(x) = r(x) + \beta \mathbb{E}_x [\max \{\bar{r}(\hat{x}), Q'(\hat{x})\}] .$$

The value function V in (5.18) is the unique solution to the Bellman equation and moreover an optimal rule is found by letting $\tau^* = \min\{t : \bar{r}(x_t) \geq V(x_t)\}$. Note by the definitions above $V(x) = \max\{\bar{r}(x), Q(x)\}$, see Section 0.6 and Theorem 36 in Section 0.4. So equivalently we can see that

$$\tau^* = \min\{t : \bar{r}(x_t) \geq Q(x_t)\}$$

gives the optimal stopping time.

Throughout this section we assume that the Markov chain $\hat{x} = (\hat{x}_t : t \in \mathbb{Z}_+)$ is positive recurrent with stationary distribution $\mu = (\mu(x) : x \in \mathcal{X})$.

Function approximation. We approximate the Q -function $Q(x)$ with a linear approximation,

$$Q(x; \mathbf{w}) = [\Phi \mathbf{w}]_x = \mathbf{w}^\top \phi(x) = \sum_{j \in \mathcal{J}} w_j \phi_j(x).$$

Here, like before, we have taken our state x and extract linearly independent features, $\phi_j = (\phi_j(x) : x \in \mathcal{X})$ for j in finite set \mathcal{J} . These features should be important in determining the function $Q(x)$. We then apply a vector of weights $\mathbf{w} = (w_j : j \in \mathcal{J})$ to each of these features. Our job is to find weights so that $Q(x; \mathbf{w})$ gives a good approximation to $Q(x)$. We can interpret the Q -function and approximation as vectors with components in \mathcal{X} given by $\mathbf{Q} = (Q(x) : x \in \mathcal{X})$ and $\mathbf{Q}_w = \Phi \mathbf{w}$ where Φ is a matrix whose x -th column is given by $\phi(x) = (\phi_j(x) : j \in \mathcal{J})$. We we can use this to define a stopping policy given by

$$\tau(\mathbf{w}) = \min \{t : \bar{r}(x) \geq Q(x; \mathbf{w}^*)\} .$$

Since $\tau(\mathbf{w})$ does not necessarily induce the same \mathcal{Q} -function as $Q(x; \mathbf{w}^*)$. We will later consider the operation

$$S_w Q(x) := r(x) + \beta \mathbb{E}_x [r(\hat{x}) \mathbb{I}[r(x) \geq \Phi \mathbf{w}(x)] + Q(x) \mathbb{I}[r(x) < \Phi \mathbf{w}(x)]]$$

Think of S_w being the value of following stopping rule $\tau(\mathbf{w})$ on the next step and there afterward following Q . It is immediate that

$$S_w \Phi \mathbf{w} = S \Phi \mathbf{w}$$

As before it is useful to consider the projection onto the space spanned by the basis functions Φ , namely,

$$\Pi(Q) := \Phi(\Phi^\top M\Phi)^{-1}\Phi^\top M Q = \underset{Q_w}{\operatorname{argmin}} \|Q - Q_w\|_\mu^2 \quad (5.19)$$

where as before $\|Q\|_\mu^2 = \sum_{x \in \mathcal{X}} \mu(x) Q(x)^2$ and M is the diagonal matrix with diagonal entries given by $\mu(x)$ for $x \in \mathcal{X}$.

Approximation Algorithm. An approximation algorithm is given by the update

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha_t \phi(x_t) (r(x_t) + \beta \max\{Q(x_{t+1}; \mathbf{w}_t), \bar{r}(x_{t+1})\} - Q(x_t; \mathbf{w}_t)) \quad (5.20)$$

here α_t is a step size parameter. Notice this is similar the TD(0) update given in Section 5.3.

If we let

$$\mathbf{g}(\mathbf{w}) = \phi(x_t) (r(x_t) + \beta \max\{Q(x_{t+1}; \mathbf{w}_t), \bar{r}(x_{t+1})\} - Q(x_t; \mathbf{w}_t))$$

Then the update (5.20) is simply, $\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha_t \mathbf{g}(\mathbf{w}_t)$.

Average Behavior. If we look at the expected change in $\mathbf{g}(\mathbf{w})$ given $x_t = x$, we get that

$$\begin{aligned} \mathbb{E}_x[\mathbf{g}(\mathbf{w})] &= \phi(x) \left(r(x) + \beta \sum_y P_{xy} \max\{\phi(y)^\top \mathbf{w}, \bar{r}(y)\} - \phi(x)^\top \mathbf{w} \right) \\ &= \phi(x) (SQ_w(x) - Q_w(x)) \end{aligned}$$

If we look at the average change this induces under stationary distribution $\mu(x)$ we get

$$\begin{aligned} \mathbb{E}_\mu[\mathbf{g}(\mathbf{w})] &= \sum_x \phi(x) \mu(x) (SQ_w(x) - Q_w(x)) \\ &= \Phi^\top M S \Phi \mathbf{w} - \Phi^\top M \Phi \mathbf{w} \end{aligned} \quad (5.21)$$

where we use the definition that $Q_w = \Phi \mathbf{w}$. So the change in the algorithm is stationary when

$$0 = \mathbb{E}_\mu[\mathbf{g}(\mathbf{w}^*)] = \Phi^\top M S \Phi \mathbf{w}^* - \Phi^\top M \Phi \mathbf{w}^*$$

Rearranging and applying Φ to both sides gives

$$\begin{aligned} \Phi \mathbf{w}^* &= \Phi (\Phi^\top M \Phi)^{-1} \Phi^\top M S \Phi \mathbf{w}^* \\ &= \Pi S \Phi \mathbf{w}^*. \end{aligned}$$

the last part follows by the definition of the projection Π (5.19).

Differential Equation Analysis. The analysis here is essentially the same as the differential equation analysis in Section 5.3. Again the change in $w(t)$ is approximated by the ODE

$$\frac{d\mathbf{w}(t)}{dt} = -A(\mathbf{w}(t))$$

where now

$$A(\mathbf{w}(t)) = \mathbb{E}_\mu [g(\mathbf{w}(t))] = \Phi^\top M(I - S)\Phi(\mathbf{w} - \mathbf{w}^*).$$

Note that A is no longer a linear function in $w(t)$. However, similar to our analysis before – where we argued $-(I - \beta P)$ was spectrally negative because P is a 1-contraction in $\|\cdot\|_\mu$ – we can argue that $-(I - S)$ is spectrally negative because S is a contraction. See Proposition 8a). Thus

$$\frac{dL}{dt} = \nabla L(\mathbf{w}) \frac{d\mathbf{w}}{dt} \leq -(\mathbf{w} - \mathbf{w}^*)^\top A(\mathbf{w} - \mathbf{w}^*).$$

This implied $L(\mathbf{w}(t)) \rightarrow 0$ and from this we can argue that $\mathbf{w}(t) \rightarrow \mathbf{w}^*$.

Approximation Error. If we let $\tau(\mathbf{w}^*)$ be the stopping rule induced by \mathbf{w}^* from the above convergence argument then we can argue that its reward behaves similarly to projecting the optimal policy:

$$\mathbb{E}[V(x_0)] - \mathbb{E}[R_{\tau(\mathbf{w}^*)}(x_0)] \leq \frac{2\beta}{(1-\beta)^2} \|\Pi Q^* - Q^*\|_\mu$$

The argument is similar to the TD(0) case, previously. However, as discussed the argument is complicated by the fact the Q -factor for $\tau(\mathbf{w})$ is not the same as $Q(x; \mathbf{w})$. The full argument is proven more formally in (8) below.

Some Formal Analysis. Here we present the main ingredients applied in this section. (Excluding the differential equation argument.)

Proposition 8.

a) The maps S , ΠS and S_w are β -contractions with respect to the norm $\|\cdot\|_\mu$, that is

1. $\|SQ - SQ'\|_\mu \leq \beta \|Q - Q'\|_\mu$.
2. $\|\Pi SQ - \Pi SQ'\|_\mu \leq \beta \|Q - Q'\|_\mu$.

$$3. \|S_w Q - S_w Q'\|_\mu \leq \beta \|Q - Q'\|_\mu .$$

b)

$$(w - w^*) \mathbb{E}_\mu [g(w)] < 0 \quad \forall w \neq w^*$$

and

$$\mathbb{E}_\mu [g(w^*)] = 0$$

where w^* solves the equation

$$\Pi(S\Phi w^*) = \Phi w^*$$

c)

$$\mathbb{E}[V(x_0)] - \mathbb{E}[R_{\tau(w^*)}(x_0)] \leq \frac{2\beta}{(1-\beta)^2} \|\Pi Q^* - Q^*\|_\mu$$

where here it is assumed that the expectation is taken with respect to the stationary distribution of x_0 , namely μ .

Proof. a) part 1 & 2) Since Π is a projection it is also a contraction. So

$$\|\Pi S Q - \Pi S Q'\|_\mu \leq \|S Q - S Q'\|_\mu .$$

So it remains to show that S is a contraction, which holds as follows.

$$\begin{aligned} \|S Q - S Q'\|_\mu &\leq \beta \|P \max\{\bar{r}, Q\} - P \max\{\bar{r}, Q'\}\|_\mu \\ &\leq \beta \|\max\{\bar{r}, Q\} - \max\{\bar{r}, Q'\}\|_\mu \\ &\leq \beta \|Q - Q'\|_\mu . \end{aligned}$$

Here we use the fact that $\|PR\|_\mu \leq \|R\|_\mu$ which we showed in Proposition 7 and a straight-forward calculation that shows that for real numbers a, b, c we have $|\max\{a, c\} - \max\{b, c\}| \leq |a - b|$.

a) part 3) Recall the definition of S_w . We let F_w be the future value given by

$$F_w Q(x) := \begin{cases} \bar{r}(x) & \text{if } \bar{r}(x) \geq \Phi w(x), \\ Q(x) & \text{otherwise .} \end{cases}$$

So $S_w Q = r + \beta F_w Q$. Now

$$\|S_w Q - S_w Q'\|_\mu = \beta \|PF_w Q - PF_w Q'\|_\mu \leq \beta \|F_w Q - F_w Q'\|_\mu \leq \beta \|Q - Q'\|_\mu$$

In the first inequality we use that $\|P\|_\mu \leq 1$, from Proposition 7, and the final inequality holds because

$$\begin{aligned} \|F_w Q - F_w Q'\|_\mu^2 &= \sum_x \mu(x) |Q(x) - Q'(x)|^2 \mathbb{I}[\bar{r}(x) < \Phi w(x)] \\ &\leq \sum_x \mu(x) |Q(x) - Q'(x)|^2 = \|Q - Q'\|_\mu^2 \end{aligned}$$

b) The following argument relies on the fact that ΠS is a contraction:

$$\begin{aligned}
& (\mathbf{w} - \mathbf{w}^*) \mathbb{E}_\mu [g(\mathbf{w})] \\
&= (\mathbf{w} - \mathbf{w}^*) [\Phi^\top M S \Phi \mathbf{w} - \Phi^\top M \Phi \mathbf{w}] \\
&= (\mathbf{w} - \mathbf{w}^*) \Phi^\top M [\Pi S \Phi \mathbf{w} - \Phi \mathbf{w}] \\
&= (\mathbf{w} - \mathbf{w}^*) \Phi^\top M [\Pi S \Phi (\mathbf{w} - \mathbf{w}^*) - \Phi \mathbf{w} + \Phi \mathbf{w}^*] \\
&= (\Phi \mathbf{w} - \Phi \mathbf{w}^*)^\top M \Pi S (\Phi \mathbf{w} - \Phi \mathbf{w}^*) - (\Phi \mathbf{w} - \Phi \mathbf{w}^*)^\top M (\Phi \mathbf{w} - \Phi \mathbf{w}^*) \\
&\leq \|\Phi \mathbf{w} - \Phi \mathbf{w}^*\|_\mu \|\Pi S \Phi \mathbf{w} - \Pi S \Phi \mathbf{w}^*\|_\mu - \|\Phi \mathbf{w} - \Phi \mathbf{w}^*\|_\mu^2 \\
&\leq -(1 - \beta) \|\Phi \mathbf{w} - \Phi \mathbf{w}^*\|_\mu^2
\end{aligned}$$

In the first equality, we apply (5.21). In the second equality, we apply that $\Phi^\top M = \Phi^\top M (\Pi + (I - \Pi)) = \Phi^\top M \Pi$. (Note Π is a projection with respect to μ onto the space spanned by Φ thus $I - \Pi$ is orthogonal to this space). The first inequality above applies the Cauchy-Schwartz Inequality. The second inequality applies that ΠS is a contraction.

c)

$$\begin{aligned}
& \beta \mathbb{E} [V(x_0)] - \beta \mathbb{E} [V_{\mathbf{w}^*}^\tau(x_0)] \\
&= \mathbb{E} [r(x_0) - \beta P V(x_0)] - \mathbb{E} [r(x_0) - \beta P V_{\mathbf{w}^*}^\tau(x_0)] \\
&= |\mathbb{E} [Q(x_0) - Q_{\mathbf{w}^*}^\tau(x_0)]| \\
&\leq \|Q - Q_{\mathbf{w}^*}^\tau\|_\mu
\end{aligned}$$

The last inequality, above, applies Jensen's Inequality. Thus

$$\mathbb{E} [V(x_0)] - \mathbb{E} [V_{\mathbf{w}^*}^\tau(x_0)] \leq \beta^{-1} \|Q - Q_{\mathbf{w}^*}^\tau\|_\mu \quad (5.22)$$

Similar to the argument in Proposition 7 we have

$$\begin{aligned}
\|Q - Q_{\mathbf{w}^*}^\tau\|_\mu &= \|Q - S(\Phi \mathbf{w}^*) + S^\tau(\Phi \mathbf{w}^*) - Q_{\mathbf{w}^*}^\tau\|_\mu \\
&\leq \|S Q - S(\Phi \mathbf{w}^*)\|_\mu + \|S^\tau(\Phi \mathbf{w}^*) - S^\tau(Q_{\mathbf{w}^*}^\tau)\|_\mu \\
&\leq \beta \|Q - \Phi \mathbf{w}^*\|_\mu + \beta \|\Phi \mathbf{w}^* - Q_{\mathbf{w}^*}^\tau\|_\mu \\
&\leq 2\beta \|Q - \Phi \mathbf{w}^*\|_\mu + \beta \|Q - Q_{\mathbf{w}^*}^\tau\|_\mu
\end{aligned}$$

Here we use that $S(\Phi \mathbf{w}^*) = \Phi \mathbf{w}^* = S^\tau(\Phi \mathbf{w}^*)$ and the contraction property for both S and S^τ .

Thus, rearranging the above gives,

$$\|Q - Q_{\mathbf{w}^*}^\tau\|_\mu \leq \frac{2\beta}{1 - \beta} \|Q - \Phi \mathbf{w}^*\|_\mu$$

Applying (5.22) to this gives

$$\mathbb{E}[V(x_0)] - \mathbb{E}[V_{w^*}^{\tau}(x_0)] \leq \frac{2}{(1-\beta)^2} \|Q - \pi(Q)\|_{\mu}.$$

□

References

This section is based on reading Tsitsiklis and Van Roy [51], but also see Bertsekas and Tsitsiklis [7]. Further approaches to optimal stopping are discussed in the book of Glasserman [19].

5.5 Cross Entropy Method

In the *Cross Entropy Method*, we wish to estimate the likelihood

$$l = \mathbb{P}(S(X) \geq \gamma).$$

Here X is a random variable whose distribution is known and belongs to a parametrized family of densities $f(\cdot, v)$. Further $S(X)$ is often a solution to an optimization problem.

It is assumed that l is a relatively rare event, say of order 10^{-5} . One way to estimate l is to take N IID samples X_1, \dots, X_N and then average

$$\hat{l} = \frac{1}{N} \sum_{i=1}^N \mathbb{I}[S(X_i) \geq \gamma].$$

This gives us an unbiased estimate. However, because the event is rare, we may require a large number of samples N in order to provide a good estimate of l . *Importance sampling* is a straightforward method that can mitigate these effects. Here sample each X_i from a different density $g(\cdot)$ and perform the estimate

$$\hat{l}_g = \frac{1}{N} \sum_{i=1}^N \mathbb{I}[S(X_i) \geq \gamma] W(x) \quad (5.23)$$

where $W(x) = f(x)/g(x)$. Such a change can improve the estimation of l . For instance, the choice $g^*(x) = \mathbb{I}[S(x) \geq \gamma]f(x)/l$ gives the correct estimate of l with certainty. Clearly g^* cannot work in practice; it requires knowledge of l , the parameter which we are estimating. Nonetheless a good estimate of g^* may be possible. The point of the cross entropy method is to find a reasonable choice for g so that importance sampling can be applied.

In particular, the *cross-entropy method* advocates estimating $g^*(\cdot)$ by $f(\cdot; v)$ according to a parameter choice that minimizes the relative entropy between g^* and $f(\cdot, v)$. Namely,

$$\min_v D(f(\cdot; v) \| g^*(\cdot)) := - \int g(x) \log \frac{f(x; v)}{g(x)} dx.$$

With a little rearrangement, we see that this is equivalent to the optimization

$$\max_v \int \mathbb{I}[S(x) \geq \gamma] f(x; u) \log f(x; v) dx.$$

With importance sampling in mind, we apply a change of measure to the above integral to get

$$\max_v \int \mathbb{I}[S(x) \geq \gamma] f(x; w) W(x; u, v) \log f(x; v) dx$$

where $W(x; u, w) = f(x; u)/f(x; w)$. The above optimization may not be explicitly solvable so we replace it with the (importance sampled) optimization

$$\max_v \frac{1}{N} \sum_{i=1}^N \mathbb{I}[S(X_i) \geq \gamma] W(X_i; u, w) \log f(x; v) dx \quad (5.24)$$

where X_i are IID samples from density $f(\cdot; w)$.

In many examples the above optimization is concave and solvable either by descent methods or by differentiating and solving for the condition

$$\sum_{i=1}^N \mathbb{I}[S(X_i) \geq \gamma] W(X_i; u, w) \nabla \log f(x; v) = 0.$$

The reason for incorporating importance sampling into the estimate (5.24) is because the event $\{\}$ could be too unlikely to form an accurate estimate under the choice $w = v$. The idea behind the cross entropy method is to apply a *multi-level* approach where we sequentially increase γ and w as follows

- We estimate the ρ -th percentile of $S(X)$ by taking by ordering the samples $S(X_1) \leq S(X_2) \leq \dots \leq S(X_N)$ and calculating the empirical ρ -th percentile, which gives the update

$$\gamma' = S_{\lceil(1-\rho)N\rceil}$$

The event ρ is chosen to be reasonably likely, e.g. $\rho = 0.01$. Note that since, in principle, the parameter γ' is such that less likely $\{S(X) \geq \gamma'\}$ under v when compared with the previous choice of γ .

- Given this γ' , we find v' as the best estimate according to our cross-entropy rule (5.24). This, in principle, should increase the likelihood of the event $\{S(X) \geq \gamma'\}$ when compared with prior choice of v .

- When γ' passes the required target level γ then the have found a parameter choice that gives reasonable importance to the event $\{S(X) \geq \gamma\}$ and we can calculate l with importance sampling rule (5.23).

Chapter 6

Reinforcement Learning with Neural Networks

6.1 Deep Q-Network (DQN)

Deep Q-Network (DQN) is a simple adaptation of Q-learning to neural networks. Recall that Q-learning performs the update

$$Q(x, a) \leftarrow Q(x, a) + \alpha \left(r + \max_{\hat{a}} Q(\hat{x}, \hat{a}) - Q(x, a) \right),$$

where the 4-tuple (x, a, r, \hat{x}) consist of the current state, action, reward and next state. When we apply function approximation (with a neural network) to Q-factor, $Q_w(x, a)$ for weights w , we must apply the appropriate gradient descent update:

$$w \leftarrow w + \alpha \left(r + \max_{\hat{a}} Q_w(\hat{x}, \hat{a}) - Q_w(x, a) \right) \nabla_w Q(x, a).$$

Notice this is the natural extension of the $TD(0)$ update under ???. The application of this algorithm to reinforcement learning has existed for some time along with its use with Neural Networks. However, along with progress in deep neural networks there have been adaptations to this basic scheme that improve stability and performance for this basic algorithm. For DQN, these are the use of fixed targets and experience replay.

Fixed Targets. The basic idea of fixed Q-targets is you fix the weights w' and then continue to update a copy of the weights w . Specifically this leads to the update:

$$w \leftarrow w + \alpha \left(r + \max_{\hat{a}} Q_{w'}(\hat{x}, \hat{a}) - Q_w(x, a) \right) \nabla_w Q(x, a).$$

After some period of time you reset $w' = w$ and then continue to update w and stays w' fixed at its new value until it is next reset. It's not a big change; notice the apostrophe now in the Q-factor that we maximize. But it helps. Here are two reasons why.

1. Notice the above step corresponds to a stochastic gradient descent step on the objective

$$\mathbb{E} \left[\left(r + \max_{\hat{a}} Q_{w'}(\hat{x}, \hat{a}) - Q_w(x, a) \right)^2 \right]$$

Here, in the context of supervised learning, $Q_w(x, a)$ is the prediction for the output given input (x, a) and $r + \max_{\hat{a}} Q_{w'}(\hat{x}, \hat{a})$ is

the output. In supervised learning the output is fixed in distribution given the input. If the parameter w' was not fixed, but was equal to w , then each step would alter the output at each step some non-trivial way. By fixing Q-targets, we can treat the reinforcement learning problem as a supervised learning problem until the next reset of w' . This simplifies the problem of fitting $Q_w(x, a)$ as standard supervised learning approaches can be to get a good fit.

2. At the point w' is updated, it holds that

$$Q_w(x, a) \approx r + \max_{\hat{a}} Q_{w'}(\hat{x}, \hat{a}).$$

Thus when we update w' , we are in essence performing a policy iteration update.

In summary, fixing Q-targets separates out the problem into a supervised learning task where we update w and a policy improvement step where we update w' . In this regard it is a very neat idea.

Experience Replay. Experience replay is the idea that we store a large number (several episodes worth) of (x, a, r, \hat{x}) in memory and then we sample from this memory, e.g. at random, and use this to do a weight update. At every simulation step we add the newest experience (x, a, r, \hat{x}) to the replay memory and remove the oldest.

Q-learning which works off-line does not really mind what order data is received. However, stochastic gradient descent tends to work much better if there is not a high degree of correlation between steps. Specifically if we send data $(\hat{x}_t, a_t, r_t, \hat{x}_{t+1})$ in the same order that it is received from the simulator, then updates might cause the weights to wander off. (They will come back but essentially we have added variance to an already noisy process.)

Additional Variations.

Prioritized Experience Replay. The idea here is to rank the 4-tuples (x, a, r, \hat{x}) in the replay memory. This is done by recording the absolute value of the TD error of each (x, a, r, \hat{x}) in memory and then forming a ranking from highest to lowest. Recall the TD error is given by

$$\delta = r + \beta \max_{\hat{a}} Q_{w'}(\hat{x}, \hat{a}) - Q_w(x, a)$$

There are two basic variants: first, you select the highest ranked error in memory; second, you select n -th highest ranked according to a probability:

$$P_n = \frac{n^{-\alpha}}{\sum_{m=1}^M m^{-\alpha}}$$

Here is a parameter $\alpha = 0$ corresponds to uniform sampling and $\alpha = \infty$ corresponds to highest ranked. In either case you need to importance sample as we know that an unbiased gradient update is the same as uniformly sampling. So the basic Q-learning step becomes:

$$w \leftarrow w + \alpha \left(\frac{1}{NP_n} \right)^\beta \delta \nabla_w Q(x, a).$$

The parameter β introduces bias, Note if $\beta = 1$ then this is the correct importance sampling, however, it could be argued that earlier in training you want to care more about the gradient updates for TDs with large (and probably previously unseen) experience. So taking $\beta \approx 0$ initially and linearly increasing to $\beta = 1$ is recommended. You can also throw caution to the wind and set $\beta = 0$, as training with ultimately be most effected by large TD errors and by biasing towards these you are attempting to deal with these errors as best as possible.

Appendix A

Appendix

A.1 Probability.

All results here can be found in Williams [57], except the Martingale Central Limit Theorem. Here instead see Hall and Heyde [22] and, for the Functional Martingale Central Limit Theorem, see Whitt [55].

Probability Inequalities

Below, unless stated otherwise, A_n , $n = 1, 2, \dots$ are an events. X is a RV with mean μ , variance σ^2 and moment generating function $M_X(\theta) := \mathbb{E}[\exp \theta X]$. $S_n = \sum_{k=1}^n X_k$, where X_k are IID instances with mean μ_k and variance σ_k^2 .

$$\mathbb{P}\left(\bigcup_{n=1}^{\infty} A_n\right) \leq \sum_{n=1}^{\infty} \mathbb{P}(A_n) \quad \text{(Union bound)}$$

$$\mathbb{P}(X \geq x) \leq \frac{\mu}{x} \quad \text{for } X \geq 0 \quad \text{(Markov's inequality)}$$

$$\mathbb{P}(|X - \mu| \geq x) \leq \frac{\sigma^2}{x^2} \quad \text{(Chebychev inequality)}$$

$$\mathbb{P}(X \geq x) \leq \exp\left\{-\max_{\theta \geq 0} (\theta x - \log M_X(\theta))\right\} \quad \text{(Chernoff Bound.)}$$

$$\mathbb{P}(S_n - \mu n \geq x) \leq \exp\left\{-\frac{x^2}{2 \sum_{i=1}^n c_i^2}\right\} \quad \text{if } |X_i| \leq c_i \quad \text{(Hoeffding's Ineq.)}$$

$$\mathbb{P}(S_n - \mu n \geq x) \leq \exp\left\{-\frac{x^2}{x + 2 \sum_{i=1}^n \sigma_i^2}\right\} \quad \text{(Bernstein's Ineq.)}$$

Heoffding is a better tail bound than Bernstein, but Bernstein's Inequality is better when $\sum_{i=1}^n \sigma_i = o(\sum_{i=1}^n c_i)$, here take $x = \sum_{i=1}^n \sigma_i$ in Bernstein's bound.

$$\begin{aligned}
 X > 0 &\implies \mathbb{E}[X] > 0 && \text{(Positivity)} \\
 f(\mathbb{E}[X]) &\leq \mathbb{E}[f(X)] \quad \text{for } f \text{ convex} && \text{(Jensen's Inequality)} \\
 \|X\|_p &\leq \|X\|_q \quad \text{for } p \leq q && \text{(Minkowski's Inequality)} \\
 \mathbb{E}[XY] &\leq \|X\|_2 \|Y\|_2 && \text{(Cauchy-Schwartz)} \\
 \mathbb{E}[XY] &\leq \|X\|_p \|Y\|_q \quad \text{for } \frac{1}{p} + \frac{1}{q} = 1 && \text{(Holder's Inequality)}
 \end{aligned}$$

Probability Limits

Here we assume A_1, A_2, \dots is a sequence of events.

$$\mathbb{P}(A_n \text{ occurs infinitely often}) = 0 \quad \text{if } \sum_n \mathbb{P}(A_n) < \infty$$

(Borel Cantelli Lemma)

$$\mathbb{P}(A_n \text{ occurs infinitely often}) = 1 \quad \text{if } \sum_n \mathbb{P}(A_n) = \infty$$

and $A_n, n \in \mathbb{N}$, are independent
(2nd Borel Cantelli Lemma)

Here we assume that X_1, X_2, \dots is a sequence of random variables (possibly not independent)

$$\mathbb{E}[X_n] \nearrow \mathbb{E}[X_\infty], \quad \text{for } X_n \leq X_{n+1}.$$

(Monotone Convergence Theorem)

$$\mathbb{E} \left[\liminf_{n \rightarrow \infty} X_n \right] \leq \liminf_{n \rightarrow \infty} \mathbb{E}[X_n], \quad \text{for } X_n \geq 0. \quad \text{(Fatou's Lemma)}$$

$$\mathbb{E}[X_n] \rightarrow \mathbb{E}[X_\infty] \quad \text{for } |X_n| \leq Y \text{ with } \mathbb{E}Y < \infty$$

(Bounded Convergence Theorem)

Conditional Expectation

In what follows, we assume that X and Y are random variables which need not be real valued, e.g. $X = (X_1, X_2, \dots, X_n)$ or $Y = (Y_t : t \in \mathbb{Z}_+)$, and Z is a real valued random variable.

Formally the conditional expectation should be defined in terms of sigma-fields. People seem to get scared of these, so we phrase these properties in terms of (vectors of) random variables and functions of these random variables.

$\mathbb{E}[Z|X] := g(X)$ such that

$$\mathbb{E}[g(X)f(X)] = \mathbb{E}[Zf(X)] \quad \text{for all } f \quad (\text{Def. } \mathbb{E}[Z|X])$$

Eg. take $g(x) = \mathbb{E}[Z|X = x]$.

$$\mathbb{E}[\mathbb{E}[Z|X]f(X)] = \mathbb{E}[Zf(X)]$$

$$\mathbb{E}[Z_1Z_2|X] = Z_1\mathbb{E}[Z_2|X] \quad \text{if } Z_1 = f(X) \quad (\text{Taking out what is known})$$

$$\mathbb{E}[\mathbb{E}[Z|X]|Y] = \mathbb{E}[Z|Y] \quad \text{if } Y = f(X) \quad (\text{Tower Property})$$

$$\mathbb{E}[Z|X] = \mathbb{E}[Z] \quad \text{if } Z, X \text{ are independent}$$

$$\mathbb{E}[Z|X, Y] = \mathbb{E}[Z|X] \quad \text{if } Y \text{ is independent of } X \text{ and } Z. \quad (\text{Role of independence})$$

$$\mathbb{E}[g(X, Y)|X = x] = \mathbb{E}[g(x, Y)] \quad \text{if } X \text{ and } Y \text{ are independent.}$$

Martingales and Stopping

We condition with respect to a sequence of random variables, in particular we let $F_n = (X_1, \dots, X_n)$ (note F_n is a function of F_{n+1} so the Tower Property of the conditional expectation applies).

Suppose M_n is a sequence of RVs such that M_n is a function of F_n and $\mathbb{E}[M_n]$ has finite expectation then we say M_n is a Martingale if

$$\mathbb{E}[M_n|F_{n-1}] = M_{n-1} \quad (\text{Martingale Property})$$

Also for supermartingales and submartingales

$$\mathbb{E}[M_n|F_{n-1}] \leq M_{n-1} \quad (\text{Supermartingale Property})$$

$$\mathbb{E}[M_n|F_{n-1}] \geq M_{n-1} \quad (\text{Submartingale Property})$$

(note b in 'sub' points up and the p in 'super' points down in the same direction of the process.) We abbreviate 'Martingale' to 'Mg'.

- If M_t is a super-Mg with $\sup_t \mathbb{E}[|M_t|] < \infty$ or $M_t \geq 0$ then the limit :

$$M_\infty := \lim_{n \rightarrow \infty} M_n \quad \text{exists} \quad (\text{Doob's Mg Convergence Thrm})$$

- If M_t is a positive sub-martingale then

$$\mathbb{P}\left(\sup_{n \leq t} M_n \geq x\right) \leq \frac{\mathbb{E}[M_t]}{x} \quad (\text{Doob's Sub-Mg Inequality})$$

(This is like Markov's Inequality)

- If M_t is a Mg and f is convex (with $\mathbb{E}[|f(M_t)|] < \infty$) then $f(M_t)$ is a submartingale.
- Suppose M is a Mg and has increments bounded by c then

$$\mathbb{P}\left(\sup_{n \leq t} M_n \geq x\right) \leq \exp\left\{-\frac{x^2}{2tc}\right\}. \quad (\text{Azuma-Hoeffding})$$

- If Y_t is an adapted process (meaning Y_t is a function of F_t for all t) then there exists Z_t a previsible process (meaning Z_t is a function of F_{t-1} for all t) and Martingale M_t such that

$$Y_t = Y_0 + M_t + Z_t \quad (\text{Doob-Meyer Decomposition})$$

moreover this decomposition is unique, with probability 1. Moreover, if X_t is a sub-Mg then Z_t is increasing. Moreover, for any Mg M_t with $\mathbb{E}M_t^2 < \infty$

$$M_t^2 - \langle M \rangle_t \quad \text{is a Mg for } .$$

where $\langle M \rangle_t := \sum_{n=1}^t \mathbb{E}[(M_n - M_{n-1})^2 | F_{n-1}]$.

- If M'_t is a positive sub-martingale then, for $p > 1$ and $p^{-1} + q^{-1} = 1$

$$\left\| \sup_t M'_t \right\|_p \leq q \sup_t \|M'_t\|_p = q \|M'_\infty\|_p \quad (\text{Doob's } \mathcal{L}^p \text{ inequality})$$

- If M_t is a super-Mg with $\sup_t \mathbb{E}[|M_t|^p] < \infty$ then the limit :

$$M_t \rightarrow M_\infty, \quad \text{w.p. 1 and in } \mathcal{L}^p \quad (\text{Doob's } \mathcal{L}^p \text{ Mg Convergence Thrm})$$

- For M_t a Mg with $\mathbb{E}M_t^2 < \infty, \forall t$

$$\frac{M_t}{\langle M \rangle_t} \rightarrow 0 \quad \text{on the event } \{\langle M \rangle_\infty = \infty\} \quad (\text{Strong Law for Martingales})$$

$$M_t \rightarrow M_\infty \quad \text{on the event } \{\langle M \rangle_\infty < \infty\}$$

where $\langle M \rangle_t := \sum_{n=1}^t \mathbb{E}[(M_n - M_{n-1})^2 | F_{n-1}]$. In other words, for any L^2 Martingale

$$M_t = o(\langle M \rangle_t) + O(1).$$

- Assume M_t is a Mg with bounded increments¹

$$\frac{M_t}{\sqrt{\langle M \rangle_t}} \Rightarrow \mathcal{N}(0, 1) \quad (\text{Mg CLT})$$

- Assume M_t is a Mg with bounded increments

$$\left(\frac{M_{nt}}{\sqrt{\langle M \rangle_{nt}}} : t \in [0, 1] \right) \Rightarrow (B_t : t \in [0, 1]) \quad (\text{Mg CLT - v1})$$

where $(B_t : t \in (0, 1))$ is a standard Brownian motion. Or, let M_t^n be a Martingale for each n and suppose $(\langle M^n \rangle_t : t \in [0, 1]) \Rightarrow (t : t \in [0, 1])$ then

$$\left(\frac{M_t^n}{\sqrt{\langle M^n \rangle_t}} : t \in [0, 1] \right) \Rightarrow (B_t : t \in [0, 1]) \quad (\text{Mg CLT - v2})$$

Random variable T is a stopping time for F_t , $t \geq 0$ if $\mathbb{I}[T \leq t]$ is a function of F_t , i.e. knowing the values of (X_1, \dots, X_t) is sufficient to know if T has happened yet or not.

¹This condition can be weakened.

A.2 Stochastic Integration

- A heuristic look at the stochastic integral.
 - heuristic derivation of Itô's formula.
-

What follows is a heuristic proof of Itô's Formula. (Rigorous proofs of the exercises are not expected.)

Ex 161 (A Heuristic look at Stochastic Integration). For $(B_t : t \geq 0)$ a standard Brownian motion argue that, for all T and for δ sufficiently small and positive,

$$\sum_{t \in \{0, \delta, \dots, T\}} (B_{t+\delta} - B_t) = B_T \quad \text{and} \quad \sum_{t \in \{0, \delta, \dots, T\}} (B_{t+\delta} - B_t)^2 \approx T$$

Ans 161. The 1st sum is an interpolating sum. By independent increments property of Brownian motion, the 2nd sum adds IIDRVs with each with mean δ . Thus the strong law of large numbers gives the approximation.

Ex 162 (Continued). Discuss why it is reasonable to expect that

$$\sum_{t \in \{0, \delta, \dots, T\}} \sigma(X_t) (B_{t+\delta} - B_t) \approx \int_0^T \sigma(X_t) dB_t$$

and

$$\sum_{t \in \{0, \delta, \dots, T\}} \mu(X_t) (B_{t+\delta} - B_t)^2 \approx \int_0^T \mu(X_t) dt.$$

Ans 162. The first sum is approximation from a Riemann-Stieltjes integral, i.e.

$$\int_0^T f(t) dg(t) \approx \sum_{t \in \{0, \delta, \dots, T\}} f(t) (g(t + \delta) - g(t)).$$

So one might expect a integral limit. (This is unrigorous because Riemann-Stieltjes Integration only applies to functions with finite variation – while Brownian motion does not have finite variation.)

The second sum is a Riemann integral upon using the approximation $(B_{t+\delta} - B_t)^2 \approx \delta$ [161].

Ex 163 (Continued). If we inductively define X_t by the recursion

$$X_{t+\delta} - X_t = \sigma(X_t)(B_{t+\delta} - B_t) + \mu(X_t)\delta, \quad t = 0, \delta, 2\delta, \dots$$

then discuss why we expect X_t to approximately obey an equation of the form

$$X_t = X_0 + \int_0^t \sigma(X_t)dB_t + \int_0^t \mu(X_t)dt.$$

Ans 163. Sum to gain $X_T - X_0$ and apply approximations from [162].

Ex 164 (Continued). Let f be a twice differentiable function, argue that

$$f(X_{t+\delta}) - f(X_t) \approx \left[f'(X_t)\mu(X_t) + \frac{\sigma(X_t)^2}{2} f''(X_t) \right] \delta + f'(X_t)\sigma(X_t)(B_{t+\delta} - B_t).$$

Ans 164. Apply a Taylor approximation

$$\begin{aligned} & f(X_{t+\delta}) - f(X_t) \\ &= f(X_t + \sigma(X_t)(B_{t+\delta} - B_t) + \mu(X_t)\delta) - f(X_t) \\ &= f'(X_t) \{ \mu\delta + \sigma \cdot (B_{t+\delta} - B_t) \} + \frac{f''(X_t)}{2} \{ \mu\delta + \sigma \cdot (B_{t+\delta} - B_t) \}^2 + o(\delta) \\ &= f'(X_t) \{ \mu\delta + \sigma \cdot (B_{t+\delta} - B_t) \} + \frac{f''(X_t)}{2} \sigma^2 \cdot (B_{t+\delta} - B_t)^2 + o(\delta) \end{aligned}$$

In the last equality we use that $(B_{t+\delta} - B_t) = o(\delta^{1/2})$, cf. [161].

Ex 165 (Continued). Argue that

$$f(X_T) - f(X_0) = \int_0^T \left[f'(X_t)\mu(X_t) + \frac{\sigma(X_t)^2}{2} f''(X_t) \right] dt + \int_0^T f'(X_t)\sigma(X_t)dB_t.$$

This is Itô's formula.

Ans 165. Apply an interpolating sum to [164] and then apply [162].

A.3 Gronwall's Lemma

We introduce a useful integration inequality due to Bellman.

Thrm 166 (Gronwall's Lemma). *If $f : \mathbb{R}_+ \rightarrow \mathbb{R}_+$ is bounded above on each closed interval $[0, T]$ and satisfies*

$$f(T) \leq a(T) + \int_0^T b(t)f(t)dt \quad (\text{A.1})$$

for increasing function $a(t)$ and positive (integrable) function $b(t)$ then

$$f(T) \leq a(T) \exp \left\{ \int_0^T b(t)dt \right\}$$

- The most common choices of a and b are constants.

Proof. Consider the function

$$v(t) = e^{-\int_0^t b(s)ds} \int_0^t b(s)f(s)ds,$$

differentiating and applying (A.1) gives

$$\begin{aligned} \frac{dv(t)}{dt} &= -b(t)e^{-\int_0^t b(s)ds} \int_0^t b(s)f(s)ds + b(t)f(t)e^{-\int_0^t b(s)ds} \\ &\leq -b(t)e^{-\int_0^t b(s)ds} \int_0^t b(s)f(s)ds + a(t)b(t)e^{-\int_0^t b(s)ds} \\ &\quad + b(t)e^{-\int_0^t b(s)ds} \int_0^t b(s)f(s)ds \\ &= a(t)b(t)e^{-\int_0^t b(s)ds}. \end{aligned}$$

Integrating and recalling that $a(t)$ is increasing gives

$$\begin{aligned} e^{-\int_0^T b(t)dt} \int_0^T b(t)f(t)dt = v(T) &\leq \int_0^T a(t)b(t)e^{-\int_0^t b(s)ds} dt \\ &\leq a(T) \int_0^T b(t)e^{-\int_0^t b(s)ds} dt \\ &= a(T) \left[1 - e^{-\int_0^T b(s)ds} \right] \end{aligned}$$

Thus, applying (A.1) and the above bound

$$\begin{aligned} f(T) \leq a(T) + \int_0^T b(t)f(t)dt &\leq a(T) + e^{\int_0^T b(t)dt} a(T) \left[1 - e^{-\int_0^T b(s)ds} \right] \\ &= a(T) \exp \left\{ \int_0^T b(t)dt \right\}. \end{aligned}$$

□

A.4 Projections

we consider the *projection* of $x \in \mathbb{R}^p$ on to closed convex set \mathcal{X} :

$$P_{\mathcal{X}}(x) = \operatorname{argmin}_{y \in \mathcal{X}} \|x - y\|^2$$

Here $\|\cdot\|$ is a norm with a dot product. The key observation is that making a projection cannot increase distances

Lemma 25.

$$\|P_{\mathcal{X}}(x) - P_{\mathcal{X}}(y)\| \leq \|x - y\|$$

Proof. For all $x' \in \mathcal{X}$, we must have $(x' - P_{\mathcal{X}}(x)) \cdot (x - P_{\mathcal{X}}(x)) \leq 0$, i.e. the plane $\mathcal{H} = \{z : (z - P_{\mathcal{X}}(x)) \cdot (x - P_{\mathcal{X}}(x)) = 0\}$ separates x from \mathcal{X} . If this were not true, then we would have a contradiction, in particular, there would be a point on the line joining x' and $P_{\mathcal{X}}(x)$ that is closer to x . We thus have

$$(P_{\mathcal{X}}(y) - P_{\mathcal{X}}(x)) \cdot (x - P_{\mathcal{X}}(x)) \leq 0 \quad \text{and} \quad (P_{\mathcal{X}}(x) - P_{\mathcal{X}}(y)) \cdot (y - P_{\mathcal{X}}(y)) \leq 0.$$

Adding together and then applying Cauchy-Schwartz implies

$$\|P_{\mathcal{X}}(y) - P_{\mathcal{X}}(x)\|^2 \leq (y - x) \cdot (P_{\mathcal{X}}(y) - P_{\mathcal{X}}(x)) \leq \|y - x\| \|P_{\mathcal{X}}(y) - P_{\mathcal{X}}(x)\|,$$

as required. \square

A.5 Misc Results

Lemma 26. *If we let $\alpha_t = t^{-\gamma}$, for $\gamma \in [0, 1]$, we have*

$$\frac{t^{1-\gamma} - 1}{1 - \gamma} \leq \sum_{s=1}^t s^{-\gamma} \leq 1 + \frac{t^{1-\gamma} - 1}{1 - \gamma}.$$

Proof. This can be obtained by simple calculations. Since $s^{-\gamma}$ is decreasing for $\gamma \in [0, 1]$, then

$$\sum_{s=1}^t s^{-\gamma} \leq 1 + \int_1^t s^{-\gamma} ds = 1 + \frac{t^{1-\gamma} - 1}{1 - \gamma},$$

and

$$\sum_{s=1}^t s^{-\gamma} \geq \int_1^t s^{-\gamma} ds = \frac{t^{1-\gamma} - 1}{1 - \gamma}.$$

\square

A.6 Utility Theory

- Utility functions; equivalence of utility functions
 - Relative risk aversion; CRRA Utility and iso-elasticity.
-

A utility function $U(x)$ is used to quantify the value that you gain from an outcome x .

Def 167 (Utility Function). For $\mathcal{X} \subset \mathbb{R}^d$, a utility function is a function $U : \mathcal{X} \rightarrow \mathbb{R}$ that is increasing, i.e. if $x \leq y$ component-wise then $U(x) \leq U(y)$. The utility of a random variable X is then its expected utility, $\mathbb{E}U(X)$. A utility function creates an ordering where an outcome X is preferred to Y if $\mathbb{E}U(X) \geq \mathbb{E}U(Y)$.

Jensen's inequality applies to a concave utility:

$$\mathbb{E}U(X) \leq U(\mathbb{E}X)$$

So we prefer a certain outcome $\mathbb{E}X$ rather than the risky outcome X that has the same mean – This is being risk averse.

Def 168 (Risk Aversion). If the function is concave then we also say that the function is risk averse. (Unless stated otherwise we assume that the utility function is risk averse).

Def 169. We say that two utility functions U and V are equivalent if they induce the same ordering. I.e. $\mathbb{E}U(X) \leq \mathbb{E}U(Y)$ iff $\mathbb{E}V(X) \leq \mathbb{E}V(Y)$.

Ex 170. Show that two utility functions are equivalent iff V the same as U up-to an affine transform, i.e.

$$V(x) = aU(x) + b$$

for constants $a > 0$ and b .

Ans 170. Define $\phi : \mathbb{R} \rightarrow \mathbb{R}$ s.t. $\phi(\mathbb{E}U(X)) = \mathbb{E}V(X)$. Let $X = x$ w.p. and $X = y$ w.p. $q = 1 - p$. Then

$$\phi(pU(x) + qU(y)) = pV(x) + qV(y) = p\phi(U(x)) + (1 - p)\phi(U(y)).$$

This implies ϕ is linear.

Def 171 (Coefficient of Relative Risk Aversion). For a utility function $U : \mathbb{R} \rightarrow \mathbb{R}$ (twice differentiable) the Coefficient of Relative Risk Aversion is

$$-x \frac{U''(x)}{U'(x)}.$$

Ex 172. You have utility function U . You are offered a bet that increases your wealth w multiplicatively by $(1 + X)$ here X is a “small” positive RV. Discuss why you would accept the bet iff

$$\frac{2\mathbb{E}X}{\mathbb{E}X^2} \geq -x \frac{U''(x)}{U'(x)}$$

I.e. You accept the bet if your mean is large but a large variance makes this less likely, and the coefficient of relative risk aversion decides the threshold.

Ans 172. Accept if

$$0 \leq \mathbb{E}[U(w(1 + X)) - U(w)] \stackrel{\text{Taylor}}{\approx} \mathbb{E}\left[U'(w)wX + \frac{1}{2}U''(w)w^2X^2\right].$$

Def 173 (CRRA Utility/Iso-elastic Utility). A Constant Relative Risk Averse utility (CRRA) takes the form

$$U(x) = \begin{cases} \frac{x^{1-R}}{1-R}, & R \neq 1, \\ \log x, & R = 1. \end{cases}$$

Def 174. A utility function is Iso-elastic if it is unchanged under multiplication: for all $c > 0$,

$$\mathbb{E}U(X) \geq \mathbb{E}U(Y) \quad \text{iff} \quad \mathbb{E}U(cX) \geq \mathbb{E}U(cY).$$

I.e. the utility only cares about the relative magnitude of the risk.

Ex 175. Show that a utility function is iso-elastic iff it is a CRRA utility (up-to an affine transform).

Ans 175. By [170], it is immediate that CRRA implies iso-elastic. Further by [170], $\forall c, U(cx) = a_c U(x) + b_c$ for constants a_c and b_c . Differentiate twice w.r.t. x and divide gives

$$\frac{cU''(cx)}{U'(cx)} = \frac{U''(x)}{U'(x)}$$

Set $x = 1$ and integrate twice w.r.t. c gives the required result.

A.7 Lagrangian Optimization and Duality

We are interested in solving the *constrained optimization problem*

$$\text{minimize } f(x) \quad \text{subject to } h(x) = b \quad \text{over } x \in \mathcal{X}. \quad (\text{P})$$

The *variables* x must belong to the *constraint region* \mathcal{X} , where \mathcal{X} is a subset of \mathbb{R}^p . The variables must satisfy the *functional constraints* $h(x) = b$, where $h : \mathcal{X} \rightarrow \mathbb{R}^d$ and $b \in \mathbb{R}^d$. Variables that belong to the constraint region and satisfy the functional constraints are called *feasible* and belong to the *feasible set*

$$\mathcal{X}(b) = \{x \in \mathcal{X} : h(x) = b\}.$$

The variables are judged according to the relative magnitude of the *objective function* $f : \mathcal{X} \rightarrow \mathbb{R}$. A feasible solution to the optimization, x^* , is called *optimal* when $f(x^*) \leq f(x)$ for all $x \in \mathcal{X}(b)$. In general, an optimum x^* need not exist. Inequality constraints $h_j(x) \leq b_j$ can be rewritten as equality constraints $h_j(x) + z_j = b_j$, where $z_j \geq 0$. Then an optimization of the form (P) can be solved over variables $(x, z) \in \mathcal{X} \times \mathbb{R}_+^{p'}$. The variables z are called *slack variables*. $\max_{x \in \mathcal{X}(b)} f(x) = -\min_{x \in \mathcal{X}(b)} -f(x)$, so maximizing is the same as minimizing.

The set \mathcal{X} could incorporate the functional constraints $\{x : h(x) = b\}$, but generally we assume \mathcal{X} is a set where the unconstrained optimization, $\min_{x \in \mathcal{X}} f(x)$, is easier to solve. For example, \mathcal{X} might be \mathbb{R}_+^p and $\min_{x \in \mathcal{X}} f(x)$ might be solved by differentiating and finding local minima. What makes the problem difficult is the functional constraints. To get some solution, we might penalize going over these constraints. If so, we could remove these constraints and solve a different optimization

$$\text{minimize } f(x) + \sum_{j=1}^d \lambda_j (b_j - h_j(x)) \quad \text{over } x \in \mathcal{X}. \quad (L_\lambda)$$

In the above optimization, the objective function

$$L(x; \lambda) = f(x) + \lambda^\top (b - h(x))$$

is called the *Lagrangian* of the optimization problem (P). We call the components of the vector, $\lambda \in \mathbb{R}^d$, *Lagrange multipliers*. We use $x^*(\lambda)$ to notate its optimum of (L_λ) , if it exists. For each inequality constraint $h_j(x) \leq b_j$, a variable $z_j \geq 0$ is added and the term $\lambda_j (b_j -$

$h_j(x)$ is replaced by $\lambda_j(b_j - z_j - h_j(x))$. Because we introduced a new term $-\lambda_j z_j$, for a finite solution to (L_λ) , we require $\lambda_j \leq 0$ and thus, after minimizing over z_j , we have $z_j^* \lambda_j = 0$. This equality is called *complementary slackness*. If (x^*, z^*) our optimal Lagrangian solution is also feasible then complementary slackness states $\lambda_j(b_j - h_j(x^*)) = 0$.

The unconstrained optimization (L_λ) is, in principle, much easier to solve and a good choice of λ can be used to penalize constraints that are not satisfied. For example, if we solve the optimization problem (L_λ) with optimum $x^*(\lambda)$ and found that $b_i > h(x_i)$ then, intuitively, we would make λ_i bigger in order to make the price of overrunning constraint i higher. Hopefully, this would then make the solution of (L_λ) closer to (P). The minimum of the Lagrangian is always smaller.

Lemma 27 (Weak Duality). *For all $\lambda \in \mathbb{R}^d$*

$$\min_{x \in \mathcal{X}} L(x; \lambda) \leq \min_{x \in \mathcal{X}(b)} f(x) \tag{A.2}$$

Proof.

$$\min_{x \in \mathcal{X}} L(x; \lambda) \leq \min_{x \in \mathcal{X}(b)} L(x; \lambda) = \min_{x \in \mathcal{X}(b)} \left[f(x) + \lambda^\top \underbrace{(b - h(x))}_{=0} \right] = \min_{x \in \mathcal{X}(b)} f(x).$$

□

In fact, if we can make the two solutions equal then we have optimality.

Theorem 28 (Lagrangian Sufficiency Theorem). *If there exists Lagrange multipliers λ^* and a solution $x^*(\lambda^*)$ to (L_{λ^*}) such that $x^*(\lambda^*)$ is feasible for (P) then $x^*(\lambda^*)$ solves the optimization problem (P).*

Proof. As $x^*(\lambda^*)$ is feasible then certainly $f(x^*(\lambda^*)) \geq \min_{x \in \mathcal{X}(b)} f(x)$. Now,

$$\begin{aligned} f(x^*(\lambda^*)) &= f(x^*(\lambda^*)) + \lambda^{*\top} (b - h(x^*(\lambda^*))) && \text{(as } x^*(\lambda^*) \text{ is feasible)} \\ &= \min_{x \in \mathcal{X}} L(x; \lambda^*) && \text{(as } x^* \text{ is optimal for } (L_{\lambda^*})) \\ &\leq \min_{x \in \mathcal{X}(b)} f(x). && \text{(by (A.2))} \end{aligned}$$

So, $f(x^*(\lambda^*)) = \min_{x \in \mathcal{X}(b)} f(x)$.

□

This result gives a simple procedure to solve an optimization:

1. Take your optimization (P) and, if necessary, add slack variables to make all inequality constraints equalities.
2. Write out the Lagrangian and solve optimization (L_λ) for $x^*(\lambda)$.
3. By solving the constraints $h(x^*(\lambda)) = b$ over λ , find a λ^* so that $x^* = x^*(\lambda^*)$ is feasible.
By Lagrangian Sufficiency Theorem, x^* is optimal.

Duality

Since weak duality holds, we want λ to make the minimized Lagrangian as big as possible. Only then can a feasible Lagrangian optimum be found to solve the optimization (P). Thus we consider the following optimization,

$$\text{maximize } g(\lambda) \text{ over } \lambda \in \Lambda. \quad (\text{D})$$

Here $g(\lambda) = \inf_{x \in \mathcal{X}} L(x; \lambda)$ and $\Lambda = \{\lambda : \inf_{x \in \mathcal{X}} L(x; \lambda) > -\infty\}$.

We call this optimization problem (D) the *dual* optimization and, in turn, we call (P) the *primal* optimization. Making dependence on b explicit, use $\rho(b)$ to denote the optimal value of the primal problem and $\delta(b)$ to denote the optimal value of the dual. By weak duality $\delta(b) \leq \rho(b)$. If $\delta(b) = \rho(b)$ then we say *strong duality* holds. Observe, $\max_\lambda L(x; \lambda) = f(x)$, if $h(x) = b$ and $\max_\lambda L(x; \lambda) = \infty$, otherwise. So, $\rho(b) = \min_{x \in \mathcal{X}} \max_\lambda L(x; \lambda)$ and by definition $\delta(b) = \max_\lambda \min_{x \in \mathcal{X}} L(x; \lambda)$. So strong duality corresponds exactly to the *saddle point condition* that

$$\min_{x \in \mathcal{X}} \max_\lambda L(x; \lambda) = \max_\lambda \min_{x \in \mathcal{X}} L(x; \lambda) \quad (\text{A.3})$$

Essentially, the Lagrangian approach will only be effective when we can find a λ^* such that $\rho(b) = \inf_{x \in \mathcal{X}} L(x; \lambda^*)$. This is equivalent to the existence of a supporting hyperplane for $\rho(b)$.

Theorem 29. $\rho(b) = \inf_{x \in \mathcal{X}} L(x; \lambda^*)$ iff λ^* and ρ , together satisfy the property

$$\rho(c) \geq \rho(b) + \lambda^{*\top}(c - b), \quad \text{for all } c \in \mathbb{R}^d. \quad (\text{A.4})$$

Proof.

$$\begin{aligned} \inf_{x \in \mathcal{X}} L(x, \lambda) &= \inf_{x \in \mathcal{X}} \{f(x) + \lambda^\top(b - h(x))\} \\ &= \inf_{c \in \mathbb{R}^d} \inf_{x \in \mathcal{X}(c)} \{f(x) + \lambda^\top(b - h(x))\} = \inf_{c \in \mathbb{R}^d} \{\rho(c) + \lambda^\top(b - c)\}. \end{aligned}$$

So $\rho(b) = \inf_{x \in \mathcal{X}} L(x; \lambda^*)$ iff $\rho(b) = \inf_{c \in \mathbb{R}^d} \{\rho(c) + \lambda^{*\top}(b - c)\}$ iff $\rho(b) \leq \rho(c) + \lambda^{*\top}(b - c)$ for all c , as required. \square

Informally, (A.4) states that there exists a tangent to ρ at point b and that the function ρ lies above this tangent. The set $H = \{(c, \rho(b) + \lambda^{*\top}(c - b)) \in \mathbb{R}^{p+1} : c \in \mathbb{R}^d\}$ defines a *hyperplane* containing $(b, \rho(b))$ and $\{(c, \rho(c)) : c \in \mathbb{R}^d\}$ lies in the half-space above H . So we call this “tangent” H a *supporting hyperplane* to $\rho(b)$ at b with slope λ^* . Any function $\rho : \mathbb{R}^d \rightarrow (-\infty, \infty]$ is convex if $q\rho(b) + (1 - q)\rho(c) \geq \rho(qb + (1 - q)c)$, for $q \in [0, 1]$. It can be verified that this is equivalent to the condition (A.4). Optimization of convex functions may be possible with Lagrangian methods.

Given (A.4), for $h > 0$, we can say that

$$\frac{\rho(b + he_i) - \rho(b)}{h} \geq \lambda_i^*, \quad \frac{\rho(b - he_i) - \rho(b)}{h} \leq \lambda_i^*.$$

Thus, if ρ is differentiable at b then

$$\frac{\partial \rho}{\partial b_i}(b) = \lambda_i^*. \tag{A.5}$$

For a maximization problem, if we interpreted $\rho(b)$ as the optimal profit (or cost for a minimization problem) made when manufacturing using b units of raw material then (A.5) can be interpreted as the change in profit made from gaining a small additional quantity of material i and thus λ_i^* is interpreted as the price the manufacturer is prepared to pay to secure those extra goods. From this interpretation, we call the optimal Lagrange multiplier λ_i^* a *shadow price*.

As noted above if $\rho(b)$ is convex then we can find a λ^* with which we can apply the Lagrangian approach. For a constrained optimization problem we say *Slater’s Condition* is satisfied if the objective function f is a convex function, if constraint region \mathcal{X} is a convex set, if for each equality constraint $h_j(x) = b_j$ the function h_j is linear, if there exist a feasible solution $x \in \mathcal{X}(b)$ such that all inequality constraint are satisfied with strict inequality $h_j(x) < b_j$. To make life easier we assume the functions f and h_j are continuous at the feasible x assumed in Slater’s condition and we assume there exists some non-optimal feasible solution in $\mathcal{X}(b)$.

Theorem 30. *If Slater’s Condition holds then $\rho(b)$ is a convex function and thus there exists λ^* such that $\rho(b) = \inf_{x \in \mathcal{X}} L(x; \lambda^*)$.*

The proof of Strong Duality following from Slater's condition is not too hard, it follows from the supporting hyperplane theorem and a few inequalities. The proof is secondary to our immediate goals so we end our discussion at this point.

A.8 Linear Algebra

We review without proof various algebraic facts about vectors and matrices.

Linear Dependence

Def 176 (Linear Dependent). *Vectors $v_1, \dots, v_r \in \mathbb{R}^n$ are linearly independent if*

$$\sum_{i=1}^r a_i v_i = 0 \implies a_i = 0 \forall i$$

otherwise the vectors are said to be linearly dependent.

Def 177 (Rank). *If $A \in \mathbb{R}^{m \times n}$ then the number of (row) vectors in A that are linearly independent is the rank of A , and we write $\text{rank}(A)$.*

Lemma 28. *$\text{rank}(A) = \text{rank}(A^T)$ i.e. the rank of the rows equals the rank of the columns.*

Eigenvalues and Eigenvectors

Def 178 (Eigenvalue and Eigenvector). *A vector $v \in \mathbb{R}^n$ and a scalar $\lambda \in \mathbb{R}$ are, respectively, an eigenvector and an eigenvalue of a matrix $A \in \mathbb{R}^{n \times n}$ if*

$$Av = \lambda v.$$

Lemma 29. *If the eigenvectors of the matrix A are linearly dependent then*

$$A = V\Lambda V^{-1}$$

where V is the matrix whose columns are given by the eigenvectors of A and Λ is the diagonal matrix whose diagonal entries are the eigenvalues of A .

The converse of the above results holds too. Further if linear independence does not hold, we can decompose but Λ is no longer diagonal: it is in Jordan normal form.

Positive Definite Matrices

We are interested in matrices A where $x^T A x$ behaves like a distance.

Def 179 (Symmetric). A matrix $A \in \mathbb{R}^{n \times n}$ is symmetric if $A^T = A$.

Def 180 (Positive Definite). A matrix is positive definite if

$$v^T A v > 0, \quad \forall v \in \mathbb{R}^n \setminus \{0\}$$

and the matrix is positive semi-definite if $v^T A v \geq 0 \forall v \in \mathbb{R}^n \setminus \{0\}$.

Def 181 (Orthogonal). A matrix $A \in \mathbb{R}^{n \times n}$ is orthogonal if

$$A^T A = I_n$$

(Here, and here after, I_n is the n -dimensional identity matrix.)

Theorem 31 (The Spectral Theorem). If A is a positive definite matrix then A has positive eigenvalues

$$0 < \lambda_1(A) \leq \dots \leq \lambda_n(A)$$

and V the matrix of eigenvectors is orthogonal.

The same result applies to positive semi-definite matrices; however the equality $0 = \lambda_1(A)$. Above, and hereafter, we will let $\lambda_1(A) \geq \dots \geq \lambda_n(A)$ be the ordered set of eigenvalues.

Corollary 3. If A is positive definite then

$$A = \sum_{k=1}^n \lambda_k v_k^T v_k$$

(where λ_k and v_k are the k th eigenvalue-eigenvector.)

Lemma 30. If A is positive definite then

$$\langle x, y \rangle_A = x^T A y$$

defines an inner product and $\|x\| = \langle x, x \rangle_A^{1/2}$.

Def 182 (Operator norm). For any norm $\|\cdot\|$ on \mathbb{R}^n the operator norm of matrix $A \in \mathbb{R}^{n \times n}$ is

$$\|A\| := \max_{x: \|x\|=1} \frac{\|Ax\|}{\|x\|} = \max_{x: \|x\|>0} \frac{\|Ax\|}{\|x\|}$$

Proposition 9. If A is a positive semi-definite matrix then

$$\|A\| = \max_{x: \|x\|=1} \frac{\|Ax\|}{\|x\|} = \lambda_{\max}(A) \quad \text{and} \quad \min_{x: \|x\|=1} \frac{\|Ax\|}{\|x\|} = \lambda_{\min}(A)$$

(where $\lambda_{\min}(A)$ and $\lambda_{\max}(A)$ are the minimum and maximum eigenvalues of A .)

By restricting the maximization and minimization above to a smaller dimension we can attain the other eigenvalues λ_k . This is called the Courant-Fischer Minimax Theorem.

The following is a consequence of the above proposition

Corollary 4. • For two positive semi-definite matrices

$$\begin{aligned} \lambda_{\max}(A+B) &\leq \lambda_{\max}(A) + \lambda_{\max}(B) \\ \lambda_{\min}(A+B) &\geq \lambda_{\min}(A) + \lambda_{\min}(B) \end{aligned}$$

- $A \mapsto \lambda_{\max}(A)$ is convex and $A \mapsto \lambda_{\min}(A)$ is concave.

The above can be generalized to intermediate eigenvalues. These are Cauchy's interlacing inequalities. Other related inequalities are Weyl inequalities and Ky Fan inequalities.

The following is a well-known eigenvalue bound.

Lemma 31. Let A, B be symmetric positive definite matrices of size $d \times d$. Then,

$$\lambda_{\min}(A) \geq \lambda_{\min}(B) - \|A - B\|.$$

Proof. Recall that

$$\lambda_{\min}(A) = \min_{x: \|x\|=1} x^T A x.$$

We can write

$$x^T A x = x^T (A - B) x + x^T B x.$$

For the first term, by Cauchy-Schwartz inequality we have

$$-x^T (A - B) x \leq |\langle x, (A - B)x \rangle| \leq \|x\| \|(A - B)x\| \leq \|A - B\|.$$

For the second term, we know

$$\mathbf{x}^\top B \mathbf{x} \geq \lambda_{\min}(B).$$

Then we have

$$\mathbf{x}^\top A \mathbf{x} = \mathbf{x}^\top (A - B) \mathbf{x} + \mathbf{x}^\top B \mathbf{x} \geq -\|A - B\| + \lambda_{\min}(B).$$

Therefore,

$$\lambda_{\min}(A) \geq \lambda_{\min}(B) - \|A - B\|.$$

We obtain the result. \square

The Trace of a Matrix

Def 183 (Trace). *The trace of $A \in \mathbb{R}^{m \times n}$ is the sum of its diagonal entries*

$$\text{Tr}(A) = \sum_i A_{ii}$$

Here is a collection of facts about the trace of a matrix.

Proposition 10.

$\text{Tr}(cA + dB) = c\text{Tr}(A) + d\text{Tr}(B)$	(Linear)
$\text{Tr}(AB) = \text{Tr}(BA) \neq \text{Tr}(A)\text{Tr}(B)$	(Commutative)
$\text{Tr}(A) = \text{Tr}(A^\top)$	(Symmetric)
$\text{Tr}(V^{-1}AV) = \text{Tr}(A)$	(Invariant)
$\text{Tr}(P_X) = \text{rank}(X), \quad \text{for } P = X(X^\top X)^{-1}X^\top$	(Projection)
$\text{Tr}(A) = \sum_i \lambda_i(A)$	(Eigenvalues)

There is a generalization for the eigenvalue sum result. Here in this extremal trace result you sum the first (or last) k eigenvalues.

The Determinant of a Matrix

The determinant of a matrix gives its volume:

Def 184 (Determinant). *The determinant of a matrix $A = [\mathbf{a}_1, \dots, \mathbf{a}_n]$ with column vectors given by $\mathbf{a}_i \in \mathbb{R}^{1 \times n}$ gives the volume of the polytope defined by the column vectors of A .² We use $\det(A)$ to denote its determinant. The determinant $\det(A)$ is the unique function such that*

1. $\det(\alpha \mathbf{a} + \beta \mathbf{b}, \mathbf{a}_2, \dots, \mathbf{a}_n) = \alpha \det(\mathbf{a}, \mathbf{a}_2, \dots, \mathbf{a}_n) + \beta \det(\mathbf{b}, \mathbf{a}_2, \dots, \mathbf{a}_n)$, for $\alpha, \beta \in \mathbb{R}$.
2. $\det(\mathbf{a}_1, \dots, \mathbf{a}_i, \dots, \mathbf{a}_j, \dots, \mathbf{a}_n) = -\det(\mathbf{a}_1, \dots, \mathbf{a}_j, \dots, \mathbf{a}_i, \dots, \mathbf{a}_n) \forall i, j$.
3. $\det(I_n) = 1$.

Here are a few equalities for the determinant

Proposition 11.

- $\det(AB) = \det(A) \det(B)$
- $\det(A^\top) = \det(A)$
- $\det(A^{-1}) = \det(A)^{-1}$
- $\det(I_n + AB) = \det(I_m + BA)$ (Sylvester's Determinant Theorem)
- $\det(A) = \prod_{i=1}^n \lambda_i(A)$
- $\det(A + \mathbf{u}^\top \mathbf{v}) = (1 + \mathbf{v}^\top A^{-1} \mathbf{u}) \det(A)$ (Matrix Determinant Lemma)

The matrix Determinant lemma works well with the Sherman-Morrison formula. Here are a few inequalities for the determinant.

Proposition 12. *For any matrix A*

- $\det(A) \leq \left(\frac{\text{Tr}(A)}{n}\right)^n$
- $\text{Tr}(I + A^{-1}) \leq \det(A) \leq \text{Tr}(I + A)$
- $\det(A) \leq \prod_{i=1}^n \|\mathbf{a}_i\|$ (Hadamard)

For any positive semi-definite matrices A and B

- $\det(A) \leq \prod_{i=1}^n A_{ii}$
- $\prod_{i=1}^n (\lambda_i(A) + \lambda_i(B)) \leq \det(A + B) \leq \prod_{i=1}^n (\lambda_i(A) + \lambda_{n+1-i}(B))$

²Though the sign of this volume ± 1 depends on the orientation of the polytope.

Singular Values

Singular values are like eigenvalues except we do not need to assume our matrix is square.

Def 185 (Singular Value). For $A \in \mathbb{R}^{m \times n}$, $s > 0$ is singular value and $u \in \mathbb{R}^m$ and $v \in \mathbb{R}^n$ are singular vectors if

$$Av = su, \quad A^\top u = sv$$

The following is immediate

Lemma 32. v and s^2 are eigenvector-eigenvalues for $A^\top A$ and u and s^2 are eigenvector-eigenvalues for AA^\top . If A is symmetric then v is an eigenvector and $s = |\lambda|$ for its eigenvalue. [Nb. because AA squares the eigenvalues.]

Theorem 32 (Singular Value Decomposition). The number of singular values $(s_1, u_1, v_1), \dots, (s_r, u_r, v_r)$ is such that $r = \text{rank}(A)$ and

$$A = \sum_{i=1}^r s_i u_i v_i^\top,$$

moreover the matrices $U \in \mathbb{R}^{m \times r}$ or $V^{n \times r}$ or singular vectors orthogonal.

We can now construct something similar to the eigenvalue decomposition.

A.9 Random Matrices

Here we recall a few results on Random Matrices so that we can apply them to covariance matrix estimation. The text of Vershynin provides an excellent source on this broad area.

The following result gives a concentration bound on covariance matrices. The proof can be found in Vershynin [53].

Lemma 33. *If $\mathbf{x}_s \in \mathbb{R}^d$ are bounded such that, for all $s \geq 1$*

$$\mathbb{E}[\mathbf{x}_s \mid \mathcal{F}_{s-1}] = \mathbf{0} \quad \text{and} \quad \mathbb{E}[\mathbf{x}_s \mathbf{x}_s^\top \mid \mathcal{F}_{s-1}] = \Sigma_s^x.$$

We assume that there exists $x_{\max} \in \mathbb{R}^+$, such that bound $\|\mathbf{x}_s\|_\infty \leq x_{\max}$ with probability 1 and Σ_s^x is positive definite. Then,

$$\mathbb{P}\left(\left\|\sum_{s=1}^t (\mathbf{x}_s \mathbf{x}_s^\top - \Sigma_s^x)\right\|_{op} \geq \varepsilon\right) \leq 2 \cdot 9^{2d} \exp\left\{-\frac{(\varepsilon/2)^2}{2 \sum_{s=1}^t (x_{\max}^2 + \|\Sigma_s^x\|_{op})^2}\right\}.$$

Proof. We show this argument in two steps. We first control $\sum_{t=1}^T \mathbf{x}_t \mathbf{x}_t^\top$ over a ε -net, and then extend the bound to the full supremum norm by a continuity argument.

Using Lemma 35 (stated below) and choosing $\varepsilon = \frac{1}{4}$ and, we can find an ε -net \mathcal{N} of the unit sphere S^{d-1} with cardinality

$$|\mathcal{N}| \leq 9^d.$$

By Lemma 36 (stated below), the operator norm of $\mathbf{x}_s \mathbf{x}_s^\top$ can be bounded on \mathcal{N} , that is

$$\begin{aligned} \left\|\sum_{s=1}^t (\mathbf{x}_s \mathbf{x}_s^\top - \Sigma_s^x)\right\|_{op} &\leq 2 \max_{\mathbf{v}, \mathbf{w} \in \mathcal{N}} \left\langle \left(\sum_{s=1}^t (\mathbf{x}_s \mathbf{x}_s^\top - \mathbb{E}[\mathbf{x}_s \mathbf{x}_s^\top])\right) \mathbf{v}, \mathbf{w} \right\rangle \\ &\leq 2 \max_{\mathbf{v}, \mathbf{w} \in \mathcal{N}} \left| \mathbf{v}^\top \left(\sum_{s=1}^t (\mathbf{x}_s \mathbf{x}_s^\top - \mathbb{E}[\mathbf{x}_s \mathbf{x}_s^\top])\right) \mathbf{w} \right|. \end{aligned} \quad (\text{A.6})$$

We first fix $\mathbf{v}, \mathbf{w} \in \mathcal{N}$, and by Azuma–Hoeffding inequality³ for any $\varepsilon > 0$ we can state that

$$\mathbb{P}\left(\left| \mathbf{v}^\top \left(\sum_{s=1}^t (\mathbf{x}_s \mathbf{x}_s^\top - \mathbb{E}[\mathbf{x}_s \mathbf{x}_s^\top \mid \mathcal{F}_{s-1}])\right) \mathbf{w} \right| \geq \frac{\varepsilon}{2}\right) \leq 2 \exp\left\{-\frac{(\varepsilon/2)^2}{2 \sum_{s=1}^t (x_{\max}^2 + \|\Sigma_s^x\|_{op})^2}\right\}.$$

³We note that Vershynin applies a Hoeffding bound. This is the only substantive difference in the proof here.

Given that $\|\mathbf{x}_s\|_\infty \leq x_{\max}$, we have $\|\mathbf{x}_s \mathbf{x}_s^\top - \mathbb{E}[\mathbf{x}_s \mathbf{x}_s^\top \mid \mathcal{F}_{s-1}]\| \leq x_{\max}^2 + \|\Sigma_s^x\|_{op}$.

Next, we unfix $\mathbf{v}, \mathbf{w} \in \mathcal{N}$ using a union bound. Since that \mathcal{N} has cardinality bounded by 9^d , we obtain

$$\begin{aligned} & \mathbb{P}\left(\max_{\mathbf{v}, \mathbf{w} \in \mathcal{N}} \left| \mathbf{v}^\top \left(\sum_{s=1}^t (\mathbf{x}_s \mathbf{x}_s^\top - \mathbb{E}[\mathbf{x}_s \mathbf{x}_s^\top \mid \mathcal{F}_{s-1}]) \right) \mathbf{w} \right| \geq \frac{\varepsilon}{2} \right) \\ & \leq \sum_{\mathbf{v}, \mathbf{w} \in \mathcal{N}} \mathbb{P}\left(\left| \mathbf{v}^\top \left(\sum_{s=1}^t (\mathbf{x}_s \mathbf{x}_s^\top - \mathbb{E}[\mathbf{x}_s \mathbf{x}_s^\top \mid \mathcal{F}_{s-1}]) \right) \mathbf{w} \right| \geq \frac{\varepsilon}{2} \right) \\ & \leq |\mathcal{N}|^2 \cdot 2 \exp\left\{-\frac{(\varepsilon/2)^2}{2 \sum_{s=1}^t (x_{\max}^2 + \|\Sigma_s^x\|_{op})^2}\right\} \\ & \leq 9^{2d} \cdot 2 \exp\left\{-\frac{\varepsilon^2}{8 \sum_{s=1}^t (x_{\max}^2 + \|\Sigma_s^x\|_{op})^2}\right\}. \end{aligned}$$

Together with (A.6), we have

$$\begin{aligned} \mathbb{P}\left(\left\| \sum_{s=1}^t (\mathbf{x}_s \mathbf{x}_s^\top - \Sigma_s^x) \right\|_{op} \geq \varepsilon\right) &= \mathbb{P}\left(2 \max_{\mathbf{v}, \mathbf{w} \in \mathcal{N}} \left| \mathbf{v}^\top \left(\sum_{s=1}^t (\mathbf{x}_s \mathbf{x}_s^\top - \mathbb{E}[\mathbf{x}_s \mathbf{x}_s^\top \mid \mathcal{F}_{s-1}]) \right) \mathbf{w} \right| \geq \varepsilon\right) \\ &\leq 2 \cdot 9^{2d} \exp\left\{-\frac{\varepsilon^2}{8 \sum_{s=1}^t (x_{\max}^2 + \|\Sigma_s^x\|_{op})^2}\right\}. \end{aligned}$$

Thus, we obtain the result. \square

A straight-forward consequence of this result is the following.

Corollary 5. *With probability 1, eventually in t it holds that*

$$\left\| \sum_{s=1}^t (\mathbf{x}_s \mathbf{x}_s^\top - \Sigma_s^x) \right\|_{op} \leq \sqrt{16 \log(t) \sum_{s=1}^t (x_{\max}^2 + \|\Sigma_s^x\|_{op})^2}.$$

Proof. Notice, if we set

$$\varepsilon_t = \sqrt{16 \log(t) \sum_{s=1}^t (x_{\max}^2 + \|\Sigma_s^x\|_{op})^2},$$

then,

$$\mathbb{P}\left(\left\| \sum_{s=1}^t (\mathbf{x}_s \mathbf{x}_s^\top - \Sigma_s^x) \right\|_{op} \geq \varepsilon_t\right) \leq \frac{2 \cdot 9^{2d}}{t^2}.$$

By the Borel–Cantelli Lemma, the result holds. \square

Lemma 34.

$$\mathbb{P}\left(\left\|\sum_{s=1}^t \alpha_s z_s \mathbf{x}_s^\top\right\|_{op} \geq \varepsilon\right) \leq 2 \cdot 9^{2d} \exp\left\{-\frac{(\varepsilon/2)^2}{2(z_{\max} x_{\max})^2 \sum_{s=1}^t \alpha_s^2}\right\}.$$

and thus, with probability 1, eventually it holds that

$$\left\|\sum_{s=1}^t \alpha_s z_s \mathbf{x}_s^\top\right\|_{op} \leq \sqrt{16z_{\max}^2 x_{\max}^2 \log(t) \sum_{s=1}^t \alpha_s^2} \quad (\text{A.7})$$

Proof. Similar to the proof of Lemma 33, we show this argument in two steps. We first control $\sum_{s=1}^t \alpha_s z_s \mathbf{x}_s^\top$ over a ε -net, and then extend the bound to the full supremum norm by a continuity argument. Notice that the summands of $\sum_{s=1}^t \alpha_s z_s \mathbf{x}_s^\top$ are a bounded martingale difference sequence.

Using Lemma 35 (stated below) and choosing $\varepsilon = \frac{1}{4}$ and, we can find an ε -net \mathcal{N} of the unit sphere S^{d-1} with cardinality

$$|\mathcal{N}| \leq 9^d.$$

By Lemma 36 (stated below), the operator norm can be bounded by terms on \mathcal{N} , that is

$$\begin{aligned} \left\|\sum_{s=1}^t \alpha_s z_s \mathbf{x}_s^\top\right\|_{op} &\leq 2 \max_{\mathbf{v}, \mathbf{w} \in \mathcal{N}} \left\langle \left(\sum_{s=1}^t \alpha_s z_s \mathbf{x}_s^\top\right) \mathbf{v}, \mathbf{w} \right\rangle \\ &\leq 2 \max_{\mathbf{v}, \mathbf{w} \in \mathcal{N}} \left| \mathbf{v}^\top \left(\sum_{s=1}^t \alpha_s z_s \mathbf{x}_s^\top\right) \mathbf{w} \right|. \end{aligned} \quad (\text{A.8})$$

We first fix $\mathbf{v}, \mathbf{w} \in \mathcal{N}$, and by Azuma–Hoeffding inequality, for any $\varepsilon > 0$ we can state that

$$\mathbb{P}\left(\left|\mathbf{v}^\top \left(\sum_{s=1}^t \alpha_s z_s \mathbf{x}_s^\top\right) \mathbf{w}\right| \geq \frac{\varepsilon}{2}\right) \leq 2 \exp\left\{-\frac{(\varepsilon/2)^2}{2 \sum_{s=1}^t (\alpha_s z_{\max} x_{\max})^2}\right\}.$$

Next, we unfix $\mathbf{v}, \mathbf{w} \in \mathcal{N}$ using a union bound. Since that \mathcal{N} has

cardinality bounded by 9^d , we obtain

$$\begin{aligned} \mathbb{P}\left(\max_{v,w \in \mathcal{N}} \left| v^\top \left(\sum_{s=1}^t \alpha_s z_s x_s^\top \right) w \right| \geq \frac{\varepsilon}{2}\right) &\leq \sum_{v,w \in \mathcal{N}} \mathbb{P}\left(\left| v^\top \left(\sum_{s=1}^t \alpha_s z_s x_s^\top \right) w \right| \geq \frac{\varepsilon}{2}\right) \\ &\leq |\mathcal{N}|^2 \cdot 2 \exp\left\{-\frac{(\varepsilon/2)^2}{2 \sum_{s=1}^t (\alpha_s z_{\max} x_{\max})^2}\right\} \\ &\leq 9^{2d} \cdot 2 \exp\left\{-\frac{\varepsilon^2}{8 \sum_{s=1}^t (\alpha_s z_{\max} x_{\max})^2}\right\}. \end{aligned}$$

Together with (A.8), we have

$$\begin{aligned} \mathbb{P}\left(\left\| \sum_{s=1}^t \alpha_s z_s x_s^\top \right\|_{op} \geq \varepsilon\right) &= \mathbb{P}\left(2 \max_{v,w \in \mathcal{N}} \left| v^\top \left(\sum_{s=1}^t \alpha_s z_s x_s^\top \right) w \right| \geq \varepsilon\right) \\ &\leq 2 \cdot 9^{2d} \exp\left\{-\frac{\varepsilon^2}{8 (z_{\max} x_{\max})^2 \sum_{s=1}^t \alpha_s^2}\right\}. \end{aligned}$$

Thus, we obtain the result. For (A.7), the argument follows in an identical manner to Corollary 5. \square

Lemmas 35 and 36 are stated below. For the proofs of Lemmas 35 and 36, we refer to Section 4 in **(author?)** [53].

Lemma 35 (Covering Numbers of the Euclidean ball). *The covering numbers of the unit Euclidean ball is such that, for any $\varepsilon > 0$*

$$\left(\frac{1}{\varepsilon}\right)^d \leq \mathcal{N} \leq \left(\frac{2}{\varepsilon} + 1\right)^d.$$

The same upper bound is true for the unit Euclidean sphere S^{d-1} .

Lemma 36 (Quadratic Form on a Net). *Let A be an $m \times n$ matrix and $\varepsilon \in [0, 1/2)$. For any ε -net \mathcal{N} of the sphere S^{n-1} and any ε -net \mathcal{M} of the sphere S^{m-1} , we have*

$$\sup_{x \in \mathcal{N}, y \in \mathcal{M}} \langle Ax, y \rangle \leq \|A\|_{op} \leq \frac{1}{1 - 2\varepsilon} \sup_{x \in \mathcal{N}, y \in \mathcal{M}} \langle Ax, y \rangle. \quad (\text{A.9})$$

Moreover, if $m = n$ and A is symmetric, then

$$\sup_{x \in \mathcal{N}} |\langle Ax, x \rangle| \leq \|A\|_{op} \leq \frac{1}{1 - 2\varepsilon} \sup_{x \in \mathcal{N}} |\langle Ax, x \rangle|.$$

Appendix B

Function Approximation

B.1 Overview of Statistical Learning

In this section we overview Statistical Learning which essentially considers finding good function approximations to noisy data.

We give a fairly general definition of a *supervised learning problem*. We assume an output $y \in \mathcal{Y}$ is a noisy function of an input $x \in \mathcal{X}$

$$y = f(x) + \epsilon,$$

here ϵ is a random variable with zero mean and we assume that the function $f : \mathcal{X} \rightarrow \mathcal{Y}$ a deterministic function. Notice this implies $f(x) = \mathbb{E}[y|x]$. (Typically \mathcal{X} and \mathcal{Y} are be finite-dimensional real-valued vector spaces.)

Data and Loss. Given data $\mathcal{D} = \{(x^{(n)}, y^{(n)}) : n = 1, \dots, N\}$ and a set of functions \mathcal{F} our goal is to selected a function $\hat{f} \in \mathcal{F}$ that approximates f . Often the set \mathcal{F} will be parameterized with a function f_θ for each parameter $\theta \in \mathbb{R}^p$. We use a real-valued *loss function* (or error function) $L(y, \hat{f}(x))$ to judge the error between an output y from an estimate from an input \hat{f} . A cannoical choice is a quadratic loss function

$$L(y, f(x)) = (y - f(x))^2,$$

but other choices can be used. Because we are provided with both input and output data, this setting is called supervised learning.

Goal. Given that the data \mathcal{D} is drawn IID with distribution equal to random variables (\hat{x}, \hat{y}) , our ultimate goal is to solve the optimization

$$\min_{g \in \mathcal{F}} \mathbb{E}[L(\hat{y}, g(\hat{x}))]$$

for a set \mathcal{F} that has a low minimum expected loss. However, we don't know the distribution of (\hat{x}, \hat{y}) . So we can only get an approximate solution of this from the available data \mathcal{D} and we must determine what set of functions \mathcal{F} makes efficient use of this finite set of data. What follows is an overview of standard approaches to this problem.

B.2 Linear Regression

Using data $(x^{(n)}, y^{(n)})$, $n = 1, \dots, N$, you want to approximate a real-valued output variable y from a p -dimensional input vector x . We

approximate y by a linear function of \mathbf{x} , namely,

$$\boldsymbol{\theta}^\top \mathbf{x} = \theta_0 + \theta_1 x_1 + \dots + \theta_K x_K .$$

Here $\mathbf{x} = (1, x_1, \dots, x_p)$. We choose the $\boldsymbol{\theta}$ that gives the least square distance between y and $\boldsymbol{\theta}^\top \mathbf{x}$ for each data point:

$$\text{minimize } \sum_{n=1}^N \left(y^{(n)} - \boldsymbol{\theta}^\top \mathbf{x}^{(n)} \right)^2 \quad \text{over } \boldsymbol{\theta} \in \mathbb{R}^{p+1} .$$

Optimal Weights. A short calculation gives that this minimization is solved by

$$\hat{\boldsymbol{\theta}} = (X^\top X)^{-1} X^\top \mathbf{y}$$

where we define the $p \times p$ matrix X , by $X_{ij} = \sum_n x_i^{(n)} x_j^{(n)}$ and we define $\mathbf{y} = (y^{(1)}, \dots, y^{(N)})^\top$. Using a singular value decomposition the complexity of finding $\hat{\boldsymbol{\theta}}$ is $O(Np^2)$.

Optimizing with big N. The inverse, $(X^\top X)^{-1}$ can involve too much calculation when N or p is big. An alternative is to do Stochastic Gradient Descent:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \left(y^{(n)} - \boldsymbol{\theta}^\top \mathbf{x}^{(n)} \right) \mathbf{x}^{(n)} .$$

So we move $\boldsymbol{\theta}$ in the direction of $\mathbf{x}^{(n)}$, one data point at a time. So we don't need to wait for our full calculation to get an estimate for $\boldsymbol{\theta}$. We could also apply asynchronous updates, so we can parallelize is the number of parameters p get big too.

Basis functions. Linear regression does not need y to be approximated by linear function of \mathbf{x} , nor does \mathbf{x} need to be finite dimensional, but we do need a linear relationship with respect to $\boldsymbol{\theta}$.

Suppose for each \mathbf{x} we take p different *features* using the *basis functions* $\phi_j(\mathbf{x})$, $j = 1, \dots, p$. We can then perform linear regression on

$$\boldsymbol{\theta}^\top \boldsymbol{\phi} = \theta_0 + \theta_1 \phi_1(\mathbf{x}) + \dots + \theta_p \phi_p(\mathbf{x}) .$$

The least squares distance is given by

$$\hat{\boldsymbol{\theta}} = (\Phi^\top \Phi)^{-1} \Phi^\top \mathbf{y}$$

where we define the $p \times p$ matrix Φ , by $\Phi_{ij} = \sum_n \phi_i(\mathbf{x}^{(n)}) \phi_j(\mathbf{x}^{(n)})$.

Polynomial Regression. A good example is polynomial regression. Here we can model y as a polynomial function of x , with the basis functions $\phi_0(x) = 1, \phi_1(x) = x, \phi_2(x) = x^2, \dots, \phi_p(x) = x^p$.

Regularization. Often it is important to reduce the complexity of a model. (We will discuss why you might want to do this in more detail in the next section.) For linear regression, this can be achieved by penalizing large values of θ as follows:

$$\text{minimize } \sum_{n=1}^N (y^{(n)} - \theta^\top \mathbf{x}^{(n)})^2 + \underbrace{\lambda \|\theta\|_2^2}_{\text{penalty term}} \quad \text{over } \theta \in \mathbb{R}^{p+1}.$$

where $\|\theta\|_2^2 = \sum_j \theta_j^2$ is the L_2 norm. This is known as *ridge regression* or L_2 -regularization.¹ We could use the L_1 norm

$$\|\theta\|_1 = \sum_j |\theta_j|.$$

This would be called Lasso or, when applied to other models, L_1 -regularization. Notice the contours of the L_1 norm are diamonds (rather than circles for the L_2 norm) so since optimal solutions tend to end up on the corners of these diamonds, we are more likely to end up setting some variables to be zero. In this way Lasso controls the number of variables used in a statistical model.

Regularization and Optimization. When L_2 regularization is applied to linear regression this optimal solution now satisfies

$$\hat{\theta} = (X^\top X + \lambda I)^{-1} X^\top \mathbf{y}$$

Notice before we could not guarantee that the inverse $(X^\top X)^{-1}$ existed but with regularization the inverse is always defined. It has a positive effect on numerical stability.

Notice when applying stochastic gradient descent we have

$$\theta \leftarrow (1 - \alpha\lambda)\theta - \alpha(y^{(n)} - \theta^\top \mathbf{x}^{(n)})\mathbf{x}^{(n)}.$$

Essentially the $(1 - \alpha\lambda)$ term means we apply a dampening effect on the stochastic gradient descent update.

¹We use ridge regression when applied to linear regression, where L_2 regularization is the idea of applying a penalty like this to any regression model.

Notice we do not explicitly limit the range of values of θ though we know (from Lagrangian optimization) that adding a penalty such as λ is equivalent to adding a constraint. So essentially we solve the optimization

$$\begin{aligned} & \text{minimize} && \sum_{n=1}^N \left(y^{(n)} - \theta^\top x^{(n)} \right)^2 \\ & \text{subject to} && \|\theta\|_2^2 \leq \kappa \\ & \text{over} && \theta \in \mathbb{R}^{p+1}. \end{aligned}$$

In this way we can think of regularization as reducing the complexity of our model. As the number of models that we are considering is reduced.

B.3 Training, Development and Test sets

Using training, development and test sets and evaluating their error is one of the most practical ways to get the most out of a machine learning algorithm.

We want to understand how much loss/error removed in from our predictions as the amount of data gets larger. The empirical loss over a data set \mathcal{D} is given by

$$\hat{\mathbb{E}}_{\mathcal{D}}[L(\hat{y}, f(\hat{x}))] := \frac{1}{|\mathcal{D}|} \sum_{(x,y) \in \mathcal{D}} L(y, x).$$

Here the expectation is taken over the empirical distribution of the data $\hat{\mathbb{P}}_{\mathcal{D}}$. Specifically we calculate mean loss over the data.

Given you have a finite set of data, if it is often worth separating your data $\mathcal{D} = \{(x^{(n)}, y^{(n)}) : n = 1, \dots, N\}$ into a *training set*, $\mathcal{D}_{\text{train}}$, a *development (dev) set* \mathcal{D}_{dev} and a *test set*, $\mathcal{D}_{\text{test}}$. The training set is the data that you give to your regression algorithm to fit with. The dev set you use to evaluate and move around other parameters, such as the number of features p or the learning rate α (and in general the size and parameters of your model). While the test set is a data that keep for later to evaluate your regression fit, once you have fit your regression model with all of your parameters fixed. Thus the test set is completely fresh data; it is not seen by your regression algorithm and you should not use it for choosing model parameters.

This is because you may want to test the level of performance of your regression fit. (In general, it is not good to use data used to fit the regression model as the regression parameters depend on this data. In order to get an unbiased evaluation of your regression fit, it is good to hold some data back for this.)

The development set is sometimes called the validation set. In a reinforcement learning setting using simulation, i.e. where you do not have a finite set of data and it is inexpensive to get new data, you can always simulate new data for training, developing, testing.

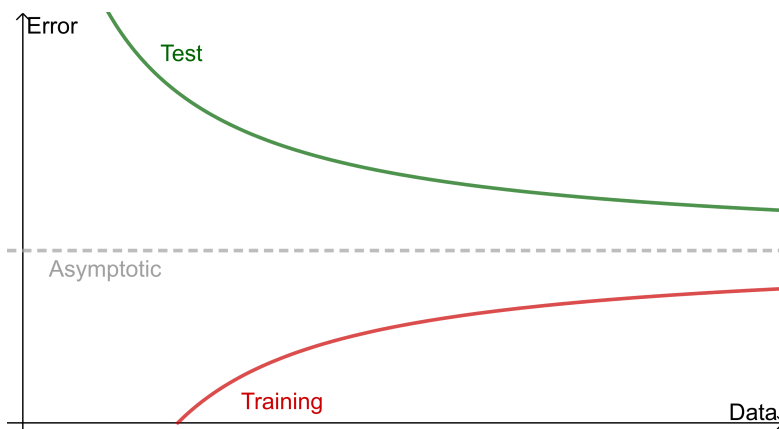


Figure B.1: Training Error (red) and Test Error (green) approach the Asymptotic Error as Data increases.

Here you fit your regression model with the training set, but *don't* use data from the test set. You can then evaluate the training and test error:

$$\begin{aligned} \text{Training Error} &= \hat{\mathbb{E}}_{\mathcal{D}_{\text{train}}} [L(\hat{y}, f(\hat{x}))] \\ \text{Dev Error} &= \hat{\mathbb{E}}_{\mathcal{D}_{\text{dev}}} [L(\hat{y}, f(\hat{x}))] \\ \text{Test Error} &= \hat{\mathbb{E}}_{\mathcal{D}_{\text{test}}} [L(\hat{y}, f(\hat{x}))] \\ \text{Asymptotic Error} &= \min_{\theta} \mathbb{E} [L(\hat{y}, f(\hat{x}))] \end{aligned}$$

Further we might have some desired level of performance. We refer to this as the target error.

The asymptotic error assumes that the data in $\mathcal{D}_{\text{test}}$ and $\mathcal{D}_{\text{train}}$ are IID samples of a random variable (\hat{x}, \hat{y}) with expectation \mathbb{E} . Under this assumption, the asymptotic error is the best fit we can get for that regression model.

Which Error is bigger? Notice we expect training error to be lower than the test and asymptotic area, since a good regression will attempt to minimize the loss of the data that it has seen (while it can't do as much with data that it has not seen). The test error should be higher than the asymptotic error, because the distribution \mathbb{E} is more representative of the training data than the test data. So we have

$$\text{Training Error} < \text{Asymptotic Error} < \text{Test Error}$$

Overfitting. We can think of the difference between the test error and asymptotic error, as the amount of additional error we introduce by having a finite data set. We could refer to the as the amount that we have overfit the data:

$$\text{overfit} = \text{Asymptotic Error} - \text{Training Error}$$

When this problem becomes chronic this could be referred to as overfitting (our model is optimizing the individual data points rather than optimizing the underlying distribution of the data). We do not usually have access to the distribution generating the data, so we cannot evaluate the amount of overfit. Instead we have to diagnose this symptomatically with other metrics such as the difference between the test and training error, sometimes referred to as the generalization error:

$$\text{Generalization Error} = \text{Test Error} - \text{Training Error}$$

What size should $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ be? There is no fixed rule and plenty more could be said here, but a common rule of thumb is to have 80% of data in your training set and 20% in your test set. Notice if we are working with simulated data that is easy to compute then we are less constrained proportioning to our training set.

Error as N increases. See Figure B.1. In general the training error increases as the amount of data, N , increases. This because our fix set of parameters have more points to fit. The test error goes down, as more the data used for train is more representative of the true distribution of (\hat{x}, \hat{y}) . Both should tend towards the asymptotic error.

Here we assume θ is the (optimal) least square fit for the data. However, if we used stochastic gradient descent, then we would not

have the optimal parameters for the data so the the relationship between test and training error is less clear cut. Generally the training and test error will decrease. If training error decreasing and test error increase this is a symptom of overfitting (to be discussed shortly).

Error as p increases. If we let the number of parameters of our regression model get big, then it is more expressive. Eg. We can fit more function to a degree 20 polynomial and a degree 2 polynomial. So we expect the asymptotic error to decrease as we let p get bigger. A good thing right? Well, no, not always. A degree 20 polynomial will wiggle around more and thus might not express the underlying structure of the distribution generating the data. See the two figures below.

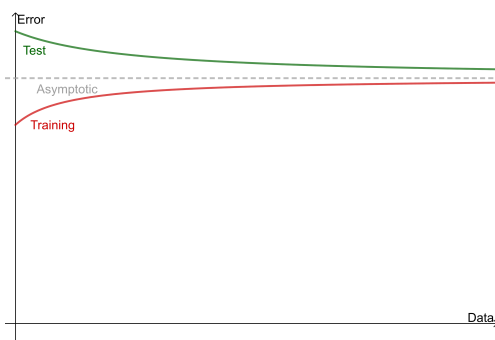


Figure B.2: A Simple Model

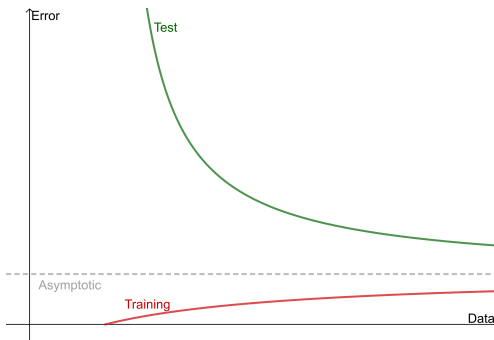


Figure B.3: A Complex Model

B.4 Bias and Variance.

Consider a statistical learning problem, $y = f(x) + \epsilon$. We are interested in how well on average our trained regression model approximates underlying model, this is called the Bias. And we are interested how much a fitted model is just learning the inherent variability in the data (rather than the underlying model), this is called the Variance. For a quadratic loss function, we can give a conceptually nice decomposition for these terms.

Bias-Variance Decomposition. Suppose as a function of data \mathcal{D} , we choose an estimate $\hat{f} \in \mathcal{F}$. We let \bar{f} be the expected value of \hat{f} ,

$$\bar{f}(x) = \mathbb{E}_{\mathcal{D}}[\hat{f}(x)].$$

Here the expectation is over a random sample of data \mathcal{D} . The bias and variance of \hat{f} at x are given by

$$\text{bias}_x(\hat{f}) = (\bar{f}(x) - f(x))^2 \quad \text{var}_x(\hat{f}) = \mathbb{E}_{\mathcal{D}} \left[(\hat{f}(x) - \bar{f}(x))^2 \right].$$

We can also take an expectation over the distribution of \hat{x} to get the bias and variance of \hat{f} . I.e.

$$\text{bias}(\hat{f}) = \mathbb{E}_{\hat{x}} \left[(\bar{f}(\hat{x}) - f(\hat{x}))^2 \right], \quad \text{var}(\hat{f}) = \mathbb{E}_{\mathcal{D}, \hat{x}} \left[(\hat{f}(\hat{x}) - \bar{f}(\hat{x}))^2 \right].$$

I.e. in the expectation above, we fit the data \mathcal{D} and then sample one more point \hat{x} so see how well the fit does.

The bias gives how close the estimator's mean value \bar{f} is to the true value f . The variance of the estimator \hat{f} gives how spread out the estimator is against its mean.

The following lemma shows that the mean of the quadratic loss is the sum of the bias and variance (plus some irreducible randomness).

Lemma 37 (Bias-Variance Decomposition).

$$\mathbb{E}_{(\hat{x}, \hat{y}), \mathcal{D}} \left[(\hat{y} - \hat{f}(\hat{x}))^2 \right] = \text{bias}(\hat{f}) + \text{var}(\hat{f}) + \text{var}(\epsilon).$$

Proof. Since $\hat{y} = f(\hat{x}) + \epsilon$, we can expand as follows

$$\begin{aligned} \mathbb{E} \left[(\hat{y} - \hat{f})^2 \right] &= \mathbb{E} \left[(\epsilon + f - \bar{f} + \bar{f} - \hat{f})^2 \right] \\ &= \underbrace{\mathbb{E}[\epsilon^2]}_{\text{var}(\epsilon)} + \underbrace{2\mathbb{E}[\epsilon]\mathbb{E}[(f - \bar{f} + \bar{f} - \hat{f})]}_{=0, \text{ independence and } E[\epsilon]=0} \\ &\quad + \underbrace{(f - \bar{f})^2}_{\text{bias}_x(\hat{f})} + 2(f - \bar{f}) \underbrace{\mathbb{E}[(\bar{f} - \hat{f})]}_{=0} + \underbrace{\mathbb{E}[(\bar{f} - \hat{f})^2]}_{\text{var}_x(\hat{f})}. \\ &= \text{bias}(\hat{f}) + \text{var}(\hat{f}) + \text{var}(\epsilon). \end{aligned}$$

□

Bias-Variance Tradeoff. We now consider this situation where the amount of data is fixed and we fit a series of models of varying complexity. Here consider adding an axis for model complexity to Figures B.2 and B.3 and take a cross section where the amount of data is fixed. We get a picture like Figure B.4.

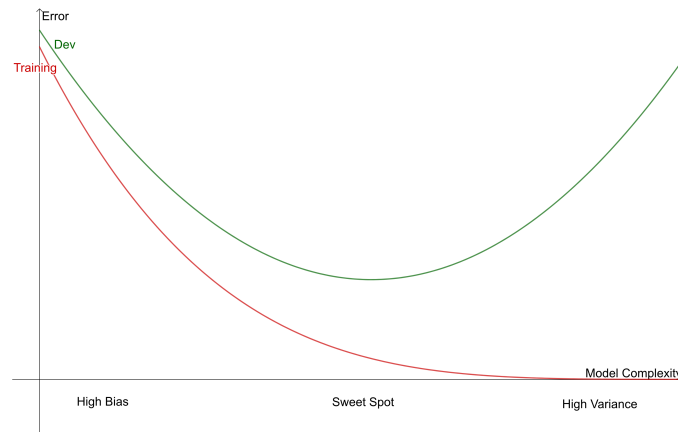


Figure B.4: Training and test error for fixed amount of data.

We see that when we fit a simple model, we are prone to have high dev and training error which are close together. I.e. High bias and low variance. If we have a complex model, then we have low training error, but high dev error. I.e. low bias but high variance. In the middle, we see there seems to be a sweet spot. This is a better predictive model for this amount of data.

Notice here we use the Dev set as we using it to decide which parameters to use. We use the test set at the end to double check that the fit is good and agrees with the errors given by the dev set.

Informal use. The bias variance decomposition is proven for a quadratic function. That said we can draw learning curves like Figure B.1 and Figure B.4 for any loss function. Thus often people refer to bias and variance more informally often in reference to some desired error that you wish to target. High bias meaning that your training error is far from your target error, and high variance meaning your test/dev error is far from your training error. (Note that bias and variance here are conflated with asymptotic error and generalization error from earlier.²)

²These ambiguities appear to be from an attempt to shoehorn the quick and

	Target Error	Train Error	Dev Error	Diagnosis	Remedy
A.	1%	15%	16%	High bias	increase model complexity with bigger model
B.	1%	0.9%	30%	High variance	Get more data if not: decrease complexity with regularization.
C.	1%	15%	30%	High bias & variance	1st. More data 2nd. Bigger model 3rd. new model.
D.	1%	0.9%	1.1%	low bias & variance	stop done.

Table B.1: Table of error symptoms, diagnosis and remedy for learning problems.

Diagnostics. We will stick with the informal terminology from above. We can use these to form some diagnostics, remedies and a procedure for getting a good model. (Here we are assuming that it is relatively straight forward to generate new data and issues such as convergence to an optimal fit have been resolved.) Below is a table with some scenarios.

- High bias, low variance: we have fitted a model. Variance between test and training is low, so we have a good fit, but high bias suggests the model is not expressive enough to represent the data. Suggestion is to add more data.
- Low bias, high variance: the test error is low but the training error is high. This is a sign the model is expressive enough to fit to the data (maybe too expressive), adding more data will bring down the variance (and potentially up the bias). If there is no more data, increasing regularization parameter is a good alternative.

conceptually easy Bias-Variance calculation into the more deep and general, but conceptually harder, theory of statistical learning from Vapnik and Chervonenkis – this is out of scope for us for now.

- High bias, high variance: a bad outcome, basically the model has not been fitted yet so you should get more data. This should reduce variance. If this still does not work you could regularize to reduce bias. If this does not work you should consider a new model to get low bias.
- Low bias and low variance: suggest you have a good working model.

Figure, gives a flow diagram where you go from high bias to low bias with high variance to a working model. The right handside discussed remedies for high bias and variance.

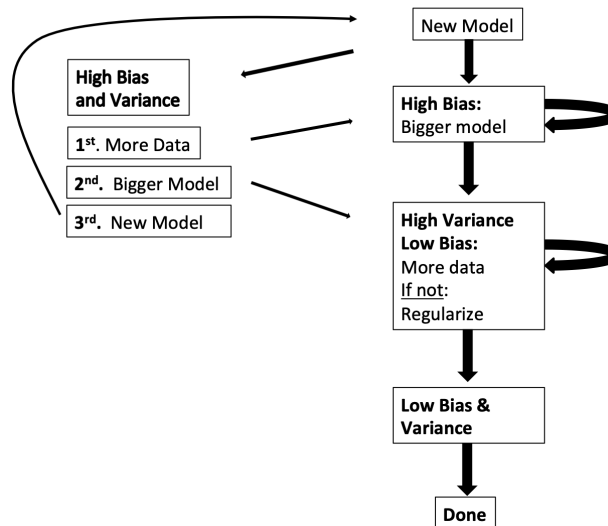


Figure B.5: Flow chart for getting a working machine learning model.

B.5 Neural Networks

A neural network (or artificial neural network) is a generalization of regression. A neural network consists of connecting together a number of artificial neurons.

Neuron. A neuron considers of a number of input parameters $x = (1, x_1, \dots, x_p)$ a weight w is applied to each of these and then this is applied to a function to give our prediction:

$$\hat{y} = f(\mathbf{w}^\top \mathbf{x})$$

The function $f(z)$ is often called an activation function. This is represented in Figure B.6

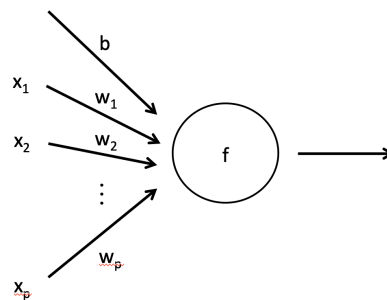


Figure B.6: A Neuron in a Neural network.

Since the amount of data used is typically quite large, neural networks are trained using stochastic gradient descent (and a number of other specialized variants). For this we need to differentiate. We can differentiate this as follows:

$$\frac{\partial \bar{y}}{\partial w_j} = x_j f'(\mathbf{w}^\top \mathbf{x}).$$

Slight more involved calculations will be used when we consider networks of neurons. But first let's quickly consider choices for the activation function $f(x)$.

Activation Functions. There are various choices of activation functions f . Some common choices are given in Table B.2

The table shows three rough categories: Binary, linear and max. Each serves a slightly different purpose. Binary helps for yes/no

Name	Formula, $f(z)$
Binary	$\begin{cases} 0, & \text{for } z < 0 \\ 1, & \text{for } z \geq 0. \end{cases}$
Logistic	$\frac{1}{1+e^{-z}}$
Tanh	$\frac{e^z - e^{-z}}{e^z + e^{-z}}$
Linear	z
ReLU	$\begin{cases} 0, & \text{for } z < 0 \\ z, & \text{for } z \geq 0. \end{cases}$
Softplus	$\log(1 + e^x)$
Max	$\max_i z_i$
Softmax	$\frac{e^{z_i}}{\sum_j e^{z_j}}$

Table B.2: List of some activation functions.

decisions. Linear (and particularly ReLU) is good for gauging the intensity of an activation (above a certain level). Max determine the most likely variable. The activations have differentiable analogues in each category. E.g. the logistic function is differentiable, unlike the binary function. This allows us to apply stochastic gradient descent.

Notice, max (and softmax) neurons are not applied to weighted sums of inputs, like $\sum_i w_i x_i$. They are directly applied to each input x_i . For this reason they are usually only used in the final layers of a deep neural network.

A Two Layer Network. We now discuss combining neurons together to make a neural network. Traditionally, a neural network would have consisted of a set of inputs, a single layer of neurons and an output layer. See Figure B.7 for an instance of a two-layer neural network.

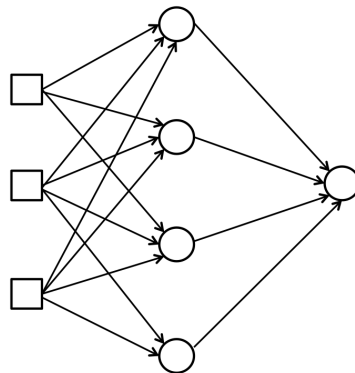


Figure B.7: A two-layer neural network. Here squares give inputs and circles are neurons.

Here the inputs are each given to different intermediate neurons. The output from each intermediate neuron is then fed into an output neuron that then gives a prediction. I.e. Given inputs x and weights w our prediction \hat{y} is

$$\hat{y} = f^{(2)}(z^{(2)}) \quad \text{where} \quad z^{(2)} = \sum_{j=1}^l a_j^{(1)} w_j^{(2)},$$

and

$$a_j^{(1)} = f_j^{(1)}(z_j^{(1)}) \quad \text{where} \quad z_j^{(1)} = \sum_k w_{jk}^{(1)} x_k$$

Here $f^{(k)}$ and $w^{(k)}$ are the activation functions and weights applied at the k -th layer. From a theoretical perspective, if l is chosen sufficiently large then appropriate weights can be chosen to any continuous function bounded function with arbitrary precision. However, in practice (and maybe in theory too), it is likely some structures work get a closer approximation with less parameters.

Deep Neural Networks. Due to success in many practical applications. People often consider neural networks with multiple layers, $l = 1, \dots, l'$. (Meaning, roughly, between 3 and 300 layers). Here a layer, l , is a set of activation functions $f_i^{(l)}$, $i = 1, \dots, n_l$, and each activation receives, as input, the output from the previous layer. These are called Deep Neural Networks. See Figure B.8.

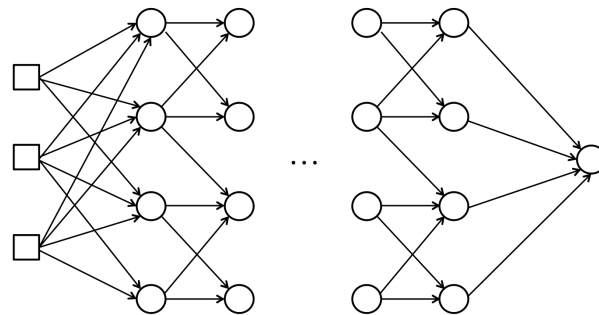


Figure B.8: A deep neural network.

As each layer is fed a weighted linear combination of the activation output from a previous layer we have:

$$a_j^{(l)} = f_j^{(l)}(z_j^{(l)}) \quad \text{where} \quad z_j^{(l)} = \sum_i a_i^{(l-1)} w_{ij}^{(l)}$$

starting initially with the inputs, $a_i^{(0)} = x_i$. (Implicitly we assume one of the activations is set equal to 1, so that we can have a constant input). Notice here really what we are doing is repeatedly composing linear combinations of functions. Notice we do not have any loops in our networks, i.e. we always work with directed acyclic graphs (aka. DAGs).

Backpropagation. To train a deep neural network we need to be able to optimize our weights. To do this different variants of stochastic gradient descent are applied. To apply (stochastic) gradient descent we need gradients. Essentially since a neural network is a

composition of functions we apply the chain rule from calculus. A conceptually useful way to organize these chain rule calculation for a neural network is Backpropagation, which we now define.

Suppose we are given data piece of data (x, y) i.e. an input vector x and an output number y . We can get a prediction \bar{y} from x by recursively applying our above formula:

$$a_j^{(l)} = f_j^{(l)}\left(\sum_i a_i^{(l-1)} w_{ij}^{(l)}\right). \quad (\text{B.1})$$

starting initially with the inputs, $a_i^{(0)} = x_i$ and ending up with a prediction $\bar{y} = a^{(l')}$ where l' is the final output layer of our neural network. We can then evaluate the loss between our prediction and the observed outcome:

$$L(y, \bar{y}).$$

We need optimize the weights of our deep neural network. Rather than recursively applying forward the formula (B.1), we must apply backward a related formula. Hence this takes its name as "back-propagation". For this, let's define

$$z_j^{(l)} := \sum_i a_i^{(l-1)} w_{ij}^{(l)} \quad \text{and} \quad \delta_j^{(l)} := \frac{\partial L}{\partial z_j^{(l)}}.$$

Note that $a_j^{(l)} = f_j^{(l)}(z_j^{(l)})$ and using notation $\dot{a}_j^{(l)} = \frac{df_j^{(l)}}{dz_j^{(l)}}$, we have that

$$\frac{\partial z_k^{(l+1)}}{\partial z_j^{(l)}} = \dot{a}_j^{(l)} w_{jk}^{(l+1)}.$$

I.e. we can calculate partial derivatives for layer $l+1$ from derivatives of the activations functions at layer l , and notice if we have the partial derivatives with respect to $z_j^{(l)}$ it is easier to calculate the other quantities that we want. Specifically,

$$\frac{\partial L}{\partial w_{ij}^{(l)}} = \frac{\partial z_j^{(l)}}{\partial w_{ij}^{(l)}} \frac{\partial L}{\partial z_j^{(l)}} = a_i^{(l-1)} \delta_j^{(l)}.$$

Now we can apply to the chain rule to get the derivatives with respect

to $z_j^{(l)}$:

$$\begin{aligned} \delta_j^{(l)} &= \sum_k \frac{\partial z_k^{(l+1)}}{\partial z_j^{(l)}} \frac{\partial L}{\partial z_k^{(l+1)}} \\ &= \sum_k \hat{a}_j^{(l)} w_{jk}^{(l+1)} \delta_k^{(l+1)} \end{aligned} \tag{B.2}$$

Notice once we have obtained the values of each $z_j^{(l)}$ in our forward pass using (B.1), we can work backwards using (B.2) from the final layer to the first to calculate each $\delta_j^{(l)}$.

The interactions between these formulas in Backpropagation can be summarized in Figure B.9. The key terms and formulas in backpropagation are summarized in Table B.3, below.

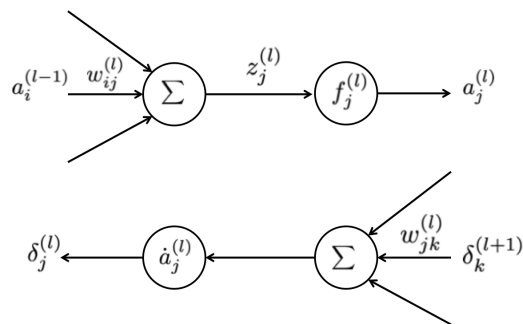


Figure B.9: Forward and backward pass of Backpropagation.

References

Everything on statistical learning, linear regression, bias-variance and neural networks is by now *very* standard machine learning. See Friedman et al. [18], Murphy [35] or Goodfellow [20].

Backpropagation Definitions	Formulas
$z_j^{(l)} := \sum_i a_i^{(l-1)} w_{ij}^{(l)}$	$a_j^{(l)} = f_j^{(l)}(z_j)$ (forward pass)
$a_j^{(0)} := \hat{x}_j$	$\delta_j^{(l)} = \sum_k \dot{a}_j^{(l)} w_{jk}^{(l+1)} \delta_k^{(l+1)}$ (backward pass)
$\delta_j^{(l)} := \frac{\partial L}{\partial z_j^{(l)}}$	$\frac{\partial L}{\partial w_{ij}^{(l)}} = a_i^{(l-1)} \delta_j^{(l)}$.

Table B.3: Backpropagation Definitions.

Index

Bellman Equation, 9, 33, 38, 44

Closed Stopping Set, 74

discount factor, 37

Dynamic Program, 9

Markov Decision Process, 32

Natural Gradient., 163

OLSA, 73

One-Step-Look-Ahead, 73

Optimal Stopping Problem, 73

Plant Equation, 8, 31

Policy Evaluation, 55

Policy Improvement, 55

Policy Iteration, 61

Q-Factor, 38

Reinforcement Learning, 208

The Secretary Problem, 76

Value Function, 32

Value Iteration, 56

Bibliography

- [1] Karl J Astrom. Optimal control of markov processes with incomplete state information. *Journal of mathematical analysis and applications*, 10(1):174–205, 1965.
- [2] Francis Bach and Eric Moulines. Non-strongly-convex smooth stochastic approximation with convergence rate $O(1/n)$ arXiv : 1306 . 2119v1 [cs . LG] 10 Jun 2013. pages 1–42, 2013.
- [3] Leemon C Baird III. Advantage updating. Technical report, WRIGHT LAB WRIGHT-PATTERSON AFB OH, 1993.
- [4] Richard Bellman. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–515, 1954.
- [5] Albert Benveniste, Michel Métivier, and Pierre Priouret. *Adaptive algorithms and stochastic approximations*, volume 22. Springer Science & Business Media, 2012.
- [6] Dimitri P Bertsekas, Dimitri P Bertsekas, Dimitri P Bertsekas, and Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA, 1995.
- [7] Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-dynamic programming*. Athena Scientific, 1996.
- [8] Jalaj Bhandari and Daniel Russo. Global optimality guarantees for policy gradient methods. *arXiv preprint arXiv:1906.01786*, 2019.
- [9] David Blackwell. Discounted Dynamic Programming. *Ann. Math. Statist.*, 36(1):226–235, 1965.

- [10] David Blackwell. Positive dynamic programming. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, volume 11, pages 415–418, Berkeley, Calif., 1967. University of California Press.
- [11] Vivek S Borkar. *Stochastic approximation: a dynamical systems viewpoint*, volume 48. Springer, 2009.
- [12] P Bremaud. *Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues*. Texts in Applied Mathematics. Springer New York, 2013.
- [13] Richard S Bucy and Peter D Joseph. *Filtering for stochastic processes with applications to guidance*. American Mathematical Soc., 1968.
- [14] Olivier Cappé, Eric Moulines, and Tobias Rydén. *Inference in hidden Markov models*. Springer Science & Business Media, 2006.
- [15] Y S Chow, H Robbins, and D Siegmund. *Great expectations: the theory of optimal stopping*. Houghton Mifflin, 1971.
- [16] Mark HA Davis. Martingale methods in stochastic control. In *Stochastic Control Theory and Stochastic Differential Systems*, pages 85–117. Springer, 1979.
- [17] J L Doob. *Classical Potential Theory and Its Probabilistic Counterpart: Advanced Problems*. Grundlehren der mathematischen Wissenschaften. Springer New York, 1984.
- [18] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.
- [19] Paul Glasserman. *Monte Carlo method in financial engineering*. 2004.
- [20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [21] M S Grewal and A P Andrews. *Kalman filtering: theory and practice with MATLAB, 4th*. Wiley, 2014.

- [22] P Hall, C C Heyde, Z W Birnbaum, and E Lukacs. *Martingale Limit Theory and Its Application*. Communication and Behavior. Elsevier Science, 2014.
- [23] Hado V Hasselt. Double q-learning. In *Advances in neural information processing systems*, pages 2613–2621, 2010.
- [24] R Howard. *Dynamic programming and Markov processes*. 1960.
- [25] Tommi Jaakkola, Michael I. Jordan, and Satinder P. Singh. On the Convergence of Stochastic Iterative Dynamic Programming Algorithms. *Neural Computation*, 1994.
- [26] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.
- [27] H.K. Khalil. *Nonlinear Systems*. Pearson Education. Prentice Hall, 2002.
- [28] Harold Kushner and G George Yin. *Stochastic approximation and recursive algorithms and applications*, volume 35. Springer Science & Business Media, 2003.
- [29] Harold Joseph Kushner and Dean S Clark. *Stochastic approximation methods for constrained and unconstrained systems*, volume 26. Springer Science & Business Media, 1978.
- [30] Joseph LaSalle. Some extensions of liapunov’s second method. *IRE Transactions on circuit theory*, 7(4):520–527, 1960.
- [31] D A Levin, Y Peres, and E L Wilmer. *Markov Chains and Mixing Times*. American Mathematical Soc.
- [32] Daniel Liberzon. *Calculus of variations and optimal control theory: a concise introduction*. Princeton university press, 2011.
- [33] Ljung. Analysis of recursive stochastic algorithms. *IEEE Transactions on Automatic Control*, 22(4):551–575, 1977.
- [34] Aleksandr Mikhailovich Lyapunov. The general problem of the stability of motion. *International journal of control*, 55(3):531–534, 1992.

- [35] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.
- [36] J.R. Norris and J.R. Norris. *Markov Chains*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1998.
- [37] M L Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley, 1994.
- [38] H Robbins and S Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22:400–407, 1951.
- [39] Herbert Robbins and David Siegmund. A convergence theorem for non negative almost supermartingales and some applications. In *Optimizing methods in statistics*, pages 233–257. Elsevier, 1971.
- [40] Jeffrey S Rosenthal. Convergence rates for markov chains. *Siam Review*, 37(3):387–405, 1995.
- [41] Burhaneddin Sandikci. Reduction of a pomdp to an mdp. *Wiley Encyclopedia of Operations Research and Management Science*, 2010.
- [42] Herbert Scarf. The optimality of (s,s) policies in the dynamic inventory problem. *Optimal pricing, inflation, and the cost of price adjustment*, 1959.
- [43] A N Shiryaev. *Optimal Stopping Rules*. Stochastic Modelling and Applied Probability. Springer Berlin Heidelberg, 2009.
- [44] Rayadurgam Srikant. *The mathematics of Internet congestion control*. Springer Science & Business Media, 2012.
- [45] Ralph E Strauch. Negative Dynamic Programming. *Ann. Math. Statist.*, 37(4):871–890, 1966.
- [46] Charlotte Striebel. Sufficient statistics in the optimum control of stochastic systems. *Journal of Mathematical Analysis and Applications*, 12(3):576–592, 1965.
- [47] Richard S. Sutton. Learning to Predict by the Methods of Temporal Differences. *Machine Learning*, 1988.

- [48] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning, Second Edition: An Introduction*. 2018.
- [49] John N Tsitsiklis. Asynchronous stochastic approximation and Q-learning. *Machine Learning*, 16(3):185–202, sep 1994.
- [50] John N. Tsitsiklis and Benjamin Van Roy. Analysis of temporal-difference learning with function approximation. In *Advances in Neural Information Processing Systems*, 1997.
- [51] John N. Tsitsiklis and Benjamin Van Roy. Optimal stopping of Markov processes: Hilbert space theory, approximation algorithms, and an application to pricing high-dimensional financial derivatives. *IEEE Transactions on Automatic Control*, 1999.
- [52] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*, 2016.
- [53] Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.
- [54] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2015.
- [55] Ward Whitt. Proofs of the martingale FCLT. *Probability Surveys*, 2007.
- [56] Peter Whittle. *Optimization over time*. John Wiley & Sons, Inc., 1982.
- [57] D. Williams. *Probability with Martingales*. Cambridge mathematical textbooks. Cambridge University Press, 1991.
- [58] Martin Zinkevich. Online Convex Programming and Generalized Infinitesimal Gradient Ascent. In *ICML*, pages 928–936, 2003.